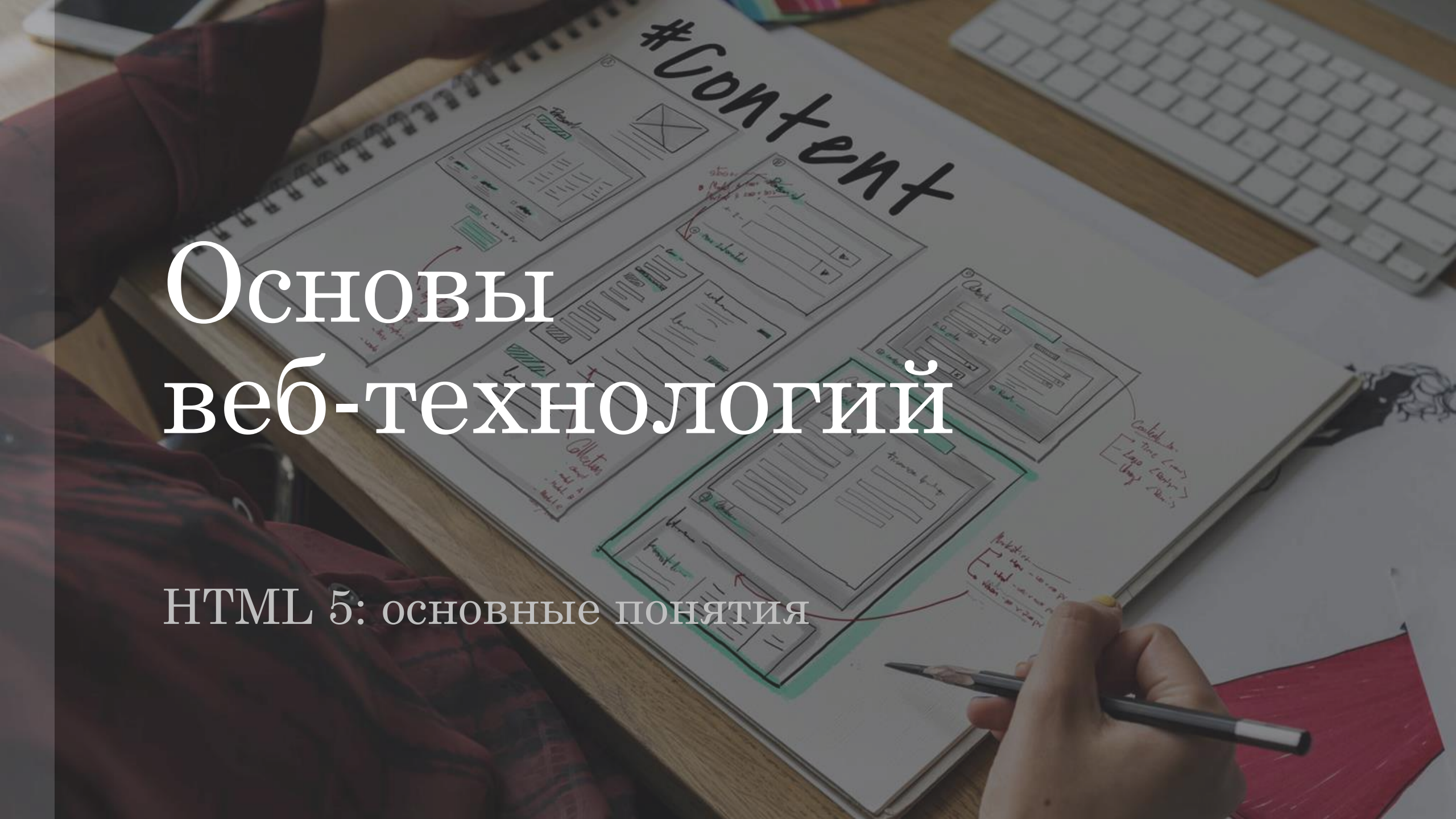


Основы веб-технологий

HTML 5: основные понятия

Visual Studio Code – редактор для кода

https://code.visualstudio.com/

Одной из особенностей VS Code является то, что он распространяется только с английским языком, а другие языки нужно устанавливать отдельно:

Откройте меню «View – Command Palette» или нажмите F1 или комбинацию клавиш Ctrl-Shift-P и начните вводить фразу «Configure Display». После появления подсказки, кликните на пункт «Configure Display Language».

Среди языков выберите «русский» и нажмите установить. После чего перезапустите редактор.

Visual Studio Code – горячие клавиши

WINDOWS

https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf

MAC

https://code.visualstudio.com/shortcuts/keyboard-shortcuts-macos.pdf

LINUX

https://code.visualstudio.com/shortcuts/keyboard-shortcuts-linux.pdf

Live Server для VS Code

Расширение Live Server позволяет запустить свой собственный локальный сервер для разработки и тестирования кода. В нем есть функция «горячей» перезагрузки кода, как для статических страниц, так и для динамических страниц.

- Установите Visual Studio Code (если еще не установлен).
- Зайдите в раздел «Extensions/Расширения» (иконка с блоками на боковой панели) и найдите «Live Server» в поиске.
- Установите расширение, нажав на кнопку «Install/Установить».
- Откройте свой проект в Visual Studio Code и нажмите правой кнопкой мыши на HTML-файле, затем выберите «Open with Live Server» .

Инструменты разработчика

Инструменты разработчика (от англ. «development tools» или сокращённо «DevTools») - это программы, которые позволяют создавать, тестировать и отлаживать (debug) программное обеспечение.

Современные браузеры имеют встроенные инструменты разработчика, позволяющие просмотреть исходный код сайта. С их помощью можно просматривать и отлаживать HTML сайта, его CSS и JavaScript. Также можно проверить сетевой трафик, потребляемый сайтом, его быстродействие и много других параметров.

Открыть инструменты разработчика в большинстве браузеров можно:

- Одной кнопкой — F12.
- Сочетанием клавиш — Ctrl + Shift + I в Windows/Linux или Command + Option + I в macOS.
- Через меню: «Дополнительные инструменты» → «Инструменты для разработчиков».

Язык разметки HTML

HTML (Hypertext Markup Language) – это код, который используется для структурирования и отображения веб-страницы и её контента. Например, контент может быть структурирован внутри множества параграфов, маркированных списков или с использованием изображений и таблиц данных.

- 1991 год – появление HTML. 6 августа 1991 года появился первый сайт (архив сайта).
- Июнь 1993 - HTML 1.2.
- 1994 год - основание W3C
- 24 уоября 1995 – HTML 2.0
- 22 марта 1995 – HTML 3.0
- 17 декабря 1996 – появление CSS
- 14 января 1997 – HTML 3.2
- 18 декабря 1997 – HTML 4.0
- 24 декабря 1999 – HTML 4.01
- **28 октября 2014 – HTML5**
- 1 ноября 2016 – HTML 5.1
- 14 декабря 2017 – HTML 5.2
- 24 декабря 2018 – HTML 5.3

Консорциум Всемирной паутины

Консорциум W3C (World Wide Web Consortium) — независимое международное сообщество, созданное для разработки и внедрения единых стандартов работы сети Интернет на основе унификации, общедоступности и безопасности. Консорциум создан в 1994 г. и в настоящее время объединяет более 400 членов.

Основная цель — разрабатывать единые принципы и стандарты, называемые рекомендациями, которые ложатся в основу разработки программ и оборудования. За счет этого достигается совместимость ПО и аппаратуры разных производителей, что делает интернет совершенным, универсальным и удобным для всех групп пользователей.

Кроме основной задачи, консорциум занимается и другими вопросами, среди которых:

- Интернационализация Сети и стремление сделать интернет максимально удобным для населения всего земного шара. Хотя английский язык по-прежнему остается универсальным языком онлайн-общения, видны некоторые подвижки, например, в виде создания кириллических URL и доменных зон типа .рф.
- Обеспечение удобства использования интернета людьми с ограниченными возможностями.
- Свод рекомендаций W3C постоянно развивается и обновляется. Каждая из рекомендаций перед одобрением проходит пять стадий проверки и рассмотрения и только после этого включается в общий свод правил.

Стандарты, утверждённые W3C

- Annotea
- CC/PP
- Compound Document Formats
- CSS
- DOM
- HTML
- HTTP
- InkML
- MathML
- OWL
- PICS
- PNG
- P3P
- RDF
- SMIL
- SOAP/XMLP
- SPARQL
- Style
- SVG
- TAG
- Timed Text
- URI/URL
- Voice Browser
- WAI
- WAI-ARIA
- WebCGM
- Web Services
- XForms
- XHTML
- XInclude
- XLink
- XML
- XML Base
- XML Binary Characterization
- XML Encryption
- XML Key Management
- XML Query
- XML Schema
- XML Signature
- XPath
- XPointer
- MXSL и XSLT

Сайт Консорциума Всемирной паутины: https://www.w3.org/

Структура HTML-документа

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="UTF-8">
    <title>...</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <script src="script.js"></script>
  </head>
  <body>
      <!-- Основная часть документа -->
  </body>
</html>
```

Язык HTML следует правилам, которые содержатся в файле объявления типа документа *(Document Type Definition, или DTD)*. DTD представляет собой XML-документ, определяющий, какие элементы, атрибуты и их значения действительны для конкретного типа HTML. Для каждой версии HTML есть свой DTD.

DOCTYPE отвечает за корректное отображение веб-страницы браузером. DOCTYPE определяет не только версию HTML (например, html), но и соответствующий DTD-файл в Интернете.

Комментарии

В HTML, как и в других языках разметки, допустимо использование комментариев. Они устанавливаются при помощи пары конструкций:

