Учреждение образования Федерации профсоюзов Беларуси
«Международный университет «МИТСО»

Факультет экономический
Кафедра информационных технологий

СОГЛАСОВАНО
Заведующий кафедрой
_____О.Б.Хорошко
30.10 2025 г.

СОГЛАСОВАНО
Декан факультета
_____А.В.Ковтунов
16.12. 2025 г.

# ЭЛЕКТРОННЫЙ УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

# WEB-ТЕХНОЛОГИИ

**для специальности 1-26 03 01 «Управление информационными ресурсами»**

Составители: Пекарь А.С., преподаватель кафедры информационных технологий учреждения образования Федерации профсоюзов Беларуси «Международный университет «МИТСО»
Пархимович А.В., преподаватель кафедры информационных технологий учреждения образования Федерации профсоюзов Беларуси «Международный университет «МИТСО»

Рассмотрено и рекомендовано к утверждению на заседании кафедры информационных технологий учреждения образования Федерации профсоюзов Беларуси «Международный университет «МИТСО»
30.10.2025 г., протокол № 3

Утверждено на заседании научно-методического совета учреждения образования Федерации профсоюзов Беларуси «Международный университет «МИТСО»
16.12.2025 г., протокол № 2

**РЕЦЕНЗЕНТЫ:**
Кафедра микропроцессорных систем и сетей Института информационных технологий УО «Белорусский государственный университет информатики и радиоэлектроники»;
Гришко Н.И., доцент кафедры экономики и менеджмента учреждения образования федерации профсоюзов Беларуси «Международный университет «МИТСО», кандидат технических наук, доцент

Регистрационный № УД-011-26/э

Регистрационное свидетельство № 1202645905 от 03.02.2026

**АКТУАЛИЗИРОВАН**
Рассмотрено на заседании кафедры информационных технологий учреждения образования Федерации профсоюзов Беларуси «Международный университет «МИТСО»
_____2025 г., протокол № ___

Утверждено на заседании научно-методического совета учреждения образования Федерации профсоюзов Беларуси «Международный университет «МИТСО»
_____2025 г., протокол № ___

# ОГЛАВЛЕНИЕ

# ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Электронный учебно-методический комплекс (далее – ЭУМК) подготовлен в соответствии с учебной программой Учреждения образования Федерации профсоюзов Беларуси «Международный университет «МИТСО» по дисциплине «WEB-технологии» для направления специальности 1-26 03 01 «Управление информационными ресурсами».

Цель ЭУМК – обеспечение максимальной доступности и конкретности изучаемого материала по дисциплине. Практическая значимость состоит в том, что данный ЭУМК позволяет конкретизировать и спланировать учебную деятельность студентов, разнообразить формы усвоения достаточно сложного материала.

Основными структурными элементами ЭУМК являются:

• учебная программа по дисциплине;

• теоретический раздел, включающий конспект лекций для теоретического изучения учебной дисциплины;

• практический раздел, содержащий учебно-методические материалы для поведения лабораторных работ;

• раздел контроля знаний, в котором представлены контрольные вопросы по дисциплине, вопросы для самостоятельного изучения по всем предусмотренным учебной программой темам дисциплины;

• вспомогательный раздел, в котором приведены рекомендации по освоению дисциплины.

Рекомендации по организации работы с ЭУМК. Для лучшего усвоения получаемых знаний слушателям рекомендуется сначала ознакомиться с учебной программой по дисциплине «WEB-технологии». Усвоение материала по каждой из тем рекомендуется начинать с изучения соответствующей лекции («Теоретический раздел»), после чего следует ответить на вопросы для самоконтроля и выполнить задания лабораторных работ («Практический раздел»). Проверка знаний проводится демонстрацией созданных программ, ответов на вопросы из перечней вопросов для опросов и вопросов к экзамену «Раздел контроля знаний». Студентам рекомендуется использовать материалы ЭУМК для осуществления активной самостоятельной работы при изучении дисциплины, подготовки к аудиторным занятиям и экзамену.

Учреждение образования Федерации профсоюзов Беларуси
«Международный университет «МИТСО»

**УТВЕРЖДАЮ**
Проректор по учебной работе
учреждения образования
Федерации профсоюзов Беларуси
«Международный университет «МИТСО»
_____ М.А.Юрочкин
_____ 2024 г.
Регистрационный № УД -111/01-24/уч.

# WEB ТЕХНОЛОГИИ

**Учебная программа учреждения высшего образования по учебной дисциплине для специальности**

1-26 03 01 Управление информационными ресурсами

2024 г.

Контрольный экземпляр

Учебная программа составлена на основе образовательного стандарта высшего образования ОСВО 1-26 03 01-2021 для специальности 1-26 03 01 «Управление информационными ресурсами», утвержденного Постановлением Министерства образования Республики Беларусь 25.04.2022 № 88, учебного плана учреждения образования Федерации профсоюзов Беларуси «Международный университет «МИТСО» для специальности 1-26 03 01 «Управление информационными ресурсами» рег. № 1-26 03 01/уч 2021 от 28.05.2021

**СОСТАВИТЕЛИ:**
А.С. Пекарь, преподаватель кафедры информационных технологий учреждения образования Федерации профсоюзов Беларуси «Международный университет «МИТСО»

**РЕКОМЕНДОВАНА К УТВЕРЖДЕНИЮ:**
Кафедрой информационных технологий учреждения образования Федерации профсоюзов Беларуси «Международный университет «МИТСО»
(протокол № 12 от 28.06.2024 г.);

Советом факультета учреждения образования Федерации профсоюзов Беларуси «Международный университет «МИТСО»
(протокол № 12 от 01.07.2024 г.)

СОГЛАСОВАНО
Декан экономического факультета _____А.В.Ковтунов

Заведующий библиотекой _____О.О.Бабарикина

Нормоконтроль
ведущий специалист УМУ
_____Г.Д.Лагунович

# I. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

В настоящее время создание сайтов и WEB-приложений является одним из наиболее интенсивно развивающихся разделов информационных технологий. Это связано как с бурным развитием интернета, так и с желанием практически любой организации во всех сферах жизнедеятельности человека иметь представительство в сети интернет.

Все это определяет важность учебной дисциплины «WEB технологии» в учебном процессе, а также обуславливает необходимость внесения своевременных изменений и дополнений в ее содержание.

Учебная программа «WEB технологии» разработана для студентов IV курса очной (дневной) формы обучения, по специальности 1-26 03 01 «Управление информационными ресурсами».

Центральной идеей образования по дисциплине «WEB технологии» является необходимость обучения студентов современным подходам и методам решения прикладных задач создания и поддержки сайтов, а также принципами их грамотного построения и продвижения.

Второй важнейшей идеей обучения является подготовка студентов к практической работе в области веб программирования.

**Цели и задачи учебной дисциплины**

Дисциплина «WEB-технологии» имеет прикладную направленность.

**Основными целями дисциплины являются:**

дать теоретическую подготовку по базовым веб-технологиям и по созданию веб-приложений;

дать практические навыки использования языка HTML5 и CSS3;

дать теоретические знания языка JavaScript;

дать практические навыки создания динамических веб-страниц;

познакомить с серверными технологиями на примере PHP.

**Основными задачами дисциплины являются:**

получить базовые знания по представлению, организации и передаче информации, и структуре Web;

изучить этапы разработки веб-сайта;

изучить популярные принципы верстки и дизайна сайтов;

изучить современные библиотеки для создания веб-приложений;

развития умения и навыков выбора адекватных методов создания веб-сайтов и веб-приложений.

Освоение учебной дисциплины должно обеспечить формирование у студентов следующих академических, социально-личностных и профессиональных компетенций.

Освоение учебной дисциплины «WEB-технологии» должно обеспечить у обучающихся формирование **специализированной** (СК) **компетенции:**

СК-8. Осуществлять разработку веб-сайтов, создание интернет-приложений для использования их в сфере маркетинга.

В результате освоения учебной дисциплины студент должен

**знать:**

основные понятия, терминологию и структуру компьютерных сетей, и интернет;

язык HTML5, основные возможности CSS3;

основы JavaScript;

основы программирования сценариев на языке PHP;

**уметь:**

проектировать интерфейс веб-приложений;

верстать веб-страницы, создавать и использовать CSS;

создавать графические изображения, преобразовывать графические объекты, создавать, редактировать элементы дизайна;

создавать динамические страницы средствами JavaScript;

использовать технологию AJAX на своей странице;

создавать html-формы и программировать их обработку на сервере с помощью языка PHP;

**владеть:**

основными приемами проектирования веб-приложений;

инструментами разработки дизайна сайтов;

навыками работы в «инструментах разработчика» основных браузеров;

средствами разработки клиентских и серверных частей сайта.

Распределение аудиторных часов по видам занятий и семестрам.
Виды и формы аттестации

| Семестр | Количество академических часов | | | | | | | Самост. работа | Форма промежуточной аттестации |
|---|---|---|---|---|---|---|---|---|---|
| | Всего | Аудит. | Из них | | | | | | |
| | | | лекции | лабор. занятия | практ. занятия | семи-нары | Управл. самост. работа | | |
| | Очная (дневная) форма получения высшего образования | | | | | | | | |
| 7 | 144 | 76 | 30 | 30 | - | - | 16 | 68 | экз. |

## II. СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

**Тема 1. Обзор современных веб-технологий**

Сеть Интернет, история создания, принципы функционирования. Архитектура «клиент-сервер». Основные виды веб-приложений. Веб-сайт. Веб-сервис. Виды веб-сервисов.

**Тема 2. Основы разработки веб-приложений**

Статические веб-сайты. Язык разметки гипертекстовых документов (HTML). Каскадная таблица стилей (CSS). Динамические веб-сайты. Язык программирования JavaScript. Проектирование веб-сервисов. Протоколы SOAP, REST. Многостраничные (MPA) и одностраничные (SPA) веб-приложения. Обзор серверных языков программирования: PHP8, Python, Node.js и др.

**Тема 3. Основы HTML и CSS**

Назначение языка HTML. Теги HTML. Правила записи и интерпретации тегов. Теги управления разметкой. Команды вставки графики, форм, таблиц и фреймов. Назначение CSS. Понятие селектора. Модель Flexbox. Grid Layout.

**Тема 4. Основы JavaScript**

Структура кода. Переменные. Типы данных. Преобразование типов. Операторы сравнения. Условное ветвление. Логические операторы. Циклы while и for. Оператор выбора. Обработка исключений – try...catch. Функциональное программирование в JavaScript. Понятие функции. Замыкание.

**Тема 5. Объектно-ориентированное программирование в JavaScript**

Класс: базовый синтаксис. Наследование классов. Статические свойства и методы. Приватные методы и свойства класса. Назначение get и set методов.

**Тема 6. Асинхронное программирование в JavaScript**

Промисы. Цепочка промисов (then…catch…finally). Промисы: обработка ошибок. Promise API. Промисификация. Async/await. Fetch API.

**Тема 7. JavaScript. BOM. DOM. События**

Понятие BOM: объект window, диалоговые окна, объект history, объект location, объект navigator. Понятие таймеров – setTimeout, setInterval. Работа с окном браузера и с документом. Понятие DOM. Объект Document. Объект Node. Навигация по DOM. Создание/ добавление/ клонирование и удаление элементов веб-страницы. Введите в обработку событий. Объект Event. События мыши и клавиатуры. Всплытие и погружение событий.

**Тема 8. Основы PHP8**

Структура кода. Переменные. Типы данных. Преобразование типов. Операторы сравнения. Условное ветвление. Логические операторы. Циклы while и for. Оператор выбора. Обработка исключений – try...catch. Функциональное программирование в PHP8. Понятие функции.

# III. УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА УЧЕБНОЙ ДИСЦИПЛИНЫ

## Очная (дневная) форма получения высшего образования

| Номер раздела, темы | Название раздела, темы | Количество аудиторных часов | | | | Количество часов УСР | Самостоятельная работа | Форма контроля |
|---|---|---|---|---|---|---|---|---|
| | | Лекции | Практические занятия | Семинарские занятия | Лабораторные занятия | | | |
| 7 семестр | | | | | | | | |
| 1. | Обзор современных веб-технологий | 2 | | | | 2 | 8 | УО |
| 2. | Основы разработки веб-приложений | 4 | | | | 2 | 8 | УО, РПЗ, Р, УД |
| 3. | Основы HTML и CSS | 6 | | | 6 | 2 | 10 | УО, РПЗ, Р, УД |
| 4. | Основы JavaScript | 4 | | | 6 | 2 | 10 | УО, РПЗ, Р, УД |
| 5. | Объектно-ориентированное программирование в JavaScript | 2 | | | 4 | 2 | 8 | УО, РПЗ, Р(ТА), УД |
| 6. | Асинхронное программирование в JavaScript | 2 | | | 4 | 2 | 8 | УО, РПЗ, Р, УД |
| 7. | JavaScript. BOM. DOM. События. | 4 | | | 4 | 2 | 8 | УО, РПЗ, Р, УД |
| 8. | Основы PHP8 | 6 | | | 6 | 2 | 8 | УО, РПЗ, Р, УД |
| | **Всего** | **30** | | | **30** | **16** | **68** | **экз.** |

# IV. ИНФОРМАЦИОННО-МЕТОДИЧЕСКАЯ ЧАСТЬ

## ОСНОВНАЯ ЛИТЕРАТУРА

1. Флэнаган, Д. JavaScript. Полное руководство : справочник по самому популярному языку программирования / Дэвид Флэнаган. – 7-е изд.. – СПб. : М. : Диалектика, 2021. – 1080 с.

2. Хавербеке, М. Выразительный JavaScript: современное веб-программирование / Марейн Хавербеке ; пер. с англ. Е. Сандицкой. – 3-е изд. – СПб. [и др.] : Питер; Прогресс книга, 2020. – 480 с.

3. Грофф, Д. Р. SQL : полное руководство / Джеймс Грофф, Пол Вайнберг, Эндрю Оппель. – 3-е изд.. – М. [и др.] : Вильямс, 2018. – 957 с.

## ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

1. Резиг, Д. JavaScript для профессионалов / Д. Резиг, Р. Фергюсон, Д. Пакстон. – СПб. : Apress, 2020. – 242 с.

2. Флэнаган, Д. JavaScript. Полное руководство / Д. Флэнаган. – 6-е изд. – СПб. : O'Reilly, 2021. – 732 с.

3. Флэнаган, Д. JavaScript. Карманный справочник. / Д. Флэнаган. – СПб. : O'Reilly, 2020. – 319 с.

4. Хавербеке, М. Выразительный JavaScript. / М. Хавербеке. – СПб. : Питер, 2019. – 482 с.

5. Мейер, Э. CSS : Полный справочник / Э. Мейер, Э. Уэйл; пер. с англ. – 4-е изд. – СПб. : Символ-Плюс, 2019. – 576 с.

6. Томсон Л., Веллинг Л. Разработка Web-приложений на PHP и MySQ : Учебник. – Киев : ДиаСофт, 2002. – 672 с.

7. Грофф Дж., Вайнберг П. Энциклопедия SQL / Дж. Грофф, П. Вайнберг. – СПб. : Питер, 2004. – 896 с.+CD .

8. Мархвида, И. Р. Создание Web-страниц: HTML, CSS, JavaSript. / И. Р. Мархвида.– Минск : Новое знание, 2002. – 352 с.

## СРЕДСТВА ДИАГНОСТИКИ РЕЗУЛЬТАТОВ УЧЕБНОЙ ДЕЯТЕЛЬНОСТИ

Диагностика результатов образовательной деятельности обучающихся осуществляется в ходе проведения всех видов занятий, самостоятельной работы и текущей аттестации по учебной дисциплине.

**Основными формами контроля знаний** по учебной дисциплине являются:

устный опрос УО;

устный доклад УД;

реферат Р;

решение практических задач РПЗ.

Текущая аттестация обучающихся проводится в течение семестра в целях периодического контроля и оценки результатов их учебной деятельности по учебной дисциплине, модулю учебного плана учреждения образования по специальности, изучаемым в семестре с выставлением отметок.

**Форма текущей аттестации** по учебной дисциплине:

реферат Р.

К промежуточной аттестации по учебной дисциплине, модулю обучающиеся допускаются при условии успешного прохождения текущей аттестации, предусмотренной учебной программой в текущем семестре.

**Формой промежуточной аттестации** является:

экзамен экз.

Оценка уровня знаний студента производится по десятибалльной шкале в соответствии с критериями, утвержденными Министерством образования Республики Беларусь.

## СРЕДСТВА ОБУЧЕНИЯ

Необходимыми средствами для реализации успешного формирования профессиональных компетенций у студентов являются: оборудованный интернет-доступом компьютерный класс с установленным специальным программным обеспечением, мультимедийная аппаратура, электронные учебно-наглядные пособия (презентации, методические материалы).

## ПЕРЕЧЕНЬ ТЕМ ЛАБОРАТОРНЫХ ЗАНЯТИЙ

Основная цель проведения лабораторных занятий состоит в закреплении теоретического материала курса, приобретении навыков выполнения эксперимента, обработки экспериментальных данных, анализа результатов, грамотного оформления отчетов.

1. Основы HTML5 и CSS3;
2. FlexBox и Grid Layout;
3. Основы JavaScript;
4. Строки. Методы обработки строк. Регулярные выражения;
5. ES6, ES8, ES2017. Проектирование типов. Классы. Наследование;
6. Core JS;
7. Основы PHP.

# ПРИМЕРНЫЙ ПЕРЕЧЕНЬ ЗАДАНИЙ И КОНТРОЛЬНЫХ МЕРОПРИЯТИЙ УСР

| № темы, (раздела) | Тема УСР | Кол-во часов | Метод. обеспечение | Форма контроля |
|---|---|---|---|---|
| 4 курс, 7 семестр (16 часов) | | | | |
| 1. | Обзор современных веб-технологий | 2 | Интернет-ресурсы | Реферат, доклад, сообщение |
| 2. | Основы разработки веб-приложений | 2 | Интернет-ресурсы | Реферат, доклад, сообщение |
| 3. | Основы HTML и CSS | 2 | Интернет-ресурсы | Реферат, доклад, сообщение |
| 4. | Основы JavaScript | 2 | Интернет-ресурсы | Реферат, доклад, сообщение |
| 5. | Объектно-ориентированное программирование в JavaScript | 2 | Интернет-ресурсы | Реферат, доклад, сообщение |
| 6. | Асинхронное программирование в JavaScript | 2 | Интернет-ресурсы | Реферат, доклад, сообщение |
| 7. | JavaScript. BOM. DOM. События. | 2 | Интернет-ресурсы | Реферат, доклад, сообщение |
| 8. | Основы PHP8 | 2 | Интернет-ресурсы | Реферат, доклад, сообщение |

Студентам предлагается выполнить одно из следующих заданий:

1. Создание интерактивной анимации на HTML5, визуализирующей несколько простейших физических процессов (бросок тела под углом к горизонту, отскок мяча от плоскости, магнитное поле и т.п.).

2. Создание страницы на HTML5, визуализирующей построение треугольника по трем элементам, которые интерактивно задаются (3 стороны, 2 стороны и угол, 2 угла и сторона, высота и 2 стороны и т.п.).

3. Создание WEB-приложения по теме. Студентам необходимо разработать проект сайта и создать SPA сайт или многостраничный сайт, используя JavaScript, AJAX, современные подходы к верстке. Предлагаемые для разработки темы сайтов:

1. Компьютерный клавиатурный тренажер.
2. Новостной сайт с подключаемыми с других сайтов информерами (курсы валют, прогноз погоды, спортивные новости, анекдоты и т.п.).
3. Сайт визитка для ученого или для научного учреждения.
4. Сайт конференции.
5. Сайт интернет-конференции.
6. Сайт научного учреждения.
7. Сайт кафедры.
8. Сайт факультета.
9. Сайт студенческой группы.
10. Сайт дошкольного учреждения.
11. Сайт школьного учреждения.
12. Сайт школьного учителя-предметника.

13. Сайт общества женщин, общества (клуба) мужчин.
14. Сайт компании по производству (например, ежедневников).
15. Сайт рекламирующий определенный товар (например, удочки)
16. Сайт болельщиков какой-либо спортивной команды.
17. Сайт поклонников какой-либо знаменитости.
18. Сайт турагентства.
19. Сайт с музыкой. Каталог + проигрыватель + мультимедиа.
20. Интернет-фотоальбом. Галерея фотографий с возможностью оценивания. Очередность отображения фотографий зависит от рейтинга.
21. Интернет-магазин (по продаже цветов, по продаже компьютерной техники, по продаже настольных игр и др.).

## ПРИМЕРНЫЙ ПЕРЕЧЕНЬ ВОПРОСОВ К ЭКЗАМЕНУ

1. Сеть Интернет, история создания, принципы функционирования. Архитектура «клиент-сервер». Основные виды веб-приложений. Веб-сайт. Веб-сервис. Виды веб-сервисов.
2. Понятие веб-сайта. Понятие веб-разработки (front-end / back-end). Виды сайтов. Понятие веб-дизайна. UI/UX дизайн. Графика в веб-дизайне. Основные принципы визуального дизайна (по Торре).
3. Понятие юзабилити (usability) сайта. Принципы юзабилити. Назначение прототипа сайта. Правила использования модульных сеток в формировании страницы сайта. Техника построения модульных сеток. Преимущества адаптивных сайтов. Внесение разметки и создание основы сайта.
4. Статические веб-сайты. Язык разметки гипертекстовых документов (HTML). Каскадная таблица стилей (CSS). Динамические веб-сайты. Язык программирования JavaScript.
5. Проектирование веб-сервисов. Протоколы SOAP, REST. Многостраничные (MPA) и одностраничные (SPA) веб-приложения. Обзор серверных языков программирования: PHP8, Python, Node.js и др.
6. Проектирование веб-сервисов. Протоколы SOAP, REST. Многостраничные (MPA) и одностраничные (SPA) веб-приложения. Технологии, используемые на стороне клиента: HTML, CSS, JavaScript (TypeScript). Обзор UI-фреймворков: Bootstrap, Foundation и т.д.
7. Статические и динамические компоненты представления веб приложений. Обзор JavaScript-фреймворков: React, Vue, Angular. Преимущества и недостатки.
8. Статические и динамические компоненты представления веб приложений. Обзор информационных систем управления контентом (CMS).
9. Назначение и приминение HTML. Элементы и атрибуты HTML. Глобальные и пользовательские атрибуты. Элементы группировки. Заголовки.
10. Назначение и приминение HTML. Форматирование текста. Изображения. Списки. Таблицы. Ссылки.

11. Назначение и приминение HTML. Работа с формами. Элементы форм (input, textarea, select и т.д.). Валидация форм.

12. Назначение и приминение HTML. Семантическая структура страницы (article, section, nav и т.д.)

13. Назначение и приминение CSS. Селекторы. Селекторы потомков. Селекторы дочерних элементов. Селекторы элементов одного уровня.

14. Назначение и приминение CSS. Псевдоклассы. Псевдоклассы дочерних элементов. Псевдоэлементы. Селекторы атрибутов.

