

Серьезное лицо - еще не признак ума, господа. Все глупости на Земле делаются именно с этим выражением. Вы улыбайтесь, господа, улыбайтесь!





СБЕРБАНК ТЕХНОЛОГИИ

HTML+CSS+JS

<https://goo.gl/k5hiC4>



СБЕРБАНК ТЕХНОЛОГИИ

Why Java?

А ЧТО ЕСЛИ Я СКАЖУ ТЕБЕ

ЧТО МОЖНО НЕ КОМПИЛИРОВАТЬ КОД

План занятия

План занятия



Почему подвиг?

НЕЛЬЗЯ ТАК ПРОСТО ВЗЯТЬ И

**ИЗУЧИТЬ FRONTEND
ЗА 3 ЛЕКЦИИ**

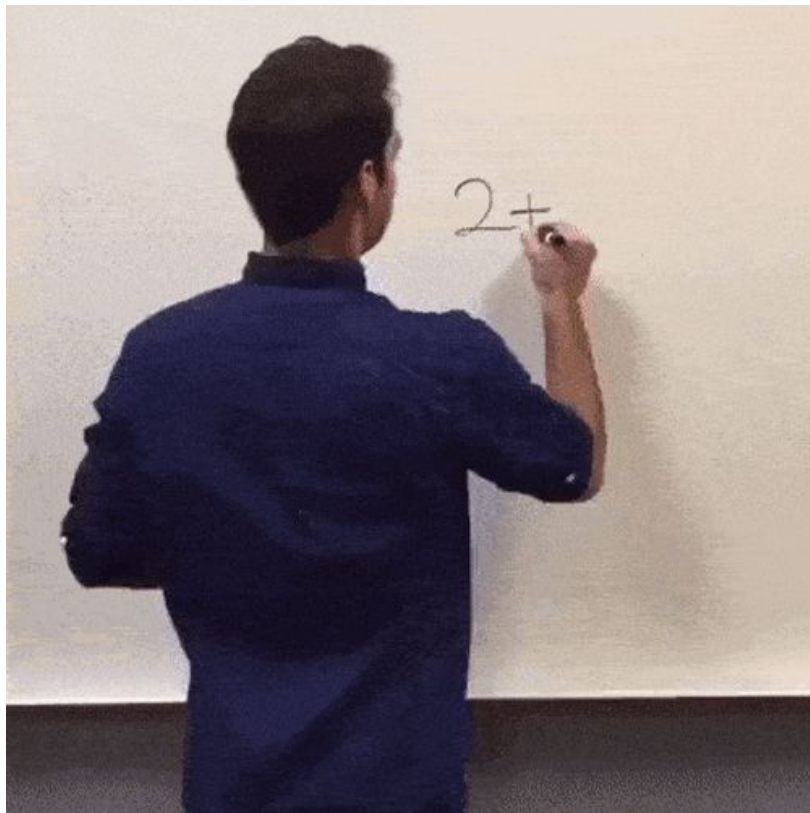


Терминология

Широко известен спор между Платоном и Диогеном. "Человек, - сказал Платон, - это двуногое животное без перьев". Тогда Диоген оципал петуха и со словами: "Вот твой человек", - поставил его перед Платоном. Пришлось Платону сделать уточнение: "Двуногое животное без перьев и имеющее ногти".

Челове́к разу́мный — вид рода Люди (Homo) из семейства гоминид в отряде приматов.

Пора пристегнуть ремни



План занятия

1. Что значит «V» в MVC?
2. Ретроспектива Web
3. Стек технологий в Web
4. HTML – лего для браузера
5. CSS – макияж для HTML
6. CSS flexbox
7. JavaScript (ECMAScript)
8. Отладка
9. Среда

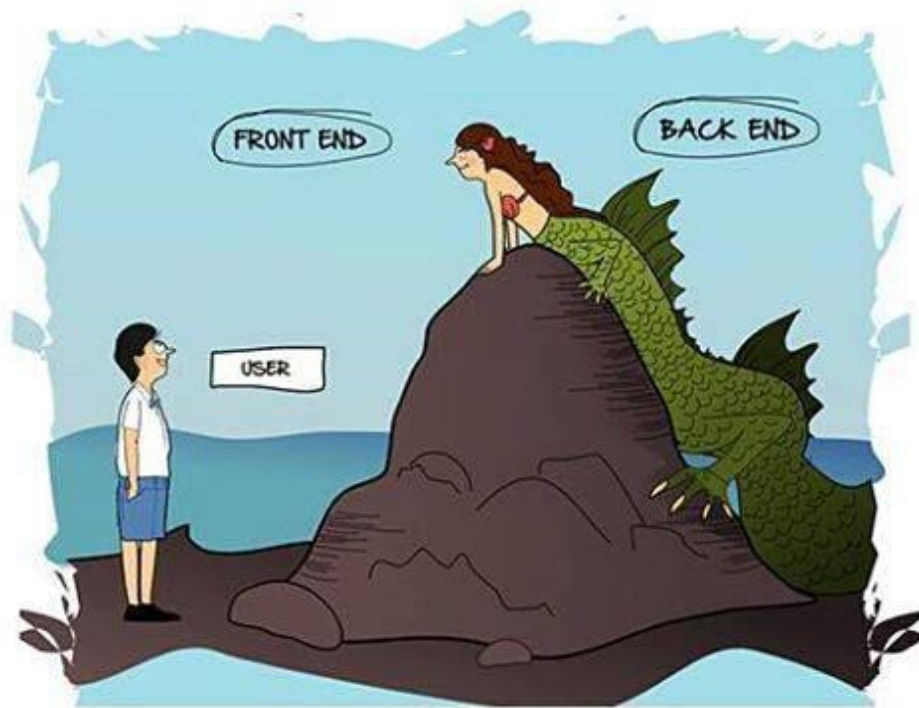


1. Что значит «V» в MVC?

