Реализация дизайна. Часть 1. HTML и CSS

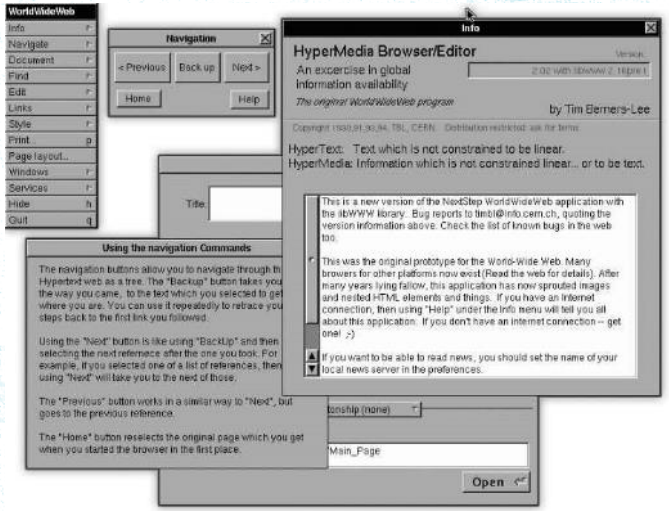
Курс «Веб-дизайн»

Язык HTML. Гипертекст

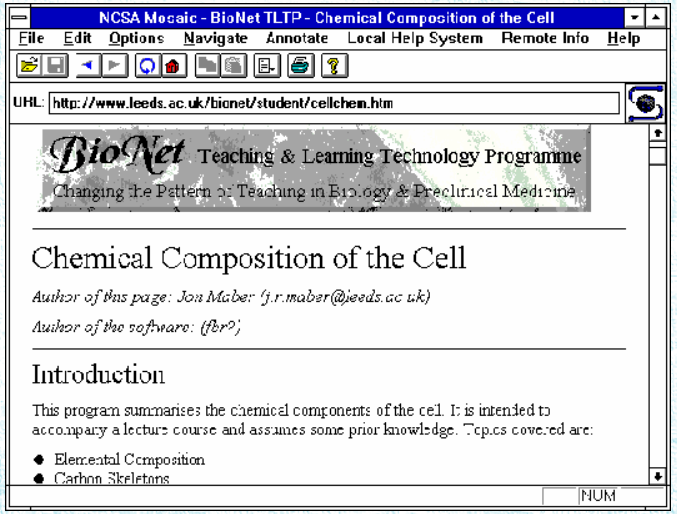
- ***Язык HTML (Hyper Text Markup Language)*** – язык разметки гипертекста.
- ***Гипертекст (hypertext)*** – технология хранения и обработки текстовых документов, которая обеспечивает возможности установления и поддержки связей между документами и/или их отдельными фрагментами и навигации пользователя в такой структуре. Термин введен Т. Нельсоном в 1965 г.
- ***Гиперссылка*** – бинарная связь между документами и/или их составными частями.

История HTML

- **1986 г.** Стандарт языка SGML
- **1990 г.** Первый браузер - WorldWideWeb (позже был переименован в Nexus)
- **1991 г.** Тим Бернерс Ли. «HTML Tags» (описано около 24 тегов для разметки веб-страниц)
- **1993 г.** Первый веб-браузер под Microsoft Windows с графическим интерфейсом
- **1994 г.** Создание W3C
- **1995 г.** HTML 2.0. Разработан IETF (Internet Engineering Task Force).
- **1997 г.** HTML 3.2
- **1999 г.** HTML 4.01.



Браузер WorldWideWeb



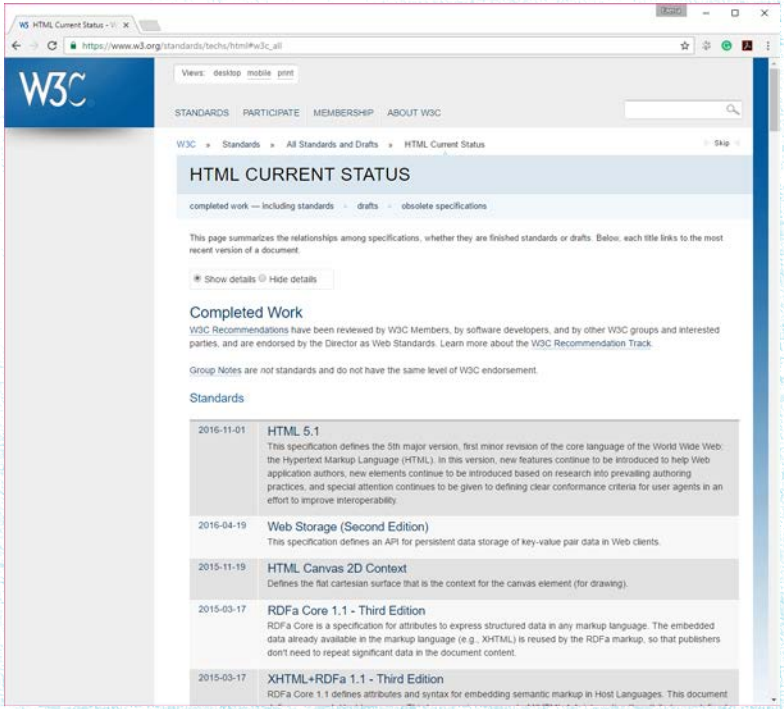
Браузер Mosaic for Windows

Консорциум Всемирной паутины (W3C)

- **Консорциум Всемирной паутины (W3C)** - организация, разрабатывающая и внедряющая технологические стандарты для Всемирной паутины.
- Консорциум возглавляет Тимоти Джон Бернерс-Ли, автор множества разработок в области информационных технологий.
- Консорциум был создан как консультативный орган для лидеров компьютерной индустрии.

История HTML

- **2000 г.** XHTML 1.0 (eXtensible Hyper Text Markup Language). Разница между HTML 4.01 и XHTML - лишь в более строгих правилах синтаксиса.
- **2001 г.** XHTML 1.1. Не поддерживался Internet Explorer.
- **2002 г.** XHTML 2.0 (первый черновой вариант). Нет обратной совместимости с существующим веб-контентом и предыдущими версиями HTML.
- **2004 г.** Работа над HTML5. Opera, Apple и Mozilla сформировали свою рабочую группу WHATWG (Web Hypertext Application Technology Working Group).
- **2006 г.** Работа W3C над HTML 5.
- **2009 г.** Официальный отказ W3C от дальнейшей работы над XHTML 2.0.
- **2016 г.** HTML 5.1



Сайт W3C

Тег, элемент, атрибут

- ***Теги (tags)*** – специальные символы, позволяющие отличать в документе описание разметки от описания данных.
  - Пример: <h1>, </h1>, <p>, <img />.

- ***Элемент*** – это тэги в совокупности с их содержанием. Элемент состоит из открывающего тега, содержимого и закрывающего тега.
  - Пример: <h1>Евгений Онегин</h1>.