15. Назначение и приминение CSS. Виды селекторов. Наследование стилей. Каскадность стилей.

16. Назначение и приминение CSS. Свойства. Цвет. Стилизация шрифтов. Форматирование текста. Стиизация адзацев.

17. Назначение и приминение CSS. Свойства. Стилизация списков. Стилизация таблиц. Внешние отступы. Внутренние отступы. Границы.

18. Назначение и приминение CSS. Свойства. Фон элемента. Тень. Обтекание элементов. Виды градиентов.

19. Назначение и приминение CSS. Трансформации, переходы и анимации.

20. Назначение и приминение CSS. Адаптивный дизайн. Метатег viewport. Media Query в CSS.

21. Назначение и приминение CSS. Flexbox. Понятия flex-контейнера и flex-элементов. Направление flex-direction. Свойство flex-wrap. Выравнивание элементов (justify-content).

22. Назначение и приминение CSS. Flexbox. Выравнивание элементов (align-items и align-self). Выравнивание строк и столбцов (align-content).

23. Назначение и приминение CSS. Flexbox. Управление элементами (flex-basis, flex-shink и flex-grow).

24. Назначение и приминение CSS. Grid Layout. Понятие grid-контейнера. Строки и столбцы.

25. Назначение и приминение CSS. Grid Layout. Функция repeat и свойство grid. Размеры строк и столбцов. Отступы между столбцами и строками.

26. Назначение и приминение CSS. Grid Layout. Области грида (grid-area).

27. Основы JavaScript. Переменные. Типы данных. Преобразование типов. Базовые операторы, математика. Операторы сравнения.

28. Основы JavaScript. Условное ветвление. Логические операторы. Оператор объединения. Циклы while и for. Конструкция "switch".

29. Основы JavaScript. Исключения. Оператор обработки исключений в JavaScript – try...catch.

30. Основы JavaScript. Встроенные функции JavaScript. Пользовательские функции JavaScript. JavaScript функции с параметрами (аргументами) и возврат значений. Способы создания пользовательских функций. Использование выражений с функциями.

31. Основы JavaScript. Область видимости переменных. JavaScript глобальные и локальные переменные в функции.

32. Основы JavaScript. Пользовательские функции JavaScript. Замыкание.

33. Основы JavaScript. Строки. Методы работы с строками.

34. Основы JavaScript. Массивы. Методы работы с массивами.

35. Основы JavaScript. Структуры данных. Map, Set, WeakMap, WeakSet и т.д.

36. Основы JavaScript. Регулярные выражения. Класс RegExp. Методы работы с регулярными выражениями.

37. Основы JavaScript. Класс: базовый синтаксис. Наследование классов.

38. Основы JavaScript. Статические свойства и методы. Приватные и защищённые методы и свойства.

39. Основы JavaScript. Расширение встроенных классов. Проверка класса: "instanceof". Примеси.

40. Основы JavaScript. Promise (промисы). Цепочка промисов. Обработка ошибок.

41. Основы JavaScript. Promise (промисы). Promise API. Промисификация.

42. Основы JavaScript. Promise (промисы). Promise API. Async/Await.

43. Основы PHP8. Основы синтаксиса. Переменные. Типы данных. Операции в PHP.

44. Основы PHP8. Условное ветвление. Логические операторы. Оператор объединения. Циклы while и for. Конструкции "switch" и "match".

45. Основы PHP8. Массивы. Ассоциативные массивы. Многомерные массивы. Операции с массивами.

46. Основы PHP8. Функции. Параметры функции. Возвращение значений и оператор return. Анонимные функции. Замыкания. Стрелочные функции.

47. Основы PHP8. Область видимости переменной. Константы. Проверка существования переменной. Проверка и установка типа переменной. Преобразование типов.

48. Основы PHP8. Отправка данных на сервер. Получение данных из строки запроса. GET-запросы.

49. Основы PHP8. Отправка данных на сервер. Отправка формы. POST-запросы.

50. Основы PHP8. ООП. Объекты и классы. Конструкторы и деструкторы. Анонимные классы.

51. Основы PHP8. ООП. Наследование. Модификация доступа. Статические методы и свойства.

52. Основы PHP8. ООП. Интерфейсы. Абстрактные классы и методы. Копирование объектов классов.

53. Основы PHP8. Обработка исключений. Конструкция try…catch…finally.

54. Основы PHP8. Обработка исключений. Генерация исключений.

55. Основы PHP8. Работа с файловой системой. Чтение и запись файлов.

56. Основы PHP8. Работа с файловой системой. Управление файлами и каталогами.

57. Основы PHP8. Работа с файловой системой. Блокировка файла. Функция flock.

58. Основы PHP8. Базовые возможности PHP. Подключение внешних файлов. Пространства имен.

59. Основы PHP8. Базовые возможности PHP. Типизация данных. Работа со строками.

60. Основы PHP8. Базовые возможности PHP. Типизация с cookie. Сессии.

# V. ПРОТОКОЛ СОГЛАСОВАНИЯ УЧЕБНОЙ ПРОГРАММЫ

| Название учебной дисциплины, с которой требуется согласование | Название кафедры | Предложения об изменениях в содержании учебной программы учреждения высшего образования по учебной дисциплине | Решение, принятое кафедрой, разработавшей учебную программу (с указанием даты и номера протокола) |
|---|---|---|---|
| Компьютерная графика | Кафедра информационных технологий | Предложений нет | Протокол заседания кафедры информационных технологий от 28.06.2024 №12 |

# I. ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

## ТЕМА 1. ОБЗОР СОВРЕМЕННЫХ ВЕБ-ТЕХНОЛОГИЙ

### 1. Сеть Интернет

Интернет −это глобальная система связанных компьютерных сетей, которая использует стандартные протоколы (например, TCP/IP) для связи между устройствами. История создания Интернета начинается с 1960-х годов, когда были разработаны первые концепции пакетной передачи данных и сетей.

Интернет работает на основе протокола TCP/IP, который обеспечивает надежную передачу данных между двумя устройствами. Протокол HTTP, который используется для передачи веб-страниц, также работает поверх TCP/IP.

World Wide Web (Web) −это сеть информационных ресурсов. Важнейшими механизмами этой сети являются:

• Единая схема наименования для поиска ресурсов в Web. Например, URL (Uniform Resource Locator) – метод разметки ресурсов. Он состоит из названия протокола, двоеточия, двух косых черт «/», названия машины и пути к ресурсу. Например, http://www.mitso.by /.

• Протоколы для доступа к именованным ресурсам через Web. Например, HTTP (Hyper Text Transfer Protocol) – протокол передачи гипертекста).

Гипертекст предложен для простого перемещения по ресурсам (например, HTML – (Hyper Text Markup Language). Гипертекст - документ, содержащий гиперссылки (ссылки на другие документы).

Web-программирование −это отдельное направление в программировании, используемое для создания web-приложений. В связи с ускоренными темпами развития Интернет-технологий постоянно появляются новые направления в этой области. Поэтому Web-программирование не сформировавшаяся наука, а скорее набор существующих программных технологий (клиентских и серверных), используемых для организации работы пользователя в сети Интернет.

Сервер – компьютер, на котором находятся документы и приложения.

Клиент – рабочая станция, используемая для просмотра документов и приложений, хранящихся на сервере.

### 2. Архитектура «клиент-сервер»

В архитектуре «клиент-сервер» клиент отправляет запрос на сервер, который затем обрабатывает запрос и возвращает ответ. Это основной принцип работы веб-приложений.

В простейшем случае структуру Интернет можно представить следующей схемой:

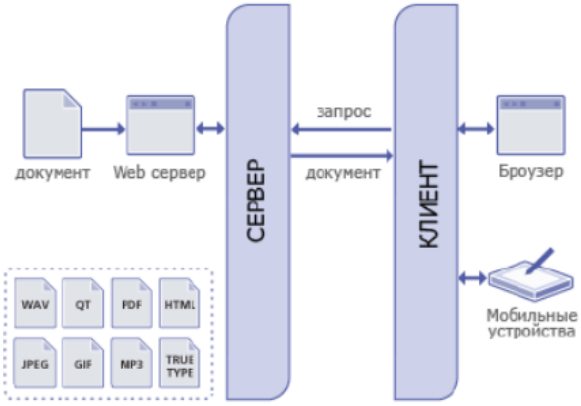


Рисунок 1.1 – Клиент/серверная архитектура Интернет

Работа такой системы осуществляется в следующей последовательности:

1. Браузер посылает запрос документа и служебную информацию на сервер.
2. Сервер находит и отправляет запрошенный документ обратно.
3. Браузер получает запрошенный документ.

Основными недостатками такой архитектуры являются:

1 Возможность просмотра только статических документами (отсутствие интерактивности).

2 Затруднения в работе с большими объемами данных.

Частично преодолеть ограничения клиент/серверной архитектуры Интернет позволяют Клиентские Web-технологии.

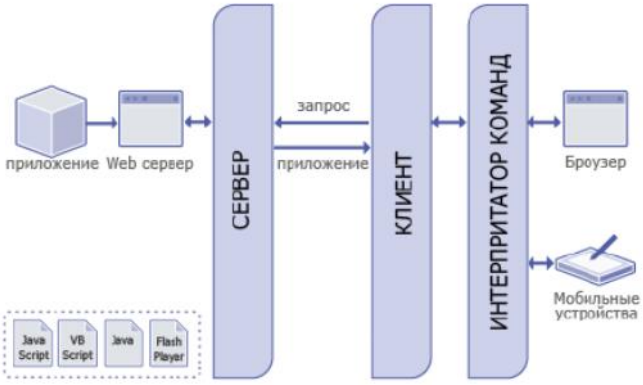


Рисунок 1.2 – Архитектура Интернет с интерпретатором команд на стороне клиента

Работа такой системы осуществляется в следующей последовательности:

Браузер посылает запрос приложения и служебную информацию на сервер.

Сервер находит и отправляет запрошенное приложение обратно.

Интерпретатор команд обрабатывает код приложения и выдает браузеру сформированный документ.

Преимуществом такой архитектуры являются возможность работы с динамическими документами (появляется интерактивность), а также возможность однажды загрузив приложение работать в нем, не обращаясь к серверу.

Однако недостатком остаются затруднения в работе с большими объемами данных.

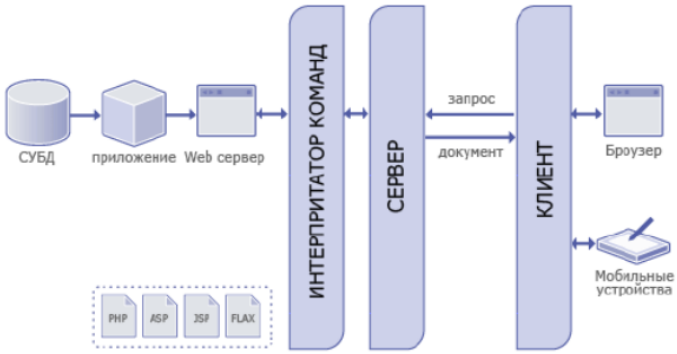


Рисунок 1.3 – Архитектура Интернет с интерпретатором команд на стороне сервера

Работа такой системы осуществляется в следующей последовательности:

1. Браузер посылает запрос приложения и служебную информацию на сервер.
2. Сервер находит и интерпретирует запрошенное приложение в документ и посылает его обратно.
3. Браузер получает сформированный документ и отображает его.

Преимуществом такой архитектуры являются:

1. Возможность работы с большими объемами данных, базами данных, не передавая их клиенту.
2. Защищенность данных.
3. Интерактивность документов.

Однако существенным недостатком такой архитектуры является то, что для совершения каждого действия над документом необходимо обращаться к серверу.

На основе рассмотренных выше архитектур строятся все современные технологии web-программирования.

Основными организациями, занимающимися разработкой новых стандартов и рекомендаций в области web-программирования являются:

- Internet Engineering Task Force (IETF). Основана в 1986 г. http://www.ietf.org
- World Wide Web Consortium (W3C). Основан в 1994 г. http://www.w3.org

**3. Основные виды веб-приложений**

Веб-приложения можно классифицировать по различным критериям, таким как тип контента, который они предоставляют, или технологии, которые они используют. Некоторые из основных типов включают статические сайты, динамические сайты, веб-приложения и веб-сервисы.

Веб-сайт −это совокупность связанных веб-страниц, обычно расположенных под одним доменным именем. Он может быть статическим (содержимое не меняется без изменения HTML-кода) или динамическим (содержимое может меняться без изменения HTML-кода).

Веб-сервис −это программное обеспечение, которое предоставляет интерфейс для взаимодействия с другими программами через сеть, обычно с использованием HTTP. Веб-сервисы могут использоваться для обмена данными между различными системами или для предоставления функциональности, которую можно использовать в других приложениях.

Существуют различные типы веб-сервисов, включая REST, SOAP и GraphQL. REST и SOAP −это два основных подхода к созданию веб-сервисов, в то время как GraphQL − это более новый подход, который предоставляет более гибкий и эффективный способ получения данных.

Примеры:

- Веб-сайт: Сайт BBC News предоставляет новости и информацию с помощью статических и динамических веб-страниц.
- Веб-сервис: Google Maps предоставляет веб-сервис, который можно использовать для получения карт и информации о местоположении.
- REST веб-сервис: Twitter API предоставляет REST веб-сервис, который можно использовать для доступа к твитам и информации о пользователях.
- SOAP веб-сервис: Сервисы, предоставляемые компанией Salesforce, часто используют SOAP для обмена данными.
- GraphQL веб-сервис: GitHub API v4 использует GraphQL для предоставления более гибкого и эффективного способа доступа к данным

Современные веб-технологии предоставляют мощные инструменты для создания разнообразных веб-приложений и веб-сервисов. Понимание основ этих технологий является ключом к эффективной разработке веб-приложений.

# ТЕМА 2. ОСНОВЫ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЙ

Приложения разделяются на три категории:

- Мобильные;
- Десктопные;
- Web.

Web-приложения:

1. Могут работать как на смартфоне, так и персональном компьютере.
2. Практически независимы от железа.
3. По функционалу скоро перестанут уступать десктопным аналогам.

Основы передачи данных:

**1 – DNS (Domain Name System, система доменных имен)**

когда в браузере вы набираете строку **google.com**, то попадаете на вполне конкретный сайт. Однако как мы выяснили ранее, **IP-адреса** – это числа, и никаких человекоудобных форм представления у них нет. Так, главная страница Гугл на самом деле имеет адрес **108.177.14.113.**

**2 – HTTP** (англ. HyperText **Transfer Protocol**

«протокол передачи гипертекста») −протокол прикладного уровня передачи данных (изначально −в виде гипертекстовых документов)**.**

На рисунке 2.1 представлено на что тратит время HTTP запрос.

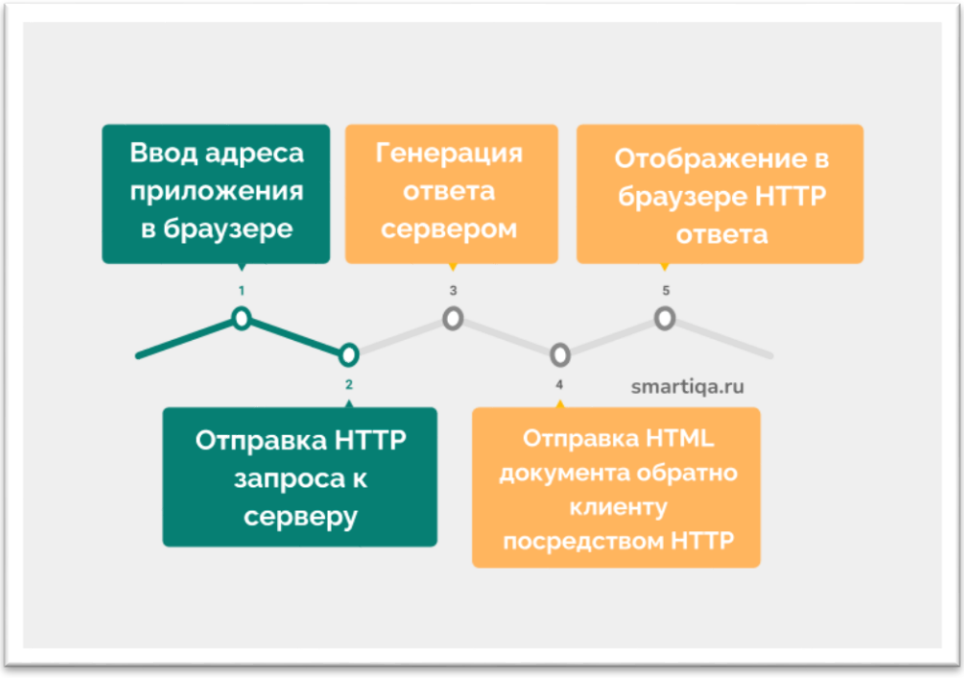


Рисунок 2.1 – Время HTTP запроса

Web-страны подразделяются на:

- **статические** – существуют на сервере в виде готовых файлов: *.htm, *.html.

• **динамические** – полностью или частично создаются на сервере в момент запроса (выбор информации из базы данных) *.asp, *.php.

+ позволяют выбирать информацию из базы данных по заранее неизвестным запросам.

– дополнительная нагрузка на сервер, загружаются медленнее.

Веб-приложения имеют клиентскую (фронтэнд) и серверную (бэкэнд) части.

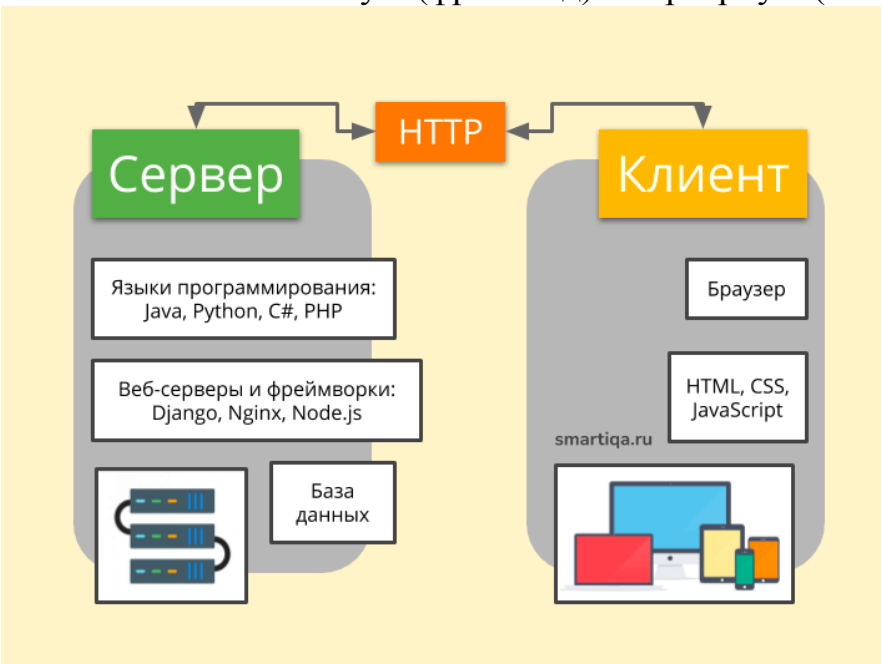


Рисунок 2.2 – Веб-приложение

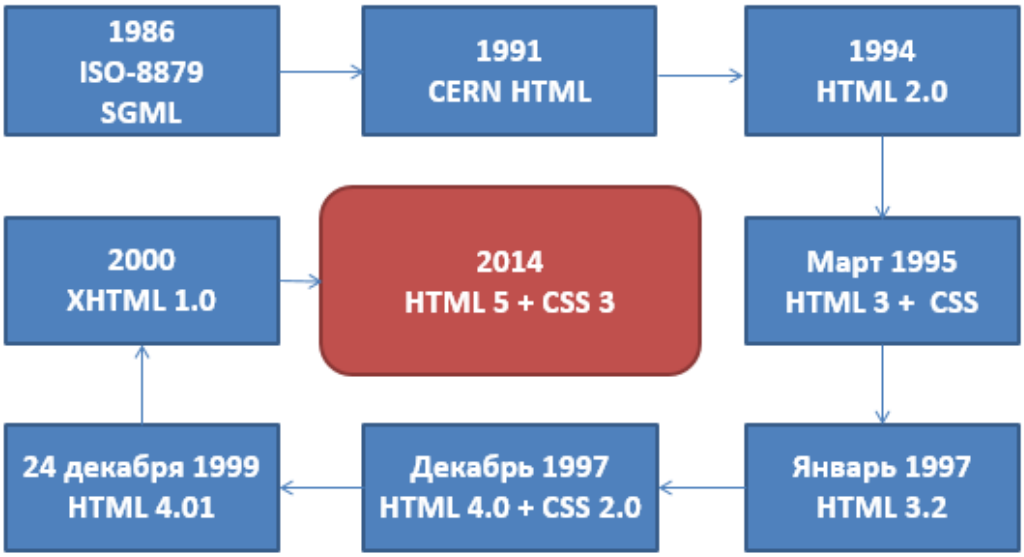


Рисунок 2.3 – Эволюция языков разметки

Основным форматом представления документов в сети Интернет является язык гипертекстовой разметки HTML (стандартный язык разметки документов во Всемирной паутине).

**HTML** – это определенная совокупность правил (тегов), по которым оформляется документ. Теги показывают Интернет-браузеру, как следует отображать текст на Web-страничке.

Тэг – это команда языка HTML, которую выполняет браузер.

## ТЕМА 3. ОСНОВЫ HTML И CSS

HTML (HyperText Markup Language) – язык создания гипертестовых документов. HTML не является языком программирования, т.к. предназначен только для представления данных (команды языка указывают клиенту, в каком виде должна быть отображена та или иная часть документа). В связи с этим язык HTML имеет следующие ограничения:

• не содержит средств обработки информации.

• не предоставляет никакой возможности сохранять данные в процессе работы приложения.

• имеет ограниченные возможности для взаимодействия с пользователем

Любая HTML страница состоит из набора объектов: текста, графики, ссылок, списков и т.д. Для представления этих объектов в языке HTML используются специальные операторы - тэги (от англ tag - ярлык).

Любой тег состоит из 3 элементов: начала тега, (открывающий тег) обозначаемого скобками <>, тела тега (например, текстовой информации) и конца тега (закрывающего тега) обозначаемого скобками </>), например:

<TITLE>Мой первый сайт</TITLE>

Таким образом, HTML документ состоит из последовательности тегов.

При написании HTML-документов нужно помнить следующие особенности:

• HTML не чувствителен к регистру. Команда <title> эквивалентна <TITLE> или <TiTlE>.

• HTML заменяет произвольные последовательности пробелов, табуляций и символов перехода на новую строку одним пробелом.

• Не все теги поддерживаются всеми Web-браузерами. Если браузер не поддерживает тег, он его просто игнорирует.

HTML документы представляют собой обычные текстовые документы и могут быть созданы в любом текстовом редакторе, например в Блокноте. Существуют специализированные WYSIWYG (What You See Is What You Get) HTML-редакторы, например, Adobe Dream Waver или MS Frontpage.

