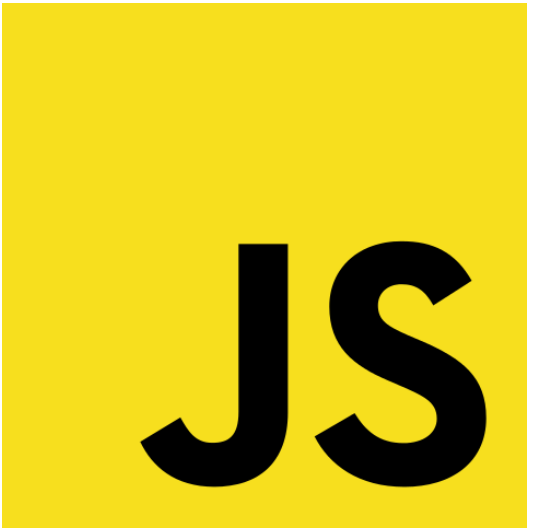
ЗНАКОМСТВО С JAVASCRIPT



ОБО МНЕ

ЗАЧЕМ ЗАНИМАТЬСЯ РАЗРАБОТКОЙ?

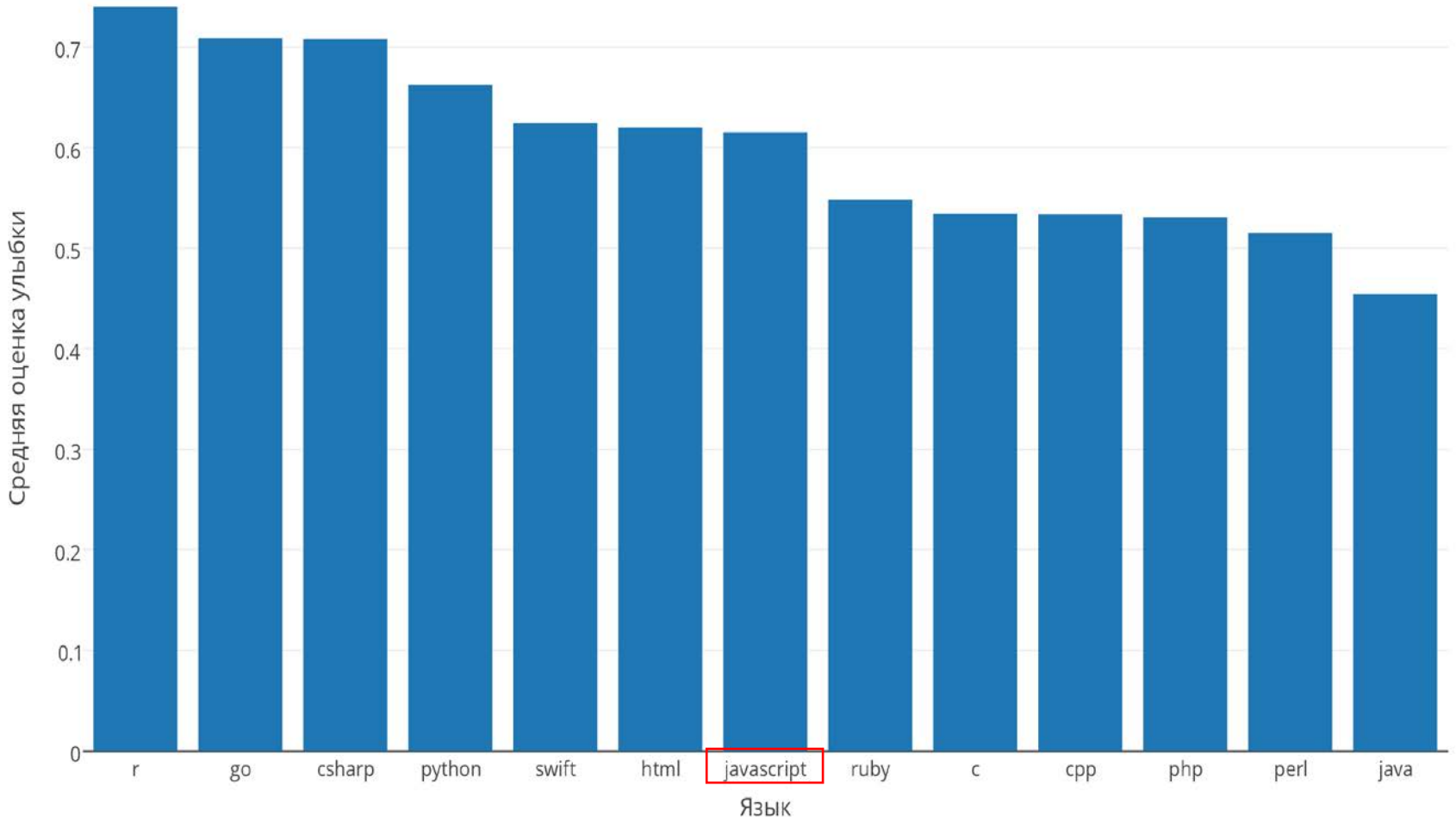
1. Полезная и интересная деятельность, реально двигающая мир вперед
2. Постоянное личное развитие и рост
3. Уникальная атмосфера, одно из лучших профессиональных сообществ
4. Конкурентная заработная плата, востребованность на рынке

3 УРОВНЯ СЧАСТЬЯ

ЗАЧЕМ УЧИТЬ JAVASCRIPT ?

1. Это очень интересный и быстро развивающийся язык программирования
2. Разнообразие задач, выполняемых языком
3. Знание этого языка всегда плюс, даже если не разрабатываешь непосредственно на нем
4. Большой рынок труда, легко начать

СРЕДНЯЯ УЛЫБЧИВОСТЬ ПРОГРАММИСТОВ



Исследование проведено по аватарам разработчиков с GitHub

** GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки*

ПЕРСПЕКТИВЫ

1. Frontend разработка
2. Backend разработка
3. Мобильная разработка
4. Десктопные приложения, VR, полеты в космос… ☺

Ты не можешь для всего использовать javascript

а вот здесь ты неправ, парень

НА JAVASCRIPT ЛЕГКО НАЧАТЬ РАЗРАБАТЫВАТЬ


VANILLA JS

ЧТО ТАКОЕ JAVASCRIPT?

Википедия:

JavaScript - мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript (стандарт ECMA-262).

Основные архитектурные черты:

- динамическая типизация,
- слабая типизация,
- автоматическое управление памятью,
- прототипное программирование,
- функции как объекты первого класса.


Вы понимаете, о чем говорят эти люди?
Лично я - нет.

