Оглавление

[***1.*** ***Обзор тегов HTML: метаданные, секционные (структурные), группирующие, табличные и текстовые элементы, элементы с встроенным содержимым, элементы форм. Сущности HTML.*** 1](#_Toc170494274)

[***2.*** ***Виды и описание тегов метаданных.*** 1](#_Toc170494275)

[***3.*** ***Виды и описание секционных (структурных) тегов.*** 1](#_Toc170494276)

[***4.*** ***Виды и описание группирующих тегов.*** 1](#_Toc170494277)

[***5.*** ***Виды и описание табличных тегов.*** 2](#_Toc170494278)

[***6.*** ***Виды и описание текстовых тегов.*** 2](#_Toc170494279)

[***7.*** ***Виды и описание тегов с встроенным содержимым.*** 2](#_Toc170494280)

[***8.*** ***Виды и описание тегов форм.*** 3](#_Toc170494281)

[***9.*** ***Селекторы CSS: виды, составные селекторы, атрибуты.*** 3](#_Toc170494282)

[***10.*** ***Способы подключения шрифтов к веб-проекту: локальное и сетевое подключение.*** 3](#_Toc170494283)

[***11.*** ***Медиа-запросы: синтаксис, применение, пример использования.*** 3](#_Toc170494284)

[***12.*** ***Создание карты изображения: назначение, этапы создания, теги.*** 4](#_Toc170494285)

[***13.*** ***Блочные и строчные элементы: отличия, способы преобразования.*** 4](#_Toc170494286)

[***14.*** ***Наследование и каскадирование: смысл терминов, примеры CSS*** 4](#_Toc170494287)

[***15.*** ***Позиционирование элементов на странице с помощью CSS.*** 5](#_Toc170494288)

[***16.*** ***Типы данных JavaScript.*** 5](#_Toc170494289)

[***17.*** ***Условия JavaScript. Условия с альтернативой. Каскад условий. Тернарный оператор для записи условий.*** 5](#_Toc170494290)

[***18.*** ***Операторы сравнения JavaScript. Приведение типов аргументов. Логические операции.*** 5](#_Toc170494291)

[***19.*** ***Цикл While и For. Синтаксис, условия не зацикливания операторов цикла.*** 6](#_Toc170494292)

[***20.*** ***Массивы JavaScript. Методы работы с массивами. Доступ к значениям массива.*** 6](#_Toc170494293)

[***21.*** *** Объекты JavaScript. Синтаксис, Доступ к значениям объекта: точечная и скобочная нотации.*** 7](#_Toc170494294)

[***22.*** ***Функции JavaScript. Синтаксис. Передача аргументов в функцию и возврат значения из функции. Способы обращения к функции из*** 8](#_Toc170494295)

[***23.*** ***Модель DOM. DOM-методы управления элементами страницы веб-приложения.*** 8](#_Toc170494296)

[***24.*** ***Что такое верстка сайтов. Виды и технологии.*** 9](#_Toc170494297)

[***25.*** ***Технологии Grid и Flex. Сходства и различия.*** 9](#_Toc170494298)

[***26.*** *** Техники графики в браузере. Элемент <canvas>.*** 9](#_Toc170494299)

# ***Обзор тегов HTML: метаданные, секционные (структурные), группирующие, табличные и текстовые элементы, элементы с встроенным содержимым, элементы форм. Сущности HTML.***

HTML-теги можно классифицировать по различным критериям:

Метаданные: <meta>, <title>, <base>, <link>, <style>, <script>

Секционные (структурные): <body>, <header>, <nav>, <main>, <article>, <section>, <aside>, <footer>

Группирующие: <div>, <span>, <p>, <ul>, <ol>, <li>

Табличные: <table>, <tr>, <td>, <th>

Текстовые: <a>, <b>, <i>, <em>, <strong>, <mark>, <abbr>, <cite>, <q>, <dfn>, <code>, <data>, <time>, <address>, <blockquote>, <del>, <ins>, <caption>, <col>, <colgroup>, <thead>, <tbody>, <tfoot>, <h1>-<h6>

Элементы с встроенным содержимым: <img>, <iframe>, <embed>, <object>, <video>, <audio>, <source>, <track>, <map>, <area>

Элементы форм: <form>, <input>, <textarea>, <button>, <select>, <datalist>, <optgroup>, <option>, <label>, <fieldset>, <legend>

Сущности HTML используются для представления специальных символов, которые не могут быть введены напрямую через клавиатуру или для замены символов в URL.

