

Пролог

в основном освоим на лекции наследие html 4

эти технологии нужны чтобы декларировать то что видит пользователь на экране пользовательского агента

язык разметки гипертекста
гипертекст -- текст с гипер ссылками, типа как википедия
но потом решили что да нужны картинки и т п

html -- структура документа. Пишите по минимуму (идеально -- ноль) того, как должно выглядеть даже **ter b (bold)** считается теперь deprecated, вместо него лучше использовать **strong** -- показывает семантически что нужно увеличить здесь внимание, например проговорить громче для слепых
css -- как должно выглядеть

История

- Официальной спецификации HTML 1.0 не существует. До 1995 года существовало множество неофициальных стандартов HTML. Чтобы стандартная версия отличалась от них, ей сразу присвоили второй номер.
- HTML 2.0 — опубликован IETF как RFC 1866 в статусе Proposed Standard (24 ноября 1995 года)
- HTML 3.0 — 28 марта 1995 года — IETF Internet Draft (до 28 сентября 1995 года)
- HTML 4.0 — 18 декабря 1997 года
- HTML 4.01 — 24 декабря 1999 года
- HTML5 — 28 октября 2014 года
- HTML 5.1 начал разрабатываться 17 декабря 2012 года. Рекомендован к применению с 1 ноября 2016 года
- HTML 5.2 был представлен 14 декабря 2017 года
- HTML 5.3 был представлен 24 декабря 2018 года

Отображение: HTML+CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Пример</title>
  <link rel="stylesheet" href="">
</head>
<body>
  <p>
    Привет,
    <span class="name">мир!</span>
  </p>
</body>
</html>
```

```
p {
  size: 16px;
}

.name {
  font-weight: bold;
}
```

первая строчка (доктайп...) называется **прологом** -- типа говорит что дальше идет html...

Html

дерево

*на самом деле корень это **document** , см слайд про **DOM***

- html
 - head -- метаинформация
 - meta (**тег** без закрытия)
 - title
 - body

узлы дерева -- элементы и текстовые ноды

все после названия тега -- **атрибуты**, в формате ключ=значение

```
html и xml похожи но не наследуются между собой, у них просто **кажется** есть общий предок
```

```
заглушка текста Lorem Ipsum
```

директива **link** -- говорит подключить данный .css файл к отображению

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

Элементы (теги?)

Элементы(теги?) можно разделить на:

- **inline**-элементы -- типа кусочек строки, могут даже начинаться на одной строке и перенестись на другую строку.
Примеры: p, div, main, header, section, article
- **блоковые** (блочные) -- по умолчанию занимает в **ширину** сколько может, а **высоту** сколько минимально нужно.
Эти режимы отображения можно насильственно менять (см [2 -- HTML + CSS > ^displaySlide](#))
по сути в html теги одного типа делают одно и то же, разные названия передают семантику
ну только **div** и **span** **не обладают семантикой**, это простой блочковый и inline элементы соответственно

Всякие теги:

- p -- paragraph
параграф текста
блочный
- a - anchor (якорь)
ссылка
есть обязательный тег **href**
inline
- span
inline-элемент
- nav -- навигация
типа менюшка
логично внутри юзать ul { li li li }

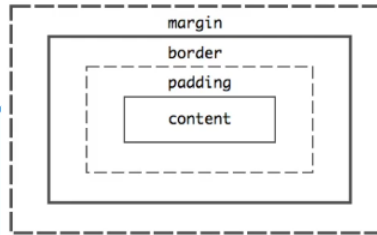
- хуйня внизу, написать копирайт

Блочные элементы (слайд)

HTML

- Блочные элементы (`display: block`)
Один в строке, можно управлять шириной и высотой. Примеры: `div`, `p`, `form`, `article`, `aside`, `main`, `nav`, `header`, `footer`, `section`, `ul`, `ol`, `li`.

По-умолчанию width и height относятся к content, но box-sizing может переопределить поведение.



margin -- внешний отступ

border -- граница

padding -- отступ контента от бордера

когда мы руками устанавливаем width/height, то по дефолту он устанавливается для контента
чтобы это поведение менять юзаем box-sizing