ПОЛЕЗНАЯ ИНФОРМАЦИЯ ПО СЛАЙДАМ

Цветовые обозначения:

Пример

Об этом будет рассказано дальше

Практика

ПОЛЕЗНАЯ ИНФОРМАЦИЯ ПО ЯЗЫКУ

/ / - комментарий однострочный

/ * комментарий
многострочный *** /**

JavaScript – язык регистрозависимый (example и Example для него разные вещи)

{ } – блок кода

HELLO WORLD

```
console.log('Hello world')
```

Среды выполнения JavaScript: браузер и NodeJs

Практика:

1) Напишем “Hello World” в консоли браузера

2) Напишем “Hello World” в node js

ПЕРЕМЕННЫЕ

Чтобы хранить информацию,
используются **переменные.**

Переменная состоит из имени и выделенной области памяти, которая ему соответствует.

Для объявления или, другими словами, создания переменной используется ключевое слово **var.**

ТИПЫ ДАННЫХ

- Number

  var n = 5; n = 23.3;

- String

  var str = “Строка в двойных кавычках"; str = ' Строка в одинарных кавычках ';

- Boolean

  var flag = true; flag = false;

- Null

  var a = null

- Undefined

  var b;

- Object

  var user = { name: “Иван" , surname: “Иванов" };

typeof - возвращает тип аргумента

ОБЪЕКТЫ

Объект – структура данных, в которой можно хранить любые данные в формате ключ-значение.

```
var person = {}; // пока пустой


person.name = 'Вася';
person.age = 25;
```

К свойству объекта можно обращаться также используя квадратные скобки объект['свойство']:

```
person['name'] = 'Вася'; // то же что и person.name = 'Вася`
```

Для перебора всех свойств из объекта используется цикл по свойствам for..in

```
for (var key in obj) {
        // этот код будет вызван для каждого свойства объекта
        // ..и выведет    имя свойства и его значение
        alert( "Ключ: " + key + " значение: " +obj[key] );
 }
```

ОПЕРАТОРЫ

У операторов есть своя терминология, которая используется во всех языках программирования.

Оператор — конструкция в языках программирования, аналогичная по записи математическим операциям, то есть специальный способ записи некоторых действий.

Операнд – то, к чему применяется оператор.

7 * 8 – оператор умножения с левым и правым операндами. Другое название: «аргумент оператора».

Унарным называется оператор, который применяется к одному выражению. Например, оператор унарный минус "-"меняет знак числа на противоположный.

```
var x = 1; x = -x; console.log( x ); // -1,
```

Бинарным называется оператор, который применяется к двум операндам. Тот же минус существует и в бинарной форме:

```
var x = 1, y = 3; console.log( y - x );  // 2
```

ОСНОВНЫЕ ОПЕРАТОРЫ

Сложение +, умножение *, вычитание и так далее.

```
var sum = 5 + 7 //13
```

Сложение строк

```
var a = «сложение" + "строк"; console.log( a ); // сложение строк
console.log( '1' + 2 ); // "12«
```

Присваивание

```
var x = 2 * 2 + 1 //5
```

ОБЪЕКТЫ: ПЕРЕДАЧА ПО ССЫЛКЕ

Фундаментальным отличием объектов от примитивов, является их хранение и копирование «по ссылке».

Обычные значения: строки, числа, булевы значения, null/undefined при присваивании переменных копируются целиком или, как говорят, «по значению».

С объектами – всё не так.

В переменной, которой присвоен объект, хранится не сам объект, а «адрес его места в памяти», иными словами – «ссылка» на него.

Вот как выглядит переменная, которой присвоен объект:

```
var user = {
  name: "Вася"
};
```

Внимание: объект – вне переменной. В переменной – лишь «адрес» (ссылка) для него.

ОБЪЕКТЫ: ПЕРЕДАЧА ПО ССЫЛКЕ

При копировании переменной с объектом – копируется эта ссылка, а объект по-прежнему остается в единственном экземпляре.

Например:

```
var user = { name: "Вася" }; // в переменной - ссылка

var admin = user; // скопировали ссылку
```

Получили две переменные, в которых находятся ссылки на один и тот же объект.

Так как объект всего один, то изменения через любую переменную видны в других переменных.

ОПЕРАТОРЫ СРАВНЕНИЯ

- Больше/меньше: **a > b, a < b.**
- Больше/меньше или равно: **a >= b, a <= b.**
- Равно **a == b**. Для сравнения используется два символа равенства '='. Один символ a = b означал бы присваивание.
- «Не равно». В математике он пишется как ≠, в JavaScript – знак равенства с восклицательным знаком перед ним !=.

Как и другие операторы, сравнение возвращает значение. Это значение имеет **логический** тип.

Существует всего два логических значения:

true – имеет смысл «да», «верно», «истина».

false – означает «нет», «неверно», «ложь».

ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

- Для операций над логическими значениями в JavaScript есть || (ИЛИ), && (И) и ! (НЕ).
- Хоть они и называются «логическими», но в JavaScript могут применяться к значениям любого типа и возвращают также значения любого типа.

Операция	Назначение	Пример
&&	*Логическое И*	(R>2) && (R<20)
||	*Логическое ИЛИ*	(L= =12) || (L>15)
!	*Логическое НЕ*	Tab != 13

ДРУГИЕ ОПЕРАТОРЫ

Побитовые операторы:

- AND(и) (&)
- OR(или) (|)
- XOR(побитовое исключающее или) (^)
- NOT(не) (~)
- LEFT SHIFT(левый сдвиг) (<<)
- RIGHT SHIFT(правый сдвиг) (>>)
- ZERO-FILL RIGHT SHIFT(правый сдвиг с заполнением нулями) (>>>)

Другие операторы:

- Остаток от деления (%)
- Инкремент и декремент (++, --)
- И другие

УСЛОВНЫЙ ОПЕРАТОР IF

Иногда, в зависимости от условия, нужно выполнить различные действия. Для этого используется оператор if.

```
var year = 2017;

if (year < 2017) {
  console.log( 'Это слишком рано..' );
} else if (year > 2017) {
 console.log( 'Это поздновато..' );
} else {
 console.log( 'Да, точно сейчас этот год!' );
}
```

SWITCH

Конструкция switch заменяет собой сразу несколько if

```
var a = 2 + 2;


switch (a) {
  case 3:
    console.log( 'Маловато' );
    break;
  case 4:
     console.log( 'В точку!' );
     break;
  case 5:
     console.log( 'Перебор' );
     break;
  default:
    console.log( 'Я таких значений не знаю' );
}
```

ЦИКЛ WHILE

Цикл while выполняется пока условие верно

```
var i = 0;
while (i < 3) {
  console.log( i );
  i++;
}
```

ЦИКЛ FOR

При написании скриптов зачастую встает задача сделать однотипное действие много раз.
Для многократного повторения одного участка кода – предусмотрены **циклы.**
Чаще всего применяется **цикл for.**

```
var i;
for (i = 0; i < 3; i++) {
        console.log( i );
}
```

ФУНКЦИИ

Зачастую нам надо повторять одно и то же действие во многих частях программы.