HTML документы принято сохраниять в файлы с расширением *.htm или *.html. Просмотр HTML документов осуществляется при помощи специальных

программ-просмоторщиков (web-браузеров), которые интерпретируют HTML код для отображения на экране.

HTML со времени своего рождения (1989 г.) постоянно развивался, непрерывно претерпевая изменения и дополнения стандартов. В данном курсе рассматривается последняя версия HTML 4.0.

Согласно спецификации HTML 4.0 структура HTML-документа выглядит следующим образом:

```
<HTML>
 <HEAD>
  <TITLE>Заголовок документа</TITLE>
 </HEAD>

 <BODY>
  Тело документа
 </BODY>
</HTML>
```

Как видно из приведенного кода, HTML документ всегда помещается в тег <HTML> и состоит из двух частей:

заголовка документа, определяемого тегом <HEAD>, тела документа, определяемого элементом <BODY>.

В заголовке HTML-документа приводится информация о документе, которая не отображается в окне браузера. Исключением является тег <TITLE>, содержимое которого выводится в заголовке окна браузера и используется для идентификации документа пользователем или поисковой машиной.

Каждый HTML документ должен иметь название, например:

```
<TITLE>Введение в HTML</TITLE>
```

В заголовок также включаются метатеги <META> для указания кодировки, ключевых слов, описания документа и т.д., например:

```
<meta name ="Content-Type" content="text/html;
charset=windows-1251">
<meta name="keywords" content="Web, программирование, HTML">
<meta name="description" content="Курс лекций по Web-
технологиям">
```

Тело HTML-документа обозначенное тегом <BODY></BODY> определяет видимую часть документа.

Открывающие теги могут содержать дополнительную информацию - параметры, которые существенно расширяют возможности представления информации. Параметры в открывающем теге записываются после названия тега в виде *параметр*=”*значение*” или *параметр*=’*значение’ и* разделяются пробелами. Порядок следования параметров в теге не произвольный. Если параметр отсутствует, его значение принимается по умолчанию согласно спецификации. Пример использования параметров в теге <BODY> приведен ниже:

```
<BODY text="black" bgcolor="white">
```

Основные параметры тега <BODY> приведены в таблице 1:

Таблица 3.1 – Основные параметры тега <BODY>

| Параметр | Значение |
|---|---|
| text | Устанавливает цвет текста документа, используя значение цвета в виде RRGGBB. Например: text=”000000” - черный цвет. |
| link | Устанавливает цвет гиперссылок, используя значение цвета в виде RRGGBB. Например: text=”FF0000” – красный цвет. |
| vlink | Устанавливает цвет гиперссылок на котох пользовтель уже побывал. Например: text=”00FF00” – зеленый цвет. |
| alink | Устанавливает цвет гиперссылок при нажатии. Например: text=”0000FF” – синий цвет. |
| bgcolor | Устанавливает цвет фона документа, используя значение цвета в виде RRGGBB. Например: text=”FFFFFF” – белый цвет. |
| background | Устанавливает изображение фона документа. Например: Background=“bg.gif” |
| Topmargin (marginheight*) | Устанавливает величину верхнего поля документа. Пример: Topmargin=”0” |
| Leftmargin | Устанавливает величину левого поля документа. Пример: |
| (marginwidth*) | Leftmargin=”10” |

Для того, чтобы таблица стилей влияла на вид документа она должна быть подключена к HTML-документу. Подключение каскадных таблиц стилей может быть выполнено одним из трех способов.

1. Внешние таблицы стилей, оформленные в виде отдельных файлов подключаются к HTML-документу при помощи тега <LINK> в заголовке документа. Например:

```
<LINK rel="stylesheet" href="style.css" type="text/css">
```

2. Внутренние таблицы стилей в составе HTML-документа помещаются в заголовок страницы при помощи тега <STYLE>. Например:

```
<HEAD>
<STYLE>
  P {color: #FF0000}
</STYLE>
</HEAD>
```

3. Локальные таблицы стилей объявляются непосредственно внутри тега, к которому они относятся при помощи параметра style. Например:

```
<P style="color: #FF0000">Каскадные таблицы стилей</p>
```

Таблицы стилей записываются в виде последовательности тегов, для каждого из которых в фигурных скобках указывается его свойства в виде **параметр: свойство**. Параметры внутри фигурных скобок разделяются точкой с запятой. Например:

```
P {color: #CCCCCC; font-size: 10px}
H1{ color: #FF0000}
```

Для увеличения гибкости контроля над элементами HTML используются классы, которые позволяют задавать различные стили для одного и того же тега. В таблицах стилей имя класса записывается после тега и отделяется от него точкой. Например:

```
P.green {color: #00FF00}
P.blue {color: #0000FF}
```

В HTML коде соответствие тега определенному классу указывается при помощи параметра class. Например:

```
<p class="green">Зеленый текст</p>
<p class="blue">Синий текст</p>
```

## ТЕМА 4. ОСНОВЫ JAVASCRIPT

JavaScript является языком клиентским скриптов. Коды программ, написанные на JavaScript, передаются в клиентский браузер и исполняются им. Поэтому важным вопросом изучения языка JavaScript является понимание объектной модели браузера.

Объектная модель браузера представляет собой строгую иерархическую структуру, позволяющую обращаться к любой части браузера или загруженных страниц с помощью языка JavaScript. Обобщенная объектная модель представлена на рисунке 4.1.

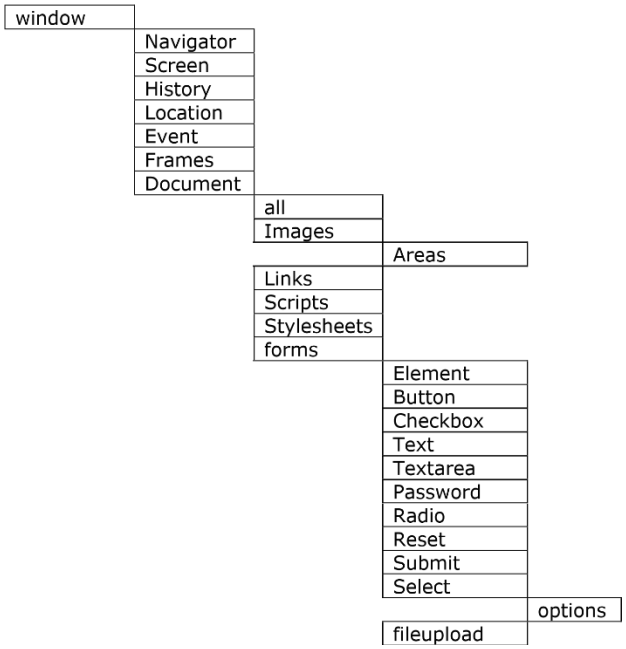


Рисунок 4.1 – Объектная модель браузера

В качестве примера на рис.5 представлена HTML-страница с указанием соответствия каждого ее элемента объектной модели.

Используя синтаксис JavaScript можно обратиться к любому элементу HTML-документа и изменить его поведение при помощи свойств и методов этого элемента, например:

```
document.forms[0].elements[0].value = “Milosh”;
```

Если мы имеем дело с большими страницами, то процедура адресации к различным объектам по номеру может стать весьма неудобной. Во избежание подобной проблемы можно присваивать различным тегам уникальные имена, например:

```
<form name="myForm">
Name: <input type="text" name="name" value=""><br>
E-mail: <input type="text" name="email" value=""><br>
<input type="submit" name="submit" value=”Push me”>
</form>
```

В этом случае обращение к элементам HTML-документа выглядет следующим образом:

```
document.myForm.name.value = “Milosh”;
```

Объект window – главный объект в иерархии объектной модели браузера. Объект window обращается к активному окну. Перечень основных свойств, методов и событий объекта window приведен в табл. 4.1, 4.2 и 4.3 соответственно.

Таблица 4.1 – Свойства объекта window

| **Свойство** | **Значение** |
|---|---|
| parent | Возвращает родительское окно |
| name | Название окна |
| status | Текст, отображаемый в строке состояния |
| document | Ссылка «только для чтения» на объект окна document |
| event | Ссылка «только для чтения» на объект окна event |
| history | Ссылка «только для чтения» на объект окна history |
| location | Ссылка «только для чтения» на объект окна location |
| navigator | Ссылка «только для чтения» на объект окна location |
| screen | Ссылка «только для чтения» на объект окна screen |

Таблица 4.2 – Методы объекта window

| Метод | Значение |
|---|---|
| alert | Вызов системного окна с простым сообщением. Например, alert(“Заполните пустые поля формы”) |
| Confirm(text) | Вызов системного окна с кнопками OK/CANCEL. Например, alert(“Продолжить поиск?”) |
| Prompt(text) | Вызов системного окна с полем для ввода текста с клавиатуры. Например, alert(“Введите Ваше имя”) |
| Open(url, name, settings) | Открывает новое окно. Например, window.open(“help.htm”, “helpWindow”, “*width=400,height=300,status=no,toolbar=no,menubar=no*”) |
| Close() | |

Таблица 4.3 –События объекта window

| Событие | Значение |
|---|---|
| onblur | Потеря фокуса |
| onfocus | Окно в фокусе |
| onhelp | Нажатие F1 или Help |
| onresize | Изменение размеров окна |
| onscroll | Прокрутка окна |
| onload | Страница полностью загружена |
| onunload | Перед выгрузкой страницы |

Объект window имеет несколько дочерних объектов, которые доступны с его помощью:

- Объект history содержит информацию о посещенных адресах.
- Объект navigator содержит информацию о браузере.
- Объект location содержит информацию о URL текущей страницы.
- Объект event содержит информацию о событиях, происходящих в браузере.
- Объект screen содержит информацию о возможностях экрана клиента.
- Объект document содержит отображаемый документ.

Перечень основных свойств, методов и событий дочерних объектов window приведен в табл. 4.5.

Таблица 4.5 – Свойства, методы и события дочерних объектов window

| Объект | Значение |
|---|---|
| **History** | |
| length | Длина списка посещенных адресов |
| back() | Загружает предыдущий адрес |
| forward() | Загружает следующий адрес |
| go(n) | Загружает адрес с номером n |
| **Navigator** | |
| appName | Название браузера |
| appVersion | Версия браузера |
| cookieEnabled | Возможность работы с cookie |
| mimeTypes | Список поддерживаемых браузером типов файлов |
| plugins | Список поддерживаемых браузером внедряемых объектов |
| javaEnabled() | Возможность запуска JavaScript |
| **Location** | |
| href | Указывет URL страницы |
| reload() | Обновляет страницу |
| **Event** | |
| X | Горизонтальная позиция курсора мыши |
| Y | Вертикальная позиция курсора мыши |
| button | Кнопка мыши, если она нажата |
| altkey | Состояние клавиши Alt |
| Ctrlkey | Состояние клавиши Ctrl |
| Shiftkey | Состояние клавиши Shift |
| Keycode | ASCII код нажатой клавиши |
| **Screen** | |
| Width | Разрешение по ширине экрана |
| Height | Разрешение по высоте экрана |
| colorDepth | Глубина цвета палитры экрана |

Перечень основных свойств, методов и событий дочерних объектов document приведен в табл. 4.6, 4.7 и 4.8.

Таблица 4.6 – Свойства объекта document

| **Объект** | **Значение** |
|---|---|
| Title | Название документа, определяемое в теге TITLE |
| Bgcolor | Цвет фона |
| Fgcolor | Цвет текста |
| linkColor | Цвет ссылок |
| alinkColor | Цвет активных ссылок |
| vlinkColor | Цвет посещенных ссылок |
| activeElement | Элемент в фокусе |

Таблица 4.7 – Методы объекта document

| **Объект** | **Значение** |
|---|---|
| open() | Открывает объект document для редактирования |
| write(text) | Запись текста в объект document |
| close() | Закрывает объект document для редактирования |
| clear | Очищает содержимое объекта document |
| alinkColor | Цвет активных ссылок |
| vlinkColor | Цвет посещенных ссылок |
| activeElement | Элемент в фокусе |

Таблица 4.8 – События объекта document

| **Объект** | **Значение** |
|---|---|
| Onclick | Щелчок мыши |
| Ondblclick | Двойной щелчок мыши |
| Onmouseout | Мышь уходит с элемента |
| Onmouseover | Мышь появляется над элементом |
| Onkeydown | Нажатие клавиши |
| onhelp | Нажатие F1 или Help |
| onload | Полная загрузка элемента |

Код скрипта JavaScript может объявляться в HTML-коде следующими способами:

1. Непосредственное размещение в любой части HTML-кода при помощи тега SCRIPT, например:

```
<script language="JavaScript"> alert("Hello,
world!")
</script>
```

2. Размещение в отдельном файле с обїявлением в заголовке HTML-документа при помощи тега SCRIPT, например:

```
<script language="JavaScript" src="script.js"> </script>
```

3. Контекстное размещение в обработчике событий. например:

```
<input type="button" onClick=" alert('Hello, world!')">
```

Часто используемые фрагменты программы можно оформить в виде функций, вызывая их по мере необходимости из различных мест программы. Функции должны быть определена перед вызовом, и размещение всех определений функций в разделе заголовка HEAD документа гарантирует доступность этих функций при обработке документа.

Формат определения функции представлен ниже:

```
function имя([параметр 1] [,параметр 2] [...,параметр N])
{ тело функции [return значение]
}
```

С помощью ключевого слова return функция может вернуть значение.

Условный оператор

```
if(i>0)
{выражение} else
{выражение}
```

Оператор выбора

```
 switch (значение)
{ case label : выражение; break;
case label : выражение; break;
... default : выражение;
}
```

Операторы цикла

```
for(i=0;i<9;i++) {выражение}
 for(i in obj)
{выражение}
 wile(условие)
{выражение}
 do {выражение}
while (условие)
```

Операторы прерывания потока вычислений

Оператор break используется для прерывания выполнения цикла, либо оператора switch, например:

```
while(i==0)
{if(j==3) break;}
```

Оператор continue используется для рестарта операторов цикла.

```
while(i==0)
{if(j==3) continue;}
```

## ТЕМА 5. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ В JAVASCRIPT

Объектно-ориентированная программа представляет собой совокупность логических конструктивных элементов – объектов. Все объекты являются экземплярами определенного класса.

Классы в свою очередь образуют иерархию наследования

**Класс** – универсальный тип данных, представляющий собой модель сущности: способ описания сущности, ее состояние, поведение, правила взаимодействия с данной сущностью, а также ее внутренний и внешний интерфейс

Классы представляют подход к проектированию, заменяющий процедурное программирование (с неизбежным “спагетти-кодом”) на хорошо структурированный, хорошо организованный код

Переменная-модель, относящаяся к определенному классу, называется экземпляром класса, или объектом

Традиционно выделяют три базовые концепции ООП: наследование, инкапсуляция, полиморфизм

JavaScript не является объектно-ориентированным языком, однако позволяет реализовать ООП- подобный подход, в частности, в функциональном стиле: роль классов выполняют функции- конструкторы

**Наследование** – позволяет описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом.

**Инкапсуляция** – свойство языка программирования, позволяющее пользователю не задумываться о сложности реализации используемого программного компонента (что у него внутри?), а взаимодействовать с ним посредством предоставляемого интерфейса (публичных методов и членов), а также объединить и защитить жизненно важные для компонента данные. При этом пользователю предоставляется только спецификация (интерфейс) объекта.

**Полиморфизм** позволяет писать более абстрактные программы и повысить коэффициент повторного использования кода. Общие свойства объектов объединяются в систему, которую могут называть по-разному – интерфейс, класс.

Создание конструктора

Функция-конструктор применяется для создания объектов определенного типа. Вызов конструктора осуществляется с помощью оператора new

Вызов функции через оператор new создает новый объект

Технически, любая функция, вызванная при помощи оператора new, становится конструктором

По общему соглашению, имя функции-конструктора должно начинаться с заглавной буквы (User)

```
function User() {}

let person = new User(); // стандартом допускается let person = new User;

console.log(person); // User {}
```

Свойства и методы класса записываются в объект this

Алгоритм работы конструктора: функция-конструктор создает новый пустой объект, this получает ссылку на этот объект, код функции выполняется, изменяя this, после чего функция возвращает новосозданный объект

```
function User(value) {
 /* this = {}; – интерпретатор создает новый пустой объект */
 /* модификация this: добавление свойств или методов */
 this.name = value;
 /* return this; – интерпретатор возвращает новый объект */
}
let person = new User('John Doe'); // User {name: "John Doe"}

alert(person.name); // "John Doe"
```

С помощью функции-конструктора можно создать любое количество объектов по определенному в конструкторе образу и подобию

Возврат значения

Как правило, конструкторы ничего не возвращают при помощи оператора return. Конструктор автоматически возвращает новый объект, заполненный в this

Однако если в конструкторе присутствует return, то значение возвращается по следующим правилам:

- вызов return с объектом возвращает этот объект, а не this
- вызов return с примитивным значением возвращает this

```
function User(value) {
 this.name = value;
 this.age = 10;
 return {name: 'Harry Potter'};
}

let person = new User('John Doe');
console.log(person); // {name: "Harry Potter"}
function User(value) {
 this.name = value;
 this.age = 10;
 return 'Harry Potter';
}

let person = new User('John Doe');
console.log(person); // {name: "John Doe", age: 10}
```

Принцип ООП: инкапсуляция

**Инкапсуляция** решает задачу разделения внутреннего и внешнего интерфейса программы

***Внутренний интерфейс (private)*** – свойства и методы, доступные только внутри класса (конструктора). Определяется посредством локальных переменных – объявленных через var, let, const

***Внешний интерфейс (public)*** – методы и свойства, доступные конечному пользователю. Определяется через контекстное значение – this

В терминологии ООП инкапсуляция означает ограничение доступа к данным и сокрытие реализации методов с целью защиты их от несанкционированного доступа

Те данные и методы, которые должны быть доступны экземплярам конструктора (публичные данные), записываются в this, вспомогательные данные объявляют локальными.

Публичные свойства (public)

```
function User(value) {
 this.name = value;
}

let person = new User('John Doe');
alert(person.name); // "John Doe"
```

Приватные свойства (private)

```
function User(value) {
 let name = value;
}

let person = new User('John Doe');
alert(person.name); // undefined
```

Инкапсуляция через замыкание

Ограничение доступа к данным класса реализуется через концепцию замыкания: параметры функции, а также переменные и методы, объявленные внутри класса, доступны извне только через публичные методы класса

```
function User(value) {
 let name = value;
 this.greeting = function() {
 alert( 'Hello, ' + name );
 };
}

let person = new User('John Doe');
person.greeting(); // "Hello, John Doe"
alert(person.name); // undefined
```

Геттеры и сеттеры

Для контроля над свойством или методом его делают приватным, а запись и чтение значения реализуются через специальные методы – getter и setter

```
function User(value) {
 let name = value.trim() || 'Anonymous';

 this.getName = function() {
 return name;
 };

 this.setName = function(newName) {
 if (!newName) alert('You forgot to specify a new name');
 else name = newName;
 };
}

let person = new User(' John Doe ');
alert( person.name ); // undefined
alert( person.getName() ); // "John Doe"
person.setName('');
alert( person.getName() ); // "John Doe"
```

Принцип ООП: наследование

**Наследование** означает создание новых классов на основе существующих

Базовый класс (прототип) содержит методы и свойства, которые будут передаваться всем его потомкам. Производные классы (наследники) могут расширять или переопределять свой базовый функционал

Наследование реализуется с помощью методов передачи контекста

```
function User() {
 let isActive = false;
 this.activate = function() {
 isActive = true;
 };
}
function Admin(value) {
 User.call(this);

 this.name = value;
}
let person = new Admin('John Doe');
person.activate();
```

Защищенные свойства (protected)

**Защищенные свойства (protected)** доступны только внутри класса и для классов-потомков

В JavaScript нет специального механизма определения защищенных свойств класса. По общему соглашению, публичные свойства, начинающиеся со знака подчеркивания _, имеют статус защищенных

```
function User() {
 this._isActive = false;

 this.activate = function() {
 this._isActive = true;
 };

 this.deactivate = function(){
 this._isActive = false;
 };
}

function Admin(value) {
 User.call(this);

 this.name = value;

 this.isActivated = function() {
  return this._isActive;
 };
```

```
}

let person = new Admin('John Doe');
alert( person.isActivated() );
```

Принцип ООП: полиморфизм

**Полиморфизм** как парадигма ООП подразумевает единый интерфейс и множество реализаций. Метод, определенный в базовом классе, может дополняться или перезаписываться в классе- наследнике

```
function User(value) {
 this.name = value || '';

 this._isActive = false;

 this.activate = function() {
  this._isActive = true;
 };

 this.deactivate = function() {
  this._isActive = false;
 };
}

function Admin(value) {
 User.call(this, value);
 /* переопределение родительского метода */
 this.activate = function() {
  this._isActive = 1;
 };

 /* расширение родительского метода */
 let parentMethod = this.deactivate;

 this.deactivate = function() {
  let agree = confirm('Are you sure?');
  if (agree) parentMethod.call(this);
 };
}
```

Схема функционального паттерна

Создание базового конструктора, в котором могут быть приватные, защищенные и публичные свойства

```
function Parent() {
 let privateProperty;

 this._protectedProperty;

 this.publicProperty;
}
```

Для наследования конструктор потомка использует методы call/apply

```
Parent.apply(this, arguments);
```

Наследник может перезаписать или расширить свойства родителя

```
let parentMethod = this.method;

this.method = function() {
 parentMethod.call(this);
};
```

Проверка принадлежности классу

Оператор instanceof позволяет проверить, принадлежит ли объект определенному классу, без учета наследования

```
function User(value) {
 this.name = value || '';
}
function Admin(value) {
 User.call(this, value);
}

let user = new Admin('John Doe');
console.log(user); // {name: "John Doe"}
console.log(user instanceof Admin); // true
console.log(user instanceof User); // false
```

# ТЕМА 6. АСИНХРОННОЕ ПРОГРАММИРОВАНИЕ В JAVASCRIPT

Асинхронное программирование стало неотъемлемой частью разработки современных веб-приложений, особенно в JavaScript. Чтобы понять важность асинхронного подхода, необходимо сначала разобраться в том, что такое синхронный и асинхронный код и чем они отличаются.

Синхронный код исполняется последовательно, строка за строкой. Когда одна операция выполняется, выполнение следующей откладывается до тех пор, пока предыдущая не завершится. Этот подход хорошо работает для простых задач, но может стать проблематичным при выполнении длительных операций, таких как запросы к серверу или чтение больших файлов. В синхронном коде, если одна из операций занимает много времени, весь процесс будет остановлен до её завершения, что приводит к низкой отзывчивости приложения.

```
function syncFunction() {
 console.log('Начало');
 let result = longRunningOperation(); // Допустим, эта операция занимает много времени
 console.log(result);
 console.log('Конец');
}

function longRunningOperation() {
 // Имитация длительной операции
 let start = Date.now();
 while (Date.now() - start < 5000) {
  // ждем 5 секунд
 }
 return 'Операция завершена';
}

syncFunction();
```

В данном примере выполнение кода остановится на вызове longRunningOperation на 5 секунд, прежде чем продолжить выполнение. Это делает приложение неотзывчивым в течение этого времени.