```
box-sizing: border-box;
```

значения `margin-box` кажется не существует

упражнение: объяснить почему будет по разному себя вести ширина блоков при добавлении width100%

```

.a {
  background-color: blue;
  margin: 1rem;
  border: 5px solid red;
  padding: 1rem;
  width: 100%;
}

.b {
  background-color: aqua;
  width: 100%;
}

```

display: (слайд)

HTML

- Строковые элементы (`display: inline`)
Являются частью строки. `width-height` автоматически. Верхние-нижние отступы не применимы. Примеры: `span`, `a`, `strong`, `button`, `input`, `label`, `select`, `textarea`, `b`, `i`.
- Не отображаемые элементы (`display: none`)
- Строково-блочные (`display: inline-block`)
Упрощенно: похожи на блочные, но не разрываются строку

к inline неприменимы верхние/нижние отступы и ширина

inline-block -- не разрывает строку, но можно указывать левые/правые отступы и т.п.

Позиционирование = position: (слайд)

Позиционирование

- position: static - по-умолчанию
- position: relative - отобразить по-умолчанию, потом сдвинуть относительно
- position: absolute - отобразить как задали, точка отчета - это ближайший относительно позиционированный предок или корень
- position: fixed - отобразить как задали относительно окна браузера (viewport-a)

речь идет про блочные элементы

relative -- нарисовали как статик, а потом вырезали и подвинули (все размеры не меняются)

absolute -- забываем про контекст, отображаем от какой-то "точки отсчета", другие элементы про него не знают (не должны знать, забывают)

точка отсчета: дефолтно -- левый верхний угол, иначе -- идем вверх по дереву до первого элемента с **position: relative**

fixed -- абсолютно относительно левого верхнего угла браузера а не документа, например для уведомлений

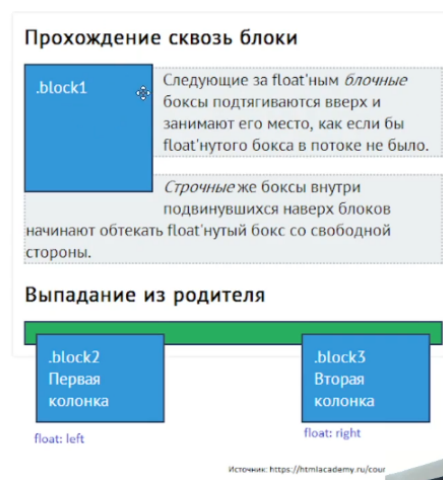
Поплавки

Поплавки

```
.block1 {  
  float: left;  
}
```

Отображение в стиле картинок в тексте, которые прижаты влево-вправо. Их обтекает content родителя, они не влияют на высоту родителя. Они всегда блочные.

Для "не выпадания" из родителя можно использовать `display: flow-root;`



ИНФОРМАЦИЯ НА СЛАЙДЕ ОЧЕНЬ ВАЖНАЯ ПРОЧИТАТЬ ВНИКНУТЬ И ПРОЧИТАТЬ ЕЩЕ РАЗ!!!!

Атрибуты элементов

все свои (не стандартные) атрибуты нужно начинать с "data-"

- **атрибут class**

говорит к какому классу принадлежит элемент

один элемент может принадлежать нескольким классам

Привет,

```
<span class="name x y u">мир!</span>
```

-- классы перечисляются через пробел

- **атрибут title**

какой текст выводится при наведении курсора

- атрибут alt

какой текст вывести вместо картинки если она не смогла загрузиться

REM

"root-em"

размер шрифта в документе

величина размера

header появился в html5 ? [#проверить](#)

проверить

CSS

CSS

CSS (Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Простые селекторы:

селектор {	* - все элементы
свойство: значение;	#id - выбор элемента по атрибуту id
свойство: значение;	.cls - выбор всех элементов по классу cls
свойство: значение;	tag - выбор всех элементов по имени тега
}	[attr=value] - выбор всех элементов по атрибуту и значению