- ***Атрибут*** используется при определении элемента, чтобы задать какие-либо параметры, уточняющие характеристики данного элемента. Атрибуты состоят из пары "название" = "значение", задаются, как правило, внутри начального тега.
  - Пример: Евгений Онегин

Из чего состоит веб-страница

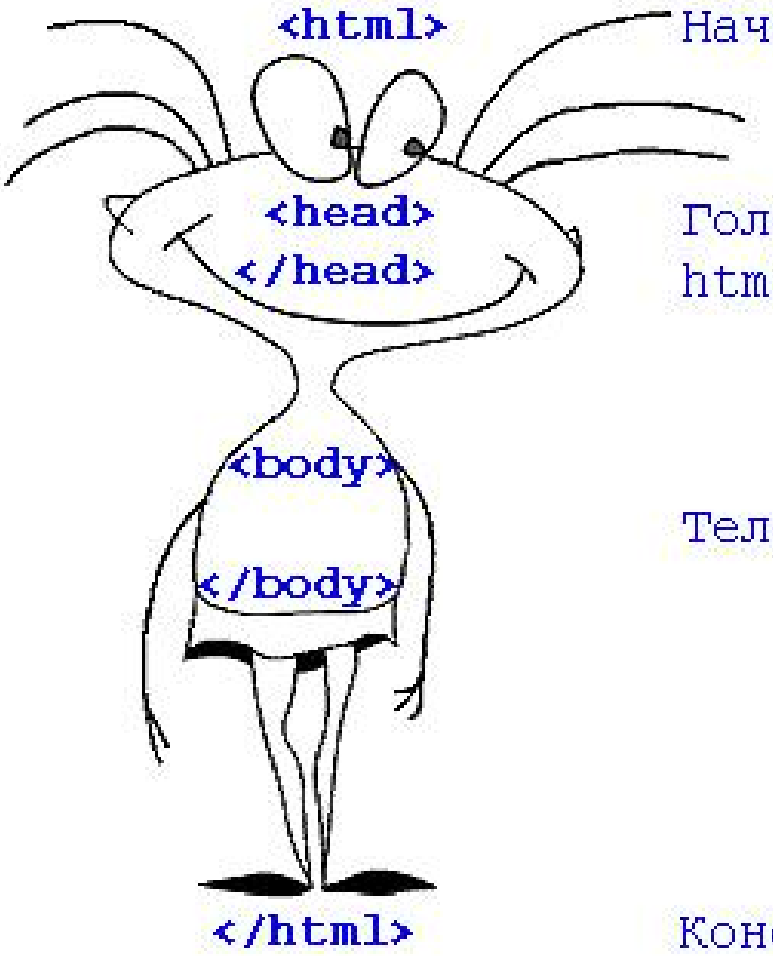
- Текст
- Списки
- Таблицы
- Картинки
- Ссылки
- Мультимедиа
- Формы, кнопки
- Фреймы (устаревший элемент)

HTML5

В HTML5 появилось множество семантических элементов, а также тегов, позволяющих вставлять аудио и видео на сайт:

- <article>
- <aside>
- <audio>
- <canvas>
- <command>
- <datalist>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <hgroup>
- <keygen>
- <main>
- <mark>
- <menu>
- <meter>
- <nav>
- <output>
- <progress>
- <rp>
- <rt>
- <ruby>
- <section>
- <source>
- <summary>
- <time>
- <video>
- <wbr>

Структура HTML-документа



Начало html-документа

Головная часть html-документа

Тело html-документа

Конец html-документа

Элемент DOCTYPE

- DOCTYPE пишется в первой строке html-файла
- Используется для задания версии HTML
  - HTML 4.01:

```
<!DOCTYPE HTML PUBLIC
    "-//W3C//DTD HTML 4.01//EN“
    "http://www.w3.org/TR/html4/strict.dtd">
```

  - HTML5:

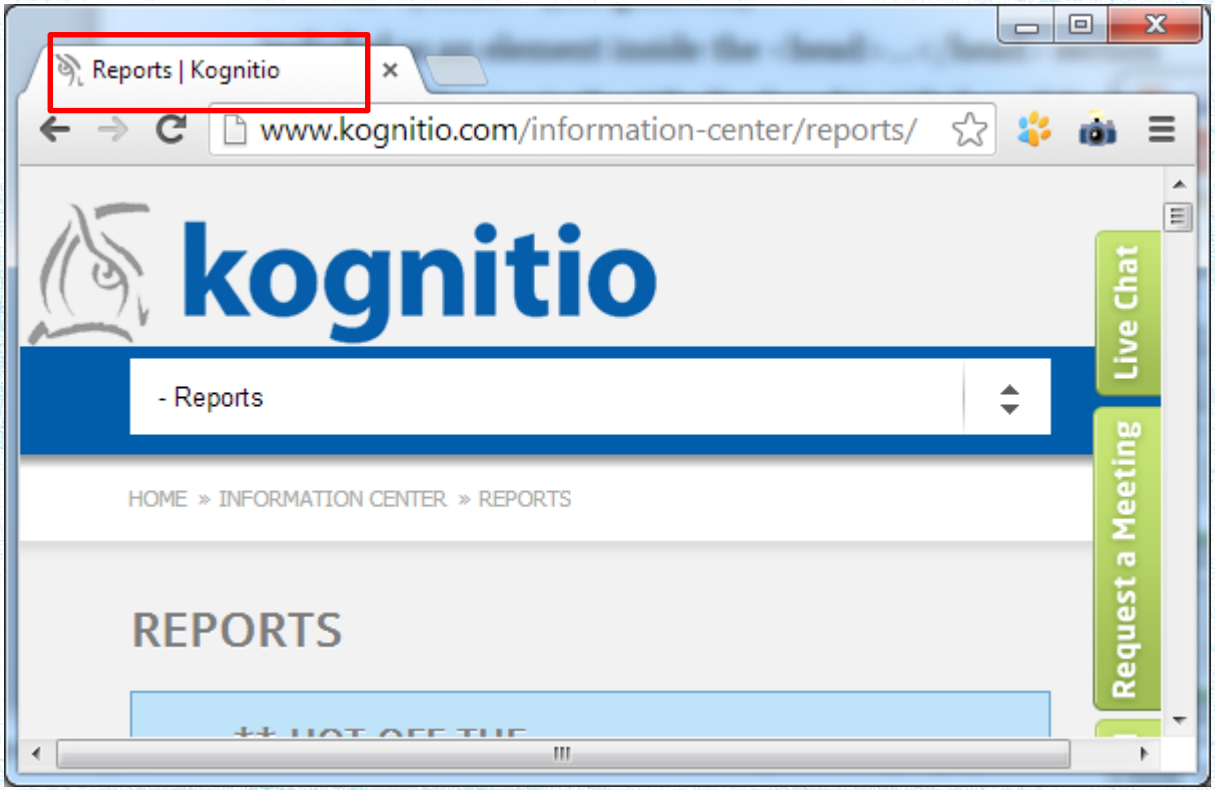
```
<!DOCTYPE html>
```

Элемент DOCTYPE. Пример

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
    </body>
</html>
```

Содержимое head: элемент title

```
<head>
  <title>Заголовок</title>
</head>
```



Содержимое head: элемент meta

- Определяет метатеги, которые используются для хранения информации предназначенной для браузеров и поисковых систем.
- Каждый метатег содержит имя параметра и его значение. Примеры:
  - <meta name ="**keywords**" content="**Ключевые слова**">
  - <meta http-equiv="**content-type**" content="**text/html; charset=Windows-1251**">
  - <meta http-equiv="**Refresh**" content="**30; URL=http://rambler.ru**">

Работа с текстом в HTML

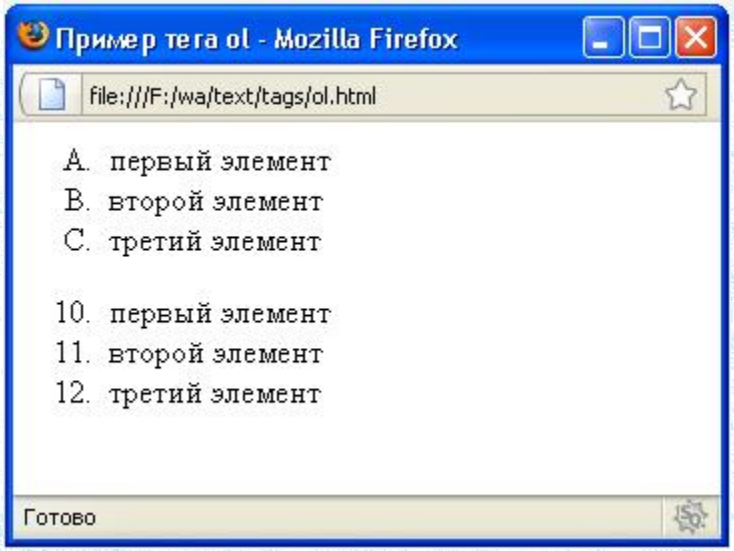
- Элементы:
  - ** **абзац** **
  - **
** (переход на новую строку)
  - **<h1>**Заголовок 1-го уровня**</h1>**
  - **<h2>**Заголовок 2-го уровня**</h2>** и т.д. до 6-го
  - ****Жирное начертание текста****
  - ****Курсивное начертание текста****
  - **^{**Верхний индекс**}**
  - **_{**Текст**}**

Списки в HTML

- Элемент **ol** создает ***нумерованный*** список.
- Синтаксис