# ***Виды и описание тегов метаданных.***

Используются для предоставления информации о веб-странице, которая не отображается напрямую на странице, но может быть использована браузерами, поисковыми системами и другими инструментами для обработки и интерпретации содержимого страницы:

<meta charset="UTF-8">: Определяет кодировку документа.

<meta name="description" content="...">: Описание страницы для поисковых систем.

<meta http-equiv="X-UA-Compatible" content="IE=edge">: Указывает браузеру использовать последнюю доступную версию IE.

<meta name="viewport" content="width=device-width, initial-scale=1">: Устанавливает масштаб и размер области просмотра для мобильных устройств.

<title>: Задает заголовок окна браузера или вкладки.

<base href="...">: Указывает базовый URL для всех относительных URL в документе.

# ***Виды и описание секционных (структурных) тегов.***

<body>: Корневой элемент HTML-документа.

<header>: Представляет начало контента, например, заголовок сайта.

<nav>: Определяет навигационное меню.

<main>: Основная часть контента веб-страницы.

<article>: Отдельный, самостоятельный контент, например, статья или блоговая запись.

<section>: Раздел внутри <article> или <main>.

<aside>: Второстепенный контент, обычно относящийся к основному контенту.

# ***Виды и описание группирующих тегов.***

<div>: Неопределенная блоковая группа.

<span>: Неопределенная инлайновая группа.

<p>: Параграф текста.

<ul>: Несуществующий список.

<ol>: Отсортированный список.

<li>: Элемент списка.

<fieldset> определяет группу

<legend> - заголовок этой группы

# ***Виды и описание табличных тегов.***

<table>: Создает таблицу.

<tr>: Столбец строки в таблице.

<td>: Ячейка таблицы.

<th>: Заголовок ячейки таблицы.

(<thead>) - заголовка

(<tbody>) - тела

(<tfoot>) – подвала

<colgroup> - создает группу столбцов

<col> - определяет отдельный столбец или набор столбцов.

# ***Виды и описание текстовых тегов.***

<a>: Ссылки на другие ресурсы.

<b>: Жирный текст.

<i>: Итальянский текст (курсив).

<em>: Эмоциональное выделение текста.

<strong>: Важность текста.

<mark>: Выделенный текст.

<abbr>: Аббревиатура или акроним.

<cite>: Цитата.

<q>: Краткая цитата.

<dfn>: Определение термина.

<code>: Код.

<data>: Связанная данные.

<time>: Дата/время.

<address>: Адрес автора контента.

<blockquote>: Цитата.

<del>: Удаленный текст.

<ins>: Вставленный текст.

<caption>: Заголовок таблицы.

<col>: Колонка таблицы.

<colgroup>: Группа колонок таблицы.

<thead>, <tbody>, <tfoot>: Части таблицы.

# ***Виды и описание тегов с встроенным содержимым.***

<img>: Изображение.

<iframe>: Встраиваемый фрейм.

<embed>: Встраиваемый объект.

<object>: Встраиваемый объект.

<video>: Видео.

<audio>: Аудио.

<source>: Исходник медиафайлов.

<track>: Метаданные для видео.

<map>: Карта кликов.

<area>: Регион карты кликов.

# ***Виды и описание тегов форм.***

<form>: Форма отправки данных.

<input>: Поле ввода формы.

<textarea>: Текстовое поле формы.

<button>: Кнопка формы.

<select>: Выпадающий список.

<datalist>: Список предопределенных значений для поля ввода.

<optgroup>: Группа опций в выпадающем списке.

<option>: Опция в выпадающем списке.

<label>: Метка для элемента формы.

<fieldset>: Группа элементов формы.

<legend>: Легенда для группы элементов формы.

# ***Селекторы CSS: виды, составные селекторы, атрибуты.***

* ***Простые селекторы***: идентификаторы (#id), классы (.class), теги (tag), псевдоэлементы (::before, ::after), псевдоклассы (:hover, :active).
* ***Составные селекторы***: комбинации простых селекторов без пробелов между ними.
* ***Атрибутные селекторы***: [attr], [attr=value], [attr~=value], [attr|=value], [attr^=value], [attr$=value].

# ***Способы подключения шрифтов к веб-проекту: локальное и сетевое подключение.***

* + **Локальное подключение**: Использование локальных файлов шрифтов через @font-face в CSS.