Чтобы не повторять один и тот же код во многих местах, придуманы функции. Функции являются основными «строительными блоками» программы

```
function showMessage() {
  console.log( 'Привет всем присутствующим!' );
}

showMessage();
```

ПАРАМЕТРЫ ФУНКЦИИ

Параметры

При вызове функции ей можно передать данные, которые та использует по своему усмотрению.

Например, этот код выводит два сообщения:

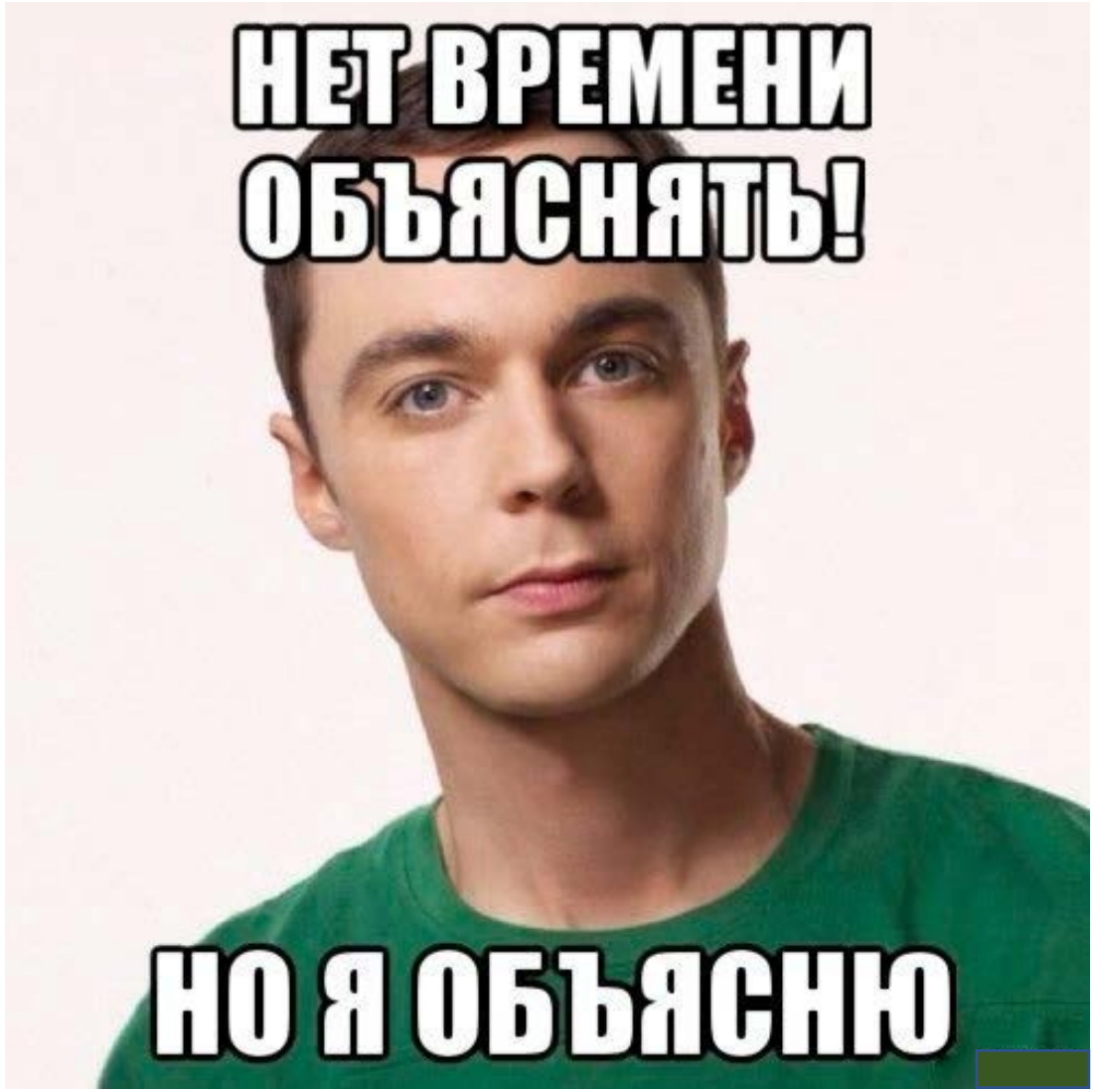
```
function showMessage(from, text) { // параметры from, text

from = "** " + from + " **";

alert(from + ': ' + text);
}

showMessage('Маша', 'Привет!');
showMessage('Маша', 'Как дела?');
```

Параметры копируются в локальные переменные функции


НЕТ ВРЕМЕНИ
ОБЪЯСНЯТЬ!
НО Я ОБЪЯСНЮ

ОБЛАСТИ ВИДИМОСТИ

Локальные переменные

Функция может содержать **локальные** переменные, объявленные через var. Такие переменные видны только внутри функции:

```
function showMessage() {
  var message = 'Привет, я - Вася!'; // локальная переменная

 console.log( message );
}

showMessage(); // 'Привет, я - Вася!'

console.log( message );
```

ОБЛАСТИ ВИДИМОСТИ

- Блоки if/else, switch, for, while, do..while не влияют на область видимости переменных.
- Неважно, где именно в функции и сколько раз объявляется переменная. Любое объявление срабатывает один раз и распространяется на всю функцию.

Внешние переменные

Функция может обратиться ко внешней переменной,

```
var userName = 'Вася';

function showMessage() {
  var message = 'Привет, я ' + userName;
  console.log(message);
}

showMessage(); / / Привет, я Вася например:
```

ВОЗВРАТ ЗНАЧЕНИЯ

Функция может возвратить результат, который будет передан в вызвавший её код.

```
function calcD(a, b, c) {
  return b*b - 4*a*c;
}

var test = calcD(-4, 2, 1);
console.log(test); // 20
```

Для возврата значения используется директива return.

Она может находиться в любом месте функции. Как только до неё доходит управление – функция завершается и значение передается обратно.

В случае, когда функция не вернула значение или return был без аргументов, считается что она вернула undefined.