```
<ol>
<li>первый элемент</li>
<li>второй элемент</li>
</ol>
```

- Атрибуты:
  - type - определяет стиль нумерации элементов списка. Может принимать значения:
    - 1 - арабские цифры
    - A - заглавные латинские буквы
    - a - строчные латинские буквы
    - I - заглавные римские цифры
    - i - строчные римские цифры
  - start - устанавливает начальное число или букву для элементов списка.

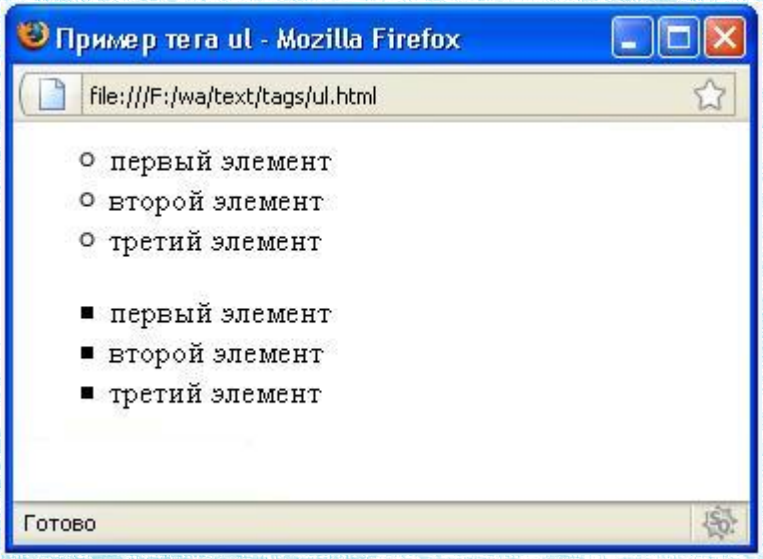


Списки в HTML

- Элемент **ul** создает ***маркированный*** список.
- Синтаксис

```
<ul>
<li>первый элемент</li>
<li>второй элемент</li>
</ul>
```

- Атрибуты:
- type - определяет вид маркеров элементов списка. Может принимать значения:
  - disc - круг
  - circle - окружность
  - square - квадрат



Таблицы в HTML

- **<table></table>** Все содержимое таблицы должно помещаться между этими двумя тегами.
- **<tr></tr>** Между этими тегами помещается строка таблицы.
- **<td></td>** Между этими тегами помещается колонка таблицы.
- **<td colspan="...">** Количество столбцев, объединенных в одной ячейке.
- **<td rowspan="...">** Количество строк, объединенных в одной ячейке.
- Пример:

```
<table>
<tr>
<td>Первая колонка первой строки</td>
<td>Вторая колонка первой строки </td>
</tr>
<tr>
<td>Первая колонка второй строки</td>
<td>Вторая колонка второй строки </td>
</tr>
</table>
```

Картинки в HTML

- Элемент **img**
- Атрибуты:
  - **src="string"** – URL к файлу с картинкой
  - **alt="string"** – альтернативный текст, который видит пользователь, если отобразить картинку не удается
  - **height="string"** – высота картинки (не обязательный)
  - **width="string"** – ширина картинки (не обязательный)
  - **title="string"** – надпись, которую видит пользователь при наведении мышью на картинку
  - др.

Картинки в HTML

- Примеры:
  - `<img src="my_photo.jpg">`
  - `<img src="http://mysite.ru/my_photo.jpg">`
  - `<img src="my_photo.jpg" alt="Иванова Е.В.">`

Гиперссылки в HTML

- Для создания гиперссылок используется элемент a:
  Текст ссылки
- Примеры:
  - ****Click here to continue****
  - ****Click here to continue****
  - ****See the box here.****
  - ****My email****

Гиперссылки в HTML

- Атрибут target

  _**blank** — загружает страницу в новое окно браузера;

  _**self** — загружает страницу в текущее окно;

- Пример:
  - **** Click here****

Изображение в качестве ссылки

```
<a href="sample.html">
  <img src="images/sample.gif" alt="Пример">
</a>
```

Закладки в HTML

- Для создания закладок также используется элемент a:
  Текст закладки
- Ссылка на закладку:
  Текст ссылки
- Пример:

  ****Перейти к заголовку****

  ****Заголовок****

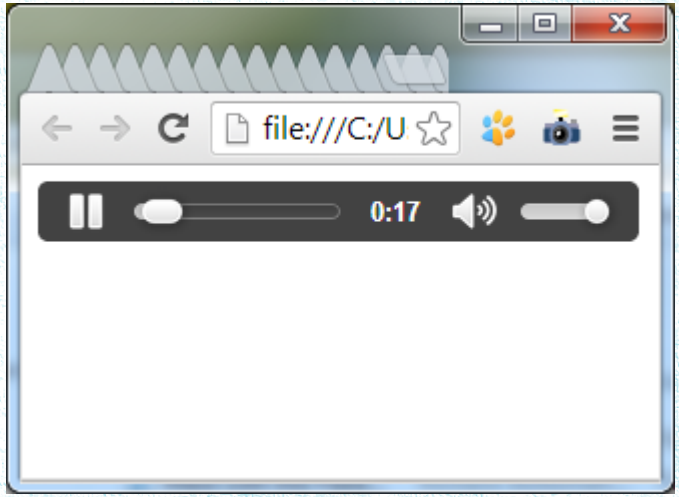
Мультимедиа в HTML

- HTML 4.01:
  - **<embed>** и **<object>** - используются для загрузки и отображения объектов (например, видеофайлов, флэш-роликов, некоторых звуковых файлов и т.д.), которые исходно браузер не понимает. Как правило, такие объекты требуют подключения к браузеру специального модуля, который называется плагин, или запуска вспомогательной программы.
  - Пример:
    <embed src="images/mouse.swf" width="400" height="300
    "
    type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplay
    er">

Мультимедиа в HTML

- HTML5:
  - **<audio>** - добавляет, воспроизводит и управляет настройками аудиозаписи на веб-странице. Путь к файлу задается через атрибут src или вложенный тег <source>. Внутри контейнера <audio> можно написать текст, который будет выводиться в браузерах, не работающих с этим тегом.
  - Пример:

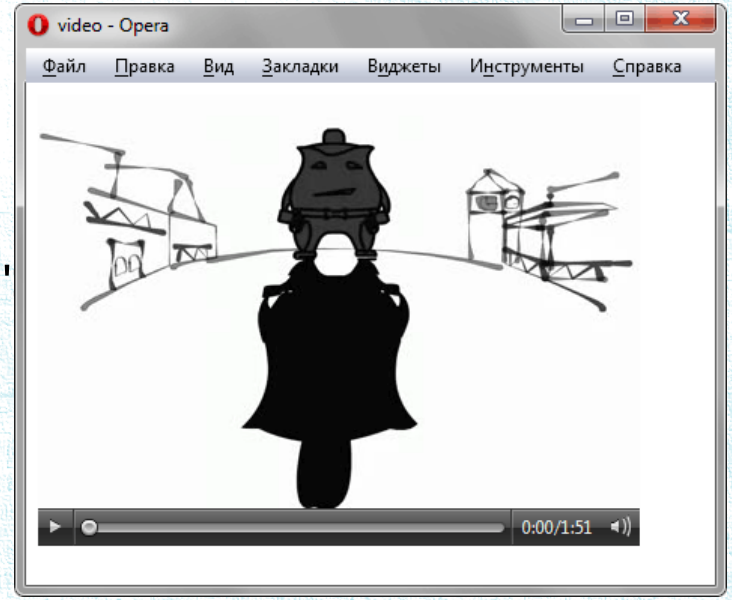
```
<audio controls>
    <source src="audio/music.ogg"
    type="audio/ogg; codecs=vorbis">
    Тег audio не поддерживается
    вашим браузером.
</audio>
```



Мультимедиа в HTML

- HTML5:
  - **<video>** - добавляет, воспроизводит и управляет настройками видеоролика на веб-странице. Путь к файлу задается через атрибут src или вложенный тег <source>.
  - Пример:

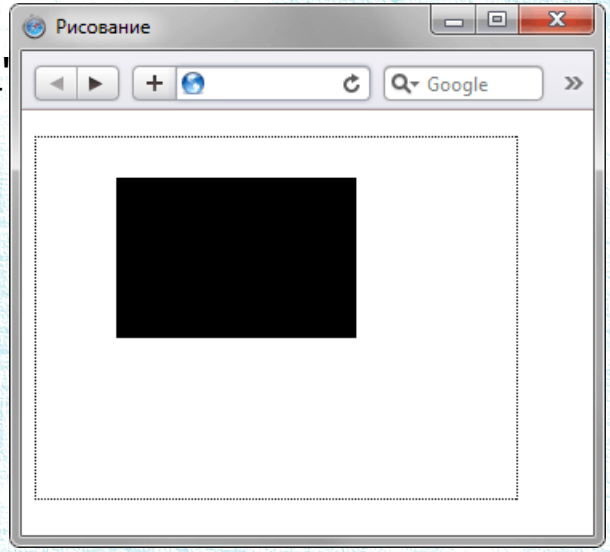
<video width="400" height="300"
controls="controls"
poster="video/duel.jpg"**>**
<source src="video/duel.ogv"
type='video/ogg; codecs="theora, vorbis"'
Тег video не поддерживается
вашим браузером.
</video>



Мультимедиа в HTML

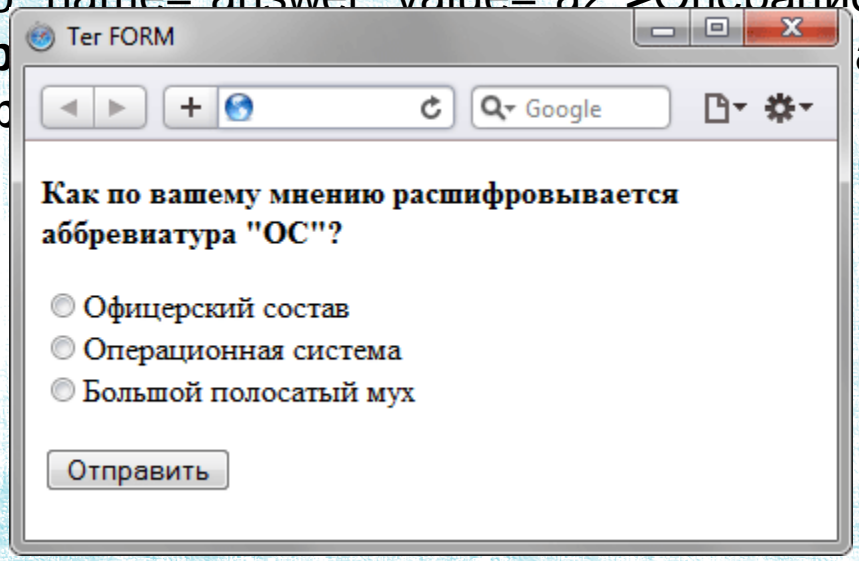
- HTML5:
  - **<canvas>** - растровый холст, который может быть использован для отображения диаграмм, игровой графики или изображений на лету. Холст это прямоугольная область на вашей странице, где с помощью JavaScript можно рисовать.
  - Пример:

```
<canvas id="a" width="300" height="225"></canvas>
<script>
    var b_canvas = document.getElementById("a"
    var b_context = b_canvas.getContext("2d");
    b_context.fillRect(50, 25, 150, 100);
</script>
```



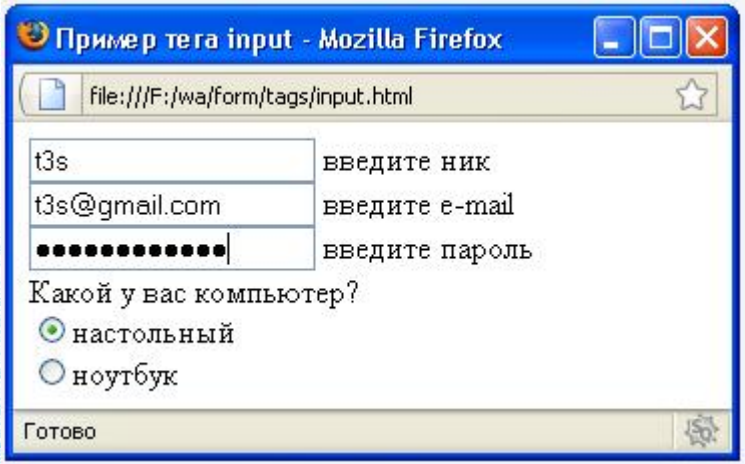
Формы и элементы управления в HTML

- Форма (элемент form) предназначена для обмена данными между пользователем и сервером.
- **Пример: <form** action="handler.php"**>**
  ** **Как по вашему мнению расшифровывается аббревиатура "ОС"?** **
  ** <input** type="radio" name="answer" value="a1"**>**Офицерский состав**
**
  <input type="radio" name="answer" value="a2"**>**Операционная система**
 <inp** a3"**>**Большой полосатый мух** **
  </form>

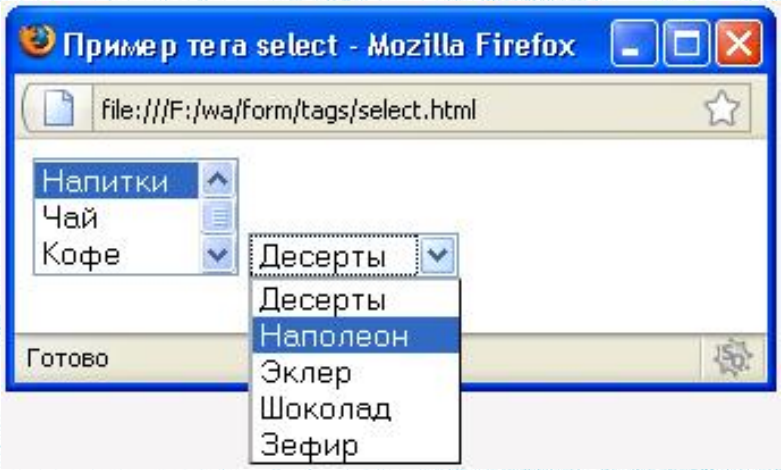


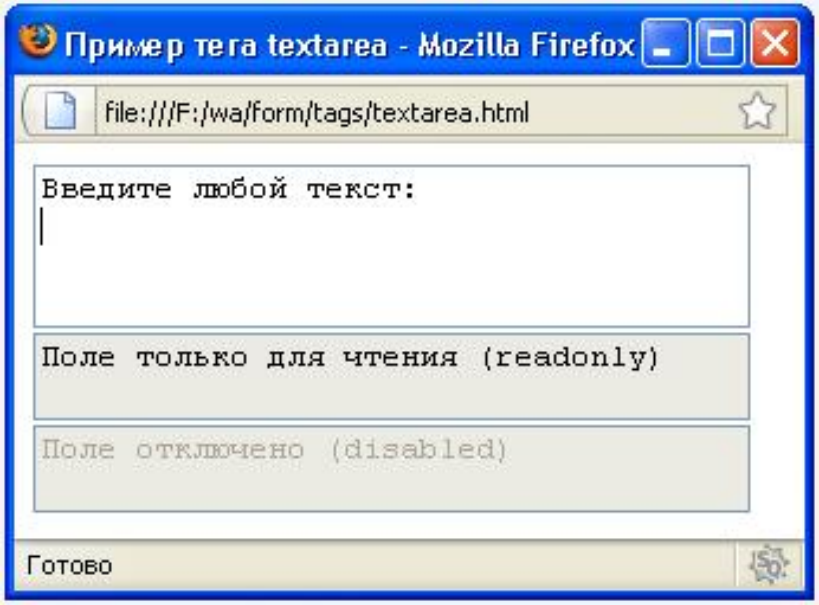
Текстовые поля

- Элемент **input** используется в основном для областей ввода данных (текстовых полей). С его помощью можно вводить практически любые данные, что делает его достаточно универсальным элементом формы.
- **Атрибут type** задает тип области (элемента) ввода, может принимать значения:
  - text - текстовая строка
  - textarea - текстовая поле из нескольких строк
  - file - строка для ввода имени файла, который пересылается на сервер.
  - password - строка для ввода пароля (введенные символы обычно заменяются звездочками)
  - radiobutton - переключатель с возможностью выбора одного значения
  - checkbox - переключатель с возможностью выбора нескольких значений
  - hidden - скрытое поле
  - button - кнопка
  - image - кнопка с изображением
  - submit - кнопка для отправки данных
  - reset - кнопка для очистки формы



Другие элементы формы





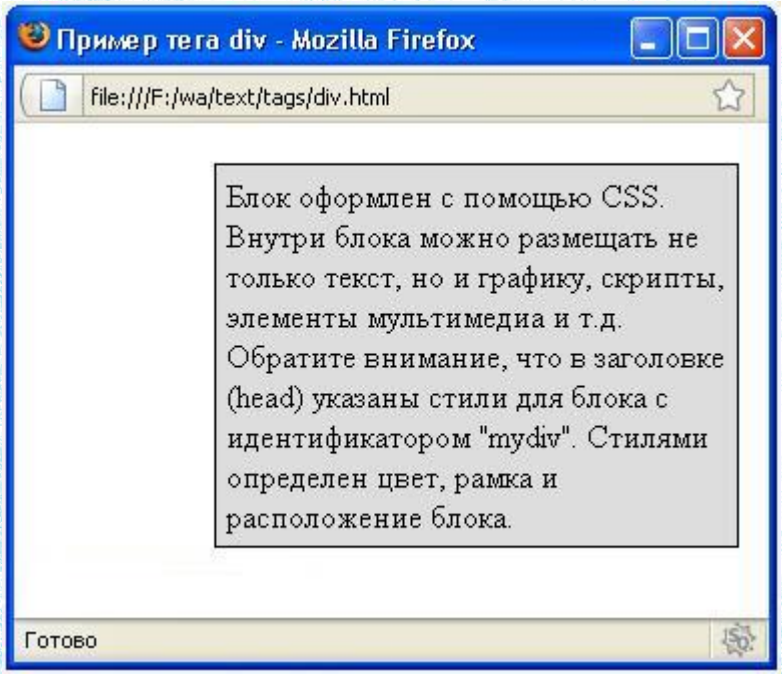
Сущности в HTML

- ***Сущность*** - ссылка на символ из набора символов

	неразрывный пробел		
£	символ фунта	&pound;	&#163;
§	знак параграфа	&sect;	&#167;
©	знак охраны авторского права	&copy;	&#169;
«	направленная влево двойная угловая кавычка	&laquo;	&#171;
»	направленная вправо двойная угловая кавычка	&raquo;	&#187;
®	знак правовой охраны товарного знака не путать с ™ — символ товарного знака	&reg;	&#174;
°	знак градуса	&deg;	&#176;
±	плюс-минус	&plusmn;	&#177;

Контейнеры

- DIV – блочный элемент общего назначения
- SPAN – строчный элемент общего назначения



Комментарии в HTML

- **<!-- comments here -->**

Cascading Style Sheets

- *CSS* (*Cascading Style Sheets)* – язык управления внешним видом HTML-документов.
- CSS используется для задания цветов, шрифтов, расположения и других аспектов представления документа.
- Разработчик спецификаций CSS W3C (http://www.w3.org)
- Текущая версия – CSS3. Начата работа над CSS4.

Пример использования CSS

	Пример без использования стилей	**Пример со стилями**

Стиль		`b {`
		`  color: red;`
		`}`

HTML-код	`<b><font color="red">Внимание!</font></b> Не переходите улицу на <b><font color="red">красный</font></b> сигнал светофора!<b> <font color="red">Это опасно!</font> </b>`	`<b>Внимание!</b> Не переходите улицу на <b>красный</b> сигнал светофора!<b>Это опасно!</b>`
Отображение	**Внимание!** Не переходите улицу на **красный** сигнал светофора! **Это опасно!**	**Внимание!** Не переходите улицу на **красный** сигнал светофора! **Это опасно!**

Основные понятия

- *Стиль CSS* = селектор + декларации форматирования
- *Селектор* определяет форматируемую порцию документа (например, абзац, картинка и др.)
- *Декларация форматирования (правило)* – набор пар "свойство-значение", позволяющий задать цвет, шрифт абзаца, рамку картинки и др.