```
<!-- Текст комментария -->
```

Дерево тегов

Элементы, находящиеся внутри элемента <html>, образуют дерево документа, так называемую объектную модель документа, DOM (document object model). При этом элемент <html> является корневым элементом.

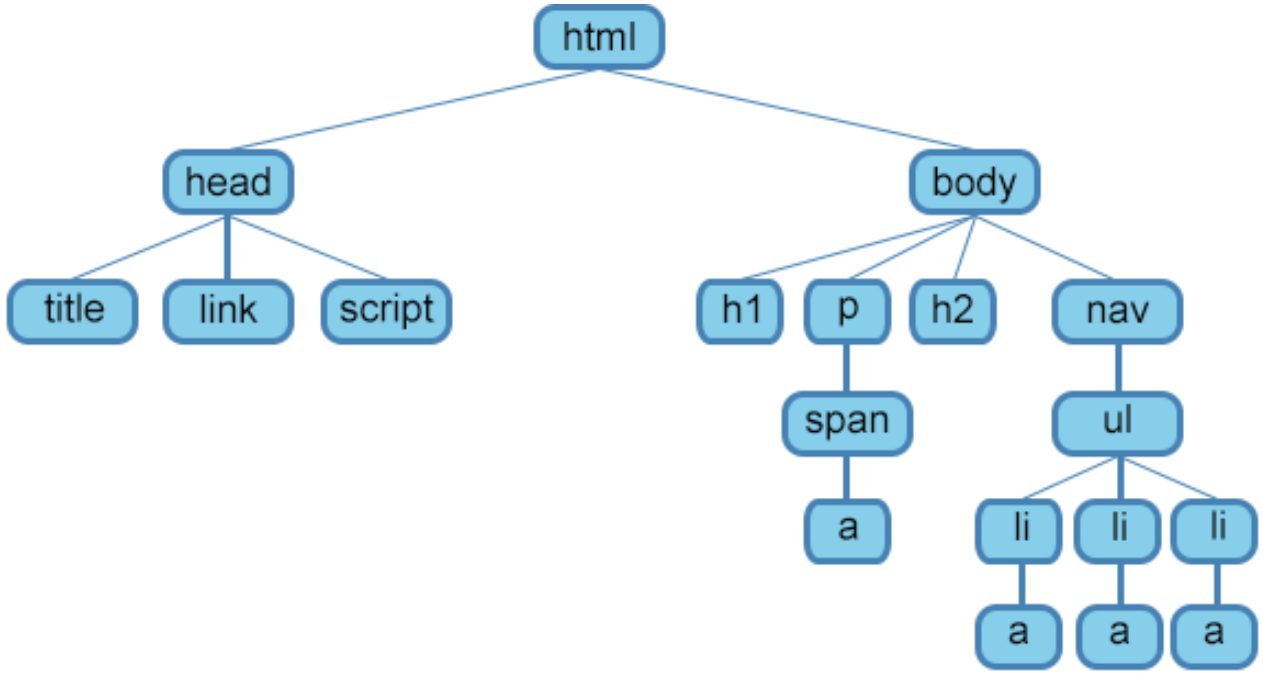
Предок — элемент, который заключает в себе другие элементы.

Потомок — элемент, расположенный внутри одного или более типов элементов.

Родительский элемент — элемент, связанный с другими элементами более низкого уровня, и находящийся на дереве выше их.

Дочерний элемент — элемент, непосредственно подчиненный другому элементу более высокого уровня.

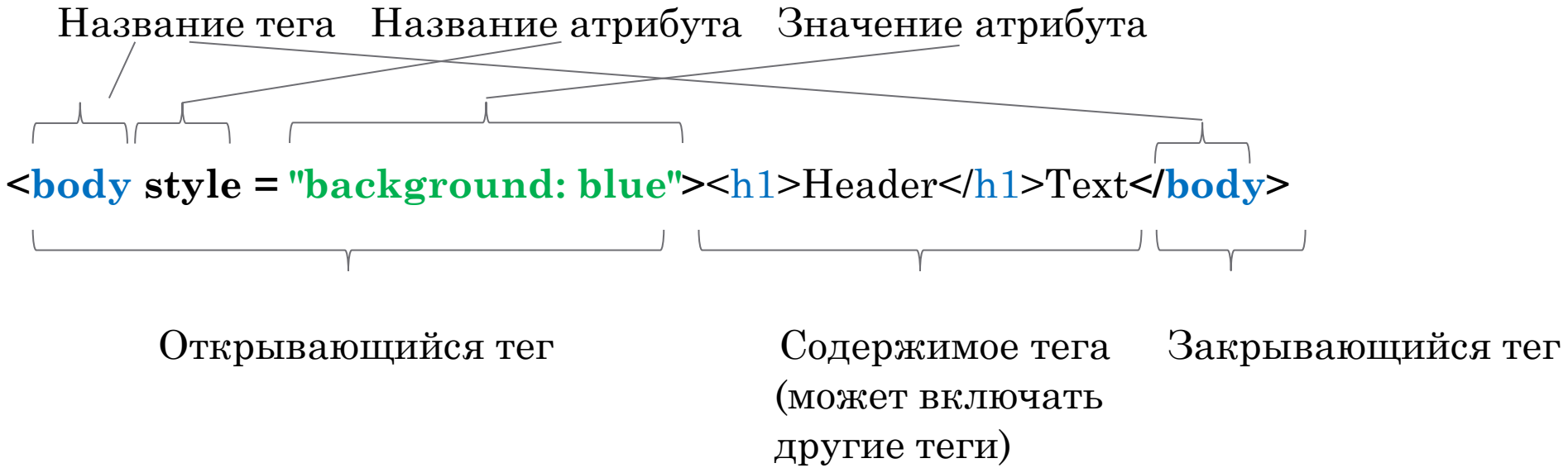
Сестринский элемент — элемент, имеющий общий родительский элемент с рассматриваемым, так называемые элементы одного уровня.



Теги

HTML-код состоит из элементов, каждый из которых может быть тегом или метатегом. У них есть дополнительные характеристики – атрибуты.

Элемент HTML — это тип компонента HTML-документа, который состоит из дерева простых узлов HTML, таких как текстовые узлы. Такие элементы позволяют HTML-документу иметь определенную семантику и форматирование.



HTML-атрибуты

HTML-атрибуты это специальные слова, которые управляют поведением HTML-элемента. Они добавляют дополнительную функциональность, либо меняют поведение элемента по умолчанию. Атрибуты элемента выражаются внутри начального тега элемента.

Атрибут имеет имя и значение. Имена атрибутов должны состоять из одного или нескольких символов, кроме управляющих, таких как пробел, ", ', >, / и =. Имена и значения атрибутов не чувствительны к регистру, но, тем не менее, рекомендуется набирать их в нижнем регистре.

Некоторые атрибуты не требуют значение, потому что у них есть только одна опция. Они называются логическими атрибутами.

Атрибуты могут быть указаны четырьмя различными способами:

- имя атрибута, например, disabled
- значение атрибута без кавычек, например, autocapitalize=sentences
- значение атрибута в одинарных кавычках, например, type='checkbox'
- значение атрибута в двойных кавычках, например, class="external icon-link"

Типы атрибутов HTML-тегов

- **Глобальные атрибуты,** которые могут быть использованы для любого HTML-элемента, хотя некоторые из них могут не оказывать на элементы никакого влияния.

- **Атрибуты событий,** для вызова обработчиков события окна, формы, клавиатуры, мыши и т.д.

- **Атрибуты тегов,** которые получили свое название от того, что они есть только у определенных тегов или набора тегов.

Глобальные атрибуты

Примеры глобальных атрибутов:

Атрибут	Описание, принимаемое значение
class	Представляет собой разделенный пробелом список классов элемента с учетом регистра. Классы позволяют CSS и Javascript выбирать и получать доступ к элементам с помощью селекторов классов или функций, таких как метод DOM document.getElementsByClassName. Синтаксис: class="toc" class="external icon-link"
hidden	Указывает браузеру, что элемент должен быть скрыт. **Синтаксис:** hidden
id	Определяет уникальный идентификатор элемента. Его цель — идентифицировать элемент при связывании (используя идентификатор фрагмента — якорь), сценарии или стилизации с помощью CSS. Значение id не должно содержать пробелов. В отличие от атрибута class, элементы могут иметь только одно единственное значение идентификатора. **Синтаксис:** id="content"
lang	Указывает основной язык для содержимого элемента и для любого из атрибутов элемента, содержащих текст. Его значение должно быть допустимым языковым тегом или пустой строкой. Установка для атрибута пустой строки означает, что основной язык неизвестен. **Синтаксис:** lang="en-GB"
style	Содержит объявления стилей CSS, которые следует применить к элементу. Рекомендуется определять стили в отдельном файле или файлах. Этот атрибут и элемент <style> в основном предназначены для быстрой настройки стилей, например, для целей тестирования. **Синтаксис:** style="color: blue; background: transparent"

Глобальные атрибуты

Атрибут	Описание, принимаемое значение
autocapitalize	Дает подсказку браузеру, каким образом вводимый текст будет автоматически писаться с заглавной буквы при вводе/редактировании пользователем. Атрибут не влияет на поведение при наборе текста на физической клавиатуре, только на поведение других механизмов ввода, таких как виртуальные клавиатуры на мобильных устройствах и голосовой ввод. Атрибут autocapitalize никогда не приводит к автоматическому включению заглавных букв для элемента <input> типа url, email или password. **Разрешенные значения:** off или none — все буквы по умолчанию в нижнем регистре. on или sentences — первая буква каждого предложения по умолчанию заглавная; все остальные буквы по умолчанию в нижнем регистре. words — первая буква каждого слова по умолчанию заглавная; все остальные буквы по умолчанию в нижнем регистре. characters — все буквы по умолчанию должны быть в верхнем регистре. **Синтаксис:** autocapitalize="sentences"
title	Содержит дополнительную информацию об элементе, задавая всплывающую подсказку для страницы. Атрибут title может содержать несколько строк, каждый символ перевода строки добавляет ее разрыв. Если элемент не имеет атрибута title, то он наследует его от своего родительского элемента, который, в свою очередь, может наследовать его от своего родительского элемента и так далее. Если для этого атрибута задана пустая строка, это означает, что заголовки его родительских элементов не будут отображаться для этого элемента. **Синтаксис:** title="Hypertext Transport Protocol"
translate	Указывает, следует ли переводить переводимые значения атрибута элемента и его дочерние текстовые узлы при переводе страницы. Хотя не все браузеры распознают этот атрибут, он учитывается системами автоматического перевода, такими как Google Translate, а также может учитываться инструментами, используемыми людьми-переводчиками. Поэтому важно, чтобы веб-авторы использовали этот атрибут, чтобы пометить контент, который не следует переводить. **Разрешенные значения:** yes или пустая строка — элемент должен быть переведен при переводе страницы. no — элемент не должен быть переведен. **Синтаксис:** translate="no"

Заголовки страницы

Раздел `<head>...</head>` содержит техническую информацию о странице: заголовок, описание, ключевые слова для поисковых машин, кодировку и т.д. Введенная в нем информация не отображается в окне браузера, однако содержит данные, которые указывают браузеру, как следует обрабатывать страницу.

Задача `<head>` — хранить метаданные документа.

Раздел заголовков может содержать следующие теги:

- `<title>`
- `<base>`
- `<link>`
- `<style>`
- `<script>`
- `<meta>`

Заголовки. Элемент <title>, <base>

Обязательным элементом раздела `<head>` является `<title>`. Текст, размещенный внутри элемента `<title>`, отображается в строке заголовка веб-браузера.

Длина заголовка должна быть не более 60 символов, чтобы полностью поместиться в заголовке. Текст заголовка должен содержать максимально полное описание содержимого веб-страницы.

Элемент `<base>` позволяет задать базу для всех гиперссылок страницы. Таким образом, ко всем гиперссылкам будет в начало добавляться адрес, указанный в элементе base.

Заголовки. Элемент <style>

Внутри этого элемента задаются стили, которые используются на странице. Для задания стилей в HTML-документе используется язык CSS.