Асинхронный код позволяет выполнять операции параллельно, не блокируя выполнение остальных частей программы. Это особенно полезно для задач, которые могут занять значительное время, таких как сетевые запросы или операции ввода-вывода. Асинхронное программирование позволяет запускать такие операции и продолжать выполнение других задач, пока первая операция не завершится.

```
function asyncFunction() {
 console.log('Начало');
 longRunningOperationAsync((result) => {
  console.log(result);
  console.log('Конец');
 });
}

function longRunningOperationAsync(callback) {
 setTimeout(() => {
  callback('Операция завершена');
 }, 5000);
}

asyncFunction();
```

Здесь используется setTimeout, чтобы имитировать асинхронную операцию, которая завершается через 5 секунд. В то время как операция выполняется, остальные части программы могут продолжать свою работу, и приложение остается отзывчивым.

JavaScript часто используется для разработки клиентских веб-приложений, где отзывчивость пользовательского интерфейса имеет критическое значение. Асинхронное программирование позволяет избежать блокировки главного потока выполнения, обеспечивая плавный и интерактивный пользовательский опыт. Кроме того, асинхронное программирование позволяет более эффективно использовать ресурсы сервера, обрабатывая запросы параллельно и увеличивая пропускную способность приложения.

Историческая перспектива развития асинхронности

Асинхронное программирование в JavaScript прошло значительный путь эволюции. От использования простых коллбеков до введения промисов и, наконец, до async/await, каждый из этих методов был разработан для решения конкретных проблем.

Коллбеки были первым и самым простым способом обработки асинхронных операций в JavaScript. Коллбек - это функция, переданная в другую функцию в качестве аргумента и вызываемая после завершения асинхронной операции.

```
function fetchData(callback) {
 setTimeout(() => {
  callback('Данные получены');
 }, 1000);
}

fetchData((data) => {
 console.log(data);
});
```

Коллбеки позволяют выполнять асинхронные задачи, не блокируя основной поток. Однако при сложных сценариях, где необходимо выполнять несколько

асинхронных операций последовательно или параллельно, код может стать сложно читаемым и поддерживаемым. Это явление известно как "ад коллбеков" (Callback Hell).

```
doSomething((result1) => {
 doSomethingElse(result1, (result2) => {
  doMore(result2, (result3) => {
   doLast(result3, (result4) => {
    console.log('Все операции завершены');
   });
  });
 });
});
```

Промисы были введены для решения проблемы коллбеков, улучшая читаемость и управляемость асинхронного кода. Промис представляет собой объект, который может находиться в одном из трех состояний: ожидание (pending), выполнено (fulfilled) и отклонено (rejected).

```
let promise = new Promise((resolve, reject) => {
 setTimeout(() => {
  resolve('Данные получены');
 }, 1000);
});

promise.then((data) => {
 console.log(data);
}).catch((error) => {
 console.error(error);
});
```

Промисы позволяют цепочками вызывать асинхронные операции, упрощая чтение и обработку кода.

```
doSomething()
 .then((result1) => doSomethingElse(result1))
 .then((result2) => doMore(result2))
 .then((result3) => doLast(result3))
 .then((result4) => console.log('Все операции завершены'))
 .catch((error) => console.error(error));
```

Промисы решают проблемы вложенности и улучшают обработку ошибок, но могут стать сложными при необходимости использовать более сложные логические конструкции, такие как циклы и условные операторы.

Async/await был введен в спецификацию ECMAScript 2017 и предоставляет синтаксический сахар для работы с промисами, делая асинхронный код выглядящим как синхронный. Ключевое слово async используется для объявления асинхронной функции, а await - для ожидания завершения промиса.

```
async function fetchData() {
 try {
  let data = await new Promise((resolve, reject) => {
   setTimeout(() => {
    resolve('Данные получены');
   }, 1000);
  });
  console.log(data);
 } catch (error) {
  console.error(error);
 }
}

fetchData();
```

Async/await значительно упрощает написание и чтение асинхронного кода, особенно при сложных последовательных операциях и в сочетании с конструкциями управления потоком, такими как циклы и условные операторы.

```
async function processOperations() {
 try {
  let result1 = await doSomething();
  let result2 = await doSomethingElse(result1);
  let result3 = await doMore(result2);
  let result4 = await doLast(result3);
  console.log('Все операции завершены');
 } catch (error) {
  console.error(error);
 }
}

processOperations();
```

Эволюция асинхронного программирования в JavaScript от коллбеков к промисам и затем к async/await демонстрирует стремление улучшить читаемость, управляемость и производительность асинхронного кода. Коллбеки, промисы и async/await решают разные проблемы и подходят для различных сценариев, предоставляя разработчикам гибкие инструменты для создания отзывчивых и эффективных приложений. Далее мы углубимся в каждый из этих методов, рассмотрим их особенности и предоставим практические примеры их использования.

# ТЕМА 7. JAVASCRIPT. BOM. DOM. СОБЫТИЯ

Сам по себе язык JavaScript не предусматривает работы с браузером. Он вообще не знает про HTML. Но позволяет легко расширять себя новыми функциями и объектами.

У этого объекта двоякая позиция – он, с одной стороны, является глобальным объектом в JavaScript, с другой – содержит свойства и методы для управления окном браузера, открытия новых окон, например:

```
// открыть новое окно/вкладку с URL http://ya.ru
window.open('http://ya.ru');
```

Объектная модель документа (DOM)

Глобальный объектdocumentдает возможность взаимодействовать с содержимым страницы.

Пример использования:

```
document.body.style.background = 'red';
alert( 'Элемент BODY стал красным, а сейчас обратно вернётся' );
document.body.style.background = '';
```

Он и огромное количество его свойств и методов описаны в стандарте W3C DOM.

По историческим причинам, когда-то появилась первая версия стандарта DOM Level 1, затем придумали еще свойства и методы и появился DOM Level 2, на текущий момент поверх них добавили еще DOM Level 3 и готовится DOM 4.

Современные браузеры также поддерживают некоторые возможности, которые не вошли в стандарты, но де-факто существуют давным-давно, и отказываться от них никто не хочет. Их условно называют «DOM Level 0».

Также информацию по работе с элементами страницы можно найти в стандарте HTML 5.

Дерево DOM

Основным инструментом работы и динамических изменений на странице является DOM (Document Object Model) – объектная модель, используемая для XML/HTML-документов.

Согласно DOM-модели, документ является иерархией, деревом. Каждый HTML-тег образует узел дерева с типом «элемент». Вложенные в него теги становятся дочерними узлами. Для представления текста создаются узлы с типом «текст».

DOM – это представление документа в виде дерева объектов, доступное для изменения через JavaScript.

Пример DOM

Построим, для начала, дерево DOM для следующего документа.

```
<!DOCTYPE HTML>
<html>
<head>
 <title>О лосях</title>
</head>
<body>
 Правда о лосях
</body>
</html>
```

В этом дереве выделено два типа узлов.

Теги образуют узлы-элементы (element node). Естественным образом одни узлы вложены в другие. Структура дерева образована исключительно за счет них.

Текст внутри элементов образует текстовые узлы (text node), обозначенные как#text. Текстовый узел содержит исключительно строку текста и не может иметь потомков, то есть он всегда на самом нижнем уровне.

Пробелы и переводы строки – это тоже текст, полноправные символы, которые учитываются в DOM.

В частности, в примере выше тег<html>содержит не только узлы-элементы<head>и<body>, но и#text(пробелы, переводы строки) между ними.

Впрочем, как раз на самом верхнем уровне из этого правила есть исключения: пробелы до<head>по стандарту игнорируются, а любое содержимое после</body>не создаёт узла, браузер переносит его внутрь, в конецbody.

В остальных случаях всё честно – если пробелы есть в документе, то они есть и в DOM, а если их убрать, то и в DOM их не будет, получится так:

```
<!DOCTYPE HTML>
<html><head><title>О лосях</title></head><body>Правда о лосях</body></html>
```

Автоисправление

При чтении неверного HTML, браузер автоматически корректирует его для показа, также это происходит и при построении DOM.

В частности, всегда будет верхний тег<html>. Даже если в тексте нет – в DOM он будет, браузер создаст его самостоятельно.

То же самое касается и тега<body>.

Например, если файл состоит из одного слова "Привет", то браузер автоматически обернет его в<html>и<body>.

Другие типы узлов

Дополним страницу новыми тегами и комментарием:

```
<!DOCTYPE HTML>
<html>
<body>
 Правда о лосях
 <ol>
 <li>Лось – животное хитрое</li>
 <!-- комментарий -->
 <li>...и коварное!</li>
 </ol>
</body>
</html>
```

В этом примере тегов уже больше, и даже появился узел нового типа – комментарий.

Казалось бы, зачем комментарий в DOM? На отображение-то он все равно не влияет. Но так как он есть в HTML – обязан присутствовать в DOM-дереве.

Все, что есть в HTML, находится и в DOM.

Даже директива<!DOCTYPE...>, которую мы ставим в начале HTML, тоже является DOM-узлом и находится в дереве DOM непосредственно перед<html>. На иллюстрациях выше этот факт скрыт, поскольку мы с этим узлом работать не будем, он никогда не нужен.

Даже сам объект document, формально, является DOM-узлом, самым-самым корневым.

Всего различают 12 типов узлов, но на практике мы работаем с четырьмя:

Документ – точка входа в DOM.

Элементы – основные строительные блоки.

Текстовые узлы – содержат, собственно, текст.

Комментарии – иногда в них можно включить информацию, которая не будет показана, но доступна из JS.

Возможности, которые дает DOM

Зачем, кроме красивых рисунков, нужна иерархическая модель DOM?

DOM нужен для того, чтобы манипулировать страницей – читать информацию из HTML, создавать и изменять элементы.

УзелHTMLможно получить как document.documentElement, а BODY– как document.body.

Получив узел, мы можем что-то сделать с ним.

Например, можно поменять цвет BODY и вернуть обратно:

```
document.body.style.backgroundColor = 'red';
alert( 'Поменяли цвет BODY' );
document.body.style.backgroundColor = '';
alert( 'Сбросили цвет BODY' );
```

Объектная модель браузера (BOM)

BOM – это объекты для работы с чем угодно, кроме документа.

Например:

Объект navigator содержит общую информацию о браузере и операционной системы. Особенно примечательны два свойства: navigator.userAgent– содержит информацию о браузере inavigator.platform содержит информацию о платформе, позволяет различать Windows/Linux/Mac и т.п.

Объект location содержит информацию о текущем URL страницы и позволяет перенаправить посетителя на новый URL.

Функции alert/confirm/promptтоже входят в BOM.

Пример использования:

```
alert( location.href ); // выведет текущий адрес
```

Большинство возможностей BOM стандартизированы в HTML 5, хотя различные браузеры и предоставляют зачастую что-то свое, в дополнение к стандарту.

## ТЕМА 8. ОСНОВЫ PHP8

PHP – это язык серверных скриптов (server scripting language), встраиваемый в HTML, который интерпретируется и выполняется на сервере.

Файлы, которые подвергаются обработке препроцессором, должны иметь определенное расширение (обычно это .php) и содержать код для препроцессора. Перед отправкой страницы PHP-код проигрывается на сервере и брозеру выдается результат в виде HTML-страницы. Обычные же страницы, имеющие расширение .html/.htm Web-сервер будет отправлять броузеру без какой-либо обработки.

Код скрипта PHP может объявляться в HTML-коде следующими способами:

1. Непосредственное размещение в любой части HTML-кода при помощи тега SCRIPT, например:

```
<script
language="PHP"> echo
"Hello, world!";
</script>
```

2. Краткая форма размещения в любой части HTML-кода при помощи конструкции <? … ?>, например:

```
<? echo "Hello,
world!";
?>
```

Текст программы на PHP состоит из последовательности операторов. Именно в такой последовательности операторы и исполняются интерпретатором (броузером). Особенности записи программ на языке PHP:

- Синтаксис языка PHP близка очень близок языку C++
- Один оператор в PHP может быть разбит на несколько строк, или, наоборот, в одной строке может быть несколько операторов

• В программе отдельные операторы записываются через точку с запятой (допускается не ставить ; если оператор является в строке последним)

• PHP не имеет жестких требований к форматированию текста программы, таким образом возможно использование символов перевода строки и отступов для придания тексту программы лучшей читабельности

• Однострочные комментарии в JavaScript записываются после симовлов //, многострочные – между символами /* */.

Допустимые синтаксисом языка PHP операции представлены в табл. 8.1.

Таблица 8.1 – Арифметические и логические операции

| **Арифметические операторы** | |
|---|---|
| +,-,*,/,% | Сложение, вычетание, умножение, деления, остаток от деления |
| ++, -- | Приращение на единицу, Уменьшение на единицу |
| - | Изменение знака |
| **Операторы присваивания** | |
| = | Присваивание значения |
| +=, -=, *=, /=, %= | Увеличение, уменьшение, умножение, деление на заданную величину, взятие модуля заданной величины |
| **Операторы сравнения** | |
| == | Эквивалентность сравниваемых объектов |
| != | Не равно |
| >, >=, <, <= | Больше, больше или равно, меньше, меньше или равно |
| **Логические операторы** | |
| &&, \|\|, ! | логическое И, ИЛИ, НЕ |

Переменные PHP могут иметь один из следующих типов:

• Числовой (целые числа или числа с плавающей точкой)

• Булевый

• Строковый

• Нулевой тип (определяется ключевым словом null)

Тип данных при объявлении переменной не указывается. Тип присваивается переменной только тогда, когда ей присваивается какое-либо значение. Имена переменных в программе всегда должны начинаться с символа $, например:

```
$x=1;
$y=”string”;
```

Тип массив введен в JavaScript для возможности манипулирования разными объектами: это список всех гипертекстовых ссылок страницы, список всех картинок на странице, список, список всех элементов формы и т.п. Пользователь

может создать и свой собственный массив, используя конструктор Array(). Например:

```
     new_array = new Array();
new_array5 = new Array(5);
     colors = new Array ("red","white","blue")
```

Размерность массива может динамически изменяться. Можно сначала определить массив, а потом присвоить одному из его элементов значение. Как только это значение будет присвоено, изменится и размерность массива:

```
      colors = new
Array(); colors[5] =
"red"
```

В данном случае массив будет состоять из 6 элементов, т.к. первым элементом массива считается элемент с индексом 0.

Перечень основных свойств и методов массивов приведен в табл. 8.2.

Таблица 8.2 – Свойства и методы массивов

| Объект | Значение |
|---|---|
| length | число элементов массива |
| join | объединяет элементы массива в строку символов, в качестве аргумента в этом методе задается разделитель, например: string = acolors.join("+") |
| reverse | изменяет порядок элементов массива на обратный |
| sort | сортирует элементов массива в порядке возрастания |

Часто используемые фрагменты программы можно оформить в виде функций, вызывая их по мере необходимости из различных мест программы. Функции должны быть определена перед вызовом, и размещение всех определений функций в разделе заголовка HEAD документа гарантирует доступность этих функций при обработке документа.

Формат определения функции представлен ниже:

```
function имя([параметр 1] [,параметр 2] [...,параметр N])
{ тело функции
[return значение]
}
```

С помощью ключевого слова return функция может вернуть значение.

Условный оператор

```
if(i>0)
{выражение} else
{выражение}
```

Оператор выбора

```
 switch
(значение)
{ case label :
выражение; break;
```

```
case label : выражение;
break;
... default :
выражение;
}
```

Операторы цикла

```
for(i=0;i<9;i++)
{выражение}
 for(i in
obj)
{выражение}
 wile(условие)
{выражение}
 do
{выражение}
while (условие)
```

Операторы прерывания потока вычислений

Оператор break используется для прерывания выполнения цикла, либо оператора switch, например:

```
while(i==0)
{if(j==3) break;}
```

Оператор continue используется для рестарта операторов цикла.

```
while(i==0)
{if(j==3) continue;}
```

# II. ПРАКТИЧЕСКИЙ РАЗДЕЛ

## ЗАДАНИЯ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ

### Лабораторная работа №1. Создание html-страницы с применением CSS

**Порядок выполнения работы.**

Выполнить задания, приведенные в колонках 2, 3, 4 и 5 таблицы 1.1. В колонке 2 указаны темы, для которых необходимо разработать веб-страницу формирования меню, в колонке 3 – параметры главной ленты меню: направ- ление главной ленты меню (гориз – горизонтальное, верт – вертикальное); фон – цвет фона ленты меню; текст – цвет текста пунктов меню. В колонках 4 и 5 указано количество пунктов второго и третьего подменю (два или три), цвет фона подменю и цвет текста его пунктов.

Главное меню должно отстоять от верхней границы окна браузера а 20 px, слева – на 12 px и содержать слева графический элемент.

Таблица 1.1 – Варианты заданий

| Номер варианта | Тема | Главная лента меню: фон/текст | Подменю 1: пунктов/фон/текст | Подменю 2: пунктов/фон/текст |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 1 | Торговая фирма | Гориз/Pink/Red | 3/Blue/Blue | 2/Aqua/Violet |
| 2 | Ремонт авто | Верт/Coral/Blue | 2/Navy/Navy | 3/Olive/Aqua |
| 3 | Расписание занятий | Гориз/Gold/Navy | 3/Olive/Aqua | 2/Navy/Navy |
| 4 | Летняя одежда | Верт/Khaki/Aqua | 2/Aqua/Violet | 3/Blue/Blue |
| 5 | Ремонт компьютеров | Гориз/Plum/Black | 3/Black/Red | 2/Tan/Gold |
| 6 | Швейное ателье | Верт/Purple/Violet | 2/Tan/Gold | 3/Violet/Blue |
| 7 | Ремонт телевизоров | Гориз/Indigo/Tan | 3/Violet/Blue | 2/Purple/Navy |
| 8 | Головные уборы | Верт/Peru/Coral | 2/Purple/Navy | 3/Blue/Red |
| 9 | Зимняя одежда | Гориз/Gray/Gold | 3/Blue/Red | 2/Olive/Aqua |
| 10 | Продукты питания | Верт/Red/Plum | 2/Olive/Aqua | 3/Peru/ Gold |
| 11 | Детские игрушки | Гориз/Lime/Peru | 3/Peru/ Gold | 2/Pink/ Blue |
| 12 | Модели авто | Верт/Aqua/Gray | 2/Pink/ Blue | 3/Blue/Blue |

Контрольные вопросы

1. Какие программные средства используются для создания веб-страниц?
2. Для чего используются CSS в веб-программировании?
3. Какова структура объявления стиля?
4. Из каких разделов состоит html-документ?
5. Как изменить параметры текста в html-документе (размер, цвет, стиль)?

6. Как создать таблицу в html-документе и вставить рисунок в ее ячейку?
7. Как построить гиперссылку, заголовок и список в html-документе?
8. Как построить гиперссылку на графическом элементе?
9. Как установить параметры ленты меню веб-страницы?
10. Как подключить таблицу CSS стилей к html-документу?

## Лабораторная работа №2. Создание простых скриптов на JavaScript

Для расширения функциональных возможностей веб-страниц используются скрипты JavaScript, которые включаются в html-документы несколькими способами:

– в теговом контейнере <body>...</body>:

```
<body>
…
<script > команды скрипта</script>
…
</body>
```

– в контейнере <head>...</head>, если скрипт представляет собой функцию, вызываемую в ответ на какое-либо событие:

```
<head>
…
<script type=”text/javascript”> команды сценария </script>
…
</head>
```

– во внешнем файле с расширением js:

```
<head>
…
<script type = "text/javascript" src = "my.js"> </script>
…
</head>
```

**Порядок выполнения работы.**

Выполнить четыре следующих задания:

Создать строку текста из 22 первых букв русского алфавита: var str = 'abcde … '. Используя функцию ***alert***, вывести символы с указанными в столбце «Задание 1» таблицы 1.2 номерами и разделить их номером варианта, например, для варианта 11: a-11-c-11-e11 и так далее.

Построить строку из цифр «Номера символов» первого задания. Из полученной строки, состоящей из 12 цифр, выделить четыре трехзначных числа. Используя полученные числа и заданные в колонке «Задание 2» таблицы 1.2 операции, построить оператор присваивания с полученным арифметическим выражением, полученный результат вывести в окно браузера.

Построить строку текста из букв, номера которых заданы в «Задание 1». Используя свойство ***innerHTML*** метода ***document.getElementById*(*id*),** вывести на страницу html построенную строку.

С помощью функции ***confirm*** построить запрос, содержащий две кнопки: Да и Нет. В зависимости от выбранной кнопки вычислить заданные в «Задании 3» таблицы 2.1 выражения: для Да – y, для Нет – f. Результат вывести на страницу html используя функцию writeln.

Таблица 2.1 – Задания к лабораторной работе

| Номер вари-анта | Задание 1 – номера символов | Задание 2 – операции | | | Задание 3 – арифметическое выражение | |
|---|---|---|---|---|---|---|
| | | | | | y = | f = |
| 1 | 01, 03, 05, 06, 08, 09 | + | – | * | $(a + b)^2/c$; | d/(f – e/2) |
| 2 | 02, 03, 04, 06, 07, 15 | + | – | / | $(a * b) – c^2$; | 2*d/(f + e/4) |
| 3 | 04, 05, 07, 09, 12, 18 | + | – | % | $(a – b/c)^2$; | d*(2*f – e/2) |
| 4 | 03, 05, 06, 12, 20, 21 | / | * | – | $(a^2 + b)/c$; | d/(f/5 + e^2) |
| 5 | 01, 02, 08, 15, 17, 19 | / | * | + | $(a – b/c)^2/c$; | 2*d/(f/1.4 – e/2) |
| 6 | 13, 15, 16, 17, 18, 20 | / | * | % | $(a + b/c^2)/c$; | d/(f*e/2) + d |
| 7 | 05, 07, 09, 13, 16, 18 | * | + | – | $(a – b/a)/c^2$; | d/(f*e + 2*d) – d |
| 8 | 03, 04, 09, 12, 14, 17 | * | + | / | $(a + b/a^2)/2*c$; | d/(f*e – 2*d) + e |
| 9 | 02, 09, 16, 17, 18, 21 | * | + | % | $(a – b/a*2)/c^2$; | d/(e*f/2) + d*e/f |
| 10 | 03, 07, 08, 09, 10, 20 | – | * | / | $(a + b/c^2)/c$-a; | d/(f*e/2 + d) – 2*f |
| 11 | 01, 03, 05, 07, 10, 11 | – | * | + | $(a – b/c/2)/c^a$; | d/(f*e + 2*f) + d |
| 12 | 06, 08, 09, 12, 13, 19 | – | * | % | (a + b/c+2)/c–2*b; | d/(f*e – 2/f) – d |

Контрольные вопросы

1. Как выделить символ из строки текста?
2. Как встроить JavaScript-код в html-документ?
3. Какие комментарии используются в языке JavaScript?
4. Какие функции вывода на страницу html Вы знаете?
5. Прокомментируйте технологию использования метода document.getElementById(id) вывода на страницу html.
6. Какие математические операции используются в JavaScripn?
7. Как извлечь символ строки по его номеру?