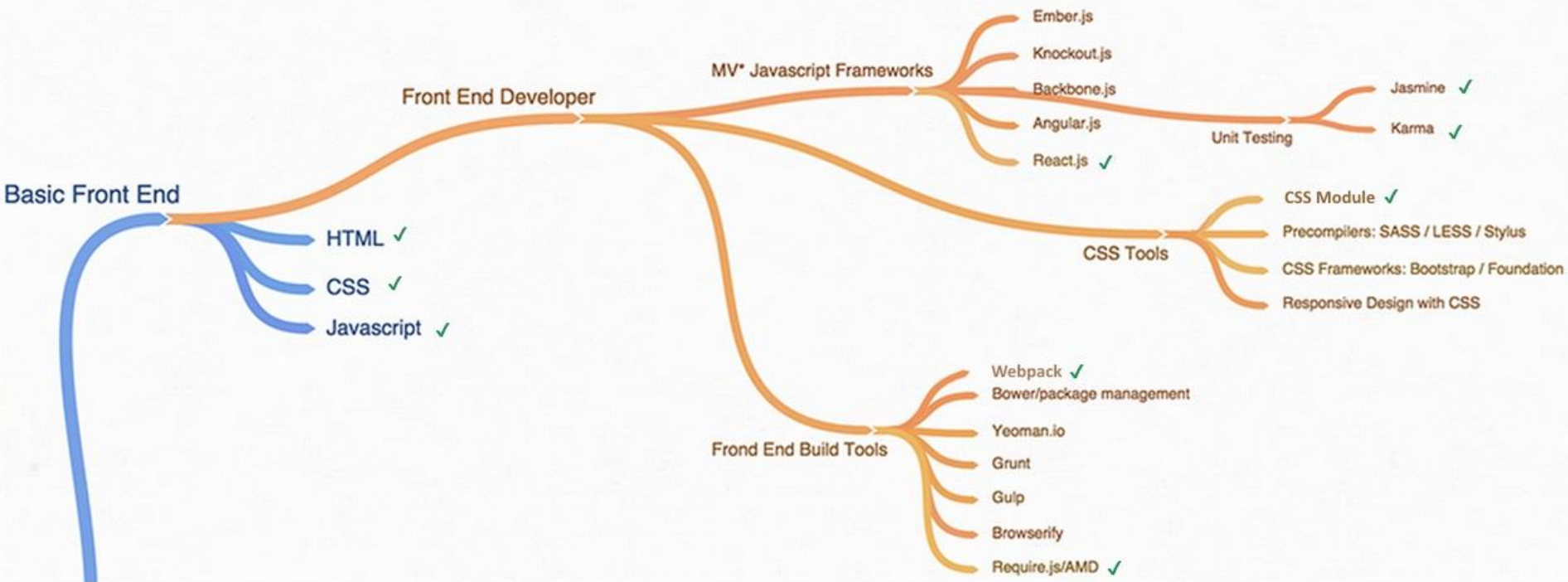
«V» значит Вендетта?



В нашем случае «V» это веб-интерфейс



Нам пригодятся следующие технологии

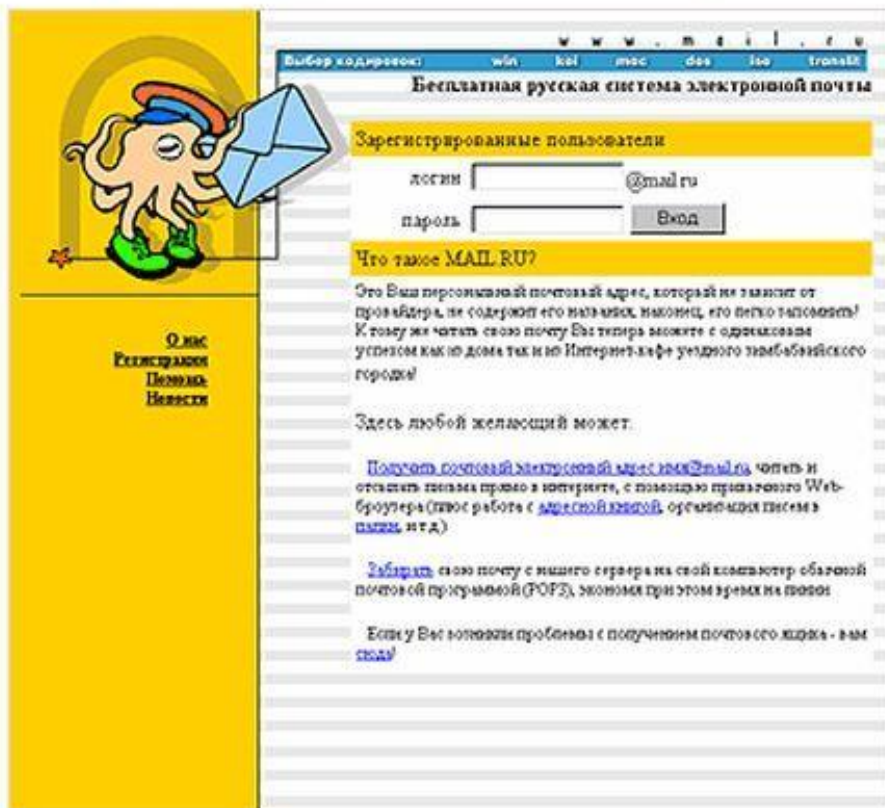




СБЕРБАНК ТЕХНОЛОГИИ

2. Ретроспектива Web

Web в 1999



Из чего состоит Web данный момент

HTML — стандартизированный язык разметки документов

CSS — формальный язык описания внешнего вида документа, написанного с использованием языка разметки