```
<style>
  .paper {
    width: 200px;
    height: 300px;
    background-color: #ef4444;
    color: #666666;
  }
</style>
```

Заголовки. Элемент <link>

Задать стили для документа можно также при помощи другого способа — записать их в отдельный файл с расширением `.css`, например, `style.css и` подключить файл со стилями используя элемента `<link>`.

```
<head>
    <link rel="stylesheet" href="style.css" type="text/css">
    <meta charset="UTF-8">
    <title>...</title>
</head>
```

Заголовки. Элемент <script>

Элемент `<script>` позволяет присоединять к документу различные сценарии. Текст сценария может располагаться либо внутри этого элемента, либо во внешнем файле. Если текст сценария расположен во внешнем файле, то он подключается с помощью атрибутов элемента.

```
<head>
   <meta charset="UTF-8">
   <title>...</title>
   <script src="script.js"></script>
</head>
```

Заголовки. Элемент <meta>

Необязательным элементом раздела `<head>` является элемент `<meta>`. С его помощью можно задать описание содержимого страницы и ключевые слова для поисковых машин, автора HTML-документа и прочие свойства метаданных.

Элемент `<head>` может содержать несколько элементов `<meta>`, потому что, в зависимости от используемых атрибутов, они несут различную информацию, например метатег charset задает кодировку символов документа html.

```
<meta charset="utf-8">
```

Метатег viewport

Viewport — это область, которую видит пользователь на экране, когда заходит на страницу сайта с любого устройства.

Раньше все было просто: разрешение экранов более или менее одинаковое, делай себе одну ширину макета, и никто не будет жаловаться. Сейчас стало труднее: пользователи заходят на страницы с разных устройств, разрешение экрана сильно различается.

Чтобы пользоваться сайтами было удобно, нужно правильно масштабировать страницы. Для этого используется метатег viewport. Он не делает верстку адаптивной. Его предназначение — контроль масштаба отображения страницы.

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8">
    <title>Viewport test</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <p>Viewport – это область, которую видит пользователь на экране, когда заходит на страницу сайта с
любого устройства.<p>
  </body>
</html>
```

Два разрешения экрана: физическое и CSS

Метатег viewport решает эту проблему адаптивного дизайна с помощью двух параметров: width и initial-scale.

Параметр width=device-width приравнивает ширину viewport к CSS-ширине устройства. CSS — это не физический размер, а некая величина, предназначенная для того, чтобы адаптивный дизайн отображался на экранах одинаково.

CSS-разрешение зависит от плотности пикселей.

- Если плотность пикселей меньше 200ppi, то коэффициент будет 1. То есть у экрана с физическим разрешением 320x480 пикселей будет CSS-разрешение 320x480 пикселей.
- Плотность пикселей 200-300ppi — коэффициент 1,5.
- Плотность пикселей больше 300ppi — коэффициент рассчитывается по формуле плотность/150, а полученное значение округляется (2, 2.5, 3 и так далее).

Размер области просмотра конкретного устройства можно узнать — тогда не придется высчитывать его самостоятельно. Но под все экраны подстроиться все равно не получится — их слишком много. Поэтому при верстке обычно пользуются универсальными значениями.

Правильное использование метатега viewport

Для области просмотра можно определить следующие параметры:

- width — ширина области viewport. Принимает значение **device-width** или фактическое число пикселей в виде целого неотрицательного числа: например, 320px.
- height — высота области viewport. Принимает значение **device-height** или фактическое число пикселей.
- initial-scale — коэффициент масштабирования начального размера viewport. Принимает значения от 0.1 до 10. Значение 1.0 задает отсутствие масштабирования.
- user-scalable — указывает, может ли пользователь масштабировать страницу жестами. Принимает значение **yes** или **no**.
- minimum-scale — минимальное значение масштабирования. Принимает значения от 0.1 до 10. Значение 1.0 задает отсутствие масштабирования.
- maximum-scale — максимальное значение масштабирования. Принимает значения от 0.1 до 10. Значение 1.0 задает отсутствие масштабирования.

Универсальное использование метатега viewport выглядит так:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Meta content для поисковых систем

Title
Фактически поле Title не относится к метаданным, но заголовок часто рассматривают в контексте meta, так как он тоже используется браузерами и поисковыми системами. Тег Title оказывает значительное влияние на кликабельность сниппета и результаты ранжирования.
Требования к **качественному** заголовку:
- интеграция в начало основной ключевой фразы;
- уникальный вариант для каждой страницы;
- привлекательность для целевой аудитории;
- объем до 60 символов.

Description
Meta name description используется алгоритмами для формирования краткого описания в сниппете. Этот тег должен соответствовать таким требованиям, как:
- Лаконичность (от 160 до 240 символов).
- Привлекательность.
- Высокая концентрация смысла.
- Семантическая релевантность.
- Для каждой страницы нужно прописать уникальный Description.

Keywords
Meta name keywords продолжительное время имел статус мощного фактора ранжирования. Но из-за частых манипуляций поисковые системы перестали учитывать его. Заполнять это поле нужно аккуратно, чтобы роботы не признали наполнение спамом.

Есть следующие актуальные подходы наполнения:
- не прописывать ничего;
- вписывать ключевые фразы через запятую;
- добавить набор релевантных слов, разделенных пробелом.

Meta content для поисковых систем

```
<meta name="description" content="Описание содержимого страницы">
<meta name="keywords" content="Ключевые,слова,через,запятую">
```

```
<meta name="description" lang="ru" content="Описание содержимого страницы">
<meta name="description" lang="en" content="Description">
<meta name="keywords" lang="ru" content="Ключевые,слова,через,запятую">
<meta name="keywords" lang="en" content="Keywords">
```

Meta content для поисковых систем

Robots
Эта группа meta-данных создана для передачи алгоритмам поисковых систем инструкций по индексации страниц. У «Яндекса» и Google существуют собственные форматы взаимодействия: YandexBot и Googlebot соответственно.

Допустимые функции:
- all – открывает доступ к индексации всех страниц и возможность перехода по всем ссылкам;
- noindex – запрет сканирования указанной директории;
- index – разрешение на сканирование;
- nofollow – запрет перехода по ссылке;
- follow – разрешение перехода по ссылкам;
- noarchive – запрет демонстрации в выдаче архивной страницы;
- noyaca (в «Яндексе») – для формирования сниппета нельзя использовать информацию из сервиса «Яндекс.Каталог»;
- nosnippet (в Google) – запрет на формирование сниппета с использованием части текста;
- noimageindex (в Google) – нельзя позиционировать страницу как первоисточник публикации графического контента;
- unavailable_after:[date] (в Google) – запрет на индексацию после определенной даты;
- none – аналог команд noindex и nofollow.

```
<meta name="robots" content="index, nofollow">
```

Формирование содержания в теге <body>. HTML-текст

HTML-текст представлен в спецификации элементами для форматирования и группировки текста. Данные элементы являются контейнерами для текста и не имеют визуального отображения.

Элементы для форматирования текста несут смысловую нагрузку и обычно задают для текста, заключенного внутрь, стилевое оформление, например, выделяют текст жирным начертанием.

Грамотно отформатированный текст дает понять поисковым системам, какие слова несут важную смысловую нагрузку, по каким из них предпочтительно ранжировать веб-страницу в поисковой выдаче.

Вся текстовая информация, отображаемая на сайте, размещается внутри элемента `<body>`.

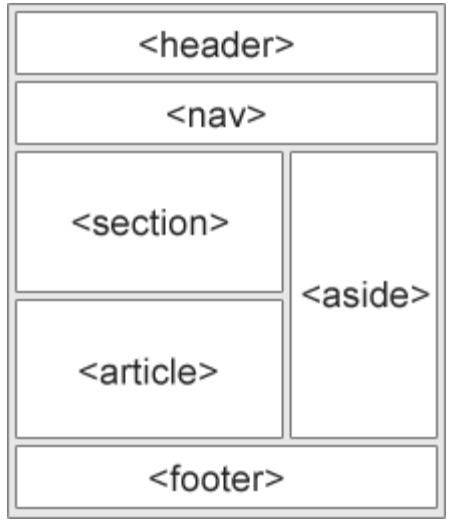
Семантические теги

Семантические теги — это теги, которые предназначены для того чтобы компьютерные программы (поисковые системы, сборщики информации, речевые браузеры и т.д.), понимали какой тип информации заложен в данных тегах.