8. Для чего используются методы document.write(), alert(), console.log()?
9. Какие способы включения JavaScript-кода в html-документ Вы знаете?
10. Для каких целей используются методы prompt и Confirm?

## Лабораторная работа № 3. Функции и обработка событий JavaScript

В JavaScript используются простые типы данных и объекты. К простым типам относятся числа, строки, логический тип, null и undefined. Среди объектов выделяют обычные и специальные объекты. Обычные объекты – это число или строка, а специальные объекты – это массивы, функции, объект даты, регулярные выражения и ошибки.

Если при наступлении события требуется произвести много действий, то удобно написать сценарий в виде функции и разместить его в контейне- ре <script> …</script>, предназначенном для сценариев. Например, для вывода модуля заданного числа используется функция Math.abs, а для округления чисел – функции Math.round, Math.ceil и Math.floor, а также методы toFixed и toPrecision.

Функции Math.min, Math.max, Math.sqrt, Math.pow, Math.random используются для определения минимального, максимального значений, вычисления квадратного корня, возведения в степень и генерации псевдослучайных чисел с равномерным законом распределения.

Для работы со строками текста используются следующие методы: length, toUpperCase, toLowerCase, substr, substring, slice, indexOf, replace, split и функ- ция join. Чтобы обратить внимание пользователя веб-сайта на определённый элемент html-документа, его свойства можно менять, например, цвет или размер, при попадании на него курсора мыши, а при снятии курсора восста- навливать прежние значения.

**Порядок выполнения работы.**

Выполнить два задания (таблица 3.1).

Таблица 3.1 – Задания к лабораторной работе

| Номер варианта | Задание 1 | | Задание 2 |
|---|---|---|---|
| | Используя Math.abs, вычислить модуль числа | Округлить число до n знаков в дробной части, используя функцию | Выполнить операции с элементами одномерного массива из n вещест- венных чисел, используя функцию |
| 1 | 2 | 3 | 4 |
| 1 | 278,785 | Math.round, n = 1 | Math.min, n = 8 |
| 2 | 547,473 | Math.ceil, n = 2 | Math.max, n = 9 |
| 3 | 182,487 | Math.floor, n = 3 | Math.sqrt, n = 5 и просуммирвать |
| 4 | 264,289 | toFixed, n = 2 | Math.pow, n = 6 и просуммирвать |
| 5 | 452,297 | Math.round, n = 2 | Math.random, n = 9 и просуммирвать |

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 6 | 984,573 | Math.ceil, n = 4 | Math.min, n = 7 |
| 7 | 893,284 | Math.floor, n = 1 | Math.max, n = 8 |
| 8 | 487,651 | toFixed, n = `3 | Math.sqrt, n = 5 и просуммирвать |
| 9 | 774,652 | Math.round, n = 3 | Math.pow, n = 9 и просуммирвать |
| 10 | 682,571 | Math.ceil, n = 1 | Math.random, n = 7 и просуммирвать |
| 11 | 455,628 | Math.floor, n = 2 | Math.min, n = 9 |
| 12 | 671,475 | toFixed, n = 4 | Math.max, n = 6 |

Контрольные вопросы

1. Как получить модуль числа?
2. Какие функции и методы округления чисел Вы знаете?
3. Прокомментируйте технологию использования функций Math.sqrt, Math.pow, Math.random.
4. Прокомментируйте технологию использования функций isNaN, isFinite, parseInt, parseFloat.
5. Какая функция используется для округления числа до необходимого количества цифр в его дробной части?
6. Какие методы для работы со строками Вы знаете?
7. В каких ситуациях возникает необходимость использования регулярных выражений при работе с текстом?
8. Как сформировать последовательность псевдослучайных чисел с равномерным распределением?
9. Как выполняется глобальный поиск и замена символа в строке текста?
10. Для чего используется метод indexOf при работе со строками текста?

## Лабораторная работа № 4. Операторы ветвлений и циклов JavaScript

Операторы ветвления используются для организации выполнения блоков кода при выполнении или невыполнении некоторых условий. К таким операторам относятся конструкции if, else, switch.

Синтаксис конструкции if:

if (***логическое выражение***) {
код если ***логическое выражение*** true
} else {
}
код если ***логическое выражение*** false

Синтаксис конструкции switch:
switch (***переменная***) { case '1':
код выполняемый, если ***переменная*** имеет значение 1;
break;
...
case 'n':
код, выполняемый, если ***переменная*** имеет значение n; break;
default:
код, выполняемый, если ***переменная*** не совпала ни с одним значением; break;
}

Синтаксис конструкции for:
for (начальное значение; условие окончания цикла; команды после прохода цикла) {
тело цикла
}

Синтаксис конструкции while:
while ( пока выражение истинно ) {
выполнять код
}

***Пример 1*** – Вывести числа от 1 до 10:

```
var i = 1;
while (i <= 10) { document.write(i + ' '); i++;
}
```

Инструкция ***break*** используется для принудительного выхода из цикла, а инструкция continue – для принудительного перехода к следующей итерации цикла.

***Пример 2*** – Вывести четные числа от 2 до 10:

```
var result = ' ';
for (var i = 2; i <= 10; i++) { if (i % 2) continue;  result += i + ' ';
}
document.write(result);
```

**Порядок выполнения работы.**

Разработать консольное приложение на языке JavaScript для решения следующих задач.

Ввести переменную ***lang***, которая может принимать значения: рус, анг, бел или нем. В переменной *msw* сформировать массив дней недели на русском, английском, белорусском или немецком языке в зависимости от варианта. Задачу

решить с помощью оператора ***if***; ***switch-case*** или многомерного массива без ***if*** и ***switch-case*** в зависимости от варианта.

Дана строка вида 'ab12cde345'. Проверить, является ли символ с заданным номером (k) этой строки буквой, а сумма ее цифр – четной.

Ввести дату и по ней определить: ВГ – время года (зима, весна, лето, осень); ДМ – декаду месяца; МГ – месяц; ВсГ – високосный / не високосный год.

Ввести массив из 25 целых вещественных чисел и определить: СЧЭ – сумма четных элементов; ПЭНН – произведение элементов с нечетными номерами; СЭКЗ – сумма элементов, номера которых кратны трем; ПНЧЭ – произведение нечетных элементов массива.

Варианты заданий приведены в таблице 4.1.

Таблица 4.1 – Варианты заданий

| Номер варианта | Задача 1 | | Задача 2 | | Задача 3 | Задача 4 |
|---|---|---|---|---|---|---|
| | lang | операт/массив | строка | k | определить | определить |
| 1 | рус | If | 'abcde12345' | 2 | ВГ; ДМ | СЧЭ; ПЭНН |
| 2 | рус | switch-case | 'ab123cde45' | 4 | ВГ; МГ | СЧЭ; СЭКЗ |
| 3 | рус | массив | 'abcd12345e' | 6 | ВГ; ВсГ | СЧЭ; ПНЧЭ |
| 4 | анг | If | 'ab12cde345' | 8 | ДМ; ВГ | ПЭНН; СЧЭ |
| 5 | анг | switch-case | 'a12bcde345' | 7 | ДМ; МГ | ПЭНН; СЭКЗ |
| 6 | анг | массив | 'a1234bcde5' | 1 | ДМ; ВсГ | ПЭНН; ПНЧЭ |
| 7 | бел | If | '123abc45de' | 3 | МГ; ВГ | СЭКЗ; СЧЭ |
| 8 | бел | switch-case | '123abcde45' | 5 | МГ; ДМ | СЭКЗ; ПЭНН |
| 9 | бел | массив | '12ab34cde5' | 7 | МГ; ВсГ | СЭКЗ; ПНЧЭ |
| 10 | нем | If | '123abc45de' | 9 | ВсГ; ВГ | ПНЧЭ; СЧЭ |
| 11 | нем | switch-case | '1ab23c45de' | 4 | ВсГ; ДМ | ПННЭ; ПЭНН |
| 12 | нем | массив | '1ab2345cde' | 7 | ВсГ; МГ | ПННЭ; СЭКЗ |

Контрольные вопросы

1. Какие операторы ветвлений Вам известны?
2. Какие варианты использования оператора if Вы знаете?
3. Прокомментируйте назначение и структуру оператора switch-case.
4. Для чего предназначена инструкция break?
5. Какие операторы цикла Вы знаете?
6. Для чего предназначен оператор for in?
7. Как реализовать досрочный выход из цикла?
8. Прокомментируйте синтаксис оператора while.
9. Для чего используется инструкция continue?
10. Прокомментируйте синтаксис оператора for.

## Лабораторная работа № 5. Методы JavaScript

Методы JavaScript – это действия, которые могут выполняться над объектами, в то же время это свойства, содержащие определение функции, например,

```
var person = { firstName: "John", lastName : "Doe", id : 5566,
fullName : function() {
return this.firstName + " " + this.lastName;
}
};
```

Ключевое слово *this* в данном примере относится к владельцу функции, т. е. к объекту *Person*, который владеет функцией *function*().

Методом объекта JavaScript может быть только функция, а значением свойства объекта – любой тип данных, за исключением функции.

Основные методы JavaScript

Методы строк. Строка в JavaScript является одновременно и объектом string и переменной, поэтому может быть создана двумя способами:

```
st1 = new String("Строка – это объект") st2 = "Строка – это переменная"
```

В JavaScript используются следующие методы строк:

charAt() – извлекает из строки символ, находящийся в указанной позиции; charCodeAt() – возвращает код юникода символа, находящегося в указанной позиции (16-разрядное целое число между 0 и 65 535);

concat() – выполняет конкатенацию одного или нескольких значений со строкой. преобразует все аргументы в строки и добавляет их по порядку в конец строки;

indexOf (подстрока, начало) – выполняет поиск в строке от начала к концу; lastIndexOf() – выполняет поиск символа или подстроки в строке с конца; match() – выполняет поиск по шаблону с помощью регулярного выражения.

Для выделения нескольких символов строки используется метод substr().

Замена подстроки текста выполняется методом replace(), построенным на использовании регулярных выражений.

Методы вывода:

alert – выводит модальное окно с сообщением;

confirm – выводит сообщение в окне с двумя кнопками: «ОК» и «ОТМЕНА» и возвращает выбор посетителя;

prompt – выводит окно с указанным текстом и полем для поль- зовательского ввода;

setInterval – выполняет код или функцию через указанный интер- вал времени.

Метод document.write() выводит на страницу переданные ему аргументы. Глобальные методы JavaScript:

alert – выводит модальное окно с сообщением;

clearInterval – останавливает выполнение кода, заданное setInterval; clearTimeout – отменяет выполнение кода, заданное setTimeout;

confirm – выводит сообщение в окне с двумя кнопками «ОК»

и «ОТМЕНА» и возвращает выбор посетителя;

decodeURI – раскодирует URI, закодированный при помощи encodeURI; decodeURIComponent – раскодирует URI, закодированный при помощи

encodeURIComponent;

encodeURI – кодирует URI, заменяя каждое вхождение определен- ных символов на escape-последовательности, представляющие символы в кодировке UTF-8;

encodeURIComponent – кодирует компоненту URI, заменяя определенные символы на соответствующие UTF-8 escape-последовательности;

eval – выполняет строку javascript-кода без привязки к конк- ретному объекту;

isFinite – проверяет, является ли аргумент конечным числом; isNaN – проверяет, является ли аргумент NaN.

*Порядок выполнения работы.*

Выполнить задания.

Заполнить массив из 14 элементов вещественными числами, используя заданную в колонке 2 таблицы 5.1 функцию. Рассортировать массив в указанном в колонке 3 порядке.

Таблица 5.1 – Варианты заданий

| Номер | Задание 1 | | Задание 2 | Задание 3 |
|---|---|---|---|---|
| варианта | Функция | Сортировать | Даты | Вычислить |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Sin(x) | По возрастанию | Год, месяц, день | Площадь усеченного конуса |
| 2 | Cos(x) | По убыванию | Месяц, день, час | Площадь трапеции |
| 3 | Tg(x) | По возрастанию | Час, минута, секунда | Площадь параллелограмма |
| 4 | Ctg(x) | По убыванию | День, час, минута | Макс. значение среди трех |
| 5 | Sin(x/2) | По возрастанию | Год, месяц, день | Макс. значение среди пяти |
| 6 | Cos(x/2) | По убыванию | Месяц, день, час | Объем конуса |
| 7 | Tg(x/2) | По возрастанию | Час, минута, секунда | Объем усеченного конуса |
| 8 | Ctg(x/2) | По убыванию | День, час, минута | Объем пирамиды |
| 9 | $Sin(x^2)$ | По возрастанию | Год, мес, день | Объем усеченной пирамиды |
| 10 | $Cos(x^2)$ | По убыванию | Месяц, день, час | Площадь шестиугольника |
| 11 | $Tg(x^2)$ | По возрастанию | Час, минута, секунда | Площадь кольца |
| 12 | $Ctg(x^2)$ | По убыванию | День, час, минута | Площадь цилиндра |

Вывести текущую дату в заданном в колонке 4 формате и определить:

используя функцию ***DataParse***, количество миллисекунд, про- шедших с 01.01.1970 г. по текущий момент;

используя метод ***getTime***, количество секунд с 01.01.1970 г. по текущий момент;

используя метод ***getDay***, номер и название дня недели, в который Вы родились.

Вычислить значение элемента, заданного в колонке 5.

*Контрольные вопросы*

1. Для чего используются методы *alert*(), *prompt*() и *document.write*()?
2. Какие методы объекта *Math* Вы знаете?
3. Прокомментируйте назначение метода *document.getElementById(id)*.
4. Какие два способа создания строки Вы знаете?
5. Что понимается под методом в *JavaScript*?
6. Какие методы вывода в JavaScript Вы знаете?
7. Какой метод используется для поиска номера символа, с которого начинается подстрока?
8. Через какой объект выполняется работа с датами в *JavaScript*?
9. Какой метод используется для округления чисел?
10. Прокомментируйте назначение функции *Date.parse*.

## Лабораторная работа № 6. Работа с массивами на JavaScript

Массивы в *JavaScript* являются нетипизированными, т. е. элементы одного и того же массива могут иметь разные типы. Элементы массива могут быть объектами или другими массивами, что позволяет создавать сложные структуры данных, такие как массивы объектов и массивы массивов.

Отсчет индексов массивов в языке JavaScript начинается с нуля, и для них используются 32-битные целые числа. Массивы в JavaScript являются динамическими: они могут увеличиваться и уменьшаться в размерах по мере необходимости, поэтому нет необходимости объявлять фиксированные размеры массивов при их создании или повторно распределять память при изменении их размеров.

Создать массив проще всего с помощью литерала, который представляет собой простой список разделенных запятыми элементов в квадратных скобках. Значениями литерала массива могут быть константы, выражения, литералы объектов:

```
var empty = []; // Пустой массив
var numbers = [2, 3, 5, 7, 11]; // Массив с пятью числовыми элементами
var misc = [ 1.1, true, "a", ]; // 3 элемента разных типов + завершающая запятая
var base = 1024;
var table = [base, base+1, base+2, base+3]; // Массив с переменными
var arrObj = [[1,{x:1, y:2}], [2, {x:3, y:4}]];  // 2 массива внутри, содержащие объекты
```

Синтаксис литералов массивов позволяет вставлять необязательную завершающую запятую, т. е. литерал [,,] соответствует массиву с двумя элементами, а не с тремя.

Другой способ создания массива – конструктор Array(), который можно вызвать тремя разными способами: без аргументов; с единственным числовым аргументом, определяющим длину массива; с явным указанием значений первых двух или более элементов или одного нечислового элемента:

```
var arr = new Array(); // пустой массив, эквивалентный литералу []
var arr = new Array(10); // пустой массив указанной длины
var arr = new Array(5, 4, 3, 2, 1, "тест"); //массив с явным указанием элементов
```

Массив является специализированной разновидностью объекта, поэтому квадратные скобки, используемые для доступа к его элементам, действуют аналогично доступу к свойствам объекта. Интерпретатор JavaScript преобразует указанные в скобках числовые индексы в строки, например, индекс 1 – в строку "1", а затем использует эти строки как имена свойств.

Следует четко отличать индексы в массиве от имен свойств объектов. Все индексы массива являются именами свойств, но только свойства с именами, представленными целыми числами, являются индексами. Массивы являются объектами, и к ним можно добавлять свойства с любыми именами. Однако если затрагиваются свойства, которые являются индексами массива, то массивы реагируют на это, обновляя при необходимости значение свойства length.

В качестве индексов массивов допускается использовать отрицательные и нецелые числа. В этом случае числа преобразуются в строки, которые используются как имена свойств.

***Добавление и удаление элементов массива***. Самый простой способ добавления элементов массива – это присваивание значений его новым индексам. Для добавления одного или более элементов в конец массива можно также использовать метод push():

```
var arr = []; // Создать пустой массив arr.push('zero'); //
Добавить значение в конец массива arr.push('one',2); //
Добавить еще два значения в массив
```

Добавить элемент в конец массива можно также, присвоив значение элементу arr[arr.length]. Для вставки элемента в начало массива можно использовать метод unshift(), при этом существующие элементы в массиве смещаются в позиции с более высокими индексами.

Удалять элементы массива можно как обычные свойства объекта – с помощью оператора delete:

```
var arr =
[1,2,'three']; delete
arr[2];
1 in arr; // false, индекс 2 в массиве не определен
arr.length; // 3: оператор delete не изменяет свойство length массива
```

***Многомерные массивы.*** JavaScript не поддерживает настоящие многомерные массивы, но позволяет имитировать их при помощи массива из массивов. Для доступа к элементу данных в массиве массивов достаточно дважды использовать оператор [ ].

*Методы класса Array*:

Array.join() – преобразует все элементы массива в строки, объединяет их и возвращает получившуюся строку.

Array.reverse() – меняет порядок следования элементов в массиве на обратный и возвращает переупорядоченный массив.

Array.sort() – сортирует элементы в исходном массиве и возвращает отсортированный массив.

Array.concat() – создает и возвращает новый массив, содержащий элементы исходного массива, для которого был вызван метод concat(), и значения всех переданных ему аргументов.

Array.slice() – возвращает фрагмент, или подмассив указанного массива.

Порядок выполнения работы**.**

Выполнить задания, варианты которых приведены в таблице 6.1.

Таблица 6.1 – Варианты заданий

| Номер варианта | Задание 1 | | Задание 2 | Задание 3 | | Задание 4 |
|---|---|---|---|---|---|---|
| | строки | k, m, n | m, n | сортировать | k, m | m, k |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | ['a', 'b', 'c'] | 3, 2, 4 | 4, 6 | По возрастанию | 7, 8 | 6, 9 |
| 2 | [1, 2, 3] | 4, 3, 2 | 5, 8 | По убыванию | 6, 5 | 7, 6 |
| 3 | ['p', 'n', 'k'] | 2, 1, 5 | 3, 5 | По возрастанию | 8, 6 | 8, 9 |
| 4 | [7, 5, 8] | 4, 2, 3 | 4, 6 | По убыванию | 5, 8 | 5,7 |
| 5 | ['r', 's', 't'] | 3, 4, 4 | 5, 7 | По возрастанию | 6, 7 | 5, 8 |
| 6 | [2, 6, 8] | 5, 3, 2 | 4, 6 | По убыванию | 5, 8 | 5, 9 |
| 7 | ['k', 'm', 's'] | 4, 1, 3 | 5, 8 | По возрастанию | 8, 7 | 6, 5 |
| 8 | [3, 7, 5] | 2, 5,4 | 3, 6 | По убыванию | 6, 8 | 6, 7 |
| 9 | ['q', 'z', 's'] | 5, 1, 3 | 4, 5 | По возрастанию | 5, 6 | 6, 8 |
| 10 | [2, 9, 4] | 3, 4, 5 | 5, 7 | По убыванию | 8, 8 | 6, 6 |
| 11 | ['t', 'r', 'n'] | 4, 2, 3 | 2, 6 | По возрастанию | 7, 7 | 7, 6 |
| 12 | [4, 7, 2] | 3, 5,2 | 3, 8 | По убыванию | 6, 5 | 7, 7 |

Создать одномерный массив R из k элементов, указанных в ко- лонке 2 таблицы 1.6. Добавить m числовых элементов в его начало и n текстовых – в конец.

Создать двухмерный массив размерностью m × n (колонка 4 таблицы 1.6) и заполнить случайными равномерно распределенными числами. Строки с четными номерами рассортировать.

Создать двухмерный массив размерностью k × m, содержащий текстовые и числовые элементы. Используя метод Array.join(), преобразовать все числовые элементы массива в строки; изменить порядок следования элементов массива на обратный.

Создать двухмерный текстовый массив размерностью m × k. Используя методы unshift() и push(), добавить k – 4 строк в начало массива и с помощью методов shift() и pop() удалить m – 3 строк из конца заданного массива.

*Контрольные вопросы*

1. Какие методы создания массивов Вы знаете?
2. Что такое динамический массив в JavaScript?
3. Что может быть элементом массива в JavaScript?
4. С какого значения ведется отсчет индексов массивов в JavaScript?
5. Как построить доступ к элементу массива?
6. Какой тип может иметь индекс элемента массива?
7. Как понимать: «массив – специализированная разновидность объекта»?

8. Какие методы используются для добавления и удаления эле- ментов массива?

9. Как в JavaScript поддерживается работа с многомерными массивами?

10. Какие методы класса Array Вы знаете?

## Лабораторная работа №7. Работа с элементами управления на JavaScript

Язык JavaScript позволяет создавать сложные веб-элементы управления сайтами, среди которых сложные меню, специализированные деревья и сложные сетки, а также два специальных – генератор всплывающих окон и динамически меняющаяся кнопка.

Всплывающие (popup-) окна – это один из способов, позволяющих показать пользователю дополнительный контент.

В недавнем прошлом всплывающими окнами злоупотребляли многие сайты, нацеленные на показ рекламы, и загружали пользователей множеством объявлений. Поэтому современные браузеры блокируют всплывающие окна.

Всплывающее окно достаточно просто отображается с помощью функции window.open() в блоке JavaScript:

```
window.open('http://www.google.com', 'myWindow',
'toolbar=0, height=500, width=800, resizable=1, scrollbars=1');
window.focus();
```

Функция window.open() принимает параметры: ссылка на новую страницу и имя фрейма окна, в которое позже должен быть загружен новый документ посредством другой ссылки. Третий параметр – разделенная запятыми строка, конфигурирующая стиль и размер всплывающего окна с помощью атрибутов:

height – высота и width – ширина в пикселях;

toolbar – панель инструментов и menuBar – строка меню, которые могут быть установлены в 1 или 0, в зависимости от того, требуется ли отобра- жение этих элементов;

resizable = 1 – рамка изменяемого размера, = 0 – фиксированного;

scrollbars = 1, если требуются линейки прокрутки, = 0 – если нет.