@font-face {

font-family: 'CustomFont';

src: url('fonts/customfont.woff2') format('woff2');}

* + **Сетевое подключение**: Использование внешних ресурсов шрифтов через CDN или другие серверы.

<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto">

# ***Медиа-запросы: синтаксис, применение, пример использования.***

Используются для адаптивного дизайна, применяя стили в зависимости от характеристик устройства пользователя.

@media screen and (max-width: 600px) {

body {

background-color: lightblue;

}

}

# ***Создание карты изображения: назначение, этапы создания, теги.***

*Карта изображения представляет собой графическое изображение, на котором определены области, каждая из которых связана с определенной ссылкой или действием.*

**Назначение**: Определяет область клика на изображении, которая может активировать ссылку или выполнить действие.

**Этапы создания**:

* Создание изображения с помощью <img src="image.jpg" alt="Image description">.
* Определение области клика с помощью <map name="mapname">.
* Определение координат области клика с помощью <area shape="rect" coords="x,y,width,height" href="url" alt="Area description">.

# ***Блочные и строчные элементы: отличия, способы преобразования.***

* **Блочные элементы** занимают всю доступную ширину родительского элемента и начинаются с новой строки.
* **Строчные элементы** не начинаются с новой строки и занимают только необходимое пространство для своего содержимого.

**Преобразование**: Можно изменять поведение элементов с помощью CSS свойства display.

* **Блочные элементы:**
  + Поведение по умолчанию: занимают всю доступную ширину по горизонтали.
  + Отображение на странице: начинаются с новой строки и занимают всю доступную ширину, выталкивая другие элементы вниз.
  + Примеры: <div>, <p>, <h1>-<h6>, <ul>, <ol>, <li>.
  + Свойства: можно задавать ширину, высоту, внутренние и внешние отступы.
* **Строчные элементы:**
  + Поведение по умолчанию: занимают только столько ширины, сколько необходимо для содержимого.
  + Отображение на странице: не начинаются с новой строки и могут быть выровнены горизонтально в строке.
  + Примеры: <span>, <a>, <strong>, <em>.
  + Свойства: не могут иметь фиксированной ширины или высоты, но можно изменять их отступы и выравнивание.
* **Способы преобразования:**
  + display: можно преобразовать блочный элемент в строчный и наоборот.

# ***Наследование и каскадирование: смысл терминов, примеры CSS***

* **Наследование**: Процесс получения свойств от родительских элементов.
* **Каскадирование**: Процесс применения стилей, когда несколько правил могут применяться к одному и тому же элементу.

/\* Пример каскадирования \*/

h1 {

color: red; /\* Стиль из этого блока будет применен ко всем h1 \*/

}

.special-heading {

color: blue; /\* Но если есть этот класс, то цвет будет синим \*/

}

# ***Позиционирование элементов на странице с помощью CSS.***

* Static: По умолчанию, элементы располагаются в потоке документа.
* Relative: Элемент смещается относительно его исходного положения.
* Absolute: Элемент позиционируется абсолютно относительно ближайшего позиционированного предка.
* Fixed: Элемент позиционируется относительно окна просмотра.
* Sticky: Элемент "липким", при прокрутке остается на месте до определенного момента.

# ***Типы данных JavaScript.***

* **Primitive types:** String, Number, BigInt, Boolean, Null, Undefined, Symbol
* **Object types:** Object, Array, Function, Date, RegExp, Map, Set, WeakMap, WeakSet

# ***Условия JavaScript. Условия с альтернативой. Каскад условий. Тернарный оператор для записи условий.***

* + Условные операторы: if, else if, else
  + Каскад условий (if...else if...else):

let time = 14;

if (time < 12) {

console.log("Доброе утро!");

} else if (time < 18) {

console.log("Добрый день!");

} else {

console.log("Добрый вечер!");