JavaScript — язык программирования



Ретроспектива Web

- 1) Текст (www) 1990
- 2) Красивый текст с картинками (html) 1993
- 3) Красивый текст с картинками и анимацией (html+js) 1995
- 4) Много красивого текста с картинками и анимацией (html+css+js) 1996

html

```
<!DOCTYPE html>
<html>
<head>
  <title>Пример страницы</title>
</head>
<body>
  <p style="color: 'navy';">Страница на HTML5 </p>
  <p style="color: 'navy';">Описание 1 </p>
  <p style="color: 'navy';">Описание 2 </p>
</body>
</html>
```

html+css

```
<!DOCTYPE html>
<html>
<head>
  <title>Пример страницы</title>
  <style type="text/css">
    p { color: 'navy'; }
  </style>
</head>
<body>
  <p>Страница на HTML5 </p>
  <p>Описание 1 </p>
  <p>Описание 2 </p>
</body>
</html>
```

html+css+js

```
<!DOCTYPE html>
<html>
<head>
  <title>Пример страницы</title>
  <style type="text/css">
    p { color: 'navy'; }
  </style>
  <script type="text/javascript">
    alert("Превед медвед");
  </script>
</head>
<body>
  <p>Страница на HTML5 </p>
  <p>Описание 1</p>
  <p>Описание 2</p>
</body>
</html>
```



3. HTML

его для браузера

HTML похож на XML

```
<!DOCTYPE html>
<html>
<head/>
<body>
  <p class="awesome">Страница на <b>HTML5</b></p>
</body>
</html>
```

1. Тэги со вложенными данными состоят из двух частей
2. Тэги без вложенных данных состоят из одной части
3. Могут иметь атрибуты
4. Имеют ограничения по вложенности (<table> только <thead> и <tbody>)

Как видит Браузер

```
<header>...</header>
```

```
<nav>...</nav>
```

```
<section>
```

```
  <article />
```

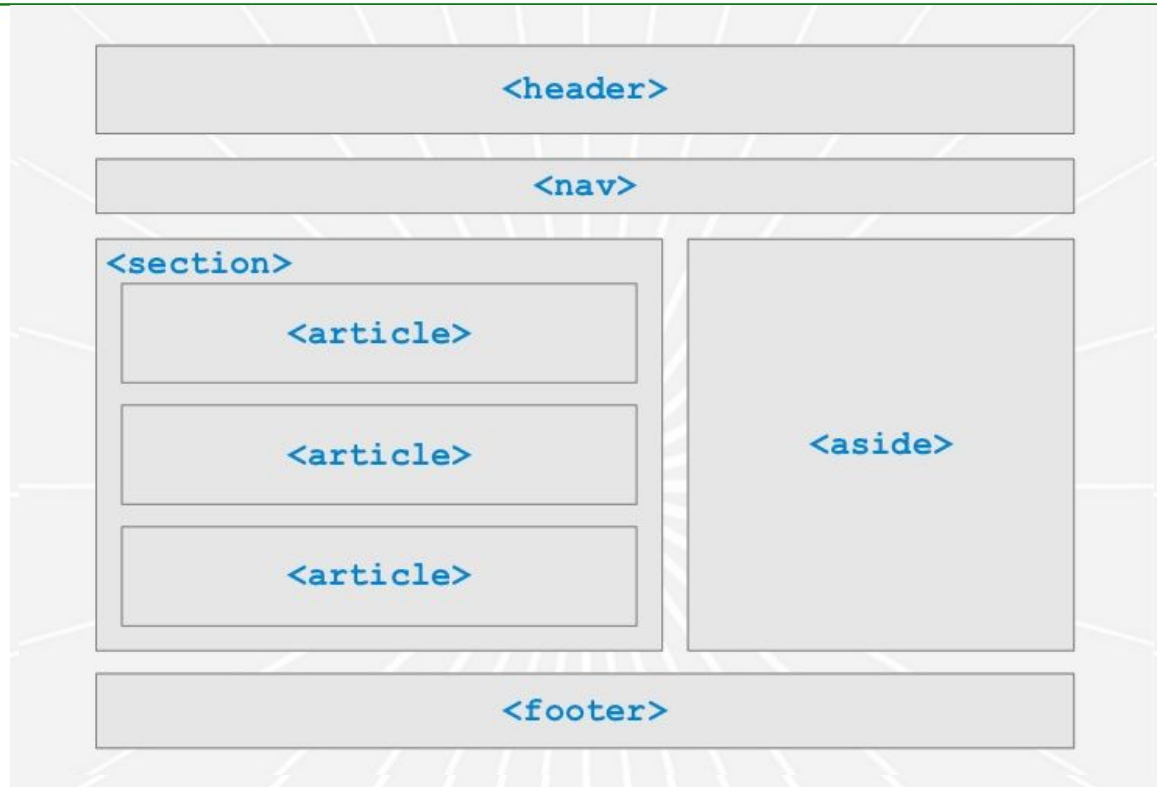
```
  ...
```

```
  <article />
```

```
</section>
```

```
<aside>...</aside>
```

```
<footer>...</footer>
```



Как пишет программист

```
<div id="header">...</div>
```

```
<div id="main-menu" />
```

```
<div id="main">
```

```
  <div class="post" />
```

```
  ...
```

```
  <div class="post" />
```

```
</div>
```

```
<div class="side">...</div>
```

```
<div id="footer">...</div>
```



DOM

```
<div id="header">...</div>
<div id="main-menu" />
<div id="main">
  <div class="post" />
  ...
  <div class="post" />
</div>
<div class="side">...</div>
<div id="footer">...</div>
```