Чтобы закрыть popup-окно, необходимо вызвать функцию newWindow.close(). Метод close() можно вызвать для любого объекта window, но window.close() игнорируется почти всеми браузерами, если окно было открыто не с помощью window.open().

Эффективным элементом управления в JavaScript является динамически меняющаяся кнопка, которая выводит на экран одно изображение, если она появляется на веб-странице впервые, при задержке над ней курсора мыши – другое, при щелчке на этой кнопке – третье.

Для обеспечения такого эффекта кнопка обычно состоит из дескрип- тора <img>, который обрабатывает JavaScript-события onclick, onmouseover и onmouseout. Эти события вызывают функции, меняющие изображения для текущей кнопки:

```
function swapImg(id, url) {
  var elm =
document.getElementById(id); elm.src = url;
}
```

В этом случае сконфигурированный дескриптор <img> выглядел бы следующим образом, html:

```
<img id="img" src="activebtn.png" style="cursor:pointer;
margin:120px" onmouseover="swapImg('img', 'hoverbtn.png')"
onmouseout="swapImg('img', 'activebtn.png')" />
```

Порядок выполнения работы**.**

Выполнить задания, варианты которых приведены в таблице 7.1.

Написать сценарий, позволяющий продемонстрировать изменения размеров и положения горизонтальной линии на странице html.

Написать сценарий формирования анкеты данных сотрудника, указанных в колонке 3 таблицы 7.1.

Написать сценарий обработки анкеты слушателя курсов повышения квалификации, содержащей: курс (первый, второй, третий); язык общения с преподавателем; изучаемые дисциплины; продолжительность курса; форму образования (очная, заочная, дистанционная); изучаемые дисциплины; стоимость. В зависимости от этих параметров определяется стоимость отдельного курса и стоимость всего обучения. Выбор значений выполнить с помощью флажков и выпадающего меню.

Таблица 7.1 – Варианты заданий

| Номер варианта | Задание 1: длина линии, %; толщина, (пикс) | Задание 2 (данные сотрудника): МР – место рождения; НАЦ – национальность; ОБР – образование; ДР – дата рождения; СП –семейное положение | Задание 3 (поля выбора): ПР – продолжительность; ФО – форма образования; ФИОП – ФИО преподавателя; ИД – изучаемые дисциплины |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 1 | 45%; 2 | Пол, МР, ОБР | Курс, язык, ПР, стоимость |
| 2 | 72%, 3 | ДР, пол, должность | Курс, язык, ФО, |
| 3 | 85%, 4 пикс | МР, возраст, пол | Курс, язык, ФИО П, стоимость |
| 4 | 57%, 2 пикс | МР, должность, пол | Курс, язык, ПР, ИД |
| 5 | 80%, 3 пикс | Пол, должность, ДР | Курс, язык, ФО, стоимость |
| 6 | 90%, 2 пикс | НАЦ, пол, возраст | Курс, язык, ФИОП, ИД |
| 7 | 50%, 2 пикс | СП, пол, должность | Курс, язык, ФО, стоимость |
| 8 | 70%, 3 пикс | Пол, НАЦ, возраст | Курс, язык, ПР, ИД |
| 9 | 80%, 4 пикс | МР, возраст, должность | Курс, язык, ФИОП, стоимость |
| 10 | 35%, 2 пикс | ДР. должность, возраст | Курс, язык, ПР, ИД |
| 11 | 45%, 3 пикс | НАЦ, должность, образование | Курс, язык, ФО, стоимость |
| 12 | 75%, 4 пикс | МР, возраст, должность | Курс, язык, ФИОП, ИД |

Контрольные вопросы

1. Какие веб-элементы управления сайтами Вы знаете?
2. Для чего используется генератор всплывающих (popup-) окон?
3. Что такое динамически меняющаяся кнопка?
4. Почему современные браузеры блокируют всплывающие окна?
5. Для чего используется функция window.open()?
6. Какие параметры содержит функция window.open()?
7. Как закрыть popup-окно?
8. Где используется и для чего свойство z-index?
9. Как работают динамически изменяющиеся кнопки?
10. Как построить динамически изменяющуюся кнопку?

## Лабораторная работа № 8. Работа с формами на JavaScript

Html-формы используются для пересылки данных от удаленного пользователя к веб-серверу. С их помощью можно организовать простейший диалог между пользователем и сервером, например, регистрацию пользователя на сервере или выбор нужного документа из представленного списка. Формы поддерживаются всеми популярными браузерами.

Для идентификации формы и ее элементов через JavaScript можно использовать два атрибута: ***name*** и ***id***.

Атрибут id используется, если не требуется отправлять данные с формы на сервер, а атрибут name – для отправки формы на сервер. При этом все элементы управления обязательно располагаются в форме.

Если в html-документе определена форма, то она доступна сценарию JavaScript как объект, входящий в объект document с именем, заданным атрибутом name тега form.

Форма имеет два набора свойств, состав одного из которых фиксированный, а другого зависит от определенных в форме элементов.

Флажок (checkbox). Свойства: name – имя объекта; value – надпись на кнопке; checked – состояние флажка (true – флажок установлен, false – не установлен); defaultChecked – отражает наличие атрибута Checked (true – есть, false – нет); метод: click( ) – меняет состояние флажка, аналогичен щелчку мышкой по кнопке.

Переключатель radio. Свойства: name – имя объекта; value – надпись на кнопке: length – количество переключателей в группе; checked – состояние переключателя: true – включен, false – выключен; defaultChecked – отражает наличие атрибута checked: true – есть, false – нет; метод: click( ) – включает переключатель. Так как группа переключателей имеет одно имя name, то адресоваться к ним необходимо как к элементам массива.

Список (select). Свойства: name – имя объекта; selectedIndex – номер выбранного элемента или первого среди выбранных (если указан атрибут multiple); length – количество элементов (строк) в списке; options – массив элементов списка, заданных тегами option; методы: focus( ) – передает списку фокус ввода; blur( ) – отбирает у списка фокус ввода.

Каждый элемент массива options является объектом со следующими свойствами: value – значение атрибута Value; text – текст, указанный после тега Option; index – индекс элемента списка; selected – присвоив этому свойству значение true, можно выбрать данный элемент; defaultSelected – отражает наличие атрибута Selected (true – есть, false – нет).

Кроме работы с готовыми списками, JavaScript может заполнять список динамически. Для записи нового элемента списка используется конструктор Option с четырьмя параметрами, первый из которых задает текст, отображаемый в списке, второй – значение элемента списка, соответствующее значению атрибута Value, третий соответствует свойству defaultSelected, четвертый – свойству selected.

Порядок выполнения работы.

Создать html-страницу с формой, содержащей элементы, описания которых приведены в таблице 8.1.

Таблица 8.1 – Варианты заданий

| Номер варианта | Описание формы |
|---|---|
| 1 | Центр изучения потребительского спроса собирает информацию о потребляемых соках в разрезе городов с населением более 1 миллиона. Используются следующие поля: фирма-производитель; название; концентрация (не более 100 % и не менее 50 %); цена; способ приготовления (переключатель), например: из концентрированного сока, из сухих материалов и т. д., содержание заменителя сахара (флажок). В случае использования заменителя сахара активизируется поле для записи его наименования. Кроме того, определена категория сока (переключатель: нектар/сок) |
| 2 | Создать форму, которую можно использовать для размещения не анонимных объявлений в глобальной сети Интернет с возможностью дальнейшего удаления автором объявления (для этого используется специальное поле-пароль). Обязательным параметром является адрес электронной почты автора объявления, который должен содержать символ @ и хотя бы одну точку |
| 3 | Пейджинговая компания желает реализовать возможность отправления сообщений абонентам через Интернет. Для этого используется следующая форма: время отправления сообщения; дата отправления сообщения, текст сообщения; номер абонента (четырехзначное число); количество повторов (1–5); автор; роуминг (флажок) по городам (список из пяти городов) |
| 4 | Создать форму для поиска книг каталога библиотеки, содержащую поле для ввода названия книги, автора, года издания. Предусмотреть поиск книг по автору, по названию книги. Для выбора темы поиска реализовать выпадающий список |
| 5 | Создать форму для заполнения электронной записной книжки, содержащую следующие поля: фамилия, имя, отчество, домашний адрес, город (выпадающий список из 10 городов), код местности (трехзначное число), номер телефона (может быть семизначным, шестизначным или пятизначным), использовать список типов контакта (друг, знакомый, коллега и т. д.). Рассмотреть 5–10 городов с разными масками телефонов |
| 6 | Один из сайтов знакомств планирует усовершенствовать свою деятельность. Для этого создана новая структура базы данных и необходимо создать форму, на основе которой будет создаваться запрос. В форме должны быть предусмотрены: информация о подающем запрос и информация об искомом человеке (две категории). Туда входят: пол (возможность выбора), возраст, цвет глаз, цвет волос (выпадающий список), рост, вес, телосложение (выпадающий список), вредные привычки (флажки) |
| 7 | Создать форму для заказов билетов на поезда, содержащую следующие поля: номер поезда; категория поезда (переключатель, два состояния – поезд местного формирования и проходящий); направление поезда (выпадающий список, например: Могилев – Минск, Могилев – Орша); название поезда (например: «Красная стрела»); количество билетов (не больше 4); класс желаемых вагонов (элементы списка: люкс, спальный, купе, плацкарт, общий) |
| 8 | Создать форму для перевода денег кредитными картами, которая должна содержать следующие поля: номер счета; номер кредитной карты; код карты (три цифры); дата окончания срока действия карты (два выпадающие списка с месяцем и годом); выбор валюты – переключатель (USD, EURO, РУБ). Форма также должна содержать кнопку «Перевести» |

Окончание таблицы 8.1

| 1 | 2 |
|---|---|
| 9 | Создать форму для дилерской регистрации на сайте торговой фирмы. Форма должна содержать поля: наименование фирмы дилера; город (поле с выпадающим списком областных центров Беларуси); адрес; телефон, E-mail, веб-сайт. В полях E-mail и веб-сайт предусмотреть автоматическое добавление символов @ и http:// |
| 10 | Создать форму для регистрации аккаунта на сервере знакомств, которая должна содержать следующие поля: имя, фамилия; пароль; подтверждение пароля; секретный вопрос; ответ; страна (выпадающий список из 10 стран). Также форма должна содержать независимый переключатель «Запомнить меня на этом компьютере» |
| 11 | Создать форму для регистрации пользователей на интернет-курсы по общеобразовательным предметам. Форма должна содержать поля: фамилия, имя, отчество; дата рождения; адрес; переключатели с выбором м/ж пола; Checkbox для выбора общеобразовательных предметов, интересующих пользователя; выпадающий список для выбора образования |
| 12 | Сотовая компания решила организовать проект по анализу рейтинга популярности тарифных планов. Требуется реализовать форму, на которой должна содержаться информация: фамилия, имя, отчество абонента; телефонный номер абонента; тарифный план абонента – выпадающий список; сумму денежных средств, затраченных на оплаты телефонных разговоров; независимый переключатель «хотите поменять тарифный план»; поле «Пожелание компании» |

Контрольные вопросы

1. Для чего используется html-формы?
2. Как организовать ввод информации в форму на JavaScript?
3. Какие существуют способы передачи данных формы?
4. Для чего предназначен метод focus()?
5. В каких целях используется программа обработки событий onSubmit?
6. Как использовать select box как навигационное меню?
7. Как использовать картинку для кнопки submit?
8. Как передавать данные между формами на различных страницах, используя JavaScript?
9. Почему document.formName.selectObject.value не отражает значение выбранного пункта в списке?
10. Как получить значение выбранной в данный момент radio button в radio group или группе checkboxes?

## Лабораторная работа № 9. Работа с изображениями на JavaScript

Изображения в html-документах представляются в виде массива, что позволяет адресоваться к ним. Они имеют определенные свойства и рассматриваются как объекты image, к которым можно обращаться из языка JavaScript. Например, можно определить размер изображения, обратившись к его свойствам width и height. То есть по записи document.images[0].width можно определить ширину первого изображения в пикселях на веб-странице.

Объект Image позволяет вносить изменения в графические образы на веб-странице, что позволяет создавать мультипликацию. Для отслеживания индекса всех изображений веб-страницы следует назначить им имена с помощью тега

```
<img src="img.gif" name="myImage" width=100 height=100>
```

Тогда для обращения к изображению необходимо написать document.myImage или document.images["myImage"].

Смена изображения на веб-странице выполняется с использованием атрибута src, который содержит адрес представленного изображения, и позво- ляет назначить ему новый адрес. В результате изображение будет загружено с этого нового адреса, заменяя на веб-странице старое:

```
<img src="img1.gif" name="myImage" width=100 height=100>
```

Здесь загружается изображение img1.gif и получает имя myImage.

В следующей строке изображение img1.gif заменяется на новое – img2.gif:

```
document.myImage.src= "img2.src".
```

При этом новое изображение всегда получает тот же размер, который был у старого, и уже невозможно будет изменить размер поля, в котором это изображение размещается.

Одной их замечательных возможностей браузеров являются слои, позволяющие позиционировать изображения, организовывать их перемещение и делать их невидимыми.

Для создания слоев можно использовать тег <layer> или тег <ilayer>. При этом можно воспользоваться следующими параметрами:

name = "layerName" – название слоя;

left = xPosition – абсцисса левого верхнего угла; top = yPosition – ордината левого верхнего угла; z-index = layerIndex – номер индекса для слоя; width = layerWidth – ширина слоя в пикселах;

clip = "x1_offset, y1_offset, x2_offset, y2_offset" – видимая область слоя; above = "layerName" – определяет, какой слой окажется под текущим; below = "layerName"

– определяется, какой слой окажется над текущим; visibility= show|hide|inherit – видимость этого слоя;

bgcolor = "rgbColor" – цвет фона, или название стандартного цвета, или rgb-запись;

background = "imageURL" – фоновая картинка.

Тег <layer> используется для слоев, которые можно точно позицио- нировать. Если не указать положение слоя с помощью параметров left и top, то по умолчанию он помещается в верхний левый угол окна.

Тег <ilayer> создает слой, положение которого определяется при формировании документа.

Поверх изображения или под ним можно показать текст.

Свойства слоев можно изменять с помощью скриптов на JavaScript. Например, в следующей строке задается горизонтальное положение слоя, смещенное на 200 пикселей:

```
document.layers["myLayer2"].left = 200;
```

Следующий пример демонстрирует, как скрипт может реагировать на сигналы о нажатии клавиш.

```
<html>
<script language="JavaScript">
window.captureEvents(Event.KEYPRESS
); window.onkeypress= pressed;
function pressed(e) {
  alert("Key pressed! ASCII-value: " + e.which); }
</script>
</html>
```

Порядок выполнения работы.

Создать html-документ, элементы и параметры которого указа- ны в таблице 9.1.

Таблица 9.1 – Варианты заданий

| Номер варианта | Количество графических элементов (слоев) | Ориентация группы элементов | Изображения меняются местами | При наведении курсора |
|---|---|---|---|---|
| 1 | 3 | Горизонтально | При наведении курсора | Увеличить на 25 % |
| 2 | 4 | Горизонтально | При наведении курсора | Увеличить на 40 % |
| 3 | 3 | Вертикально | По щелчку мыши | Уменьшить на 30 % |
| 4 | 4 | Вертикально | По щелчку мыши | Уменьшить на 40 % |
| 5 | 3 | Горизонтально | При наведении курсора | Увеличить на 35 % |
| 6 | 4 | Горизонтально | При наведении курсора | Увеличить на 45 % |
| 7 | 3 | Вертикально | По щелчку мыши | Уменьшить на 35 % |
| 8 | 4 | Вертикально | По щелчку мыши | Уменьшить на 50 % |

| Номер варианта | Количество графических элементов (слоев) | Ориентация группы элементов | Изображения меняются местами | При наведении курсора |
|---|---|---|---|---|
| 9 | 3 | Горизонтально | При наведении курсора | Увеличить на 30 % |
| 10 | 4 | Горизонтально | При наведении курсора | Увеличить на 45 % |
| 11 | 3 | Вертикально | По щелчку мыши | Уменьшить на 20 % |
| 12 | 4 | Вертикально | По щелчку мыши | Уменьшить на 45 % |

Контрольные вопросы

1. Для чего предназначен объект Image?
2. Как в html-документах представляются изображения?
3. Как из JavaScript можно адресоваться к изображениям?
4. Для чего предназначен атрибут src тега <img>?
5. Какие условия следует соблюдать, чтобы скрипт смены изображений сохранял свою гибкость?
6. Какие свойства определяют размеры объекта image?
7. Какой объект позволяет вносить изменения в графические образы на веб-странице для создания мультипликации?
8. Что позволяет делать изображения в браузерах невидимыми?
9. Какой тег используется для создания слоев?
10. Как изменить свойство слоя?

## Лабораторная работа № 10. Обработка событий на JavaScript

Событие – это сигнал от браузера о том, что что-то произошло, например, load – загрузка страницы и unload – выгрузка ее. Клиентские программы на языке JavaScript основаны на модели программирования, когда выполнение программы управляется событиями. При таком стиле программирования веб- браузер генерирует событие, когда с документом или некоторым его элементом что-то происходит. Например, веб-браузер генерирует событие, когда завершает загрузку документа, когда пользователь наводит указатель мыши на гиперссылку или нажимает клавишу на клавиатуре.

События мыши: click – клик; dblclick – двойной клик; mouseover – наведение курсора мыши на элемент; mousemove – перемещение курсора мыши над элементом; mouseout – уведение курсора мыши с элемента; mousedown – нажатие левой кнопки мыши; mouseup – отпускание левой кнопки мыши; contextmenu – нажатие правой кнопки мыши и вывод контекстного меню.

Для того чтобы обратить внимание пользователя на определённый элемент html-документа, можно менять свойства этого элемента при попадании на него курсора мышки, а при снятии курсора восстанавливать прежние значения свойств.

Например, можно менять цвет или размер элемента. Попадание курсора мышки на элемент фиксируется событием onMouseOver. Парное для него событие onMouseOut происходит при снятии курсора мышки с элемента.

Другая пара событий onMouseDown и OnMouseUp происходит при нажатии и отпускании левой кнопки мышки. Эту пару событий удобно применять для изменения свойств элементов или замены элементов на время удержания кнопки мышки нажатой.

События клавиатуры: keyup – возникает при отпускании клавиши клавиатуры; keydown – возникает при нажатии клавиши клавиатуры и длится, пока нажата клавиша; keypress – возникает при нажатии клавиши клавиатуры, но после события keydown и до события keyup.

Для того чтобы написать ответную реакцию на событие, создают обработчик события (event handler), который, как правило, представляет собой функцию.

Назначить обработчик события можно в виде атрибута элемента, безымянной функции, именованной функции или с помощью метода addEventListener().

Тип события – это строка, определяющая тип действия, вызвавшего событие. Тип mousemove, например, означает, что пользователь переместил указатель мыши, тип keydown – что была нажата клавиша на клавиатуре, а тип load – что завершилась загрузка документа или какого-либо другого ресурса из сети.

Обработчик события – это функция, которая обрабатывает событие или откликается на него. Приложения должны зарегистрировать свои функции обработчиков событий в веб-браузере, указав тип события и цель.

Реакция на событие в отдельном элементе. Так как в объектной модели объекты могут быть вложены друг в друга, то событие, происходящее в дочернем объекте, одновременно происходит и в родительском. JavaScript предоставляет различные способы локализации влияния события на иерархию объектов. Простейшей способ локализации заключается в размещении сценария в теге, на который должно воздействовать событие.

Порядок выполнения работы.

Создать html-документ, содержащий строку и изображение с подписью, параметры которого указаны в таблице 10.1.

Таблица 10.1 – Варианты заданий

| Номер вари-анта | Создать строку, при наведении курсора на которую меняется | | | Создать изображение с надписью на нем и подписью | |
|---|---|---|---|---|---|
| | шрифт | цвет шрифта | цвет фона | При щелчке на изображении меняется | Цвет при щелчке на подписи |
| 1 | Увеличить до 48 pt | Белый | Голубой | Изображение и надпись | Синий |
| 2 | Увеличить до 64 pt | Синий | Белый | Цвет шрифта и надпись | Красный |
| 3 | Увеличить до 36 pt | Красный | Голубой | Цвет фона и надпись | Зеленый |
| 4 | Увеличить до 42 pt | Зеленый | Белый | Цвет шрифта и изображение | Голубой |
| 5 | Увеличить до 52 pt | Синий | Зеленый | Цвет фона и изображение | Желтый |
| 6 | Увеличить до 56 pt | Белый | Голубой | Фон и цвет надписи | Зеленый |
| 7 | Уменьшить на 12 pt | Синий | Белый | Изображение и надпись | Синий |
| 8 | Уменьшить на 16 pt | Красный | Голубой | Цвет шрифта и надпись | Желтый |
| 9 | Уменьшить на 24 pt | Зеленый | Белый | Цвет фона и надпись | Красный |
| 10 | Уменьшить на 20 pt | Синий | Зеленый | Цвет шрифта и изображение | Голубой |
| 11 | Уменьшить на 10 pt | Красный | Голубой | Цвет фона и изображение | Зеленый |
| 12 | Уменьшить на 14 pt | Зеленый | Белый | Фон и цвет надписи | Синий |

Контрольные вопросы

1. Какие виды событий используются в JavaScript?
2. Какие события мыши Вы знаете?
3. Как назначить обработчик событий?
4. Какие события JavaScript относятся к системным?
5. Какие события клавиатуры Вы знаете?
6. Что понимается под типом события?
7. Что означают типы событий mousemove и keydown?
8. Что понимается под целью события?
9. Что такое обработчик событий?
10. Что означает тип события load?

## Лабораторная работа № 11. Основные методы JQuery

Методы jQuery позволяют манипулировать содержимым веб-страницы. Они присваивают элементам, отобранным в jQuery-объектах, заданные действия. В результате этого происходит динамическое изменение элементов и их содержимого.

Каждый метод jQuery либо сам что-либо возвращает, либо получает параметр и выполняет указанные в параметре действия.

В общем виде синтаксис для вызова метода jQuery имеет следующий вид:

```
$("селектор").имяМетода(параметры);
```

Динамическое изменение элементов веб-страниц. Библиотека jQuery упрощает процесс отбора элементов html-страниц. С помощью методов jQuery производятся манипуляции с объектной моделью документа DOM. Чтобы отобрать группу элементов, нужно передать селектор функции jQuery. В качестве селектора элемента может выступать сам элемент, его идентификатор или класс, а также комбинация селекторов: $("a"); $("#some- id"); $(".someclass"); $("header > ul:has(a)").

Функция $() возвращает объект jQuery, содержащий массив элементов DOM – так называемый обернутый набор, соответствующий указанному селектору. Большинство методов по завершении действий возвращает первоначальный набор элементов.