Политика спецификации HTML5 заключается в том, чтобы каждый HTML-тег имел свою семантику.

Новые новые семантические теги HTML5 :

```
<article>, <aside>,
<details>,<figcaption>,
<figure>, <footer>,
<header>, <main>,
<mark>, <nav>,
<section>, <summary>,
<time>
```



Семантические теги

Тег `<main>` определяет основное содержимое документа. Содержимое внутри элемента `<main>` должно быть уникальным для данного документа. Здесь не должно быть ничего, что повторяется где-либо, вроде сайдбаров, навигационных ссылок, информации об авторских правах, логотипов сайта, поисковых форм и т.п. В документе может быть только один элемент `<main>`, и он не должен быть вложен в элементы `<article>`, `<aside>`, `<footer>`, `<header>` или <nav>.

Элемент `<section>` определяет раздел в документе, который представляет собой тематическую группировку контента, обычно с заголовком.

Элемент `<article>` определяет независимое, автономное содержимое и должна быть возможность читать его независимо от остальной части веб-сайта. Семантический акцент: поисковая система считает, что информация расположенная между этими тегами, является основной на HTML-странице.

Элемент `<header>` задает заголовок для документа или раздела. Элемент `<header>` должен использоваться в качестве контейнера для вступительного содержания. В одном документе может быть несколько элементов `<header>`.

Элемент `<footer>` указывает нижний колонтитул для документа или раздела. Элемент `<footer>` должен содержать сведения о содержащем его элементе. Нижний колонтитул обычно содержит автора документа, информацию об авторском праве, ссылки на условия использования, контактные данные и т.д. В одном документе может быть несколько элементов `<footer>`.

Элемент `<nav>` определяет набор навигационных ссылок.

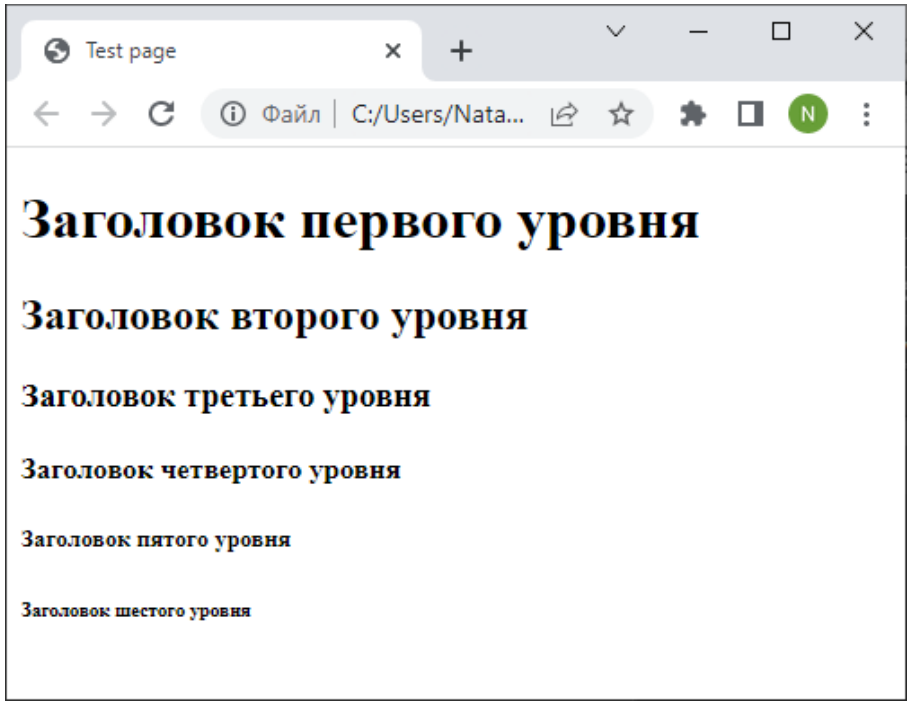
Элемент `<aside>` определяет блок сбоку от основного контента. Контентом, заключенным в этот тег, могут быть, например, список рубрик, архивных ссылок, метки и другая информация.

HTML-элементы для заголовков

Заголовки являются важными элементами веб-страницы, они упорядочивают текст, формируя его визуальную структуру и используются для структурирования контента и помощи поисковым системам для понимания иерархии информации на странице. Элементы `<h1>...<h6>` должны использоваться только для выделения заголовков нового раздела или подраздела.

При использовании заголовков необходимо учитывать их иерархию, т.е. за `<h1>` должен следовать `<h2>` и т.д. Также не рекомендуется вкладывать в заголовки другие элементы.

```
<body>
  <h1>Заголовок первого уровня</h1>
  <h2>Заголовок второго уровня</h2>
  <h3>Заголовок третьего уровня</h3>
  <h4>Заголовок четвертого уровня</h4>
  <h5>Заголовок пятого уровня</h5>
  <h6>Заголовок шестого уровня</h6>
</body>
```

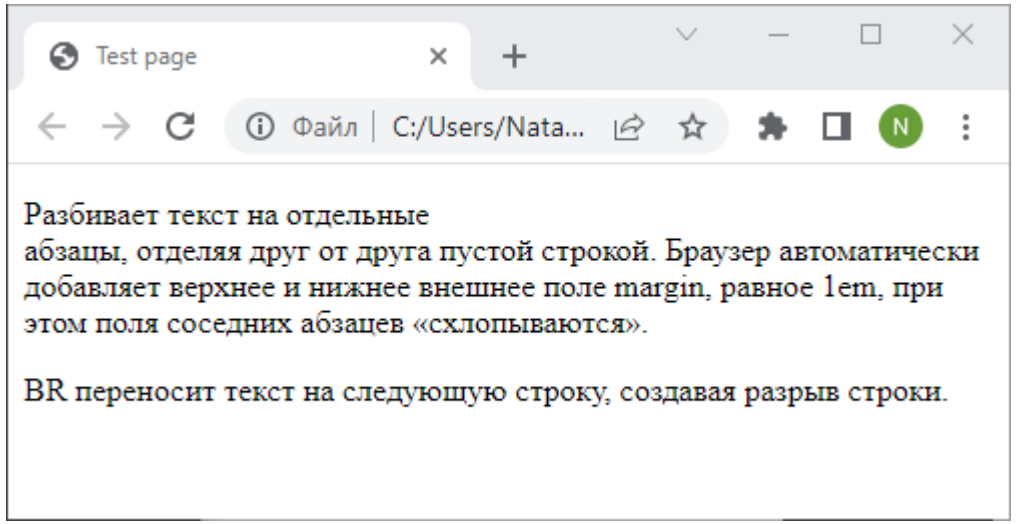


Абзацы, средства переноса текста

Элемент <p>
Разбивает текст на отдельные абзацы, отделяя друг от друга пустой строкой. Браузер автоматически добавляет верхнее и нижнее внешнее поле margin, равное 1em, при этом поля соседних абзацев «схлопываются».

**Элемент
**
Переносит текст на следующую строку, создавая разрыв строки.

```
<body>
<p>
Разбивает текст на отдельные <br> абзацы, отделяя друг от друга пустой строкой. Браузер автоматически добавляет верхнее и нижнее внешнее поле margin, равное 1em, при этом поля соседних абзацев «схлопываются».
</p>
<p>
BR переносит текст на следующую строку, создавая разрыв строки.
</p>
</body>
```

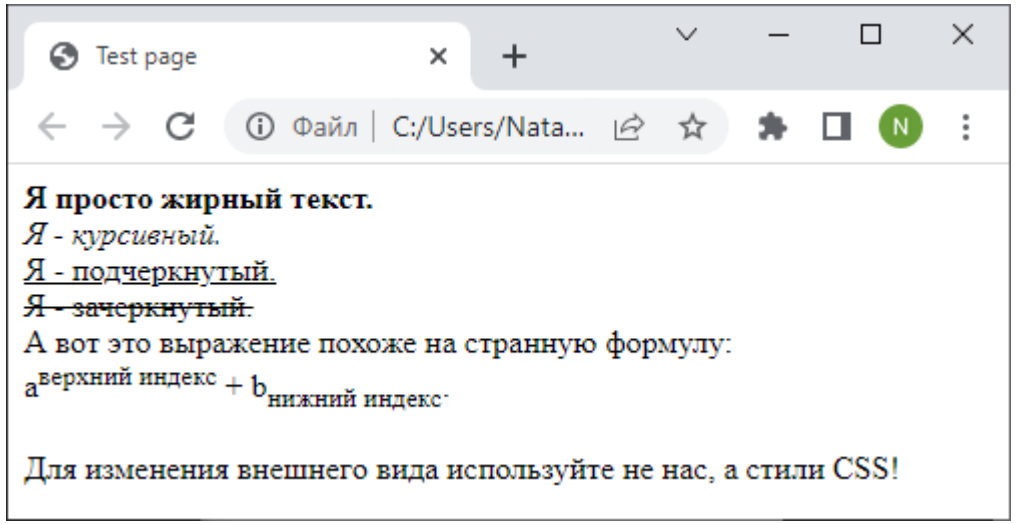


Физическое форматирование текста

В предыдущих версиях HTML элементы, отвечающие за форматирование документа, делились на две группы: элементы физического и логического форматирования. Однако в пятой версии языка все элементы физической разметки, которые не были удалены, изменили свое назначение и стали относиться к группе элементов логического форматирования.

Теперь HTML5 используется в веб-программировании практически полностью как инструмент логической разметки документа, а за внешний вид отвечает CSS и применять какие-либо теги только лишь для изменения внешнего вида текста крайне не рекомендуется. В HTML5 все теги имеют логическую нагрузку, хотя отношение некоторых из них к логическим и выглядит несколько натянутым.

```
<body>
<b>Я просто жирный текст.</b> <br>
  <i>Я - курсивный.</i> <br>
  <u>Я - подчеркнутый.</u> <br>
  <s>Я - зачеркнутый.</s> <br>
  А вот это выражение похоже на странную
формулу:<br>
  a<sup>верхний индекс</sup> + b<sub>нижний
индекс</sub>.<br><br>
  Для изменения внешнего вида используйте не нас, а
стили CSS!
<body>
```



Логическое форматирование текста

Логическое форматирование текста – это процесс форматирования html-кода при помощи набора соответствующих элементов разметки, которые предназначены главным образом для структурной, логической разметки документа, определяя степень важности своего содержимого и его отношение к тому или иному типу данных, а также выделяя смысловое отличие своего содержимого от окружающего контекста.