ПРАКТИКА (VANILLA JS)

ПРАКТИКА (VANILLA JS)

Написать функцию, которая принимает на вход 2 объекта и возвращает true, если все их свойства равны, и false если нет.


ES6

ES6

01.

Let и const

Ключевое слово let позволяет объявлять переменные с ограниченной областью видимости — только для блока {...}, в котором происходит объявление. Это называется блочной областью видимости. Вместо ключевого слова var, которое обеспечивает область видимости внутри функции, стандарт ES6 рекомендует использовать let.

```
var a = 2; { let a = 3; console.log(a); // 3 } console.log(a); // 2
```

01.

Другой формой объявления переменной с блочной областью видимости является ключевое слово const. Оно предназначено для объявления переменных (констант), значения которых доступны только для чтения. Это означает не то, что значение константы неизменно, а то, что идентификатор переменной не может быть переприсвоен.

```
{
  const ARR = [5, 6];
  ARR.push(7);
  console.log(ARR); // [5,6,7]
  ARR = 10; // TypeError
  ARR[0] = 3; // значение можно менять
  console.log(ARR); // [3,6,7]
}
```

02.

Arrow function

Стрелочные функции представляют собой сокращённую запись функций в ES6. Стрелочная функция состоит из списка параметров (...), за которым следует знак => и тело функции.

```
// Классическое функциональное выражение
let addition = function(a, b) {
return a + b;
};

// Стрелочная функция
let addition = (a, b) => a + b;
```

03.

Многое другое

http://es6-features.org/

BABEL

Babel.JS – это транспайлер, переписывающий код на ES6 в код на предыдущем стандарте ES5.

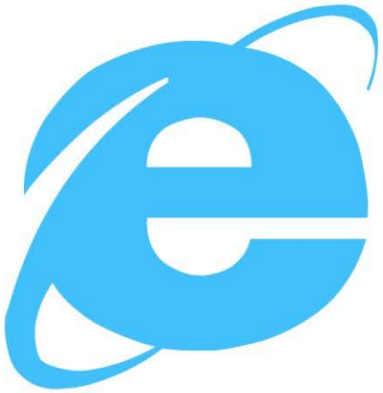
Он состоит из двух частей:

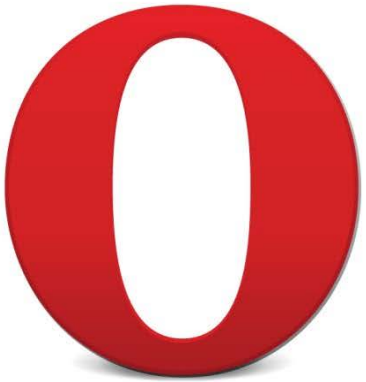
- Собственно транспайлер, который переписывает код.
- Полифилл, который добавляет методы Array.from, String.prototype.repeat и другие.

На странице https://babeljs.io/repl/ можно поэкспериментировать с транспайлером: слева вводится код в ES6, а справа появляется результат его преобразования в ES5.

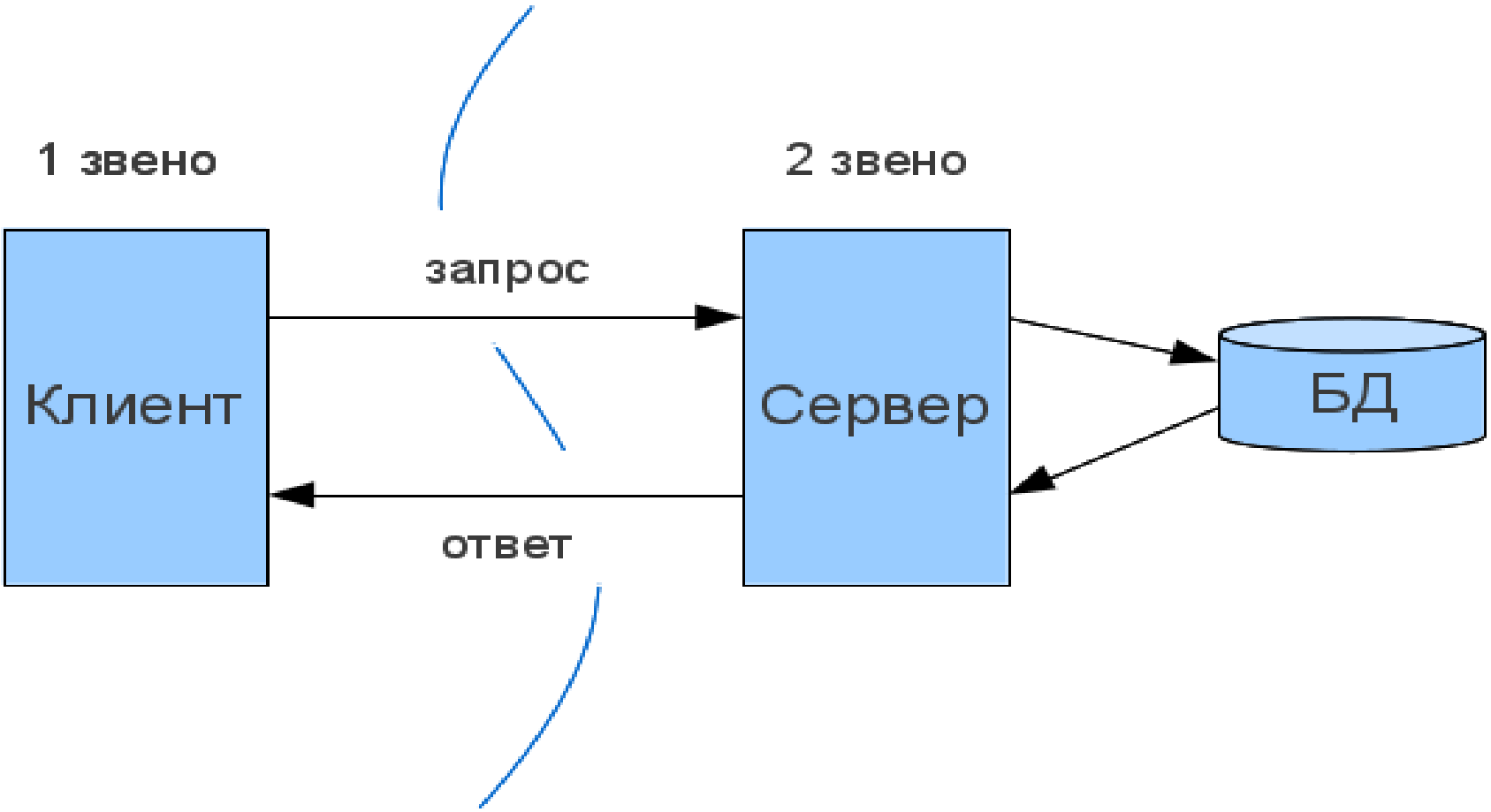
КЛИЕНТСКИЙ JS

КЛИЕНТСКИЙ JAVASCRIPT





КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА



Клиентская часть реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него.