JQuery – это одна из наиболее известных библиотек, написанная на языке JavaScript для упрощения программирования веб-страниц. Это файл с расширением .js, который подключается к веб-странице как фрагмент скрипта и загружается в браузер вместе с веб-страницей.

Чтобы включить JQuery в веб-страницу, достаточно скачать последнюю версию библиотеки, например, файл jquery-1.9.1.js с сайта jquery.com, положить его в ту же папку, где лежит текст веб-страницы, а в текст веб-страницы вставить <script src="jquery-1.9.1.js"></script>

Файл jquery-1.9.1.js при этом имеет объем более 200 Кбайт, что может замедлять загрузку веб-страницы в браузере пользователя.

Если веб-страница маленькая и важно, чтобы все «летало» и загрузка библиотеки JQuery ничего не замедляла, то существует альтернативный метод загрузки JQuery – с сайта Google:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js">
</script>
```

Второе важное характерное для jQuery применение состоит в создании AJAX элементов, т. е. тех элементов страницы, которые отсылают на сервер данные

и получают ответ без перезагрузки страницы. К таким элементам можно отнести форму «Управление корзиной для интернет-магазина», пагинацию (нумерацию страниц сайта), вывод информера погоды и многое другое.

Порядок выполнения работы.

Используя библиотеку Jquery, создать сайт. Варианты заданий указаны в таблице 11.1.

Таблица 11.1 – Варианты заданий

| Номер варианта | Используя библиотеку Jquery, создать сайт, содержащий |
|---|---|
| 1 | Выпадающее меню |
| 2 | Плавающее меню |
| 3 | Фото галерею |
| 4 | Всплывающие окна |
| 5 | Слайдеры (блоками на странице, в пределах которых с установленной периодичностью происходит демонстрация анонсов новостей, статей или изображений) |
| 6 | Перемещающиеся блоки |
| 7 | Изменение прозрачности элементов |
| 8 | Подсвечивание текста |
| 9 | Переливание цвета текста разными оттенками |
| 10 | Плавающее меню |
| 11 | Всплывающие окна |
| 12 | Изменение прозрачности элементов |

Контрольные вопросы

1. Что такое JQuery?
2. Что такое слайдер?
3. Какие действия выполняют методы JQuery?
4. Прокомментируйте синтаксис вызова методов jQuery.
5. Что понимается под селектором jQuery?
6. Какие основные правила использования jQuery Вы знаете?
7. Как подключить библиотеку jQuery к веб-странице?
8. Где найти файл библиотеки jQuery?
9. Что такое альтернативный метод загрузки jQuery?
10. Какие основные методы jQuery Вы знаете?

## Лабораторная работа № 12. Основные события JQuery

События jQuery, представляющие собой момент, в который что-либо происходит, например щелчок кнопки мыши, помогают сделать веб-страницы интерактивными, реагирующими на простейшие действия пользователя.

Момент, в который произошло событие, называется запуском события. События могут срабатывать при выполнении различных операций с веб- страницей. Помимо этого, и сам браузер может стать источником событий.

Управление веб-страницей производится с помощью следующих событий: мыши; документа/окна; форм; клавиатуры; jQuery.

События мыши:

.click() – запускается при нажатии и отпускании кнопки мыши, применяется к ссылкам, картинкам, кнопкам, абзацам, блокам и т. д.;

.dblclick() – запускается при двойном нажатии и отпускании кнопки мыши, например, при открытии какой-либо папки;

.mousedown() – происходит во время нажатия кнопки мыши, например, при перетаскивании элементов;

.mousemove() – запускается при перемещении указателя мыши по элементу;

.mouseout() – запускается при отпускании кнопки мыши. События документа/окна:

.load() – запускается, когда браузер загрузит все файлы веб-страницы: html-файлы, внешние css- и Javascript-файлы, медиафайлы;

.resize() – запускается, когда пользователь изменяет размер окна браузера;

.scroll() – запускается, когда пользователь использует полосы прокрутки, либо прокручивает веб-страницу с помощью колесика мыши, либо использует для этих целей клавиши клавиатуры (pgup, pgdn, home, end);

.unload() – запускается, когда пользователь собирается покинуть страницу, щелкая по ссылке для перехода на другую страницу, закрывает вкладку страницы или окно браузера.

События форм:

.blur() – запускается, когда поле формы выводится из фокуса, например, при переходе в другое поле формы;

.change() – запускается при изменении статуса поля формы, например при выборе пункта из выпадающего меню;

.focus() – запускается при переходе в поле формы при щелчке на нем кнопкой мыши или клавишей табуляции;

.reset() – позволяет вернуть форму в первоначальное состояние, отменив сделанные изменения;

.select() – запускается при выделении текста внутри текстового поля формы;

.submit() – запускается при отправлении заполненной формы с помощью щелчка по кнопке «Отправить» или нажатии клавиши «Enter», когда курсор помещен в текстовом поле.

События клавиатуры:

.keydown() – запускается при нажатии клавиши перед событием keypress.

.keypress() – запускается при нажатии на клавишу до тех пор, пока клавиша не будет отпущена;

.keyup() – запускается при отпускании клавиши. События jQuery:

.hover() – позволяет одновременно решить две задачи, связанные с событием наведения указателя мыши и событием снятия указателя мыши в отношении выбранного объекта;

.toggle() – работает аналогично событию hover() с разницей в том, что оно запускается от щелчка кнопкой мыши. Например, можно открыть выпадающее меню одним щелчком и скрыть вторым.

Объект события: при запуске события браузер сохраняет информацию о нём в объекте события, который содержит данные, собранные в момент, когда событие произошло. Обработка события происходит с помощью функции, при этом объект передается функции как аргумент-переменная evt.

Объект события имеет различные свойства, наиболее распространённые из которых следующие:

pageX – расстояние (px) от указателя мыши до левого края окна браузера; pageY – расстояние (px) от указателя мыши до верхнего края окна браузера; screen – расстояние (px) от указателя мыши до левого края монитора; screenY – расстояние (px) от указателя мыши до верхнего края монитора;

shiftKey – TRUE, если была нажата клавиша SHIFT, когда происхо- дило событие;

which – используется для определения числового кода нажатой клавиши (вместе с shiftKey);

target – означает, что по объекту события щелкнули кнопкой мыши (например, для события click());

data – объект, использованный с функцией bind() для передачи данных функции, управляющей событием.

Порядок выполнения работы.

Используя события, указанные в таблице 12.1, и библиотеку Jquery, создать сайт.

Таблица 12.1 – Варианты заданий

| Номер варианта | Используя библиотеку Jquery, создать сайт, содержащий |
|---|---|
| 1 | События мыши .click() и .mousedown() |
| 2 | События мыши .dblclick() и .mousemove() |
| 3 | События документа/окна .load() и .scroll() |
| 4 | События документа/окна .resize() и .unload() |
| 5 | События форм. blur(), .focus() и .select() |
| 6 | События форм .change(), .reset() и .submit() |
| 7 | События клавиатуры .keydown() и .keyup() |
| 8 | События клавиатуры .keypress() и .keyup() |
| 9 | Событие jQuery .hover() |
| 10 | Событие jQuery .toggle() |
| 11 | События мыши .click() и .mousemove() |
| 12 | События форм. blur(), .focus() и .submit() |

Контрольные вопросы

1. Что представляют собой события jQuery?
2. С помощью каких событий производится управление веб-страницей?
3. Какие события мыши Вы знаете?
4. Что такое объект события?
5. Какие события документа/окна Вы знаете?
6. Какие события клавиатуры Вы знаете?
7. Какие события форм Вы знаете?
8. Какие события jQuery Вы знаете?
9. В каком объекте браузер сохраняет информацию о событии при его запуске?
10. Какие свойства объекта события Вы знаете?

## Лабораторная работа №13. Изучение строковых функций языка PHP

В языке PHP используются три способа задания строк: с помощью одинарных кавычек, двойных кавычек и с использованием heredoc-синтаксиса.

Строки, содержащие заключенные в одинарные кавычки переменные и управляющие последовательности специальных символов, не обрабатываются.

Важнейшим свойством строк в двойных кавычках является обработка содержащихся в них переменных.

Определение строк с использованием heredoc-синтаксиса начинается с символа <<<, после которого следует идентификатор. Заканчивается строка

этим же идентификатором, который должен начинаться с первой пози- ции новой строки.

Heredoc-текст ведет себя так же, как и строка в двойных кавычках. Это означает, что в heredoc нет необходимости экранировать кавычки, но можно использовать управляющие последовательности. Переменные внутри heredoc также обрабатываются.

Для работы со строками в PHP имеется более ста функций (таблица 13.1).

Таблица 13.1 – Некоторые функции обработки строк

| Номер функции | Функция |
|---|---|
| 1 | 2 |
| 1 | chunk_split – разбивает строку на фрагменты |
| 2 | echo – выводит одну или более строк |
| 3 | explode – разбивает строку с помощью разделителя |
| 4 | implode – объединяет элементы массива в строку |
| 5 | lcfirst – преобразует первый символ строки в нижний регистр |
| 6 | ltrim – удаляет пробелы или другие символы из начала строки |
| 7 | print – выводит строку |
| 8 | printf – выводит отформатированную строку |
| 9 | rtrim – удаляет пробелы или другие символы из конца строки |
| 10 | similar_text – вычисляет степень похожести двух строк |
| 11 | sprintf – возвращает отформатированную строку |
| 12 | sscanf – разбирает строку в соответствии с заданным форматом |
| 13 | stripos – возвращает позицию первого вхождения подстроки без учета регистра |
| 14 | strlen – возвращает длину строки |
| 15 | str_pad – дополняет строку другой строкой до заданной длины |
| 16 | str_replace – заменяет все вхождения строки поиска на строку замены |
| 17 | str_shuffle – переставляет символы в строке случайным образом |
| 18 | str_split – преобразует строку в массив |
| 19 | str_word_count – возвращает информацию о словах, входящих в строку |
| 20 | strip_tags – удаляет теги html и PHP из строки |
| 21 | strpbrk – ищет в строке любой символ из заданного набора |
| 22 | strpos – возвращает позицию первого вхождения подстроки |
| 23 | strrchr – находит последнее вхождение символа в строке |
| 24 | strrev – переворачивает строку задом наперед |
| 25 | stripos – возвращает позицию первого вхождения подстроки без учета регистра |
| 26 | strlen – возвращает длину строки |
| 27 | strpbrk – ищет в строке любой символ из заданного набора |

| 1 | 2 |
|---|---|
| 28 | strpos – возвращает позицию первого вхождения подстроки |
| 29 | strrchr – находит последнее вхождение символа в строке |
| 30 | strrev – переворачивает строку задом наперед |
| 31 | strripos – возвращает позицию последнего вхождения подстроки без учета регистра |
| 32 | strrpos – возвращает позицию последнего вхождения подстроки в строке |
| 33 | strspn – возвращает длину участка в начале строки, соответствующего маске |
| 34 | strstr – находит первое вхождение подстроки |
| 35 | strtok – разбивает строку на токены |
| 36 | strtolower – преобразует строку в нижний регистр |
| 37 | strtoupper – преобразует строку в верхний регистр |
| 38 | strtr – преобразует заданные символы или заменяет подстроки |
| 39 | substr_count – возвращает число вхождений подстроки |
| 40 | substr_replace – заменяет часть строки |
| 41 | substr – возвращает подстроку |
| 42 | trim – удаляет пробелы или другие символы из начала и конца строки |
| 43 | ucfirst – преобразует первый символ строки в верхний регистр |
| 44 | ucwords – преобразует в верхний регистр первый символ каждого слова строки |
| 45 | vfprintf – записывает отформатированную строку в поток |
| 46 | vprintf – выводит отформатированную строку |
| 47 | vsprintf – возвращает отформатированную строку |
| 48 | wordwrap – переносит строку по указанному количеству символов |
| 49 | serialize() – создает строковое представление текущего состояния массива или объекта |
| 50 | unserialize() – восстанавливает состояние массива / объекта из строки |

Порядок выполнения работы.

Работа со строками:

– определить строку с использованием синтаксиса одинарных кавычек;

– определить строку с использованием синтаксиса двойных кавычек;

– определить строку с использованием heredoc-синтаксиса;

– создать массив из трех-пяти элементов, вывести его с использованием echo, print, print_r, serialize и пояснить полученные результаты.

Составить программу на языке PHP с использованием функций, указанных в таблице 13.2, согласно варианту.

Таблица 13.2 – Варианты заданий

| Номер вари-анта | Номер функции | Номер вари-анта | Номер функции | Номер вари-анта | Номер функции |
|---|---|---|---|---|---|
| 1 | 1, 13, 25, 37 | 5 | 5, 17, 29, 41 | 9 | 9, 21, 33, 45 |
| 2 | 2, 14, 26, 38 | 6 | 6, 18, 30, 42 | 10 | 10, 22, 34, 46 |
| 3 | 3, 15, 27, 39 | 7 | 7, 19, 31, 43 | 11 | 11, 23, 35, 47 |
| 4 | 4, 16, 28, 40 | 8 | 8, 20, 32,44 | 12 | 12, 24, 36, 48 |

Контрольные вопросы

1. Каковы особенности строк, записанных в одинарных кавычках?
2. В чем особенности строк, записанных в двойных кавычках?
3. Что понимается под heredoc-синтаксисом?
4. Как объединить элементы массива в строку?
5. Как удалить пробелы или другие символы из конца строки?
6. Как преобразовать строку в массив?
7. Какая функция переворачивает строку задом наперед?
8. Какие функции преобразуют символы строки в верхний регистр?
9. Как удалить пробелы из начала и конца строки?
10. Что такое heredoc-текст?

## Лабораторная работа № 14. Изучение операторов цикла языка PHP

В языке PHP существует несколько конструкций, позволяющих выполнять повторяющиеся действия в зависимости от условия. Это циклы while, do ...while, foreach и for.

while – это простой цикл. Он имеет две формы записи: while (выражение) { блок_выполнения }

либо

while (выражение): блок_выполнения endwhile;

Циклы do..while похожи на циклы while, но в них истинность выражения проверяется в конце цикла. Форма записи:

do {блок_выполнения} while (выражение);

Цикл for со счетчиком используется для выполнения тела цикла определенное число раз. Синтаксис цикла for:

for (инициализирующие_команды; условие; команды_после_итерации)

{тело_цикла;}

Цикл for начинает свою работу с выполнения инициализирующих_команд, которые выполняются только один раз. Затем проверяется условие_цикла, и если оно истинно (true), то выполняется тело_цикла. После того как будет выполнен

последний оператор тела, выполняются команды_после_итерации. Затем снова проверяется условие_цикла. Если оно истинно (true), выполняется тело_цикла и команды_после_итерации и т. д. Например:

```
<?php
for ($x=0; $x<10; $x++) echo $x;
?> // выводит: 0123456789
```

Если необходимо указать несколько команд, то их можно разделить запятыми, например:

```
<?php
for ($x=0, $y=0; $x<10; $x++, $y++) echo $x;
?> // Выводит 0123456789
```

Пример использования нескольких команд в цикле for:

```
<?php
for($i=0,$j=0,$k="Точки"; $i<10; $j++,$i+=$j) { $k=$k."."; echo $k; }
                    // Выводит Точки.Точки..Точки...Точки....
?>
```

Цикл for имеет альтернативный синтаксис:

```
for(инициализирующие_команды;                                    условие;
команды_после_итерации); операторы;
endfor;
```

Цикл foreach перебора массивов имеет синтаксис:

```
foreach (массив as $ключ=>$значение)
команды;
```

Пример цикла foreach:

```
<?php
$names["Иванов"] = "Андрей"; $names["Петров"] = "Борис";
$names["Волков"] = "Сергей"; $names["Макаров"] = "Федор";
foreach ($names as $key => $value) {
echo "<b>$value $key</b><br>";
}
?>
```

выводит: Андрей Иванов
Борис
Петров Сергей
Волков Федор
Макаров

Порядок выполнения работы**.**

Написать и отладить скрипт, выполняющий действия, указанные в таблице 14.1, согласно варианту.

Таблица 14.1 – Варианты заданий

| Номер вари-анта | С помощью цикла |
|---|---|
| 1 | 2 |
| 1 | Найти сумму корней чисел от 1 до 15. Результат округлить до двух знаков в дробной части |
| 2 | Найти сумму тех чисел от 1 до 100, которые делятся на 7 |
| 3 | Создать строку из шести символов, состоящую из случайных чисел от 1 до 9 |
| 4 | Дан массив с числами, найти сумму квадратов его элементов |
| 5 | Дан массив с числами, найти корень из суммы квадратов элементов этого массива, а результат округлить в меньшую сторону до целых |
| 6 | Дан массив с числами, найти сумму тех его чисел, которые больше 0 и меньше 10 |
| 7 | Заполнить двумерный массив, содержащий 10 подмассивов, случайными числами от 1 до 10. В каждом подмассиве должно быть по 10 элементов |
| 8 | Преобразовать строку 'var_text_hello' в 'varTextHello'. Скрипт должен работать с любыми строками такого типа |
| 9 | Дан массив с произвольными числами. Сделать так, чтобы элемент в массиве повторился количество раз, соответствующее его значению. Например, [1, 3, 2, 4] должен превратиться в [1, 3, 3, 3, 2, 2, 4, 4, 4, 4] |
| 10 | Дана строка, удалить из этой строки четные символы |
| 11 | Дана строка, поменять ее первый символ на второй и наоборот, третий на четвертый и наоборот, пятый на шестой и наоборот и т. д., т. е. из строки '12345678' необходимо сформировать строку '21436587' |
| 12 | Написать скрипт, который проверяет, являются ли заданные числа простыми, т. е. делящимися только на единицу и сами на себя |

Контрольные вопросы

1. Какие операторы цикла Вы знаете?
2. Какие формы записи конструкции while Вы знаете?
3. Прокомментируйте работу цикла do..while.
4. Для чего используется конструкция for?
5. Какие формы записи конструкции for Вы знаете?
6. Для чего используется оператор break в цикле for?
7. Прокомментируйте работу конструкции foreach.
8. Какие конструкции цикла используют для работы с массивами?
9. Как прервать выполнение цикла?
10. Для чего используется оператор break?

## Лабораторная работа № 15. Изучение приемов работы с массивами на языке PHP

В языке PHP в одном массиве допускается хранение переменных различных типов, а также массивов и объектов. Для обращения к элементу массива используется его индекс (ключ).

PHP поддерживает работу с индексными и ассоциативными массивами, индексами которых являются строки.

Для обращения к элементам индексных массивов используются числовые индексы, а ассоциативных – строковые.

Для создания массивов можно использовать конструкцию array() или способ приведения скалярной переменной типа int, float, string или boolean к типу array, а также специализированные функции:

array([…]) – создает массив из значений, переданных конструкции в качестве параметров array_fill($start_index, $num, $value), которая возвращает массив, содержащий $num элементов, имеющих значение $value. Нумерация индексов при этом начинается со значения $start_index;

range($low, $high [, $step]) – создает массив со значениями из интервала от $iow до $high и шагом $step;

explode($delimiter, $str [, $limit) – возвращает массив из строк, каждая из которых соответствует фрагменту исходной строки $str, находящемуся между разделителем, определяемым аргументом $delimiter. Необязательный параметр $limit определяет максимальное количество элементов в массиве, при этом последний элемент будет содержать остаток строки $str.

В качестве элементов массива могут выступать другие массивы, в этом случае говорят о многомерных массивах. Массивы можно создавать, обращаясь к элементам или используя вложенные конструкции array(). Для вывода массива используется функция print_r().

Работу с ассоциативными массивами удобно выполнять с использованием специализированного оператора цикла foreach.

При манипуляции с массивами и их элементами часто возникает необходимость определения количества элементов в массиве. Для решения этой задачи используются следующие функции:

count($array [,$mode]) – возвращает количество элементов массива $array. Если $mode принимает значение count_recursive, функция рекурсивно обходит многомерный массив, в противном случае подсчитывается количество элементов только на текущем уровне;

sizeof() – синоним для функции count();

array_count_values($input) – подсчитывает количество уникальных значений среди элементов массива и возвращает ассоциативный массив, ключами которого являются значения массива, а значениями – количество их вхожде-

ний в массив $input.

Порядок выполнения работы**.**

Написать и отладить скрипт, выполняющий действия, указанные в таблице 15.1, согласно варианту.

Таблица 15.1 – Варианты заданий

| Номер вари анта | Задание |
|---|---|
| 1 | 2 |
| 1 | В массиве из n строк проверить, начинается ли каждая строка символом "*", а строки без "*" перенести в другой массив |
| 2 | В массиве из n строк проверить, содержит ли k-я строка символ @. Если не содержит, то вставить этот символ в конец строки |
| 3 | В массиве строк удалить все html-теги, заключенные в скобки < > |
| 4 | В массив случайным образом поместить строки, содержащие «цитата дня». Для выбора строки из массива случайным образом можно использовать функции Shuffle(array arr) или arrayrand(array arr, int num) |
| 5 | Создать многомерный массив: Факультет, Курс, Группа, Студенты. Вывести список студентов в алфавитном порядке |
| 6 | Создать многомерный массив: Факультет, Кафедра, Преподаватель, Ученое_звание. Вывести список преподавателей в алфавитном порядке |
| 7 | Создать двухмерный массив, в первой строке которого записаны номера и названия месяцев года в произвольном порядке. Во второй строке рассортировать месяцы года в алфавитном порядке, а в третьей – в порядке возрастания номера месяца |
| 8 | Заполнить элементы квадратной матрицы возрастающими числами начиная с единицы по спирали, начиная с элемента [1, 1] по часовой стрелке |
| 9 | Заполнить элементы квадратной матрицы возрастающими числами начиная с единицы по спирали, начиная с элемента [n, n] против часовой стрелки |
| 10 | Перемножить две числовые матрицы, размеры и значения матриц выбрать самостоятельно |
| 11 | Найти максимальную сумму диагональных элементов квадратной матрицы, размер и значения матрицы выбрать самостоятельно |
| 12 | Найти суммы элементов двух квадратных матриц и выбрать матрицу с большей суммой, размеры и значения матриц выбрать самостоятельно |

Контрольные вопросы

1. Допускается ли хранение в одном массиве значений разных типов?
2. Что такое ассоциированный массив?
3. Какие конструкции используются для создания массивов?
4. Что такое индексный массив?
5. Как создать двухмерный массив?
6. Для чего используется функция Shuffle(array arr)?

7. Прокомментируйте назначение функции arrayrand(array arr, int num).
8. Можно ли хранить в массиве другие объекты?
9. Какие специализированные функции создания массивов Вы знаете?
10. Может ли быть элементом массива другой массив?

## Лабораторная работа № 16. Изучение условных операторов PHP

К условным операторам языка PHP относятся операторы if с расши- рениями else, elseif, а также конструкция switch, позволяющая проверить условие и выполнить в зависимости от его истинности определенные действия.

Структура оператора if:

```
if (выражение) блок_выполнения
```

Здесь выражение – это любое правильное PHP-выражение, которое в процессе обработки скрипта преобразуется к логическому типу. Если в резуль- тате преобразования значение выражения истинно, то выполняется блок_выполнения, в противном случае блок_выполнения игнорируется. Если блок_выполнения содержит несколько команд, то он заключается в фигурные скобки.

Структура оператора switch:

```
switch (выражение или
переменная){ case значение1:
  блок_действий1
break;
case значение2:
  блок_действий2
  break;
  ...
  default:
    блок_действий_по_умолчанию
  }
```

Для конструкции switch, как и для if, возможен альтернативный синтаксис, где открывающая switch фигурная скобка заменяется двоеточием, а закры- вающая – endif и endswitch соответственно.

Порядок выполнения работы.

Написать и отладить скрипт, выполняющий действия, указанные в таблице 16.1, согласно варианту.

Таблица 16.1 – Варианты заданий

| Номер варианта | Задание |
|---|---|
| 1 | В массиве $a хранятся целые числа и слова. Определить количество и сумму положительных, отрицательных чисел, а также количество слов в массиве |
| 2 | В массиве $x хранятся целые и вещественные числа. Определить произведение всех целых и сумму вещественных чисел, больших 25,5 |
| 3 | Переменная $s содержит строку, состоящую из положительных и отрицательных чисел и слов, разделенных запятыми. Сформировать массив чисел и определить среднее их абсолютных значений, а также строку слов |
| 4 | Переменная $min содержит строку разделенных пробелом чисел из интервала от 0 до 59. Определить, сколько чисел попадает в какую четверть часа, а также суммы чисел в каждой четверти |
| 5 | Переменная $lang может принимать два значения: "ru" и "en". Если она имеет значение "ru", то в переменную $arr записать массив дней недели на русском языке, а если "en" – то на английском. Задачу решить с использованием if, switch-case и многомерного массива без if и switch |
| 6 | В переменной $a содержатся целые числа, разделенные пробелами. Если выделяемые значения равны или меньше пяти, то в переменную $b записать их сумму, а если больше или равны девяти, то разность |
| 7 | Если i-я переменная массива $a больше двух и меньше девяти или переменная $b больше или равна шести и меньше девяти, то вывести "Верно", в противном случае "Неверно" |
| 8 | В массиве $day содержатся числа из интервала от единицы до 31. Определить, в какую декаду месяца попадают эти числа |
| 9 | Переменная $month содержит числа из интервала от единицы до 12. Определить, в какую пору года попадает этот месяц: зима, весна, лето или осень |
| 10 | В переменной $year хранится номер год. Определить, является ли он високосным. Год является високосным, если он делится на 4, но при этом не делится на 100 либо делится на 400 |
| 11 | Дана строка с цифрами, например, "12345". Проверить, является ли первым символом этой строки цифра 1, 2 или 3. Если является, то вывести "да", в противном случае вывести "нет" |
| 12 | Дана строка из шести цифр. Проверить, равняется ли сумма первых трех цифр сумме вторых трех цифр. Если это так, вывести "да", в противном случае "нет" |

Контрольные вопросы

1. Какие операторы в языке PHP относятся к условным?
2. Какие расширения используются в операторе if?
3. Для чего используется оператор switch?
4. Прокомментируйте структуру оператора switch.
5. Для чего используется расширение else в операторе if?
6. Для чего используется расширение elseif в операторе if?
7. Что понимается под альтернативным синтаксисом оператора if?
8. Для чего используется break в операторе switch?

9. Как оформить несколько команд в блоке выполнения оператора if?
10. Что понимается под альтернативным синтаксисом оператора switch?

## Лабораторная работа №17. Изучение технологии работы с функциями PHP

В PHP существуют две основные формы функций: встроенные и пользовательские.

Полный список встроенных PHP функций можно просмотреть в окне редактора кода, нажав кнопку «*Поиск»* в правой колонке при пустой строке поиска «PHP-поиск». Для просмотра подробного описания с примером конкретной PHP-функции необходимо указать ее имя в строке PHP-поиск.

Пользовательская функция создается с помощью оператора function, после которого через пробел указывается имя функции и круглые скобки, которые могут быть пустыми либо содержать параметры, переменные PHP, принимаемые функцией.

После объявления функции в фигурных скобках записывается код, выполняемый функцией. Общий синтаксис пользовательской функции имеет вид:

```
function имя_функции (параметры)
{
  //тело функции
}
```

Вызывается функция по ее имени, после которого обязательны круглые скобки, даже если они пустые.

Для возврата значения, являющегося результатом работы функции, используется оператор return, который прекращает выполнение текущей функции и возвращает ее значение. При этом оператор return может быть расположен в любом месте функции.

В PHP допускается использование **динамических функций**. Это означает, что если некоторой переменной присвоено имя функции, то с этой переменной можно обращаться точно так же, как с самой функцией.

В функциях допускается использование глобальных переменных, созданных с помощью инструкции **global** вне функции.

Чтобы переменная сохраняла свое значение между вызовами функции, нужно объявить ее статической с помощью инструкции **static.**

**Порядок выполнения работы.**

Написать и отладить скрипт, выполняющий действия, указанные в таблице 17.1, согласно варианту.

Таблица 17.1 – Варианты заданий

| Номер варианта | Задания |
|---|---|
| 1 | 2 |
| 1 | 1 Создать пользовательскую функцию, которая принимает два аргумента и возвращает их произведение. Вызвать функцию, передав ей в качестве аргументов два числа, и результат вывести на экран.<br>2 Создать пользовательскую функцию, формирующую массив делителей (чисел, на которое оно делится без остатка) заданного числа |
| 2 | Создать три переменные, присвоить им числовые значения и вывести на экран их произведение. Создать пользовательскую функцию, принимающую два аргумента по ссылке и один по значению, которая должна присваивать переменным другие числовые значения. Вызвать функцию и вывести на экран произведение всех переменных |
| 3 | Создать две переменные, присвоить им числовые значения и создать пользовательскую функцию, принимающую два аргумента со значениями по умолчанию и выводящую произведение своих аргументов. Вызвать функцию, передав ей в качестве аргументов сначала значения двух переменных, затем значение одной из переменных и, наконец, вообще без аргументов |
| 4 | Создать пользовательскую функцию, принимающую аргументы в массив переменной длины и выводящую их на экран. Для доступа к элементам массива использовать цикл foreach. Вызвать функцию, передав ей в качестве значения две строки и число |
| 5 | Создать пользовательскую функцию, которая вычисляет корень из заданного четырехзначного числа и округляет его в большую и меньшую стороны. В мас- сив $arr первым элементом записать заданное число, вторым – корень из этого числа, третьим – округление в меньшую сторону, четвертым – округление в большую сторону |
| 6 | Заполнить массив 30 случайными числами от 1 до 10. Найти квадратный корень из каждого числа. Округлить результаты в большую и меньшую сторону и записать результаты округления в ассоциативный массив с ключами 'floor' и 'ceil' |
| 7 | Дано целое двухзначное число. Создать функцию, определяющую делители этого числа, т. е. числа, на которое оно делится без остатка. Сформировать массив делителей заданного числа |
| 8 | Задать режим строгой типизации, использовать инструкцию declare(strict_types=1), после этого создать пользовательскую функцию, которая будет принимать два целочисленных аргумента и выводить на экран их сумму. Вызвать функцию, передав ей в качестве аргументов сначала два целых числа, а затем одно из них в виде строки |
| 9 | Задать режим строгой типизации, используя инструкцию declare(strict_types=1), затем создать пользовательскую функцию my_func(), которая будет принимать два целочисленных аргумента и возвращать их произведение. Создать переменную $count_apples и присвоить ей строку с именем функции. Обратиться к функции через переменную и вывести на экран общую массу яблок, зная, что имеет- ся 25 корзин по семь килограмм яблок в каждой |
| 10 | Создать переменную и присвоить ей целое число. Создать еще одну переменную и присвоить ей анонимную функцию, наследующую эту переменную и выводящую на экран ее инкрементированное значение. Выполнить вызов функции, затем изменить значение внешней переменной и снова вызвать функцию. Изменить скрипт, задав наследование переменной по ссылке |
| 11 | 1 Создать функцию, вычисляющую квадратный корень из заданного числа. Результат округлить в большую и меньшую сторону, результаты округления записать |

| 1 | 2 |
|---|---|
| | в ассоциативный массив с ключами "floor" и "ceil".<br>2 Создать функцию, определяющую, сколько первых элементов массива [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] нужно сложить, чтобы сумма получилась больше 10 |
| 12 | Создать пользовательскую функцию, вычисляющую корень квадратный из заданного четырехзначного числа. Результат округлить в большую и меньшую стороны. В массив $arr записать первым элементом корень из числа, вторым – округление в меньшую сторону, третьим – округление в большую сторону. Для решения задачи можно использовать функции: sqrt – корень из числа, floor – округление в меньшую сторону и ceil– округление в большую сторону |

Контрольные вопросы

1. Какие две основные формы PHP-функций Вы знаете?
2. Как посмотреть полный список встроенных PHP-функций?
3. Как просмотреть подробное описание конкретной PHP-функции?
4. Как создаются пользовательские PHP-функции?
5. Что записывается в фигурных скобках в пользовательской
6. PHP-функции?
7. Прокомментируйте общий синтаксис PHP-функции.
8. Как вызвать PHP-функцию?
9. Как вернуть результат пользовательской PHP-функции?
10. Что такое динамические PHP-функции?
11. Как использовать глобальные переменные в PHP-функциях?

## Лабораторная работа №18. Изучение технологии работы с файлами PHP

В PHP-документ с помощью инструкции include() можно включать файлы кода. Аргументом этой инструкции является путь к файлу. Чтобы содержимое этого файла обрабатывалось как PHP-программа, его необходимо обрамлять открывающим и закрывающим тегами PHP.

PHP содержит множество функций управления файлами, наиболее употребительными из которых являются:

touch() – создает пустой файл с заданным именем, например: touch("ex1.txt"). Если такой файл уже существует, то функция изменит дату модификации;

copy() – копирует файл. Для копирования файлов в PHP используется функция copy($source, $result), в которой используются два параметра – источник $source и имя файла-копии $result. При этом следует указывать полные адреса к файлам;

unlink() – удаляет заданный файл.

Например:

```
<?php
if (unlink('filename.txt'))
  { echo "Файл удален"; }
else
    { echo "Ошибка при удалении файла"; }
?>
```

fopen() – открывает локальный или удаленный файл и возвращает указатель на него. Указатель используется во всех операциях с содержимым файла. Аргументами функции fopen() являются имя файла и режим открытия;

fclose() – закрывает файл. Аргумент: указатель файла, полученный ранее от функции fopen();

feof() – проверяет конец файла. Аргумент: указатель файла;

fgetc() – читает очередной символ из файла. Аргумент: указатель файла; fgets() – читает очередную строку файла. Аргументы: указатель файла и

длина считываемой строки. Операция прекращается после считывания заданного количества символов или при обнаружении конца строки или файла;

fread() – общая функция чтения из файла. Аргументы: указатель файла и количество считываемых символов;

fseek() – отступ от начала файла. Аргументы: указатель файла и смещение; fputs() – записывает строку в файл. Аргументы: указатель файла и строка; fwrite() – полный аналог функции fputs( );

flock() – блокирует файл, т. е. не позволяет другим пользователям читать этот файл или писать в него, пока тот, кто наложил блокировку, не закончит работу с данным файлом. Аргументы: указатель файла и номер режима блокировки.

Порядок выполнения работы.

Написать и отладить скрипт, выполняющий действия, указанные в таблице 18.1 согласно варианту.

Таблица 18.1 – Варианты заданий

| Номер варианта | Задание |
|---|---|
| 1 | 2 |
| 1 | Дан массив со строками. Создать папку 'test', а в ней – папки, названиями которых служат элементы этого массива |
| 2 | Найти все файлы из папки 'test' и вставить в начало каждого файла полный путь к нему. Текст файла должен остаться в нем и начинаться с новой строки после пути |
| 3 | Вывести на экран имена всех папок из папки 'test' и их подпапок. Уровень вложенности папок может быть любым |
| 4 | Вывести на экран содержимое всех файлов из папки 'test' и ее подпапок. Уровень вложенности папок может быть любым |
| 5 | Найти все файлы из папки 'test' и ее подпапок любого уровня вложенности и вставить в начало каждого файла полный путь к нему (текст файла должен остаться в нем и начинаться с новой строки после пути) |
| 6 | Удалить из папки 'test' все файлы размером более 1 Мбайт |

| 1 | 2 |
|---|---|
| 7 | Имеется папка с файлами, определить и вывести на экран размер этой папки |
| 8 | Имеется папка с подпапками, определить размеры всех подпапок папки и вывести их на экран |
| 9 | Вывести на экран название всех файлов с расширением txt из папки 'test' |
| 10 | В папке 'test' есть файлы и подпапки. Вывести на экран содержимое всех файлов, которые лежат непосредственно в папке 'test' |
| 11 | Вывести на экран название всех файлов, но не подпапок из папки 'test' |
| 12 | Вывести на экран название всех файлов и подпапок из папки 'test' |

Контрольные вопросы

1. С помощью какой инструкции можно включать файлы кода в PHP-документ?
2. С помощью какой PHP-функции можно создать пустой файл?
3. Какая PHP-функция используется для открытия файла?
4. Какие параметры используются в PHP-функции copy()?
5. Какая PHP-функция используется для чтения очередной строки файла?
6. Какая PHP-функция используется для записи строки в файл?
7. Для чего используется PHP-функция fputs()?
8. Как в качестве элемента массива создать другой массив?
9. Как вывести на экран имена всех папок из папки?
10. Как вывести на экран название всех файлов из папки?

# III. РАЗДЕЛ КОНТРОЛЯ ЗНАНИЙ

## ПРИМЕРНЫЙ ПЕРЕЧЕНЬ ВОПРОСОВ К ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

1. Сеть Интернет, история создания, принципы функционирования. Архитектура «клиент-сервер». Основные виды веб-приложений. Веб-сайт. Веб-сервис. Виды веб-сервисов.
2. Понятие веб-сайта. Понятие веб-разработки (front-end / back-end). Виды сайтов. Понятие веб-дизайна. UI/UX дизайн. Графика в веб-дизайне. Основные принципы визуального дизайна (по Торре).
3. Понятие юзабилити (usability) сайта. Принципы юзабилити. Назначение прототипа сайта. Правила использования модульных сеток в формировании страницы сайта. Техника построения модульных сеток. Преимущества адаптивных сайтов. Внесение разметки и создание основы сайта.
4. Статические веб-сайты. Язык разметки гипертекстовых документов (HTML). Каскадная таблица стилей (CSS). Динамические веб-сайты. Язык программирования JavaScript.
5. Проектирование веб-сервисов. Протоколы SOAP, REST. Многостраничные (MPA) и одностраничные (SPA) веб-приложения. Обзор серверных языков программирования: PHP8, Python, Node.js и др.
6. Проектирование веб-сервисов. Протоколы SOAP, REST. Многостраничные (MPA) и одностраничные (SPA) веб-приложения. Технологии, используемые на стороне клиента: HTML, CSS, JavaScript (TypeScript). Обзор UI-фреймворков: Bootstrap, Foundation и т.д.
7. Статические и динамические компоненты представления веб приложений. Обзор JavaScript-фреймворков: React, Vue, Angular. Преимущества и недостатки.
8. Статические и динамические компоненты представления веб приложений. Обзор информационных систем управления контентом (CMS).
9. Назначение и приминение HTML. Элементы и атрибуты HTML. Глобальные и пользовательские атрибуты. Элементы группировки. Заголовки.
10. Назначение и приминение HTML. Форматирование текста. Изображения. Списки. Таблицы. Ссылки.
11. Назначение и приминение HTML. Работа с формами. Элементы форм (input, textarea, select и т.д.). Валидация форм.
12. Назначение и приминение HTML. Семантическая структура страницы (article, section, nav и т.д.)
13. Назначение и приминение CSS. Селекторы. Селекторы потомков. Селекторы дочерних элементов. Селекторы элементов одного уровня.
14. Назначение и приминение CSS. Псевдоклассы. Псевдоклассы дочерних элементов. Псевдоэлементы. Селекторы атрибутов.

15. Назначение и приминение CSS. Виды селекторов. Наследование стилей. Каскадность стилей.

16. Назначение и приминение CSS. Свойства. Цвет. Стилизация шрифтов. Форматирование текста. Стиизация адзацев.

17. Назначение и приминение CSS. Свойства. Стилизация списков. Стилизация таблиц. Внешние отступы. Внутренние отступы. Границы.

18. Назначение и приминение CSS. Свойства. Фон элемента. Тень. Обтекание элементов. Виды градиентов.

19. Назначение и приминение CSS. Трансформации, переходы и анимации.

20. Назначение и приминение CSS. Адаптивный дизайн. Метатег viewport. Media Query в CSS.

21. Назначение и приминение CSS. Flexbox. Понятия flex-контейнера и flex-элементов. Направление flex-direction. Свойство flex-wrap. Выравнивание элементов (justify-content).

22. Назначение и приминение CSS. Flexbox. Выравнивание элементов (align-items и align-self). Выравнивание строк и столбцов (align-content).

23. Назначение и приминение CSS. Flexbox. Управление элементами (flex-basis, flex-shink и flex-grow).

24. Назначение и приминение CSS. Grid Layout. Понятие grid-контейнера. Строки и столбцы.

25. Назначение и приминение CSS. Grid Layout. Функция repeat и свойство grid. Размеры строк и столбцов. Отступы между столбцами и строками.

26. Назначение и приминение CSS. Grid Layout. Области грида (grid-area).

27. Основы JavaScript. Переменные. Типы данных. Преобразование типов. Базовые операторы, математика. Операторы сравнения.

28. Основы JavaScript. Условное ветвление. Логические операторы. Оператор объединения. Циклы while и for. Конструкция "switch".

29. Основы JavaScript. Исключения. Оператор обработки исключений в JavaScript – try...catch.

30. Основы JavaScript. Встроенные функции JavaScript. Пользовательские функции JavaScript. JavaScript функции с параметрами (аргументами) и возврат значений. Способы создания пользовательских функций. Использование выражений с функциями.

31. Основы JavaScript. Область видимости переменных. JavaScript глобальные и локальные переменные в функции.

32. Основы JavaScript. Пользовательские функции JavaScript. Замыкание.

33. Основы JavaScript. Строки. Методы работы с строками.

34. Основы JavaScript. Массивы. Методы работы с массивами.

35. Основы JavaScript. Структуры данных. Map, Set, WeakMap, WeakSet и тд.

36. Основы JavaScript. Регулярные выражения. Класс RegExp. Методы работы с регулярными выражениями.

37. Основы JavaScript. Класс: базовый синтаксис. Наследование классов.

38. Основы JavaScript. Статические свойства и методы. Приватные и защищённые методы и свойства.

39. Основы JavaScript. Расширение встроенных классов. Проверка класса: "instanceof". Примеси.

40. Основы JavaScript. Promise (промисы). Цепочка промисов. Обработка ошибок.

41. Основы JavaScript. Promise (промисы). Promise API. Промисификация.

42. Основы JavaScript. Promise (промисы). Promise API. Async/Await.

43. Основы PHP8. Основы синтаксиса. Переменные. Типы данных. Операции в PHP.

44. Основы PHP8. Условное ветвление. Логические операторы. Оператор объединения. Циклы while и for. Конструкции "switch" и "match".

45. Основы PHP8. Массивы. Ассоциативные массивы. Многомерные массивы. Операции с массивами.

46. Основы PHP8. Функции. Параметры функции. Возвращение значений и оператор return. Анонимные функции. Замыкания. Стрелочные функции.

47. Основы PHP8. Область видимости переменной. Константы. Проверка существования переменной. Проверка и установка типа переменной. Преобразование типов.

48. Основы PHP8. Отправка данных на сервер. Получение данных из строки запроса. GET-запросы.

49. Основы PHP8. Отправка данных на сервер. Отправка формы. POST-запросы.

50. Основы PHP8. ООП. Объекты и классы. Конструкторы и деструкторы. Анонимные классы.

51. Основы PHP8. ООП. Наследование. Модификация доступа. Статические методы и свойства.

52. Основы PHP8. ООП. Интерфейсы. Абстрактные классы и методы. Копирование объектов классов.

53. Основы PHP8. Обработка исключений. Конструкция try…catch…finally.

54. Основы PHP8. Обработка исключений. Генерация исключений.

55. Основы PHP8. Работа с файловой системой. Чтение и запись файлов.

56. Основы PHP8. Работа с файловой системой. Управление файлами и каталогами.

57. Основы PHP8. Работа с файловой системой. Блокировка файла. Функция flock.

58. Основы PHP8. Базовые возможности PHP. Подключение внешних файлов. Пространства имен.

59. Основы PHP8. Базовые возможности PHP. Типизация данных. Работа со строками.

60. Основы PHP8. Базовые возможности PHP. Типизация с cookie. Сессии.

# IV. ВСПОМОГАТЕЛЬНЫЙ РАЗДЕЛ

## ПЕРЕЧЕНЬ РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Грофф Дж., Вайнберг П. Энциклопедия SQL / Дж. Грофф, П. Вайнберг. – СПб. : Питер, 2004. – 896 с.+CD .

2. Грофф, Д. Р. SQL : полное руководство / Джеймс Грофф, Пол Вайнберг, Эндрю Оппель. – 3-е изд.. – М. [и др.] : Вильямс, 2018. – 957 с.

3. Мархвида, И. Р. Создание Web-страниц: HTML, CSS, JavaSript. / И. Р. Мархвида.– Минск : Новое знание, 2002. – 352 с.

4. Мейер, Э. CSS : Полный справочник / Э. Мейер, Э. Уэйл; пер. с англ. – 4-е изд. – СПб. : Символ-Плюс, 2019. – 576 с.

5. Резиг, Д. JavaScript для профессионалов / Д. Резиг, Р. Фергюсон, Д. Пакстон. – СПб. : Apress, 2020. – 242 с.

6. Томсон Л., Веллинг Л. Разработка Web-приложений на PHP и MySQ : Учебник. – Киев : ДиаСофт, 2002. – 672 с.

7. Флэнаган, Д. JavaScript. Карманный справочник. / Д. Флэнаган. – СПб. : O'Reilly, 2020. – 319 с.

8. Флэнаган, Д. JavaScript. Полное руководство / Д. Флэнаган. – 6-е изд. – СПб. : O'Reilly, 2021. – 732 с.

9. Флэнаган, Д. JavaScript. Полное руководство : справочник по самому популярному языку программирования / Дэвид Флэнаган. – 7-е изд.. – СПб. : М. : Диалектика, 2021. – 1080 с.

10. Хавербеке, М. Выразительный JavaScript. / М. Хавербеке. – СПб. : Питер, 2019. – 482 с.

11. Хавербеке, М. Выразительный JavaScript: современное веб-программирование / Марейн Хавербеке ; пер. с англ. Е. Сандицкой. – 3-е изд. – СПб. [и др.] : Питер; Прогресс книга, 2020. – 480 с.