}

* + Тернарный оператор: condition? exprIfTrue : exprIfFalse

# ***Операторы сравнения JavaScript. Приведение типов аргументов. Логические операции.***

* + Операторы сравнения в JavaScript:
    - * - == (равенство) - сравнивает значения, приводя их к общему типу данных.
      * - === (строгое равенство) - сравнивает значение и тип данных.
      * - != (неравенство) - сравнивает значения, приводя их к общему типу данных.
      * - !== (строгое неравенство) - сравнивает значение и тип данных.
      * - > (больше)
      * - < (меньше)
      * - >= (больше или равно)
      * - <= (меньше или равно)
  + Приведение типов аргументов:
    - * - При использовании оператора ==, JavaScript пытается привести типы сравниваемых значений к одному типу данных.
      * - Например, '5' == 5 вернёт true, так как строка '5' будет приведена к числу 5.
      * - При использовании оператора ===, приведения типов не происходит, сравниваются значение и тип данных.
  + Логические операции:
    - * - && (логическое И)
      * - || (логическое ИЛИ)
      * - ! (логическое НЕ)
      * - Результат логических операций зависит от приведения операндов к логическому типу данных (true или false).

# ***Цикл While и For. Синтаксис, условия не зацикливания операторов цикла.***

* ***Цикл while:***

let i = 0;

while (i < 5) {

console.log(i);

i++;

}

* ***Цикл for:***

Шаг: выражение, выполняемое после каждой итерации цикла.

for (let i = 0; i < 5; i++) {

console.log(i);

}

* ***Условия предотвращения зацикливания:***

**Обновление переменной управления циклом**: важно обновлять переменную управления циклом (i), чтобы гарантировать выход из цикла.

**Правильное условие завершения**: убедиться, что условие завершения цикла в конечном итоге станет ложным, чтобы избежать зацикливания.

**Избегайте бесконечных циклов**: проверить, что код не содержит ошибок или условий, которые могут привести к бесконечному циклу.

# ***Массивы JavaScript. Методы работы с массивами. Доступ к значениям массива.***

Массивы позволяют добавлять, удалять, изменять и получать доступ к их элементам

* **Создание массива:** 
  + let myArray = [1, 2, 3, 4, 5];
* **Доступ к значениям массива:**
  + // Получение значения по индексу
  + let value = myArray[2]; // Получить третий элемент (индекс 2)
  + // Изменение значения по индексу
  + myArray[0] = 6; // Заменить первый элемент на 6

**Методы работы с массивами**

* **Добавление элементов:**
  + **push():** Добавляет элемент(ы) в конец массива.
  + **unshift():** Добавляет элемент(ы) в начало массива.
* **Удаление элементов:**
  + **pop():** Удаляет последний элемент массива.
  + **shift():** Удаляет первый элемент массива.
  + **splice():** Удаляет, заменяет или добавляет элементы в любом месте массива.
* **Доступ к элементам:**
  + **indexOf():** Возвращает индекс первого найденного элемента в массиве.
  + **find():** Возвращает первый найденный элемент в массиве в соответствии с заданным условием.
  + **findIndex():** Возвращает индекс первого найденного элемента в массиве в соответствии с заданным условием.
* **Изменение массива:**
  + **concat():** Объединяет два или более массивов и возвращает новый массив.
  + **slice():** Возвращает выбранный участок массива в виде нового массива.
* **Итерация по массиву**
  + **forEach():** Выполняет указанную функцию один раз для каждого элемента массива.
  + **map():** Создает новый массив с результатами вызова указанной функции для каждого элемента.
* **Другие методы:**
  + **filter():** Создает новый массив с элементами, прошедшими проверку на соответствие условию.
  + **sort():** Сортирует элементы массива.

# ***Объекты JavaScript. Синтаксис, Доступ к значениям объекта: точечная и скобочная нотации.***

**Объекты** – основной способ хранения и организации данных.

Объекты представляют собой коллекции пар ключ-значение, где ключом является строка (или символ), а значением может быть любое другое значение, включая другой объект.

let person = {

name: "John Doe",

age: 30,

city: "New York"

};

**Доступ к свойствам объекта:**

* + - * - Точечная нотация:

console.log(person.name); // "John Doe"

* + - * Скобочная нотация:

console.log(person["age"]); // 30

# ***Функции JavaScript. Синтаксис. Передача аргументов в функцию и возврат значения из функции. Способы обращения к функции из***

Функция в JavaScript объявляется с использованием ключевого слова function, за которым следует имя функции, список параметров в круглых скобках и тело функции в фигурных скобках.

function functionName(parameters) {

// Тело функции

}

**Аргументы передаются в функцию через список параметров при её вызове.**

function greet(name) {

console.log("Hello, " + name);

}

greet("Alice"); // Выводит: Hello, Alice

**Значение может быть возвращено из функции с помощью оператора return.**

function add(a, b) {

return a + b;