```
селектор, селектор {
  свойство: значение;
  свойство: значение;
  свойство: значение;
}
```

Правила

Селекторы CSS

- Селектором могут быть:
  - Элементы HTML: <P>, ...
  - Классы: <P class="myclass">, ...
  - Идентификаторы: <P id="myid">, ...
  - Смешанный селектор
  - Контекстный селектор

Селектор по элементу

Тег { свойство1: значение; ... }

	Пример без использования стилей	Пример со стилями

Стиль		`b {`
		`  color: red;`
		`}`

HTML-код	`<b>< font color="red">Внимание! </font></b> Не переходите улицу на <b>< font color="red">красный </font></b> сигнал светофора!<b> <font color="red">Это опасно!</font> </b>`	`<b>Внимание!</b> Не переходите улицу на <b>красный</b> сигнал светофора!<b>Это опасно!</b>`
Отображение	**Внимание!** Не переходите улицу на **красный** сигнал светофора! **Это опасно!**	**Внимание!** Не переходите улицу на **красный** сигнал светофора! **Это опасно!**

Селектор по классу

```
Тег.Имя класса { свойство1: значение; ... }
```

	Пример использования классов
Стиль	`.red { color: red; }`
HTML-код	`<b class="red">Внимание!</b> Не переходите улицу на <b>красный</b> сигнал светофора!<b class="red">Это опасно!</b>`
Отображение	**Внимание!** Не переходите улицу на **красный** сигнал светофора! **Это опасно!**

Селектор по идентификатору

```
#Имя идентификатора { свойство1: значение; ... }
```

	Пример использования идентификатора

Стиль	`#red {`
	`  color: red;`
	`}`

HTML-код	`<b>Внимание!</b> Не переходите улицу на <b id="red">красный</b> сигнал светофора!<b>Это опасно!</b>`
Отображение	**Внимание!** Не переходите улицу на **красный** сигнал светофора! **Это опасно!**

Смешанный селектор

	Пример использования классов
Стиль	`.red { color: red; }` `b.red#red { font-size: 120%; }`
HTML-код	`<b class="red" id="red">Внимание!</b> Не переходите улицу на <b>красный</b> сигнал светофора!<b class="red">Это опасно!</b>`
Отображение	**Внимание!** Не переходите улицу на **красный** сигнал светофора! **Это опасно!**

Контекстный селектор

Тег1 Тег2 { ... }

Стиль будет применяться к Тегу2 когда он размещается внутри Тега1

	Пример использования классов
Стиль	`p b { color: red; }`
HTML-код	`<p class="my"><b>Внимание!</b></p> Не переходите улицу на <b>красный</b> сигнал светофора!<b>Это опасно!</b>`
Отображение	**Внимание!** Не переходите улицу на **красный** сигнал светофора! **Это опасно!**

Соседний селектор

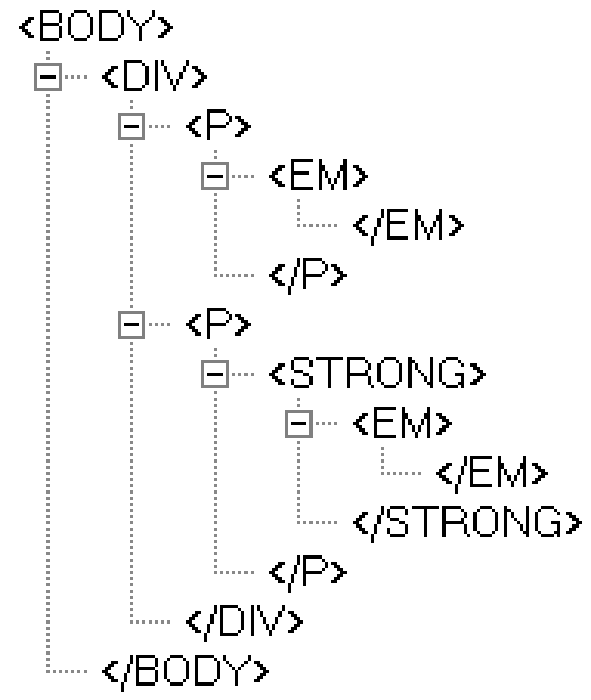
```
<p>Lorem <b>ipsum </b> dolor sit amet, <i>consectetuer</i>
adipiscing <tt>elit</tt>.</p>
```

Соседними здесь являются теги **<b>** и **<i>**, а также **<i>** и **<tt>**. При этом **<b>** и **<tt>** к соседним элементам не относятся из-за того, что между ними расположен контейнер **<i>**.

Селектор 1 + Селектор 2 { Описание правил стиля }

Стиль будет применяться к Селектору 2, но только в том случае, если он является соседним для Селектора 1 и следует сразу после него.

Дочерние селекторы



Например, дочерние элементы для элемента DIV – элементы P.

`Селектор 1 > Селектор 2 { … }`

Стиль применяется к Селектору 2, но только в том случае, если он является дочерним для Селектора 1.

Селекторы атрибутов