- (от англ. emphasis – акцент) – предназначен для акцентирования внимания на фрагменте текста (обращает внимание на его важность) и отображает свое содержимое курсивом;
- (от англ. strong – сильный) – предназначен для еще большего акцентирования текста (делает его еще более важным) и отображает свое содержимое полужирным шрифтом;
- <cite> (от англ. cite – цитировать) – предназначен для выделения сносок на другой материал и отображает свое содержимое курсивом;
- <code> (от англ. code – код) – предназначен для выделения текста программного кода и отображает свое содержимое моноширинным шрифтом;
- <kbd> (от англ. keyboard – клавиатура) – предназначен для выделения текста, который должен быть введен с клавиатуры или используется для названия клавиш клавиатуры; элемент отображает свое содержимое моноширинным шрифтом;
- <var> (от англ. variable – переменная) – предназначен для выделения переменных компьютерных программ и отображает свое содержимое курсивом;
- <samp> (от англ. sample – пример, образец) – предназначен для выделения текста, который является результатом вывода компьютерной программы или скрипта, и отображает свое содержимое моноширинным шрифтом;
- <dfn> (от англ. definition – определение) – предназначен для выделения терминов, которые встречаются в тексте впервые, и отображает свое содержимое курсивом;

Логическое форматирование текста

- <abbr> (от англ. abbreviation – аббревиатура) – предназначен для выделения аббревиатур и обычно используется с атрибутом title, содержащим расшифровку аббревиатуры; текст данного элемента браузерами никак не выделяется, сохраняя исходное форматирование;
- <q> (от англ. quote – цитата, кавычки) – предназначен для выделения в тексте небольших цитат и отображает свое содержимое в кавычках;
- <ins> (от англ. inserted – вставленный) – предназначен для выделения текста, который был добавлен в новую версию документа, и отображает свое содержимое подчеркнутым;
- <del> (от англ. deleted – удаленный) – предназначен для выделения текста, который был удален в новой версии документа, и отображает свое содержимое зачеркнутым;
- <small> (от англ. small – маленький) – предназначен для выделения текста, который можно отнести к надписям мелким шрифтом или пометкам, как, например, второстепенная информация в конце юридических документов об отказе от ответственности или же информации о лицензии; элемент отображает свое содержимое моноширинным шрифтом уменьшенного размера;
- <time> (от англ. time – время) – предназначен для создания контейнера, содержимое которого помечается как дата, время или дата и время; обычно содержимое элемента «time» представляет собой дату и время в удобочитаемом для пользователей формате; браузеры никак не выделяют содержимое элемента «time».

Логическое форматирование текста

- `<address>` (от англ. address – адрес) – элемент разметки «address», сформированный данным парным тегом, предназначен для выделения информации об авторе, например, ссылка на его ресурс, адрес и т.д.; элемент отображает свое содержимое курсивом, а браузеры отображают **как блочный элемент**.

- `<blockquote>` (от англ. blockquote – блок с цитатой) – элемент разметки «blockquote», сформированный данным парным тегом, предназначен для выделения длинных цитат, в отличие от элемента «q», при помощи которого выделяются небольшие цитаты; браузеры отображают **элемент как блочный**.

```
<body>
   <h3>Пушкин А.С.</h3>
   <blockquote>
     У лукоморья дуб зелёный; <br>
     Златая цепь на дубе том:<br>
     И днём и ночью кот учёный<br>
     Всё ходит по цепи кругом  <br>
   </blockquote>
   <address>Связаться с автором цитаты не получится.</address>
 </body>
```

HTML-списки

HTML-списки используются для группировки связанных между собой фрагментов информации. Существует три вида списков:

- маркированный список `<ul>` — каждый элемент списка `<li>` отмечается маркером,
- нумерованный список `<ol>` — каждый элемент списка `<li>` отмечается цифрой,
- список определений `<dl>` — состоит из пар термин `<dt> - <dd>` определение.

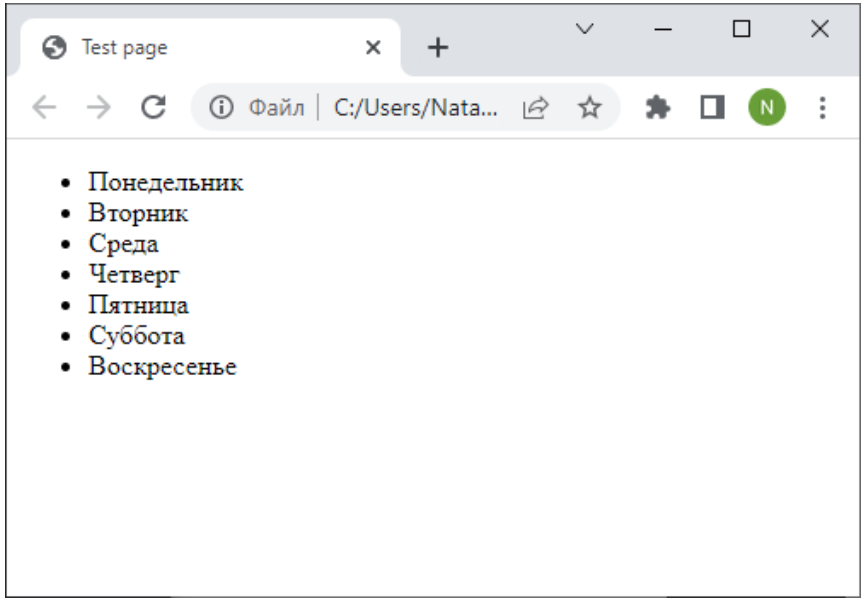
Каждый список представляет собой контейнер, внутри которого располагаются элементы списка или пары термин-определение.

Элементы списка ведут себя как блочные элементы, располагаясь друг под другом и занимая всю ширину блока-контейнера. Каждый элемент списка имеет дополнительный блок, расположенный сбоку, который не участвует в компоновке.

Маркированный список

Маркированный список представляет собой неупорядоченный список (от англ. Unordered List). Создаётся с помощью элемента `<ul>`. В качестве маркера элемента списка выступает метка, например, закрашенный кружок.

```
<ul>
  <li>Понедельник</li>
  <li>Вторник</li>
  <li>Среда</li>
  <li>Четверг</li>
  <li>Пятница</li>
  <li>Суббота</li>
  <li>Воскресенье</li>
</ul>
```



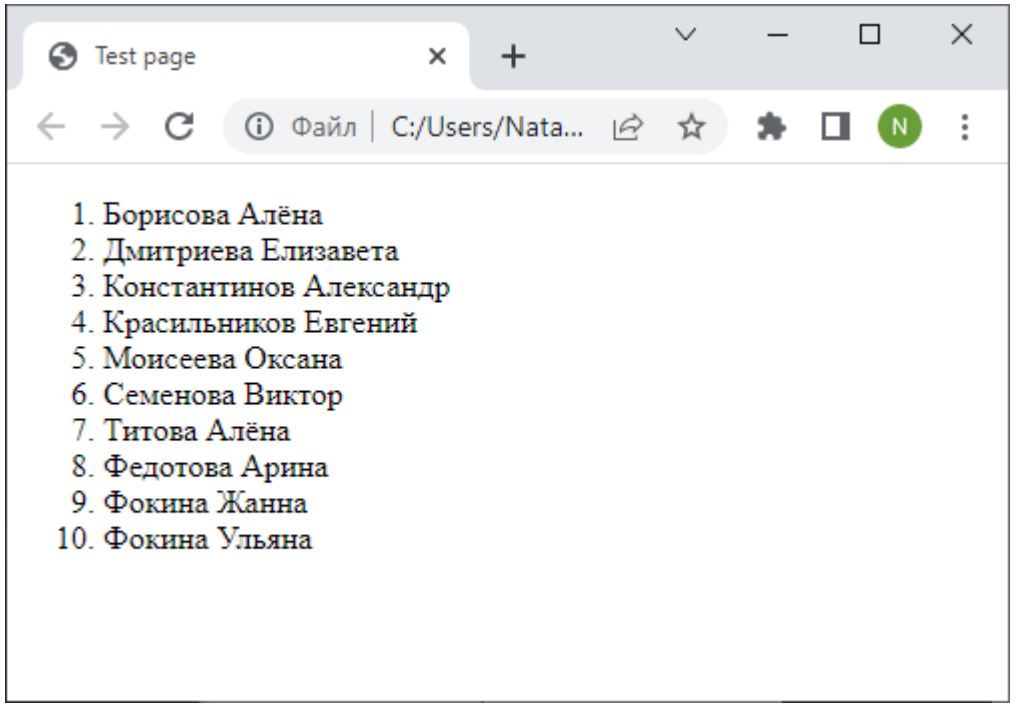
Нумерованный список

Нумерованный список создаётся с помощью элемента `<ol>`. Каждый пункт списка также создаётся с помощью элемента `<li>`. Браузер нумерует элементы по порядку автоматически и если удалить один или несколько элементов такого списка, то остальные номера будут автоматически пересчитаны.

Для элемента `<li>` доступен атрибут `value`, который позволяет изменить номер по умолчанию для выбранного элемента списка. Например, если для первого пункта списка задать `<li value="10">`, то остальная нумерация будет пересчитана относительно нового значения.

Атрибут `reversed` задает отображение списка в обратном порядке. Атрибут `start` задает начальное значение, от которого пойдет отсчет нумерации.