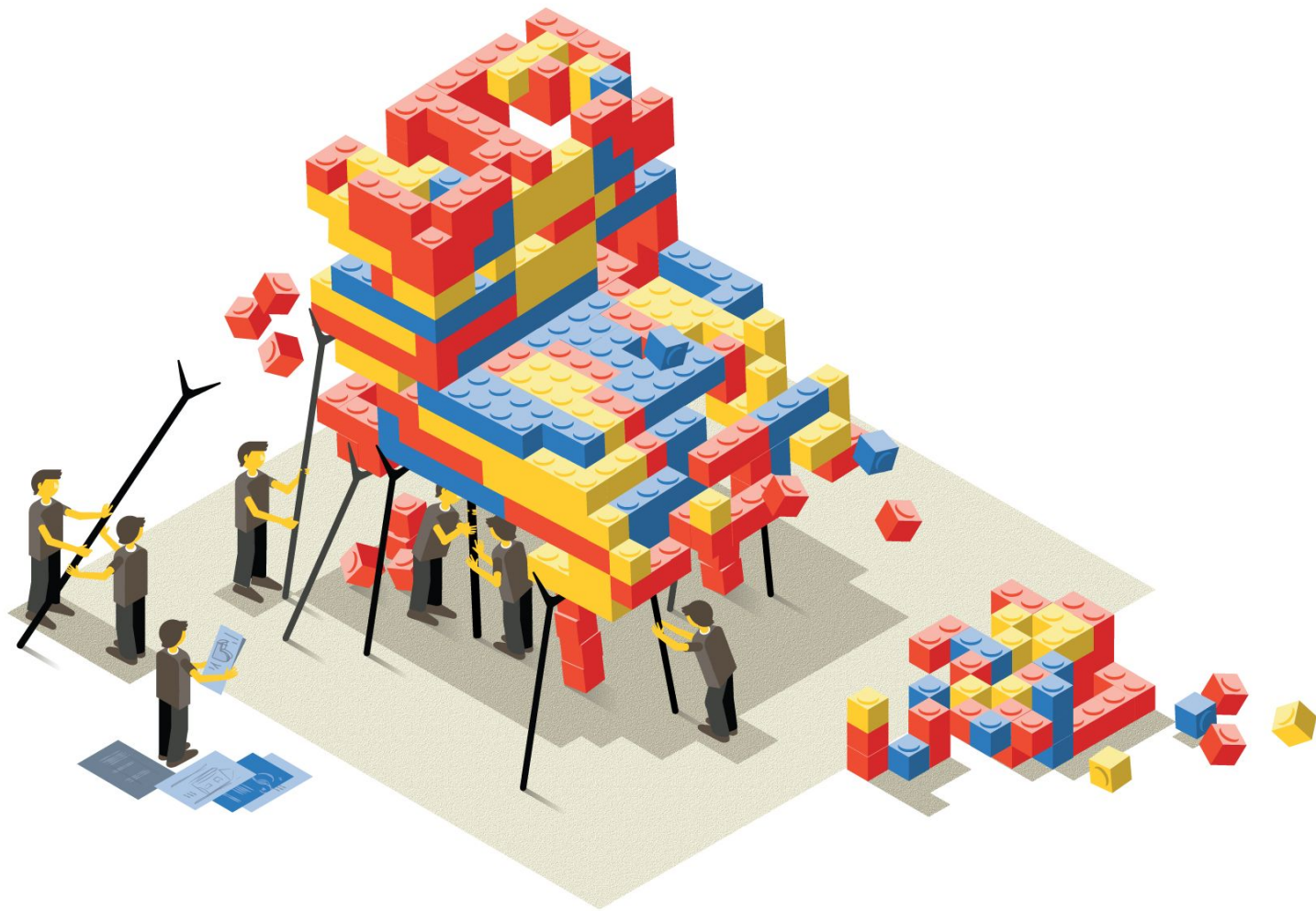




СБЕРБАНК ТЕХНОЛОГИИ

Где Iego?