```
[атрибут] { Описание правил стиля }
Селектор[атрибут] { Описание правил стиля }
```

- Стиль применяется к тем элементам, внутри которых добавлен указанный атрибут.

Универсальный селектор

```
* { Описание правил стиля }
```

- Используется для обозначения любого селектора.
- Записи *.class и .class являются идентичными по своему результату.

Псевдоклассы

- ***Псевдоклассы*** определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также положение в дереве документа.
  - Примером такого состояния служит текстовая ссылка, которая меняет свой цвет при наведении на неё курсора мыши.

Селектор:Псевдокласс { Описание правил стиля }

Пример: псевдокласс :hover активизируется, когда курсор мыши находится в пределах элемента, но щелчка по нему не происходит.

```
A: hover {color: red; text-decoration: none;}
```

Группирование

- При создании стиля для сайта, когда одновременно используется множество селекторов, возможно появление повторяющихся стилевых правил. Чтобы не повторять дважды одни и те же элементы, их можно сгруппировать для удобства представления и сокращения кода.

```
Селектор 1, Селектор 2, ... Селектор N { … }
```

правила стиля применяются ко всем селекторам, перечисленным в одной группе.

Комментарии в CSS

- Комментарии в CSS оформляются символами
  /* … */

Включение стилей в документ

- Стили на уровне документа
- Внешние стили
- Встроенные стили (inline-стили)

Стили на уровне документа

```
<html>
<head>
<title>Документ со стилями, определенными на уровне документа.</title>
<style>
  h1 {
    color: blue;
  }
  h2 {
    color: red;
  }
</style>
      <body>
        <h2>Красный заголовок</h2>
        <h1>Синий заголовок</h1>
      </body>
</html>
```

Внешние стили

Стили располагаются в отдельном файле *.css (например, style.css).

Index.html:

```
<html>
<head>
   <link rel="stylesheet" type="text/css"
         href="http://somehost/style.css">
</head>
...
</html>
```

Style.css:

```
P {
   color: red;
}
.myclass {
   margin: 0px;
   padding: 10px 2px;
   border-style: none;
}
```

Встроенные стили (inline-стили)

- *Стиль можно встраивать в html-документ через* атрибут *style* со списком свойств и их значений.
- Пример:

```
<h1 style="color: blue; font-style: italic">
   Title
</h1>
```

Правила каскадирования CSS

- Стандарт CSS определяет приоритеты, в порядке которых применяются правила стилей, если для какого-то элемента подходят несколько правил одновременно. Это называется «***каскадом***», в котором для правил рассчитываются приоритеты или «веса», что делает результаты предсказуемыми.
- Таблицы стилей имеют три источника:
  - авторские стиль,
  - пользовательские стили,
  - стили браузера.

Параметр !important

- !important позволяет повысить приоритет декларации форматирования.
- Пример использования:
  - BODY {
    color: #000 !important;
    }

Правила каскадирования CSS

- Алгоритм каскадирования:
  1. Найти все декларации форматирования, применяемые к данному элементу/атрибуту.

ПРИМЕР

Определить цвет текста: <P class=”color”>text</P>

Авторские стили:

P#color_red {color: red;}

P {color: green;}

P.color {color: blue}

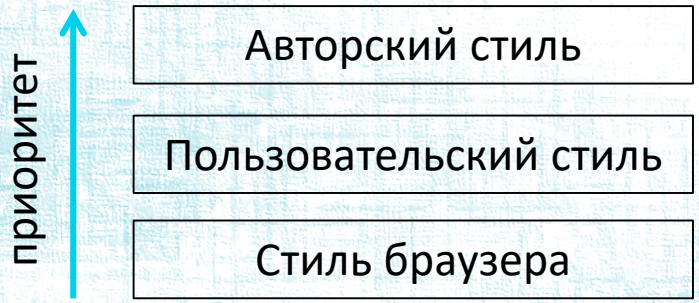
Браузерный стиль:

P {color: black}

Результат шага 1: P {color: green;}, P.color {color: blue}, P {color: black}

Правила каскадирования CSS

- Алгоритм каскадирования:
  2. Первичная сортировка объявлений выполняется по приоритету и источнику.



ПРИМЕР

Результат шага 1: P {color: green;}, P.color {color: blue}, P {color: black}

Результат шага 2: P {color: green;}, P.color {color: blue}, P {color: black}

Правила каскадирования CSS

- Алгоритм каскадирования:
  3. Вторичная сортировка выполняется по специфичности селектора.

Специфичность селектора

- Если декларация содержится в атрибуте style, выбирается значение 1, в противном случае – 0 (= a).
- подсчитывается число атрибутов ID в данном селекторе (= b);
- подсчитывается число классов и псевдоклассов в данном селекторе (= c);
- подсчитывается число имен элементов в данном селекторе (= d);
  - Объединение этих значений в последовательность a-b-c-d (в системе счисления с большим основанием) определяет **специфичность**.

Основание системы счисления определяется самым большим числом в любой из категорий. Например, если a=14, можно использовать шестнадцатеричную систему. Если a=17 (что маловероятно), потребуется система счисления по основанию 17.

Специфичность селектора: примеры

- LI {} /* a=0 b=0 c=1 -> специфичность = 1 */
- UL LI {} /* a=0 b=0 c=2 -> специфичность = 2 */
- UL OL+LI {} /* a=0 b=0 c=3 -> специфичность = 3 */
- UL OL LI.red {} /* a=0 b=1 c=3 -> специфичность = 13 */
- LI.red.level {} /* a=0 b=2 c=1 -> специфичность = 21 */
- #x34y {} /* a=1 b=0 c=0 -> специфичность = 100 */

Правила каскадирования CSS

- Алгоритм каскадирования:
  3. Вторичная сортировка выполняется по специфичности селектора.

ПРИМЕР

Результат шага 2: P {color: green;}, P.color {color: blue}, P {color: black}

Результат шага 3: P.color {color: blue}, P {color: green;}, P {color: black}

Правила каскадирования CSS

- Алгоритм каскадирования:
  4. Сортировка в соответствии с порядком следования: если два правила имеют одинаковые приоритет, источник и специфичность, то будет использоваться правило, описанное последним.

ПРИМЕР

Результат шага 3: P {color: green;}, P.color {color: blue}, P {color: black}

Результат шага 4: **P.color {color: blue},** P {color: green;}, P {color: black}

Задачи

Авторские стили:

P#color_red {color: red;}

P {color: green;}

P.color {color: blue}

Браузерный стиль:

P {color: black}

Определить цвет текста:

- <P>text</P>
- <P class=”color” id=”color_red”> text</P>
- <P style=”color: crey” id=”color_red”> text</P>

Блочные и строчные элементы в HTML

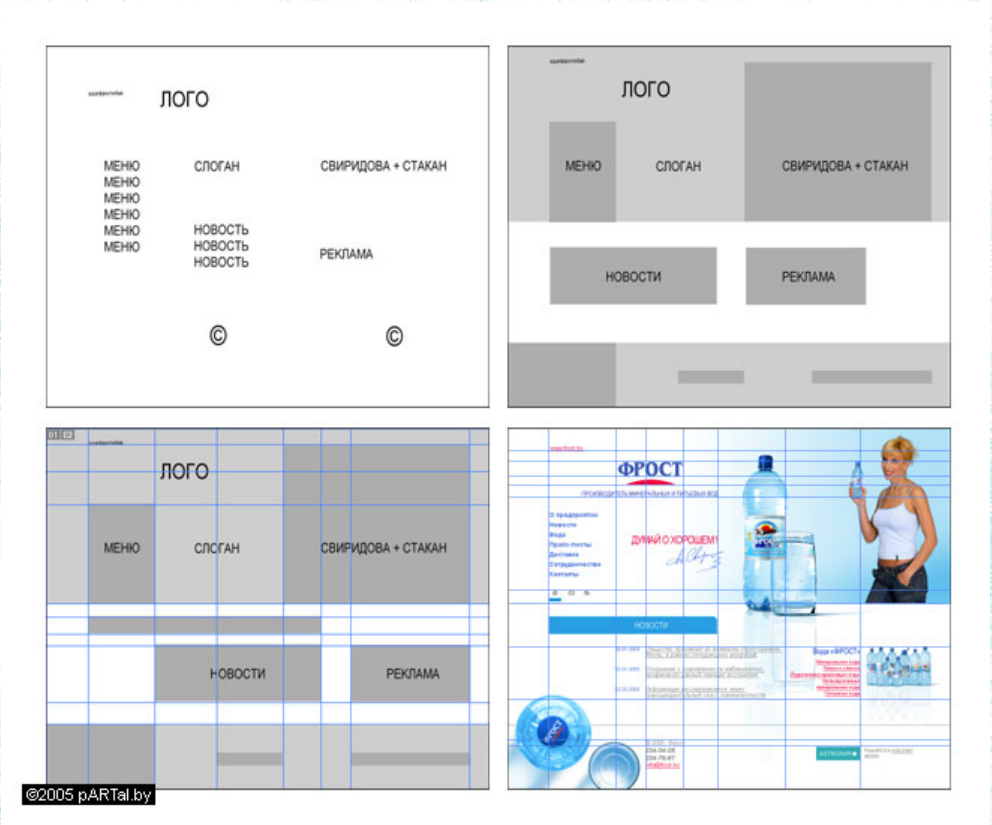
- *Блочные элементы* создают визуально самостоятельную структурную единицу - **блок**. Примером блочных элементов могут быть элементы **H1-P6**, **P**, **DIV**.
- *Строчные элементы* не создают визуальной самостоятельной структурной единицы и **выводятся линейной строкой**. Примеры: элементы **I**, **B**, **U**, **S**, **EM** и др.

Верстка

- **Верстка веб-страниц** — процесс формирования веб-страниц в текстовом либо WYSIWYG-редакторе, а также результат этого процесса, то есть собственно веб-страницы.
- Задачи верстки:
  - соблюсти исходный замысел автора
  - реализовать веб-страницы кроссбраузерными и валидными
  - выполнить семантическую верстку
  - провести SEO-оптимизацию

Исходный замысел автора эскизов

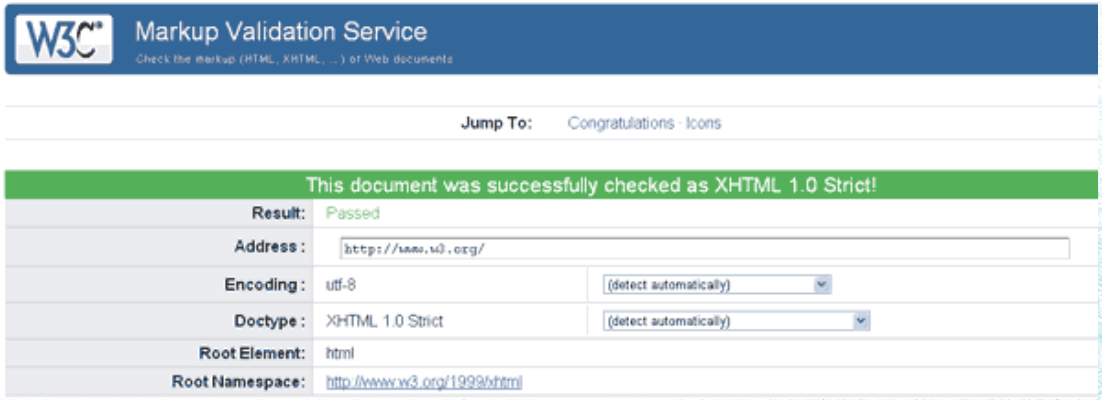
- Верстка выполняется на базе брифа/ТЗ, лейаута и эскизов.



Валидность верстки по стандартам W3C

- **Валидность верстки -** соответствие верстки определенному формату.
- **Валидатор** - компьютерная программа, которая проверяет валидность.
  - **Markup Validation Service: http://validator.w3.org/**

    Проверка документов HTML, XHTML и др.



  - **CSS Validation Service: http://jigsaw.w3.org/css-validator/**

    Проверка каскадных таблиц стилей (CSS) и документов (X)HTML с таблицами стилей

Кроссбраузерность

- **Кроссбраузерность** – свойство сайта отображаться и работать во всех популярных браузерах идентично.
- Под идентичностью понимается отсутствие развалов верстки и способность отображать материал с одинаковой степенью читабельности.
- Понятие «кроссбраузерность» не равно попиксельному соответствию.
- **Хак** — набор приемов, когда определенному браузеру «подсовывают» код, который понимается только этим браузером, а остальными игнорируется.
  - Например: -moz-background-size
    -o-background-size
    -webkit-background-size

Семантическая верстка

- **Семантический HTML-код** — это верстка с правильным использованием HTML тегов, использованием их по назначению, так как их задумывали разработчики языка HTML и веб-стандартов.
  - Например: тег <b> — это выделение текста жирным (не подходит для выделения заголовков)

Зачем нужна семантическая верстка?

1. Повышение читабельности кода, упрощение локализации ошибки

```
<table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr>
<td colspan="2" id="main-container">
  <table width="100%" border="0" cellpadding="0" cellspacing="0">
  <tr>
  <td id="left-container">
    <table width="300%" border="0" cellspacing="0" cellpadding="0">
    <tr>
    <td id="left1"><a href=http://esso-mc.ru><img src="http://esso-mc.ru/images/1/log
    </tr>
    <tr>
    <td id="left2"><div class="item"><a href="/history
      <div class="item"><a href="/materials/" title="C
    </td>
    </tr>
    <tr>
    <td id="left3"><table border="0" cellspacing="0" c
    <tr>
    <td><img src="http://esso-mc.ru/images/1/ico1.gif"
    <td><a href="/catalog/group1/">Продукция для<br />
                автомобильной<br />
                 техники</a></td>
    </tr>
    <tr>
    <td><img src="http://esso-mc.ru/images/1/ico2.gif"
    <td><a href="/catalog/group2/">Индустриальная<br /
                продукция</a></td>
    </tr>
    <tr>
```