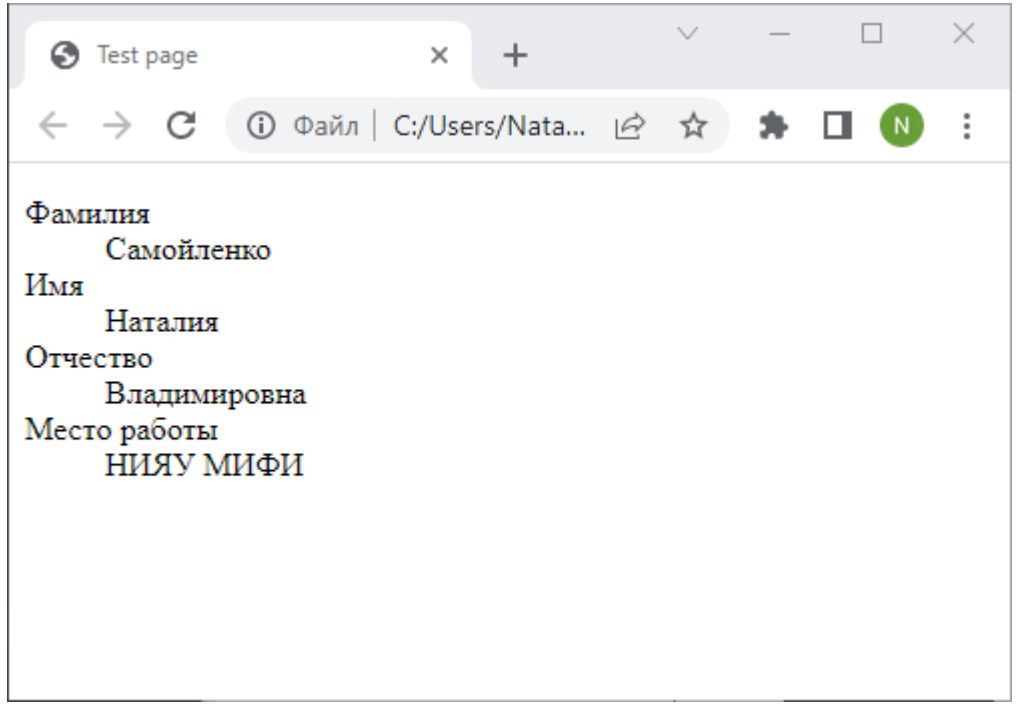
```
<ol>
  <li>Борисова Алёна</li>
  <li>Дмитриева Елизавета</li>
  <li>Константинов Александр</li>
  <li>Красильников Евгений</li>
  <li>Моисеева Оксана</li>
  <li>Семенова Виктор</li>
  <li>Титова Алёна</li>
  <li>Федотова Арина</li>
  <li>Фокина Жанна</li>
  <li>Фокина Ульяна</li>
</ol>
```



Список определений

Списки определений создаются с помощью элемента `<dl>`. Для добавления термина применяется элемент `<dt>`, а для вставки определения — элемент `<dd>`.

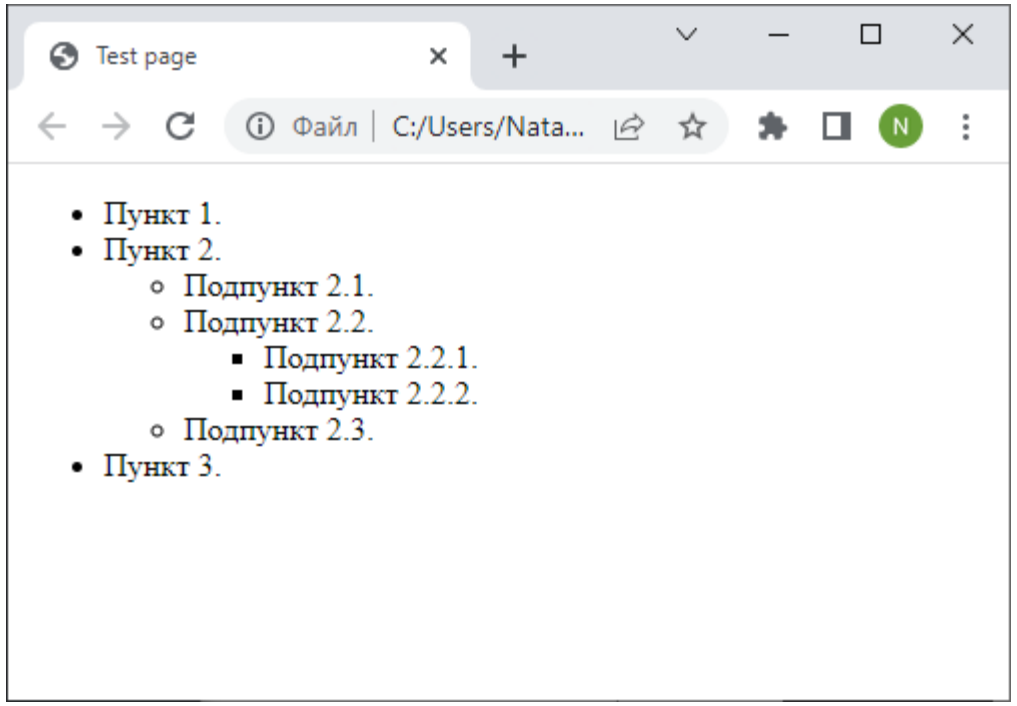
```
<dl>
  <dt>Фамилия</dt>
  <dd>Самойленко</dd>
  <dt>Имя</dt>
  <dd>Наталия</dd>
  <dt>Отчество</dt>
  <dd>Владимировна</dd>
  <dt>Место работы</dt>
  <dd>НИЯУ МИФИ</dd>
</dl>
```



Вложенный список

Зачастую возможностей простых списков не хватает, например, при создании оглавления никак не обойтись без **вложенных пунктов**. Разметка для вложенного списка будет следующей:

```
<ul>
 <li>Пункт 1.</li>
  <li>Пункт 2.
    <ul>
      <li>Подпункт 2.1.</li>
       <li>Подпункт 2.2.
        <ul>
          <li>Подпункт 2.2.1.</li>
          <li>Подпункт 2.2.2.</li>
        </ul>
       </li>
      <li>Подпункт 2.3.</li>
    </ul>
  </li>
 <li>Пункт 3.</li>
</ul>
```



Ссылки

HTML-ссылки создаются с помощью элементов `<a>` и `<link>`. Ссылки представляют собой связь между двумя ресурсами, одним из которых является текущий документ.
Ссылки можно поделить на две категории:

- ссылки на внешние ресурсы — создаются с помощью элемента `<link>` и используются для расширения возможностей текущего документа при обработке браузером;
- гиперссылки — ссылки на другие ресурсы, которые пользователь может посетить или загрузить.

Гиперссылки создаются с помощью элемента `<a></a>`. Внутрь помещается текст, который будет отображаться на веб-странице. Текст ссылки отображается в браузере с подчёркиванием, цвет шрифта — синий, при наведении на ссылку курсор мыши меняет вид.
Обязательным параметром элемента `<a>` является атрибут `href`, который задает URL-адрес веб-страницы.

```
<a href="https://mephi.ru">указатель ссылки</a>
```

Ссылки

Ссылка состоит из двух частей — **указателя** и **адресной части**. **Указатель ссылки** представляет собой фрагмент текста или изображение, видимые для пользователя. **Адресная часть** ссылки пользователю не видна, она представляет собой адрес ресурса, к которому необходимо перейти.

Адресная часть ссылки состоит из **URL** (Uniform Resource Locator) — унифицированный адрес ресурса.

Абсолютные ссылки

Абсолютная ссылка — это адрес ресурса целиком. Обычно такие ссылки ставят на сторонние ресурсы и надеются, что адрес не изменится.

```
http://site.ru/pages/tips/tips1.html
```

При отсутствии имени файла в адресе абсолютной ссылки будет загружаться веб-страница, которая задана по умолчанию в настройках веб-сервера (так называемый индексный файл).

```
http://site.ru/
```

Обычно в качестве индексного файла выступает документ с именем index.html.

Если вы используете абсолютные ссылки на сайте, то при смене структуры папок или перемещении файлов большинство ссылок могут сломаться.

Относительные ссылки

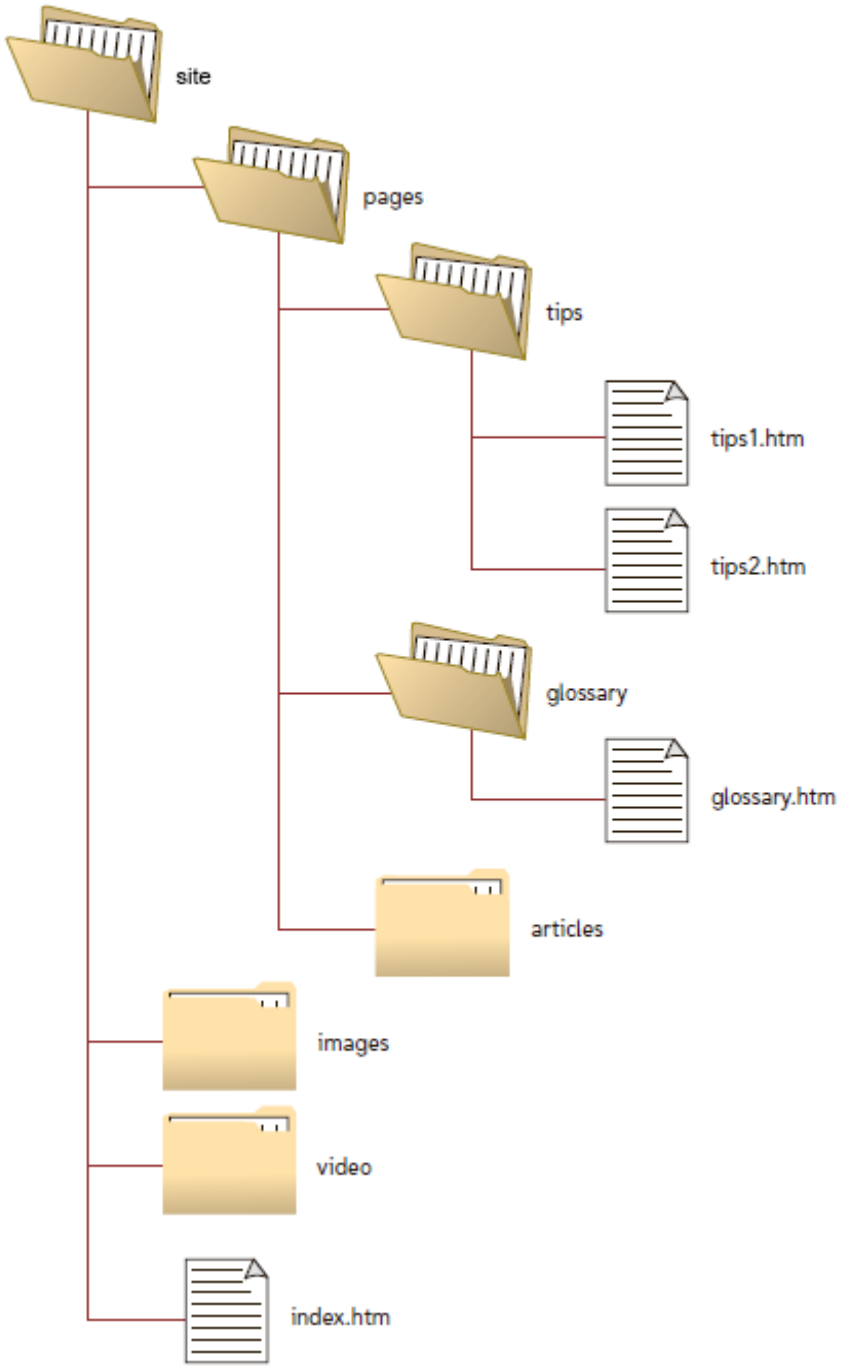
Путь относительной ссылки определяется с учётом местоположения веб-страницы, на которой находится ссылка. Относительные ссылки используются при создании ссылок на другие документы на одном и том же сайте. Когда браузер не находит в ссылке протокол, он выполняет поиск указанного документа на том же сервере.