```
#user {
  margin: 1rem;
}
.postcard {
  border: 1px solid black;
}
```



состоит в основном из правил

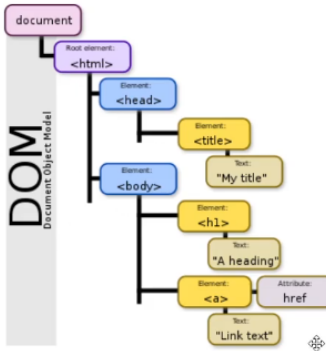
как выглядит правило:

```
селектор : {
  ключ1: значение1;
  ключ2: значение2;
  ...
}
```

Селектор, DOM

Селектор -- декларативный запрос к **DOM** (*Document Object Model* = дереву документа) "найди набор элементов из этого дерева"

DOM



DOM (от англ. Document Object Model — «объектная модель документа») — это независимый от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML-, XHTML- и XML-документов, а также изменять содержимое, структуру и оформление таких документов.

Любой документ известной структуры с помощью DOM может быть представлен в виде дерева узлов, каждый узел которого представляет собой элемент, атрибут, текстовый, графический или любой другой объект.

Составные селекторы

"более сложные способы запрашивать узлы из DOM-а"

контролировать связки скобками **нельзя**))

- пробел = "предок->потомок"

note: потомок, а не сын

```
div span {
  color: red;
}
```

- больше = "родитель->ребенок"

```
div > span {
  color: red;
}
```

- запятая=объединение путей, "или"

```
div .name div > span, p {
  color: red;
}
```

- двоеточие=псевдоклассы -- всякие штуки))

CSS

Псевдоклассы

- :nth-child(even) - отфильтрует только такие, которые являются четным ребенком. Другие примеры :nth-child(odd), :nth-child(1)
- :visited - посещенная ссылка
- :hover - мышь сверху

```
li:nth-child(odd) {
  color: #aae;
}

a:visited {
  color: gray;
}

.comment:hover {
  cursor: pointer;
}
```

```
li:nth-child(3n+1) {
  color: red;
}

li:nth-child(odd) {
  color: red;
}
```

```

:root {
  font-size: 16px;
  font-family: Verdana, serif;
}

```

Не путать!

"пэшка" с id="mike":

```

p#mike {

```

id="mike" у которого есть предок "пэшка":

```

p #mike {

```

Примеры

```

p : { -- все теги <p>
}

```

```

* : { -- ВСЕ элементы
}

```

```

#[идентификатор] {
} -- для
<[тег] id="[идентификатор]">...</>

```

предполагается что id уникальны

```

.[класс] {
} -- для всех
<[тег] class="[класс1 класс ...]">...</>

```

```

<p data-key="value">
  Привет,
  <span class="name">мир!</span>
</p>

```

```

[data-key=value] {
  color: beige;
}
[data-key="value"] {
  color: beige;
}

```

вроде можно и без кавычек

Еще фишки

```

html {
  min-width: 800px;
}

```

min_width 800px -- ширина не меньше 800 пикселей (если окно меньше то будет прокрутка а не еще большее сжатие)

```
.middle aside section .header:before {  
  content: "11";  
}
```

псевдо-элементы before after

normalize.css

у браузера есть встроенные типа-css-ки которые говорят какие отступы делать на элементах по умолчанию

хотим ? верстку которая будет одинаково смотреться во всех браузерах в мире

есть два пути:

1. ресетнуть нахуй все
2. сделать нормалайз -- сбросить стили до нормального одинакового четкого значения во всех браузерах

#непонел

КОЛЛИЗИИ

css -- при коллизиях правил движки обычно стараются применить правило более специфичного селектора (например по id специфичнее чем по tag-name)

но **вроде** четких правил нет

можно повышать значимость правила))

```
[data-key="value"] {  
  color: beige !important;  
}
```