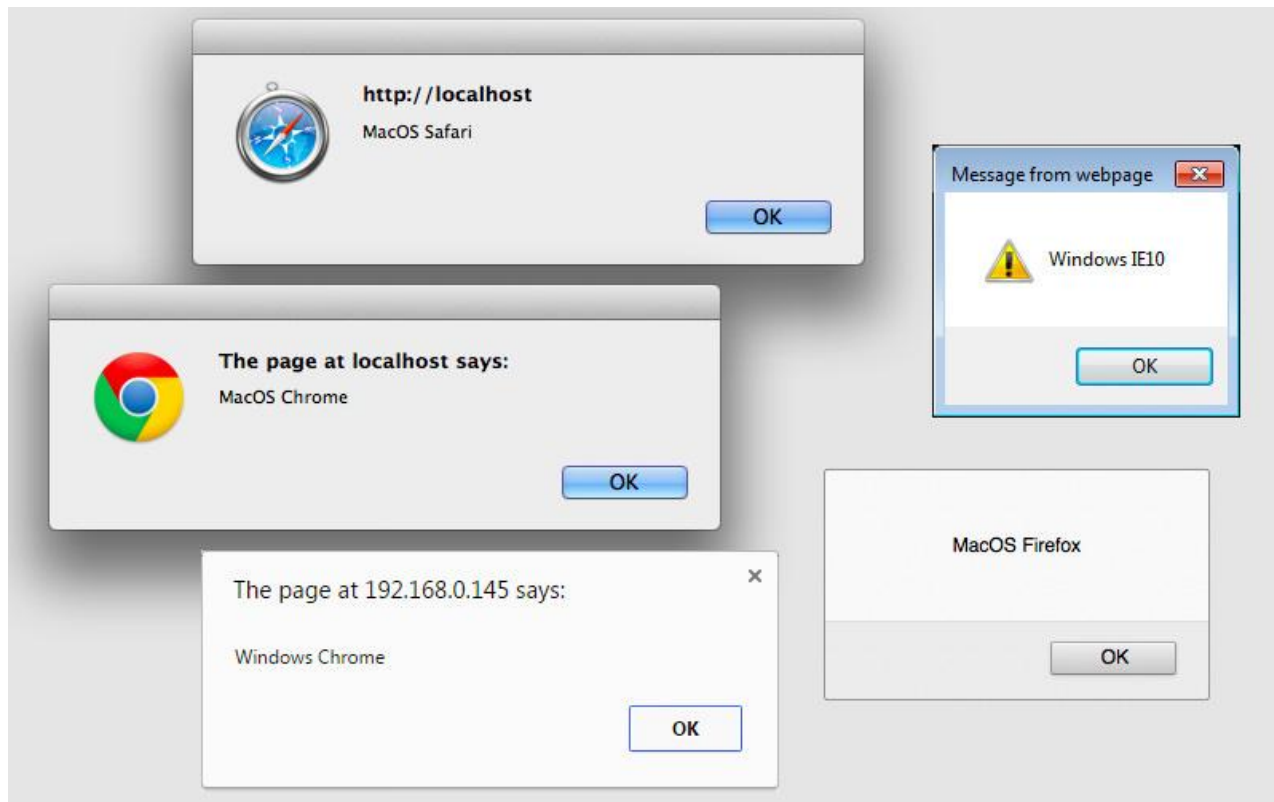
Обзорно тэги

Всего тегов > 300

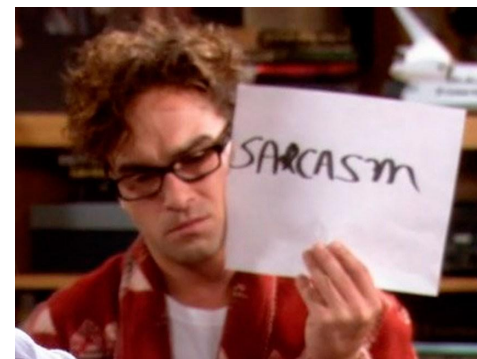
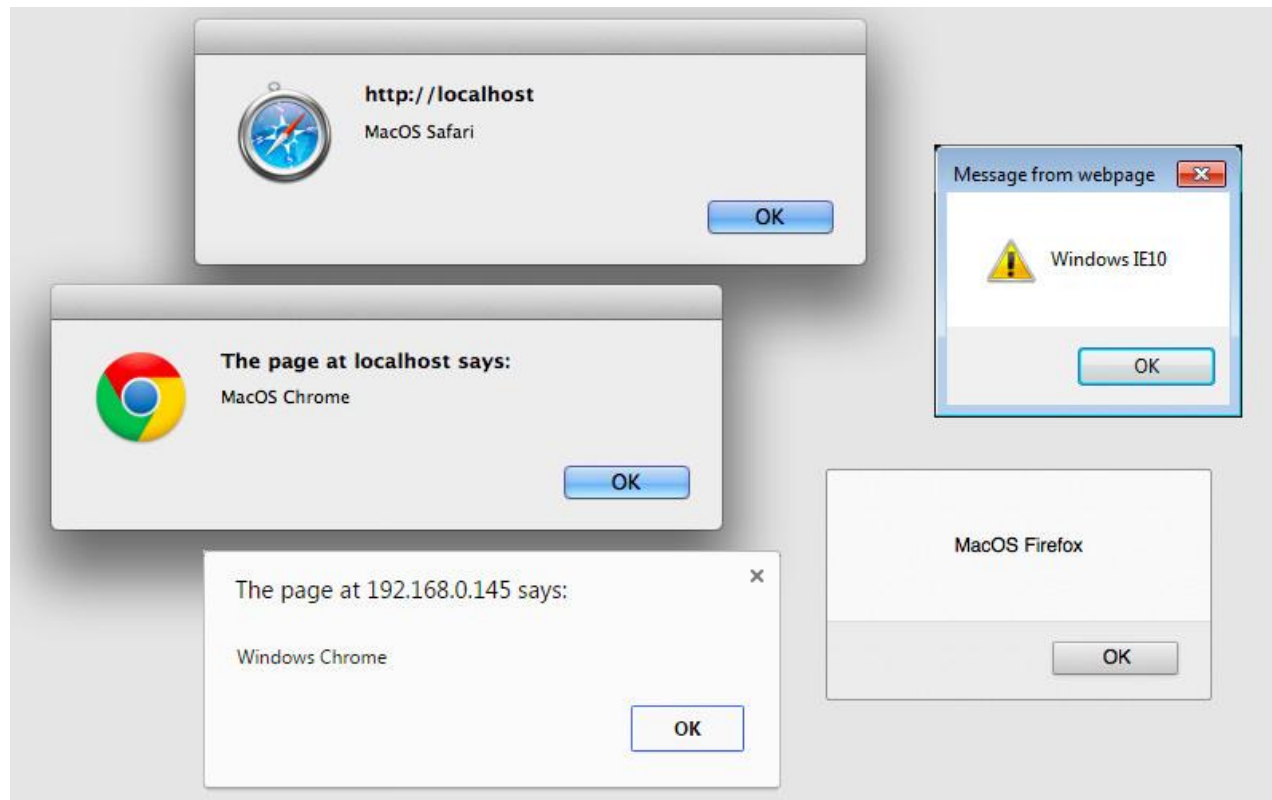
Но пользуются в основном одним - `<div>`

Потому что в разных браузерах теги отображаются по разному.

Стандарты для слабаков



Стандарты для слабаков



На самом деле тэги делятся на группы

Обеспечивающие функциональность:

Основные: `<html>`, `<body>`

Специфичные: `<iframe>`, `<video>`, `<input>`

Обеспечивающие отображение:

`<div>`, `<table>`, `<h1>`, `<p>`, ``

= `<div>` + стили (имя служит только для простоты понимания)

Demo

<https://jsfiddle.net/8wchdnfa/1/> - теги



4. CSS

макияж для html



CSS немного похож на JSON

```
.redText {  
    color: "red";  
    align: "center";  
}
```

color: red;

декларация
declaration

align: center;

свойство
property

значение
value

правило
rule



Селекторы и их комбинации

Основные

tag

#id

.class

.class1.class2

*

:pseudoclass

::pseudoelement

Комбинации

A B

A > B

A + B

A ~ B

Атрибуты

[attr]

[attr='value']

[attr^='value']

[attr\$='value']

[attr*='value']

[attr~='value']

[attr|='value']

Часто используемые комбинации

Основные	Комбинации	Атрибуты
tag	A B	[attr]
#id	A > B	[attr='value']
.class	A + B	[attr^='value']
.class1.class2	A ~ B	[attr\$='value']
*		[attr*='value']
:pseudoclass		[attr~='value']
::pseudoelement		[attr ='value']

CSS свойства (> 150 различных)

```
.example {
```

```
display: flex;
```

```
transition: all .5s;
```

```
user-select: none;
```

```
.example {
```

```
display: -webkit-box;
```

```
display: -webkit-flex;
```

```
display: -moz-box;
```

```
display: -ms-flexbox;
```

```
display: flex;
```

```
-webkit-transition: all .5s;
```

```
-o-transition: all .5s;
```

```
-moz-transition: all .5s;
```

```
transition: all .5s;
```

```
-webkit-user-select: none;
```

```
-moz-user-select: none;
```

```
-ms-user-select: none;
```

```
user-select: none;
```

```
}
```

```
}
```



PostCSS

Demo

<https://jsfiddle.net/8wchdnfa/2/> - классы и селекторы



5. flexbox

Какая боль, какая боль IE 8.0

Что он умеет?