Относительные ссылки

Путь для относительных ссылок имеет три специальных обозначения:

- `/` указывает на корневую директорию и говорит о том, что нужно начать путь от корневого каталога документов и идти вниз до следующей папки
- `./` указывает на текущую папку
- `../` подняться на одну папку (директорию) выше

Главное отличие относительного пути от абсолютного в том, что относительный путь не содержит имени корневой папки и родительских папок, что делает адрес короче, и в случае переезда с одного домена на другой не нужно прописывать новый абсолютный адрес.



id-ссылки (якоря)

Якоря, или внутренние ссылки, создают переходы на различные разделы текущей веб-страницы, позволяя быстро перемещаться между разделами. Это оказывается очень удобным в случае, когда на странице слишком много текста. Внутренние ссылки также создаются при помощи элемента `<a>` с разницей в том, что атрибут `href` содержит имя указателя — так называемый якорь, а не URL-адрес. Перед именем указателя всегда ставится знак #.

Следующая разметка создаст оглавление с быстрыми переходами на соответствующие разделы:

```
<h1>Времена года</h1>
<h2>Оглавление</h2>
<a href="#p1">Лето</a> <!--создаём якорь, указав #id элемента-->
<a href="#p2">Осень</a>
<a href="#p3">Зима</a>
<a href="#p4">Весна</a>
<a href="../files/file2.html#p5">????</a>
<p id="p1">...</p> <!--добавляем соответствующий id элементу-->
<p id="p2">...</p>
<p id="p3">...</p>
<p id="p4">...</p>
```

Ссылки на телефонный номер, скайп или адрес электронной почты

У ссылок в HTML5 появились новые возможности — по клику можно не только переходить на другие страницы и скачивать файлы, но и совершать звонки на телефоны, отправлять сообщения или звонить по скайпу.

```
ссылка на телефонный номер
<a href="tel:+74951234567">+7 (495) 123-45-67</a>

ссылка на адрес электронной почты
<a href="mailto:example@mail.ru">example@mail.ru</a>

ссылка на скайп (позвонить)
<a href="skype:имя-пользователя?call">Skype</a>

ссылка на скайп (открыть чат)
<a href="skype:имя-пользователя?chat">Skype</a>

ссылка на скайп (добавить в список контактов)
<a href="skype:имя-пользователя?add">Skype</a>

ссылка на скайп (отправить файл)
<a href="skype:имя-пользователя?sendfile">Skype</a>
```

Атрибуты ссылок. Атрибут target

По умолчанию ресурсы, на которые ведут ссылки, открываются в том же окне. С помощью атрибута target можно переопределить это действие. Атрибут target может принимать следующее значение:

`_blank`: открытие html-документа в новом окне или вкладке браузера

```
<a href="https://mephi.ru" target="_blank">НИЯУ МИФИ</a>
```

HTML-изображения

HTML-изображения добавляются на веб-страницы с помощью элемента <img>. Использование графики делает веб-страницы визуально привлекательнее. Изображения помогают лучше передать суть и содержание веб-документа.

Элемент `<img>` представляет изображение и его резервный контент, который добавляется с помощью атрибута `alt`. Так как элемент `<img>` является строчным, то рекомендуется располагать его внутри блочного элемента, например, `<p>` или `<div>`.

Элемент `<img>` имеет обязательный атрибут `src`, значением которого является абсолютный или относительный путь к изображению:

```
<img src="image.png" alt="Пример">
```

Атрибут height задает высоту изображения в px. Синтаксис: height="300".
Атрибут width задает ширину изображения в px. Синтаксис: width="500".

Нормальный поток

Поток — одно из важнейших базовых понятий в вёрстке. Это принцип организации элементов на странице при отсутствии стилей: если мы напишем HTML и не напишем CSS, то отображение в браузере будет предсказуемо благодаря тому, что мы абсолютно точно знаем, как браузер располагает элементы в потоке.

Если вообще не применять никаких стилей, браузер формирует из элементов нормальный поток. Поведение блочных элементов в нормальном потоке отличается от поведения строчных.

Блочные элементы в нормальном потоке располагаются друг под другом, всегда занимая всю доступную ширину родителя. Высота блочного элемента по умолчанию равна высоте его содержимого. Три абзаца, идущие друг за другом в HTML, будут располагаться точно в таком же порядке и на странице.

Строчные элементы располагаются друг за другом, как слова в предложении. В зависимости от направления письма в конкретном языке элементы могут располагаться слева направо (например, в русском языке), справа налево (как, например, в иврите) и даже сверху вниз (как иероглифы и знаки слоговых азбук в японском вертикальном письме). Ширина и высота строчного элемента равна ширине и высоте содержимого. В отличие от блочного элемента, мы не можем управлять шириной и высотой строчного элемента через CSS. Несколько строчных элементов будут стремиться уместиться на одной строке, насколько хватает ширины родителя. Если ширины родителя не хватает, то лишний текст строчного элемента переносится на следующую строку.

Нормальный поток

В нормальном потоке:

- Вывод элементов на страницу браузера осуществляется в том порядке, в котором они следуют в HTML коде.
- В коде элементы вложены друг в друга, и чтобы это учитывать при выводе используют так называемые **воображаемые слои для отображения элементов**. При этом слой элемента тем выше (ближе к нам), чем данный элемент является более вложенным в коде, т.е. глубже расположен в нём.
- Положение элемента в потоке зависит от значения свойства display.

Все элементы по умолчанию находятся в нормальном потоке. Но это поведение можно поменять при помощи некоторых CSS-свойств. При изменении значений этих свойств элемент перестаёт взаимодействовать с остальными блоками в потоке. Говорят, что он «вышел из потока». Элементом «вне потока» может быть плавающий, абсолютно позиционированный или корневой элемент.

Тут нужно отметить, что элементы, вышедшие из потока, создают внутри себя своего рода мини-поток. Их дочерние элементы будут подчиняться правилам взаимодействия в потоке в пределах родителя.

Элементы «span» и «div»

Данные элементы представляют собою обобщенные, универсальные контейнеры, первый из которых отображается браузером как строчный элемент, а второй – как блочный. Другого логического смысла они не имеют. Оба элемента формируются парными тегами, соответственно, `<span>` (от англ. span – охватывать) и `<div>` (от англ. division – раздел).

Элемент «span» в основном применяется для выделения небольших фрагментов текста, рисунков или даже отдельных символов и букв, для которых применение тегов с каким-то логическим смыслом нецелесообразно. Например, если нам нужно выделить слово зеленым цветом, но оно не имеет особой смысловой нагрузки и важности, то нецелесообразно применять для этого, скажем, элемент «strong», который подразумевает важность своего содержимого и, кроме того, отображает свое содержимое полужирным шрифтом. А вот элемент «span» сам по себе свое содержимое никак не изменяет, но зато легко объединяется со стилями CSS через атрибуты style, class или id, позволяя делать с ним практически все, что пожелает разработчик.