```
<div class="catalogFullDesc">
        <h1><a href="#">Citation I</a></h1>
        <dl>
                <dt>Класс ВС:</dt>
                <dd>Light Jet</dd>
                <dt>Количество пассажиров:</dt>
                <dd>5-7 человек</dd>
                <dt>Высота потолка салона:</dt>
                <dd>1,46 м</dd>
                <dt>Наличие туалета:</dt>
                <dd>Да</dd>
                <dt>Крейсерская скорость:</dt>
                <dd>640 км/ч</dd>
                <dt>Дальность полета по времени:</dt>
                <dd>3 часа 45 минут</dd>
                <dt>Дальность полета по протяженности:</dt>
                <dd>2400 км</dd>
                <dt>Размер багажного отделения:</dt>
                <dd>1,96 куб.м</dd>
        </dl>
        <a href="#" id="innerBronLink"><span>Забронировать</span></a>
</div>
```

Зачем нужна семантическая верстка?

2. Предоставление специальных возможностей для пользователей (голосовые браузеры, плагины браузеров для перемещения по документу, др.).

Зачем нужна семантическая верстка?

3. Адаптированность для поисковых систем

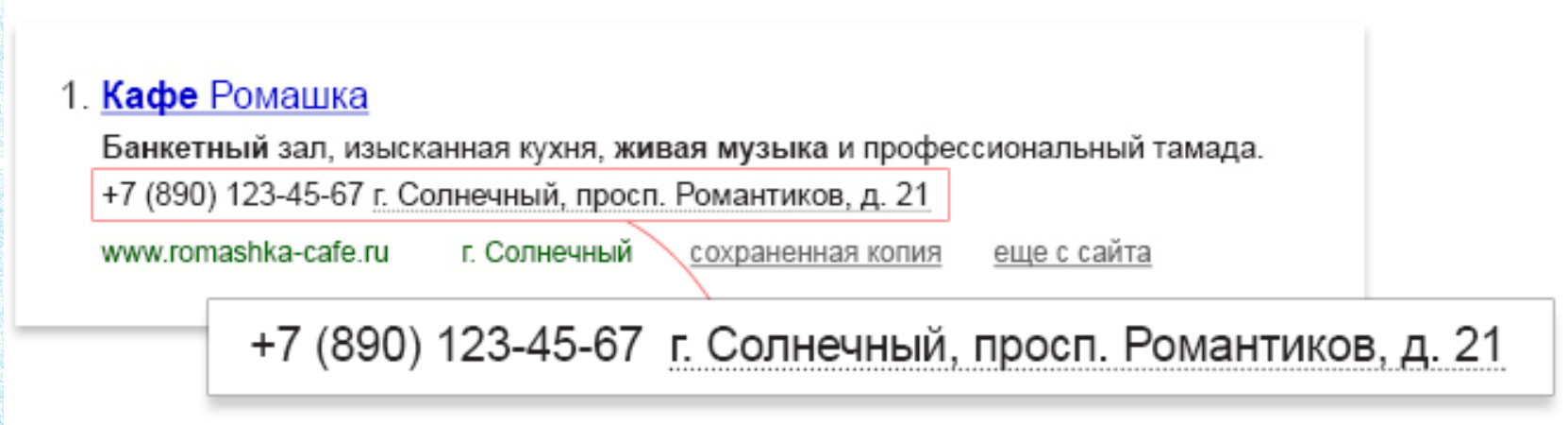
- **Микроформаты** — это форматы семантической разметки (X)HTML-страниц, позволяющие сделать контент доступным для обработки роботами. Микроформаты дают возможность явно указать смысловое значение отдельных блоков текста, дополнив существующую HTML-разметку специальными блоками.
- Например, можно обозначить, что конкретная строка является адресом организации:

```
<div class="adr"> Москва, ул. Льва Толстого, 16 </div>
```

В примере HTML-элементу добавлен атрибут class, значение которого представляет собой имя свойства, предусмотренного микроформатом **hCard**. В результате поисковый робот сможет интерпретировать содержимое элемента как значение этого свойства.

Как Яндекс работает с hCard

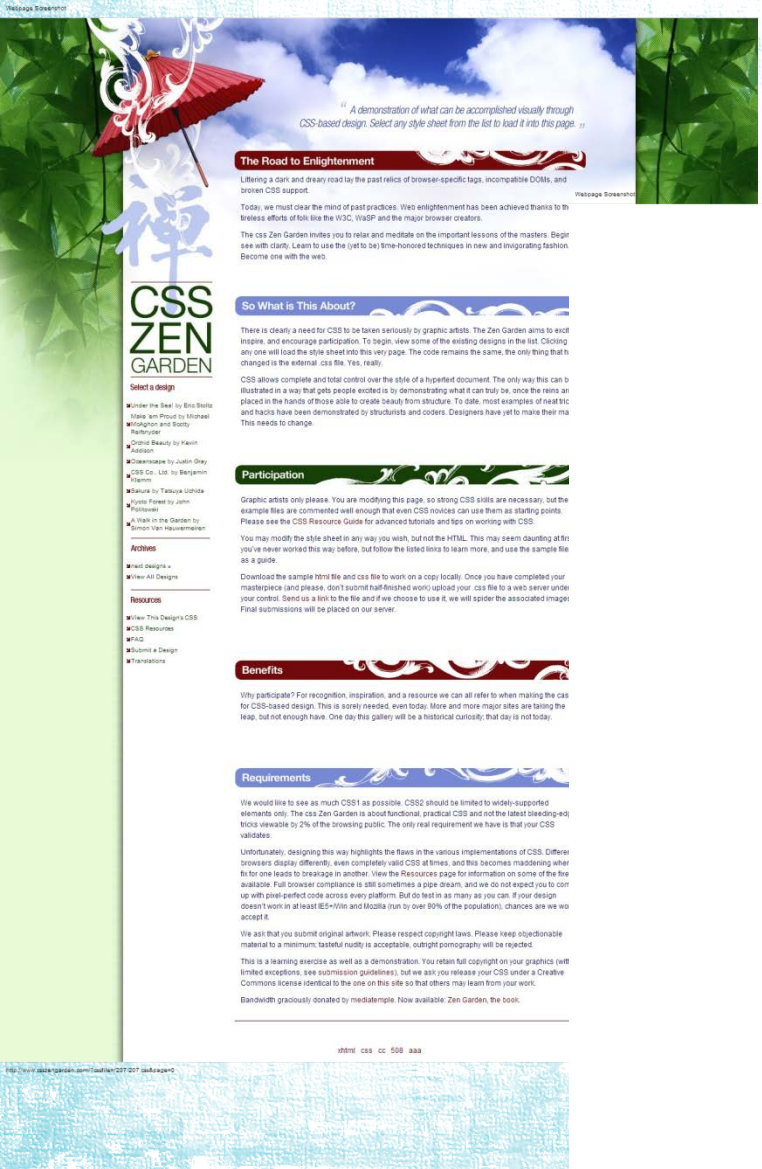
- hCard-разметка обрабатывается одновременно с индексацией сайта поисковым роботом Яндекса. Извлеченные адресные данные отображаются в различных сервисах Яндекса.

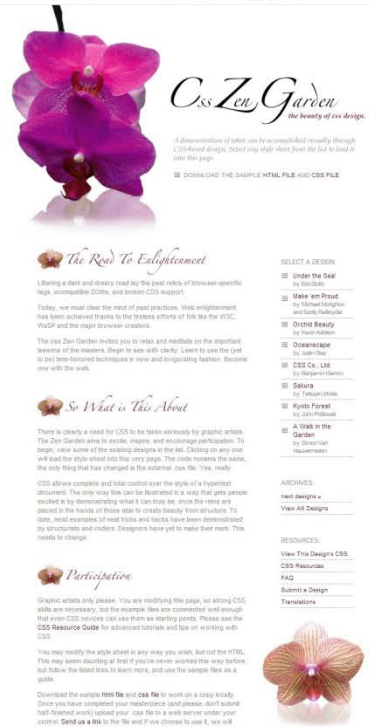


Виды верстки (по способу верстки)

- **Табличная верстка** — метод верстки HTML-документов, при котором в качестве структурной основы для расположения текстовых и графических элементов документа используются таблицы (то есть HTML-элементы <table>, <tr>, <td> и др.). Табличная верстка не является семантической.
- **Верстка слоями** заключается в конструктивном использовании для верстки веб-страниц тегов ** ** и стилей. Таблицы при данном виде верстки применяются только для представления табличных данных.

Пример верстки слоями: **Один HTML-код - разные CSS (http://www.csszengarden.com/)**





Тестирование верстки

- Тестирование на соответствие ТЗ и эскизу
- Корректность верстки по стандартам W3C
- Просмотр сайта на разных разрешениях экрана и в разных браузерах
- Тестирование без картинок
- Тестирование с настройками браузера пользователем