Однострочная обертка

- flex-direction - определение осей
- justify-content - выравнивание по главной оси
- align-items - выравнивание по поперечной оси

Многострочная обертка

- flex-flow - многострочный режим
- align-content - выравнивание рядов
- flex-basis - размер элемента

Параметры элемента

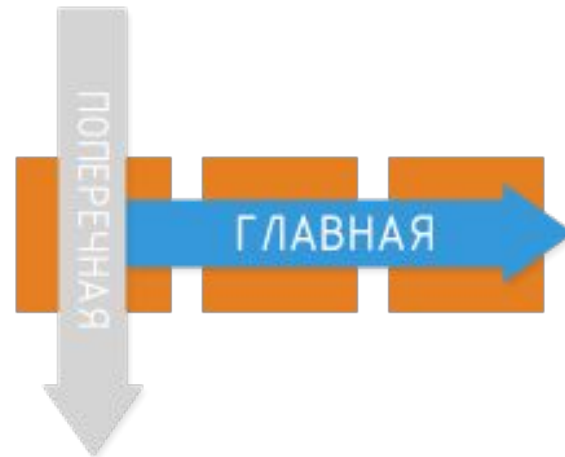
- flex – размер элемента
- order - порядок элемента

flex-direction

- row
- row-reverse
- column
- column-reverse



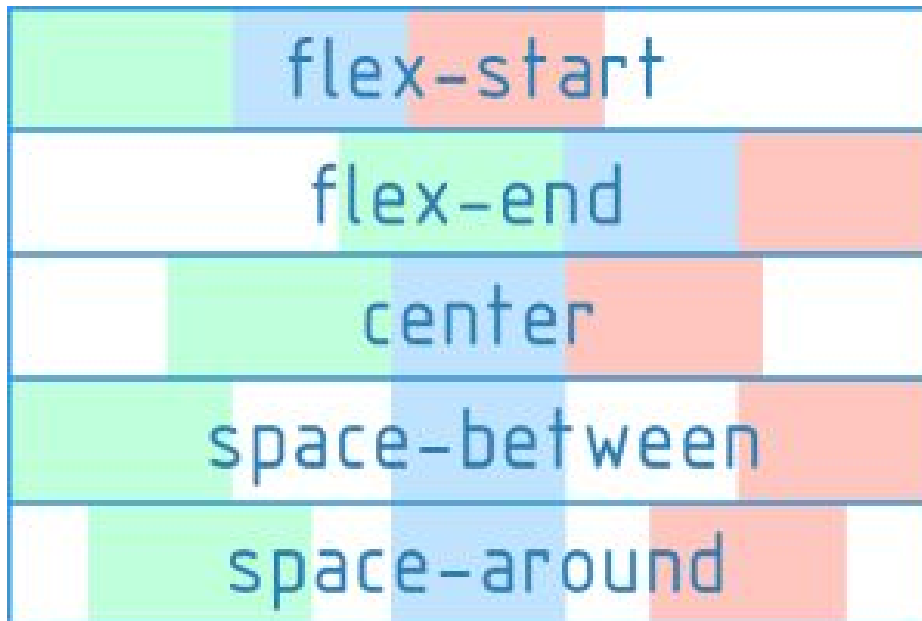
flex-direction: column



flex-direction: row

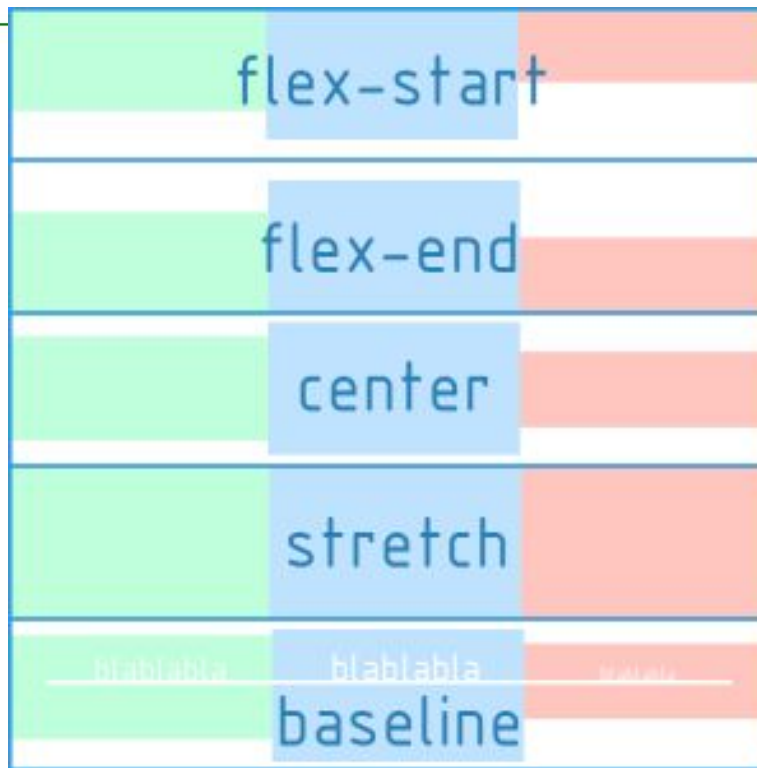
justify-content

- flex-start
- flex-end
- center
- space-between
- space-around



align-items

- **flex-start**
- **flex-end**
- **center**
- **baseline**
- **stretch**



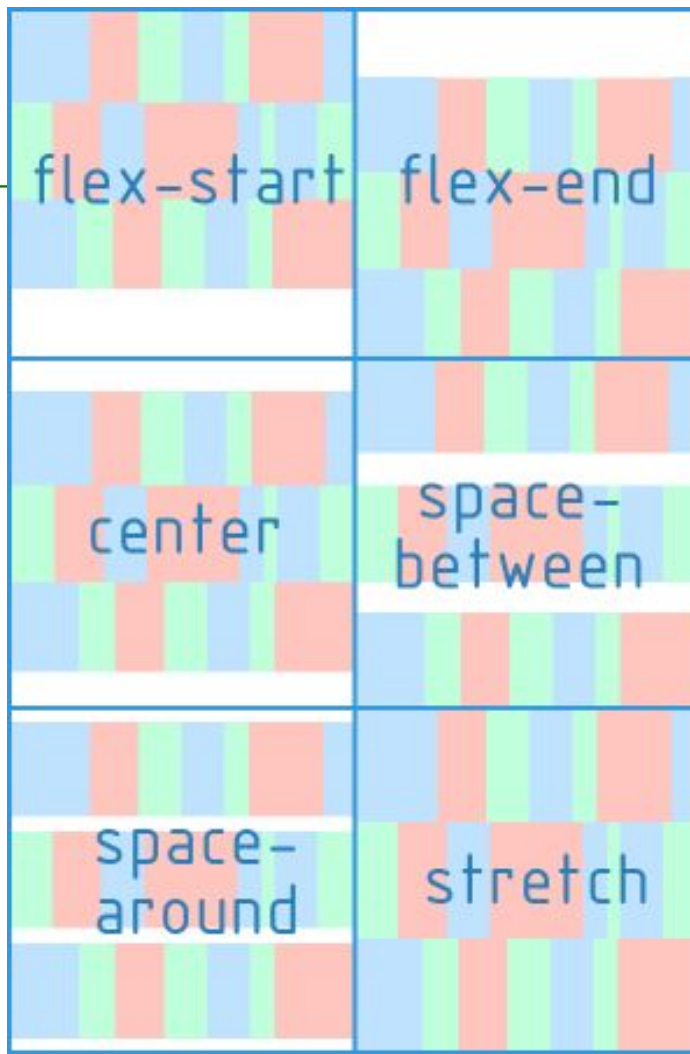
Первый параметр аналогичен **flex-direction**

Второй параметр тип обертки:

- **nowrap** (значение по умолчанию) : блоки расположены в одну линию слева направо (в rtl справа налево)
- **wrap**: блоки расположены в несколько горизонтальных рядов (если не помещаются в один ряд). Они следуют друг за другом слева направо (в rtl справа налево)
- **wrap-reverse**: тоже что и wrap, но блоки располагаются в обратном порядке.

align-content

- flex-start
- flex-end
- center
- space-between
- space-around
- stretch



Demo

<https://jsfiddle.net/8wchdnfa/3/> - justify-content

<https://jsfiddle.net/8wchdnfa/5/> - align-items

<https://jsfiddle.net/8wchdnfa/6/> - когда нужно несколько строк

<https://jsfiddle.net/8wchdnfa/7/> - align-content

<https://jsfiddle.net/8wchdnfa/8/> - заготовка layout

<https://jsfiddle.net/8wchdnfa/9/> - готовый вариант layout

<https://jsfiddle.net/8wchdnfa/10/> - пример с просторов интернета layout