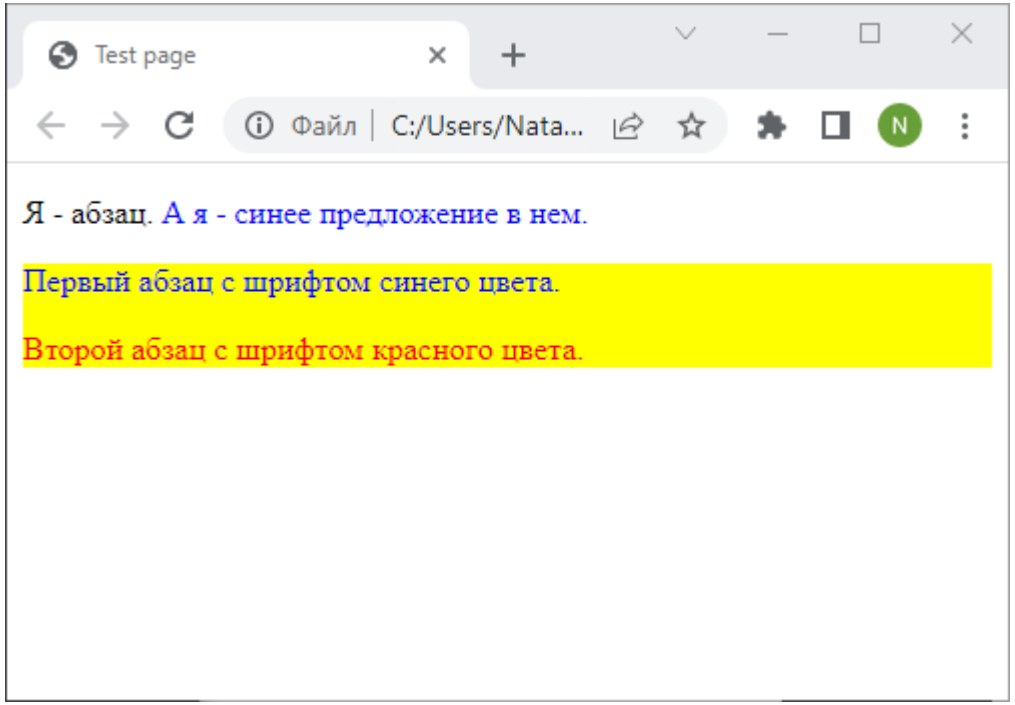
Элемент «div» является универсальным блочным элементом. Поэтому он применяется для изменения внешнего вида целых блоков веб-документа, для которых применение тегов с каким-то логическим смыслом нецелесообразно. Также, как и элемент «span», он свое содержимое никак не изменяет, но легко объединяется со стилями CSS через атрибуты `style, class` или `id`, являясь по сути аналогом элемента «span», но только среди блочных элементов

Элементы «span» и «div»

```
<body>
  <p title="Применение универсального элемента
«span»">
      Я - абзац.
      <span style="color: blue">А я – синее
предложение в нем.<span>
  </p>

  <div style="background-color: yellow">
    <p style="color: blue">
      Первый абзац с шрифтом синего цвета.
    </p>

    <p style="color: red">
      Второй абзац с шрифтом красного цвета.
    </p>
  </div>
</body>
```



Задание цвета

Существуют несколько основных способов представления цветов:

- В виде #123ABC. Представление в виде трёх пар шестнадцатеричных цифр, где каждая пара отвечает за свой цвет:
  - Две первые цифры — красный
  - Две в середине — зелёный
  - Две последние цифры — синий

  Возможно также краткое представление цвета в виде #ABC, что будет интерпретировано как #AABBCC.
- Представление ключевыми словами, например green, black. Во избежание случаев, когда указанное ключевое слово «не понимается» браузером, следует использовать лишь небольшой набор основных цветов, используемых во всех браузерах.
- В виде RGB(*,*,*), где «*» — числа от 0 до 255, обозначающих количество соответствующего цвета (красный, зелёный, синий) в получаемом цвете.
- Возможен и RGBA(*,*,*,*), где первые 3 «*» — компоненты цвета, задающиеся в диапазоне 0 до 255, а последняя «*» — уровень непрозрачности (альфа-канал), задающийся рациональными числами от 0 до 1.

Примеры цветов: https://www.w3schools.com/colors/colors_picker.asp

Справочники

- https://www.w3schools.com/html/default.asp
- https://developer.mozilla.org/ru/docs/Web/HTML
- https://html5css.ru/html/default.php

Дополнительные материалы

Адаптивные (отзывчивые) изображения

Технологии отзывчивых изображений были реализованы недавно для решения проблем, связанных с размерами изображения на экранах с различным разрешением. Они позволяют вам предоставить браузеру несколько изображений, каждое из которых отображает одно и то же, но содержит разное количество пикселей (resolution switching), или разные изображения с отдельными областями основного изображения (art direction).

```
<img srcset="img-320w.jpg 320w,
             img-480w.jpg 480w,
             img-800w.jpg 800w"
     sizes="(max-width: 320px) 280px,
            (max-width: 480px) 440px,
            800px"
     src="img-800w.jpg" alt="Example">
```

strset
название_изображения пробел реальная_ширина_изображения w
`img-320w.jpg 320w`

sizes определяет перечень медиавыражений и указывает предпочтительную ширину изображения
медиа-условие пробел ширина_слота
`(max-width: 480px) 440px`

Браузер сделает следующее: посмотрит на ширину экрана устройства, затем попытается определить подходящее медиа-условие из списка в атрибуте sizes, затем посмотрит на размер слота к этому медиавыражению и загрузит изображение из списка из srcset, которое имеет тот же размер, что и выбранный слот, или, если такого нет, то первое изображение, которое больше размера выбранного слота.

Карта изображения

Элемент <map> служит для представления графического изображения в виде карты с активными областями. Активные области определяются по изменению вида курсора мыши при наведении. Щелкая мышью на активных областях, пользователь может переходить к связанным документам.

Для элемента доступен атрибут name, который задает имя карты. Значение атрибут name для элемента <map> должно соответствовать имени в атрибуте usemap элемента <img>.

Элемент <map> содержит ряд элементов <area>, определяющих интерактивные области в изображении карты.

```
<img src="flowers_foto.png" alt="flowers_foto" width="680" height="383" usemap="#flowers">

<map name="flowers">
   <area shape="circle" coords="70,164,50" href="https://ru.wikipedia.org/wiki/Гербера"
alt="gerbera" target="_blank">
   <area shape="poly" coords="191,13,240,98,143,98,191,13"
href="https://ru.wikipedia.org/wiki/Гиацинт" alt="hyacinth" target="_blank">
   <area shape="circle" coords="318,93,50" href="https://ru.wikipedia.org/wiki/Ромашка"
alt="camomiles" target="_blank">
   <area shape="circle" coords="425,129,45"
href="https://ru.wikipedia.org/wiki/Нарцисс_(растение)" alt="narcissus" target="_blank">
   <area shape="rect" coords="480,3,572,89" href="https://ru.wikipedia.org/wiki/Тюльпан"
alt="tulip" target="_blank">
</map>
```

Что такое HTML-валидация и зачем она нужна

Валидация — это проверка HTML-кода: соответствует ли он общепринятым правилам и нет ли в нём ошибок. Хороший код называют валидным. Он быстро загружается, браузеры корректно обрабатывают его синтаксис, а поисковикам кажется, что сайт просто замечательный, и его позиции растут.

Чем опасны ошибки в разметке

Ошибки портят впечатление о сайте, мешают ему нормально работать и продвигаться в поисковых системах. Вот конкретные примеры:

- Страницы загружаются слишком медленно.
- Посетители видят только часть текстов и иллюстраций. А значит, контент для них почти бесполезный — и посетитель, скорее всего, уйдет к конкуренту.
- Если поисковый робот запнется хотя бы об одну ошибку — битую ссылку, пропущенный знак в коде или неправильную верстку, — он может не проиндексировать страницу.
- Сайт некорректно отображается на разных устройствах. Он может хорошо выглядеть на экране компьютера, а с телефона не будут видны тексты, кнопки или весь контент «съедет».
- На сайте много скрытой рекламы и вирусов, а разработчик или владелец сайта не в курсе.

Почему важно проверять наличие HTML-ошибок?

- Ошибки в коде HTML могут быть критическими и несущественными, которые не ведут к серьезным потерям. Что касается критических, то одни из них отрицательно сказываются на функционировании сайта, а другие — на работе поисковых систем.

- Современные браузеры автоматически исправляют 99% критических ошибок при загрузке сайта. Однако некоторые из них браузер исправить не может. Например, если тег <a> для создания ссылки не содержит адреса, то браузер не сможет определить, куда её направить. Или в теге <img> для размещения картинки не указан путь к ней, тогда браузер не сможет её подгрузить. Наличие таких ошибок в коде может привести к серьезным последствиям — например, не загрузятся фото товара или не будет работать корзина.

- Поисковые системы также автоматически исправляют часть HTML-ошибок, но у них возникает следующая проблема: если браузеры в состоянии потратить несколько секунд на исправление ошибок, то у поисковых роботов нет такой возможности. Им приходится сканировать сотни миллиардов страниц ежемесячно, поэтому боты не могут тратить время на устранение всех ошибок. Некоторые из них поисковые системы игнорируют, а также могут не включать в индекс содержащие их страницы или проиндексировать только часть контента таких страниц.

- Веб-мастера и пользователи просматривают сайты в браузере, где большая часть HTML-ошибок исправляется автоматически, и поэтому не придают им большого значения. Зачастую даже разработчики не исправляют элементарных грубых ошибок в разметке. Это приводит к тому, что критические для поисковых систем ошибки остаются на сайтах и могут стать причиной неправильной индексации страниц. В результате бюджеты на продвижение будут потрачены неэффективно, а источник проблемы так и остается неустановленным.

Валидатор W3C

Валидатор — это сервис проверки валидности HTML, который быстро находит ошибки в коде и помогает их исправить. Подобных сервисов несколько, но разработчики часто используют официальный валидатор W3C. В нём можно найти ошибки тремя способами: указать URL сайта, загрузить HTML-документ или HTML-код.

https://validator.w3.org/

Сервис проверяет синтаксис кода: например, верно ли указаны тип документа и кодировка, нет ли в коде пропущенных элементов. Также происходит проверка соответствию DTD (Document Type Definition) — валидатор смотрит, соответствует ли код типу документа.

Валидатор делит проблемные части кода на предупреждения и ошибки. Для удобства они выделены разными цветами, чтобы сразу было понятно, каким проблемам стоит уделить особое внимание.

Предупреждения

Незначительные проблемы. Сайт, скорее всего, не сломается, но всё равно не соответствует спецификациям W3C.

Это означает, что при прочих равных сайты конкурентов будут лучше работать на разных устройствах и эффективнее продвигаться в поисковых системах.

Ошибки

Серьезные проблемы. Есть риск, что сайт будет отображаться некорректно, часть контента окажется скрытой или пользователь вообще не сможет просматривать страницы.

Ошибки следует исправлять в первую очередь, но хорошим тоном будет избавиться вообще от всех проблем: это поможет сайту работать нормально.