Серверная часть получает запрос от клиента, выполняет вычисления, после этого формирует веб-страницу и отправляет её клиенту по сети с использованием протокола HTTP.

HTML

HTML (от англ. HyperText Markup Language — «язык гипертекстовой разметки») — стандартизированный язык разметки документов во Всемирной паутине. Большинство веб-страниц содержат описание разметки на языке HTML. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Любой документ на языке HTML представляет собой набор элементов, причём начало и конец каждого элемента обозначается специальными пометками — тегами.

Элементы могут быть пустыми, то есть не содержащими никакого текста и других данных (например, тег перевода строки
). В этом случае обычно не указывается закрывающий тег. Кроме того, элементы могут иметь атрибуты, определяющие какие-либо их свойства (например, id или класс элемента)

Справочник html тэгов: http://htmlbook.ru/

ПРИМЕР HTML

```
<!DOCTYPE html>
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>HTML Document</title>
</head>
<body>
        <p>
                Здесь написан
                <span class= " test " >тестовый текст</span>
        </p>
</body>
</html>
```

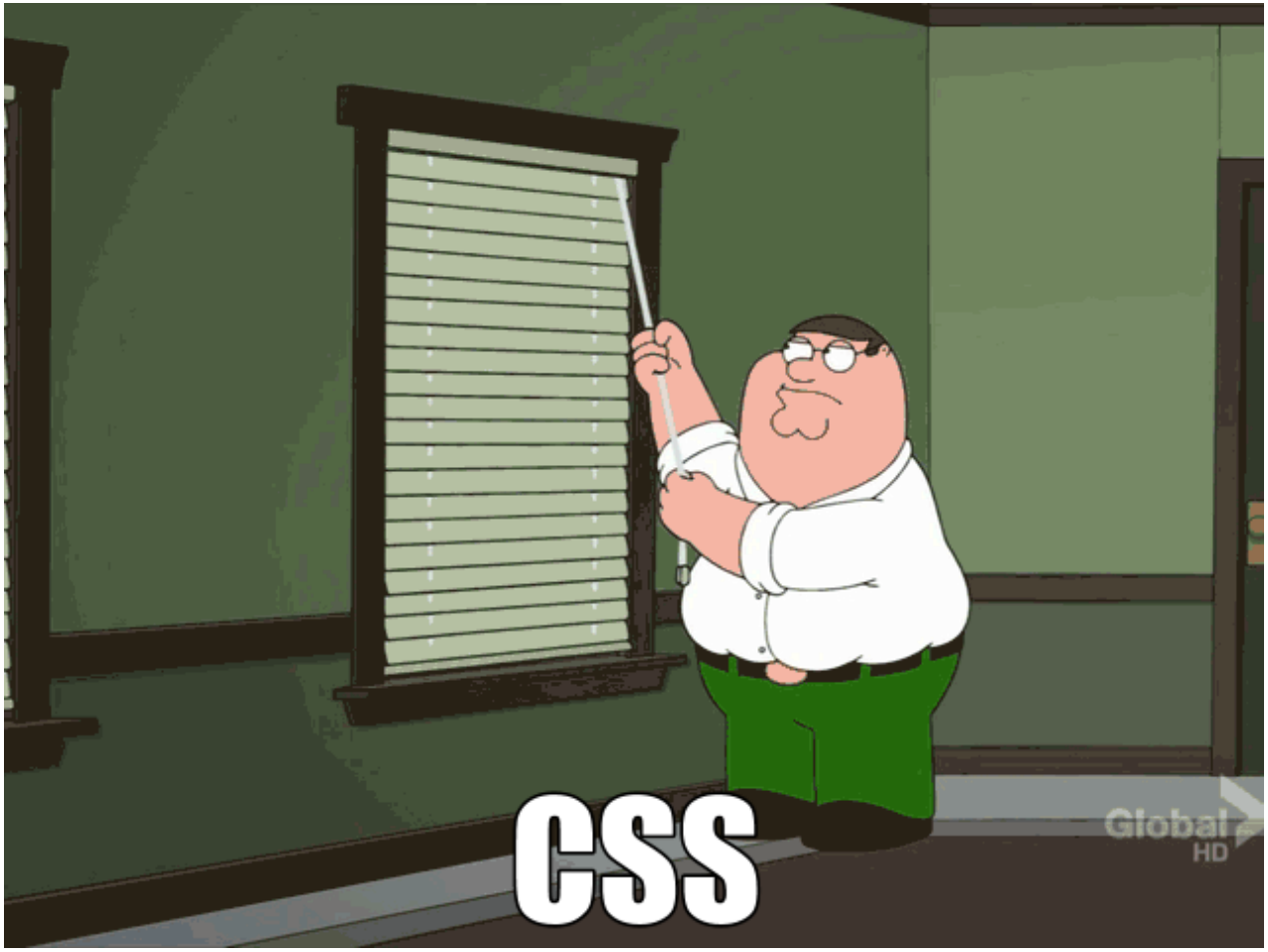
CSS

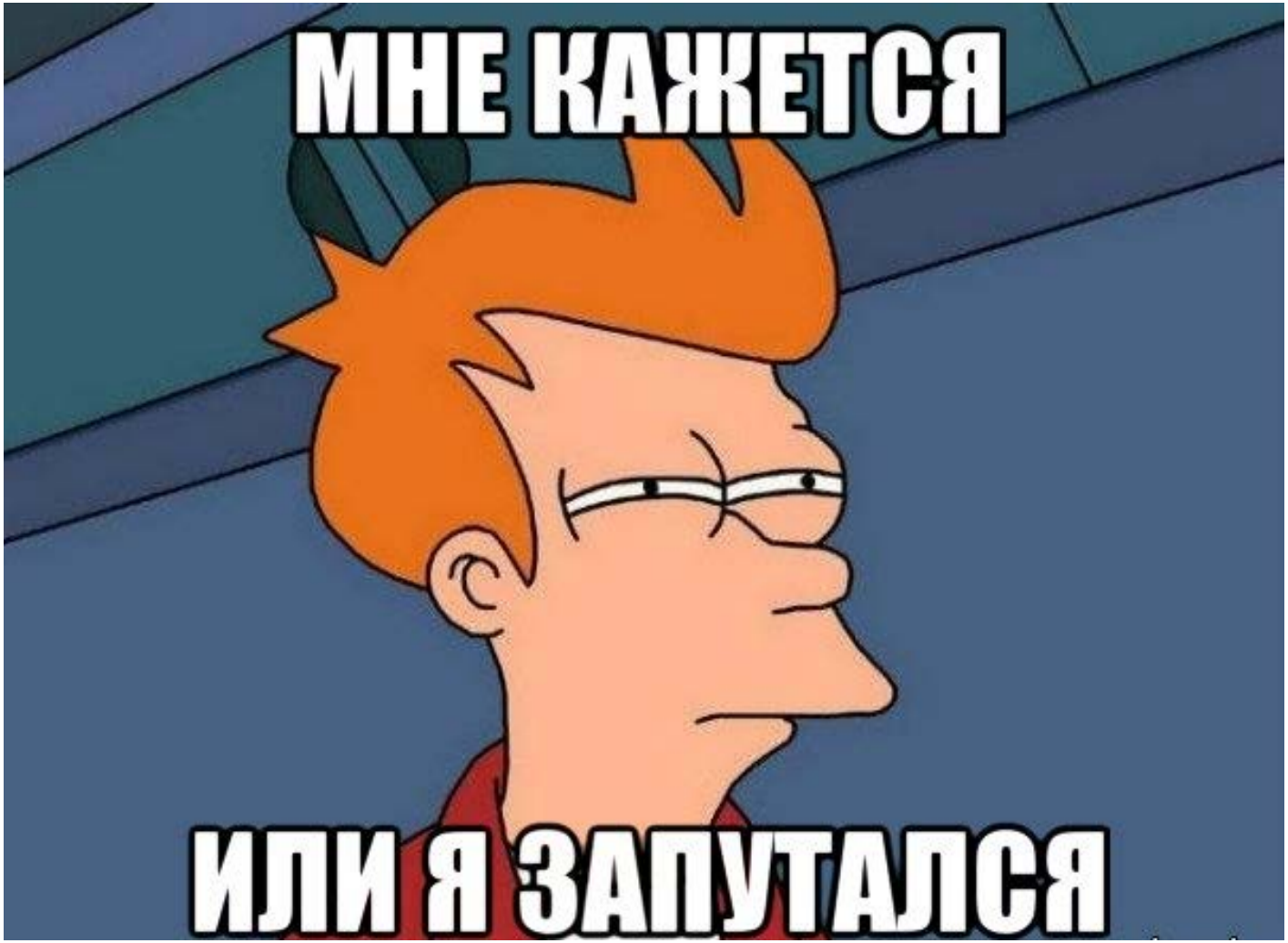
CSS (/siːɛsɛs/ англ. Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Пример CSS:

```
p {
  font-size: 20pt;
  color: red;
  background: white;
}
.test {
  color: red;
  background-color: yellow;
  font-weight: bold;
}
```

А ВООБЩЕ CSS ЭТО СЛОЖНО




МНЕ КАЖЕТСЯ
ИЛИ Я ЗАПУТАЛСЯ

ЗОНЫ ОТВЕТСТВЕННОСТИ

- HTML – структура документа
- CSS – внешний вид и дизайн
- JavaScript – логика и поведение

ПОДКЛЮЧЕНИЕ JS В HTML

Обычно javascript-код в HTML подключают следующим образом:

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

При этом файл /my/script.js содержит javascript-код, который иначе мог бы находиться просто внутри тега <script>.

Это очень удобно, потому что один и тот же файл со скриптами можно подключать на разных страницах. При правильных настройках сервера браузер закеширует его и не будет скачивать каждый раз заново.

Чтобы подключить несколько скриптов - используйте несколько таких тегов:

```
<script src="/js/script1.js"></script>
<script src="/js/script2.js"></script>
```