6. JavaScript

В лучших традициях ES5

Создавался за 10 дней



Brendan Eich (создатель языка JavaScript)

... JS был обязан «выглядеть как Java», только поменьше, быть эдаким младшим братом-тупицей для Java. Кроме того, он должен был быть написан за 10 дней, а иначе мы бы имели что-то похуже JS....

...10 дней на то, чтобы сделать лексер, парсер, компилятор в байткод (bytecode emitter), интерпретатор, встроенные классы и декомпилятор...

...Простите, времени было мало для того, чтобы сделать правильную оптимизацию хвостовой рекурсии. 10 дней почти без сна, чтобы сделать JS с чистого листа, заставить его «выглядеть как Java» (я сделал, чтобы он выглядел как C) и тайком протащить туда его спасительные фишки: first class functions и прототипы (примерно как в

JavaScript вчера (ES5)

```
var Shape = function () {  
    function Shape(id, x, y) {  
        _classCallCheck(this, Shape);  
        this.id = id;  
        this.move(x, y);  
    }  
    Shape.prototype.move = function move(x, y) {  
        this.x = x;  
        this.y = y;  
    };  
    return Shape;  
}();
```

JavaScript завтра (ES6 / ES2015)

```
class Shape {  
  constructor (id, x, y) {  
    this.id = id  
    this.move(x, y)  
  }  
  move (x, y) {  
    this.x = x  
    this.y = y  
  }  
}
```

А что сегодня



Типы данных

В JavaScript есть три основных типа данных, два основных типа данных и два специальных типа данных. Итого 6 типов. Мощно? – Мощно!

Основные

- String
- Number
- Boolean

Составные (ссылочные):

- Object

Специальные:

- null
- undefined

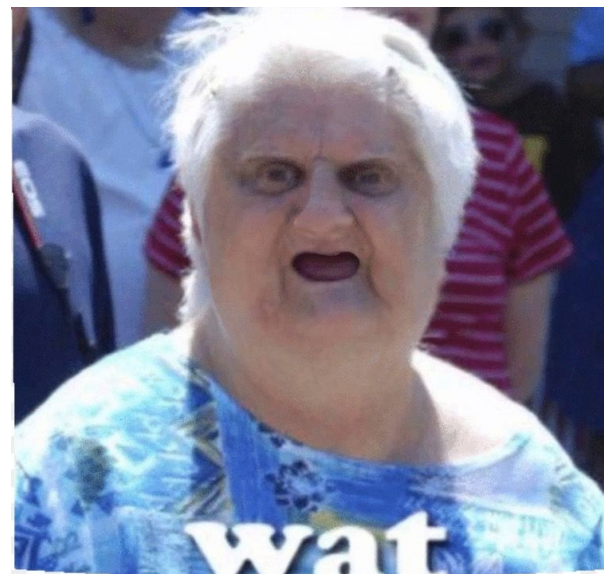
Есть 5 «примитивных» типов: number, string, boolean, null, undefined и 6-й тип – объекты object.