}

let result = add(5, 3); // result будет равно 8

***Способы обращения к функции***

Вызов функции: functionName(arguments)

* + - * Рекурсивный вызов: functionName()
      * Привязка контекста: functionName.call(context, arguments) или functionName.apply(context, arguments)

# ***Модель DOM. DOM-методы управления элементами страницы веб-приложения.***

*DOM (Document Object Model) представляет структуру HTML-документа как дерево объектов. Это позволяет программно изменять содержимое, стиль и структуру веб-страницы.*

***DOM-методы управления элементами***

* createElement(tagName): Создает новый элемент с заданным тегом.
* appendChild(childNode): Добавляет дочерний узел в конец списка дочерних узлов родительского элемента.
* removeChild(childNode): Удаляет указанный дочерний узел из родительского элемента.
* replaceChild(newNode, oldNode): Заменяет старый дочерний узел новым в родительском элементе.
* insertBefore(newNode, referenceNode): Вставляет newNode перед существующим childNode.

# ***Что такое верстка сайтов. Виды и технологии.***

Верстка сайтов — это процесс создания макета веб-страницы с использованием HTML и CSS. Она включает в себя размещение элементов на странице, их стилизацию и управление их расположением.

# ***Технологии Grid и Flex. Сходства и различия.***

***Сходства***

* Обе технологии предоставляют гибкие способы управления расположением элементов.
* Используются для создания адаптивных макетов.

***Различия***

* Grid предназначена для двухмерной компоновки элементов (строки и столбцы). Она идеально подходит для сложных макетов, таких как сетки с несколькими рядами и колонками.
* Flex предназначена для одномерной компоновки элементов вдоль одной оси (строка или столбец). Она лучше всего подходит для упорядочивания элементов внутри контейнера, например, для создания горизонтального или вертикального списка.

# ***Техники графики в браузере. Элемент <canvas>.***

Одним из способов реализации графического интерфейса в веб-браузерах является использование элемента <canvas>. Этот элемент позволяет рисовать графику с помощью JavaScript и API Canvas.

Элемент <canvas>

Элемент <canvas> используется для создания графики на веб-страницах. Он предоставляет контекст рендеринга, который может быть использован для рисования графики с помощью JavaScript.

<canvas id="myCanvas" width="500" height="500"></canvas>

const canvas = document.getElementById('myCanvas');

const ctx = canvas.getContext('2d');

ctx.fillStyle = 'red';

ctx.fillRect(10, 10, 100, 100);

***создать форму с полями и там проверки сделать***

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Форма Регистрации</title>

</head>

<body>

<form id="registrationForm">

<label for="name">Имя:</label><br>

<input type="text" id="name" name="name" required><br>

<label for="email">Email:</label><br>

<input type="email" id="email" name="email" required><br>

<label for="password">Пароль:</label><br>

<input type="password" id="password" name="password" required minlength="6"><br>

<input type="tel" id="phoneInput" placeholder="+7 (XXX) XXX-XX-XX">

<button type="submit">Зарегистрироваться</button>

</form>

<script src="script.js"></script>

</body>

</html>

document.addEventListener("DOMContentLoaded", function() {

const form = document.getElementById('registrationForm');

form.addEventListener('submit', function(event) {

event.preventDefault(); // Предотвращаем стандартное поведение формы

// Валидация имени

const name = document.getElementById('name').value;

if (!name.trim()) {

alert('Введите свое имя.');

return false;

}

// Валидация электронной почты

const email = document.getElementById('email').value;

if (!email.includes('@')) {

alert('Введите действительный адрес электронной почты.');

return false;

}

// Валидация пароля

const password = document.getElementById('password').value;

if (password.length < 6) {

alert('Пароль должен состоять минимум из 6 символов.');

return false;

}

var input = document.getElementById('phoneInput');

input.addEventListener('input', function(e) {

var x = e.target.value.replace(/\D/g, '').match(/(\d{0,3})(\d{0,3})(\d{0,2})/);

e.target.value =!x[2]? x[1] : '(' + x[1] + ') ' + x[2] + '-' + x[3];

});

// Удаление маски при фокусировке

input.addEventListener('focus', function(e) {

this.value = this.value.replace(/\D/g, '');

});

// Если все проверки пройдены, можно продолжить обработку формы

alert('Форма отправлена успешно!');

return true; // Продолжаем выполнение формы

});

});