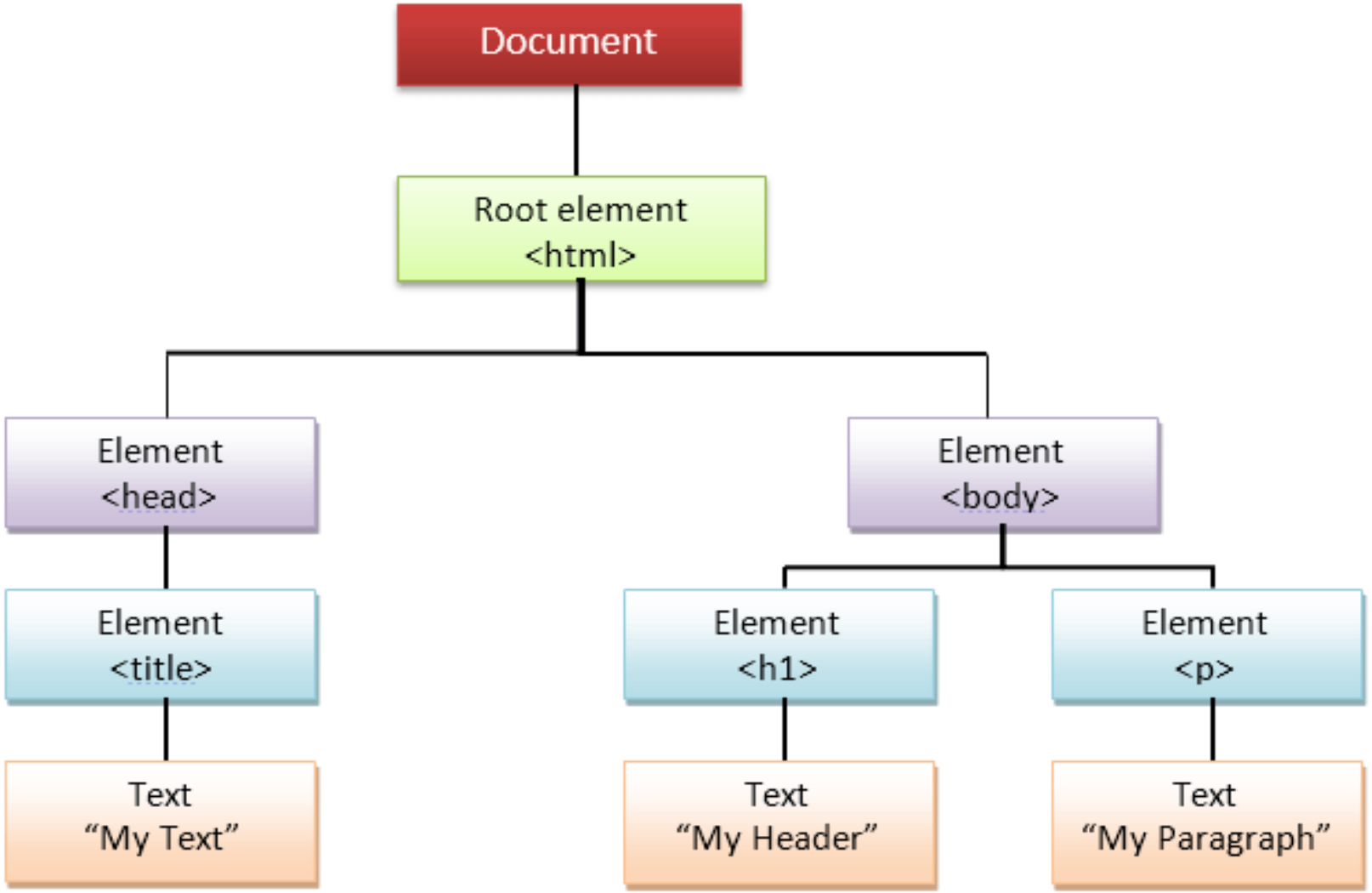
DOM

Основным инструментом работы и динамических изменений на странице является DOM (Document Object Model) - объектная модель, используемая для XML/HTML-документов.

Согласно DOM-модели, документ является иерархией.
Каждый HTML-тег образует отдельный элемент-узел, каждый фрагмент текста - текстовый элемент, и т.п.

Проще говоря, DOM - это представление документа в виде дерева тегов. Это дерево образуется за счет вложенной структуры тегов плюс текстовые фрагменты страницы, каждый из которых образует отдельный узел.

```
<html>
  <head>
    <title>
       My text
    </title>
  </head>
  <body>
    <h1>
        My header
    </h1>
    <p>
        My paragraph
    </p>
  </body>
</html>
```


Document
Root element
<html>
Element
<head>
Element
<body>
Element
<title>
Element
<h1>
Element
<p>
Text
“My Text”
Text
“My Header”
Text
“My Paragraph”

МАНИПУЛЯЦИИ DOM

Рассмотрим основные способы изменять DOM.

Чтобы создать новый элемент - используется метод document.createElement(тип).

```
var newDiv = document.createElement('div')
```

Добавить новый элемент к детям существующего элемента можно методом appendChild, который в DOM есть у любого тега.

```
// элемент-список UL
var list = document.getElementById('list')

// новый элемент
var li = document.createElement('LI')
li.innerHTML = 'Новый элемент списка'
// добавление в конец
list.appendChild(li)
```

Чтобы убрать узел из документа - достаточно вызвать метод removeChild из его родителя.

```
list.removeChild(elem)
```

ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ

01.

alert

Выводит модальное окно с сообщением. Посетитель не сможет продолжить работу, пока не нажмет на кнопку "ОК" в модальном окне.

```
alert('Добрый день')
```

02.

prompt

Выводит сообщение в окне с текстовым полем и двумя кнопками: "ОК" и "ОТМЕНА".

Возвращает введенное значение или null, если посетитель нажал на кнопку "ОТМЕНА".

Как и в alert, окно - модальное, то есть посетитель не может делать ничего другого, пока не выберет одну из кнопок.

```
var years=prompt('Сколько вам лет?',100)
alert('Вам '+years+' лет!')
```

ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ

03.

confirm

Выводит сообщение в окне с двумя кнопками: "ОК" и "ОТМЕНА".

Возвращает true/false в зависимости от того, куда нажмет посетитель.

Как и alert, окно - модальное, то есть посетитель не может делать ничего другого, пока не выберет одну из кнопок.

```
if (confirm("Сказать привет?")) {
  alert("Привет!")
} else {
  alert("Вы нажали кнопку отмена")
}
```

СОБЫТИЯ

Практически все JavaScript-приложения выполняют те или иные действия, откликаясь на различные события.

Событие - это сигнал от браузера о том, что что-то произошло.

Есть множество самых различных событий.

DOM-события, которые инициируются элементами DOM. Например, событие click происходит при клике на элементе, а событие mouseover - когда указатель мыши появляется над элементом,

События окна. Например событие resize - при изменении размера окна браузера,

Другие события, например load, readystatechange. Они используются, скажем, в технологии **AJAX.**

УСТАНОВКА СОБЫТИЯ

Для этого нужно:

- получить элемент
- назначить обработчик

Пример:

```
document.getElementById('myElement').onclick = function() {
        alert('Спасибо')
}
<input id="myElement" type="button" value="Нажми меня"/>
```

ВЗАИМОДЕЙСТВИЕ С СЕРВЕРОМ

AJAX, или, более длинно, Asynchronous Javascript And Xml - технология для взаимодействия с сервером без перезагрузки страниц.

За счет этого уменьшается время отклика и веб-приложение по интерактивности больше напоминает десктоп.

```
var request = $.ajax({ url: "script.php", type: "POST", data: {id : menuId},
dataType: "html" });
```

```
request.done(function(msg) { $("#log").html( msg ); });

request.fail(function(jqXHR, textStatus) { alert( "Request failed: " +
textStatus ); });
```

JQUERY

jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX.

$("#id") –получить элемент по id

```
$( document ).ready(function() {
// код который нужно выполнить по загрузке страницы
}
);
```

$('#id').on(‘click‘, handler– повесить обработчик handler на событие click элемента с id “id”

ПРАКТИКА (КЛИЕНТСКАЯ ЧАСТЬ)

ПРАКТИКА (КЛИЕНТСКАЯ ЧАСТЬ)

Создать страницу онлайн-теста, на который есть любые вопросы по JavaScript и поля для ввода ответа на них.

Так же на странице должна быть кнопка, по нажатию которой происходит отправка ответов на сервер по адресу