JavaScript не типизирован

- Нет типа для представления одиночного символа
- В if ложью будут: **''**, **0**, **false**, **null**, **NaN**, **undefined**
- Мягкое сравнение **false == 0**, **0 == '0'**, **'0' == true**

JavaScript не типизирован

- Нет типа для представления одиночного символа
- В if ложью будут: `''`, `0`, `false`, `null`, `NaN`, `undefined`
- Мягкое сравнение `false == 0`, `0 == '0'`, `'0' == true`



JavaScript не типизирован

- Нет типа для представления одиночного символа
- Нет отдельного класса с плавающей запятой
- В if ложью будут: **“, 0, false, null, NaN, undefined**
- Сравнение без учета типа **false == 0, 0 == '0', '0' == true**
- При этом **false != undefined**
- Сравнение с учетом типа **false !== 0**
- Тонкости сложения: **[] + {} = "[object Object]" ; {} + [] = 0**

JavaScript не типичное ООП

- Быстрое создание объекта { **foo**: 'bar' }
- Быстрое создание массива ['baz']
- Объекты не имеют классов
- Функция это объект который можно исполнить
- Конструктором объекта может выступать почти любая функция, достаточно её вызвать через **new**
- **this** определяется в момент исполнения

Demo

<https://jsfiddle.net/8wchdnfa/11/> - примеры по js



7. ES6 (ES2015)

Назад в будущее

Переменные

let переменная которую **можно**
переприсвоить

```
let foo = {bar : "baz" };  
foo = {bar : "biz" } // можно  
foo.bar = "biz" // можно
```

const переменная которую **нельзя**
переприсвоить

```
const foo = {bar : "baz" };  
foo = {bar : "biz" } // нельзя  
foo.bar = "biz" // можно
```

Сравнения

ES6

ES7

JS

Деструкторы

```
let { op, lhs, rhs } = getASTNode();
```

```
var tmp = getASTNode();
```

```
var op = tmp.op;
```

```
var lhs = tmp.lhs;
```

```
var rhs = tmp.rhs;
```



BABEL

Анонимные стрелочные функции

```
odds = evens.map(v => v + 1)
```

```
pairs = evens.map(v => (  
  { even: v, odd: v + 1 }  
))
```

```
nums = evens.map((v, i) => v + i)
```

Стрелочные функции сохраняют this

```
odds = evens.map(function (v) {  
  return v + 1;  
});
```

```
pairs = evens.map(function (v) {  
  return {even: v, odd: v + 1};  
});
```

```
nums = evens.map(function (v, i) {  
  return v + i;  
});
```

Анонимные не сохраняют this

Классы

```
class Shape {  
  constructor (id, x, y) {  
    this.id = id  
    this.move(x, y)  
  }  
  move (x, y) {  
    this.x = x  
    this.y = y  
  }  
}
```

```
var Shape = function () {  
  function Shape(id, x, y) {  
    _classCallCheck(this, Shape);  
    this.id = id;  
    this.move(x, y);  
  }  
  Shape.prototype.move = function move(x, y) {  
    this.x = x;  
    this.y = y;  
  };  
  return Shape;  
}();
```

Многоточие

```
let params = [ "hello", true, 7 ]  
let other = [ 1, 2, ...params ]  
// [ 1, 2, "hello", true, 7 ]  
f(1, 2, ...params) === 9
```

```
let str = "foo"  
let chars = [ ...str ]  
// [ "f", "o", "o" ]
```

```
var params = [ "hello", true, 7 ];  
var other = [ 1, 2 ].concat(params);  
// [ 1, 2, "hello", true, 7 ]  
f.apply(null, [ 1, 2 ].concat(params)) === 9;
```

```
var str = "foo";  
var chars = str.split("");  
// [ "f", "o", "o" ]
```

Demo

Отладка

- **alert(<строка>)** - останавливает исполнение кода и показывает строку
- **console.log(<данные>)** - не останавливает исполнение выводит в консоль браузера
- **debugger** - аналог breakpoint при достижении браузер переходит в режим пошаговой отладки



СБЕРБАНК ТЕХНОЛОГИИ

Среда

в понедельник?

Инструменты

- **Y.браузер** – браузер с хорошим режимом отладки
- **NodeJS** - серверный js для запуска инструментов
- **Npm** – пакетный менеджер для библиотек и инструментов разработки

Инструменты

node.js Необходимо скачать node.exe с сайта:

<http://nodejs.org/dist/latest/win-x64/> (для 64-битной сборки)

И распаковать в локальную папку, например:

C:\Users\OUT-Surname-NP\node\

npm Зайти на сайт <https://github.com/npm/npm/releases>, скачать актуальную версию в виде zip-архива и распаковать в локальную папку, например:

C:\Users\OUT-Surname-NP\npm-3.10.7

Browser

<https://www.google.ru/chrome/browser/desktop/index.html?standalone=1>

Инструменты

После того, как все нужные пакеты распакованы в локальную папку, необходимо добавить папки, содержащие их exe в локальную переменную PATH. Для этого:

- Пуск → Выполнить
- `rundll32 sysdm.cpl,EditEnvironmentVariables`
- Переменные среды пользователя для ... (username)
- Изменить PATH (или создать, если ее нет)
- В качестве значения через точку с запятой указать папки с node, npm

Например:

`C:\Users\OUT-Surname-NP\node;C:\Users\OUT-Surname-NP\npm-3.10.7\bin;`