http://localhost:3000

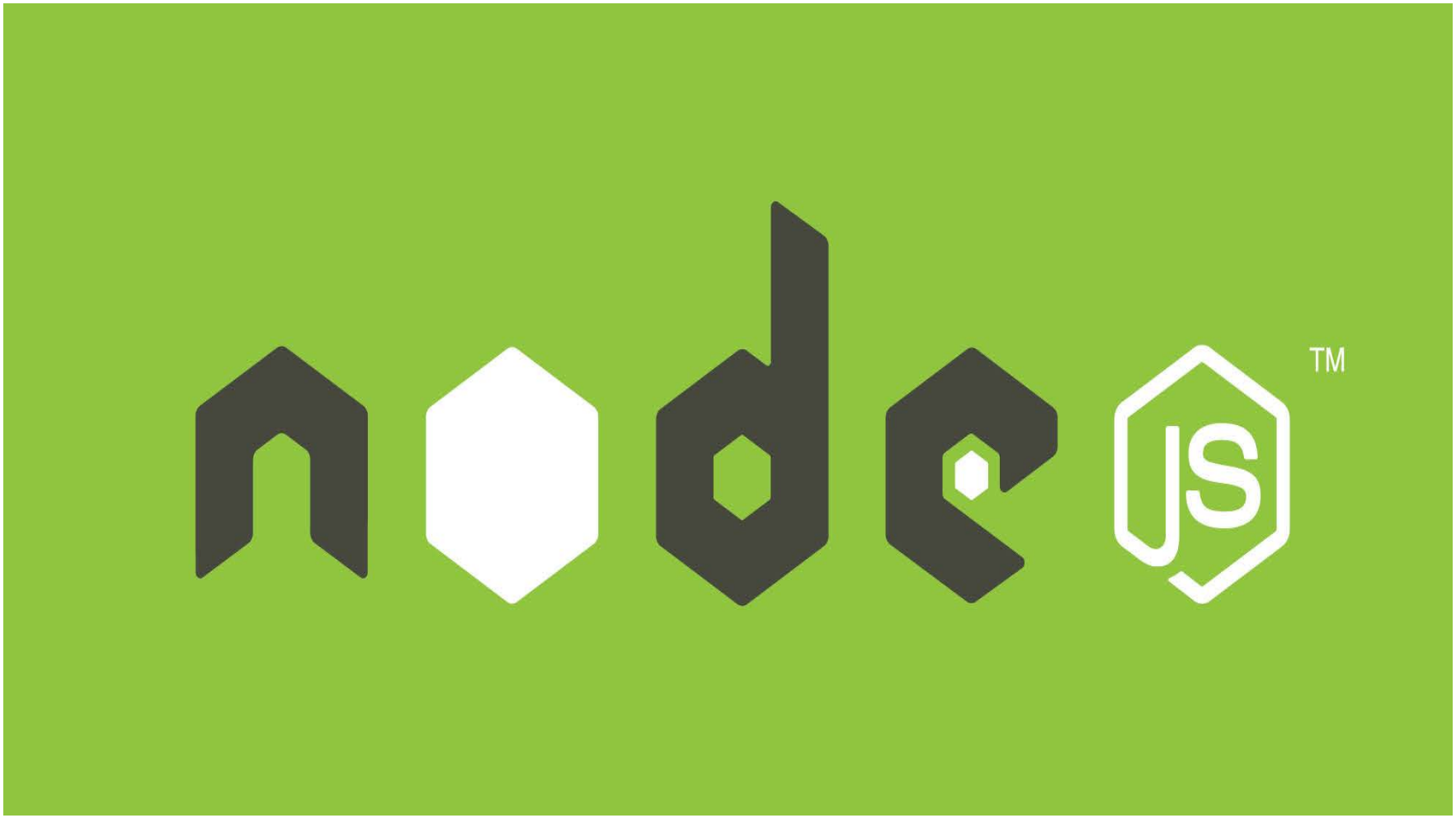
ИНТЕРЕСНЫЕ ВОЗМОЖНОСТИ БРАУЗЕРА

HTML 5

- Geolocation
- Canvas
- Video, audio
- Web storage
- Web sockets
- и многое другое...

СЕРВЕРНЫЙ JS

СЕРВЕРНЫЙ JS



ЧТО ТАКОЕ NODE JS?

Node или Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения.

Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода.

Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и Mac OS) и даже программировать микроконтроллеры (например, tessel и espruino).

NPM

В состав Node.js входит собственный установщик пакетов **npm.** Установка производится при помощи команды:

npm install <packagename>

Все доступные для установки пакеты и их краткое описание:

npm search

Этой же командой можно производить выборочный поиск пакетов.

PACKAGE.JSON

Файл package.json содержит в себе информацию о вашем приложении: название, версия, зависимости и тому подобное.

Пример :

```
{ "name": “test",
  "description": “test app",
  "version": "1.0.0",
  "devDependencies": {
    "coffee-script": "~1.6.3"
  },
  "scripts": {
    "prepare": "coffee -o lib/ -c src/waza.coffee"
  },
  "main": "lib/waza.js"
}
```


app
node_modules
bower
package.json
gulp
.Gitignore

EXPRESS

Фреймворк Node.js, предоставляющий набор функций для разработки мобильных и веб-приложений. Позволяет отвечать на HTTP-запросы, используя промежуточное ПО.

npm install --save express

ПРОСТОЙ СЕРВЕР

server.js

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

Запуск

```
node server.js
```

ПРАКТИКА (СЕРВЕРНАЯ ЧАСТЬ)

ПРАКТИКА (СЕРВЕРНАЯ ЧАСТЬ)

Написать серверный код, который слушает на 3000 порту клиентскую часть из предыдущего практического задания.

И сравнивает с правильными ответами на сервере (можно взять функцию сравнения из практики в блоке Vanilla JS)

Добавить в клиентскую часть вывод alert(‘Тест пройден’), если ответы совпали.

ДРУГИЕ ТЕХНОЛОГИИ

- KOA
- MongoDB
- Socket.io
- Webpack
- Jasmine

ДРУГИЕ ВОЗМОЖНОСТИ


Mobile
Development

МОБИЛЬНАЯ РАЗРАБОТКА

React Native

Позволяет писать приложения для Android и iOS на JavaScript.

Минус: технология еще молодая, иногда чтобы довести приложение до ума приходится все таки лезть в java и objective C

Кто уже использует: **Facebook, Instagram, Сбербанк**

ДЕСКТОПНАЯ РАЗРАБОТКА

Electron - фреймворк, разработанный GitHub.

Позволяет разрабатывать нативные графические приложения для настольных операционных систем с помощью веб-технологий.

Фреймворк включает в себя Node.js, для работы с back-end, и библиотеку рендеринга из Chromium

Кто уже использует: **Atom, Slack, Github Desktop.**


Oculus VR
HD Prototype

VR

WebVR

WebVR API — это программный интерфейс для работы с устройствами. Он ничего не знает про 3D графику. Работа с графикой, отрисовка сцены, установка источников света и все прочее лежит на суровых плечах программистов. WebVR API всего лишь позволяет абстрагировать доступ к устройствам.

Данное API предоставляет инструменты для рендеринга картинки, для получения информации об устройстве, его возможностях и технических характеристиках, но саму картинку и 3D мир нужно рисовать, используя уже ставшие привычными веб-технологии, такие как: HTML, CSS, WebGL, Canvas, etc…

ReactVR

React VR использует существующие технологии, такие как WebGL и WebVR. Важно отметить, что фреймворк предназначен не для написания сложных VR-игр и иных приложений.

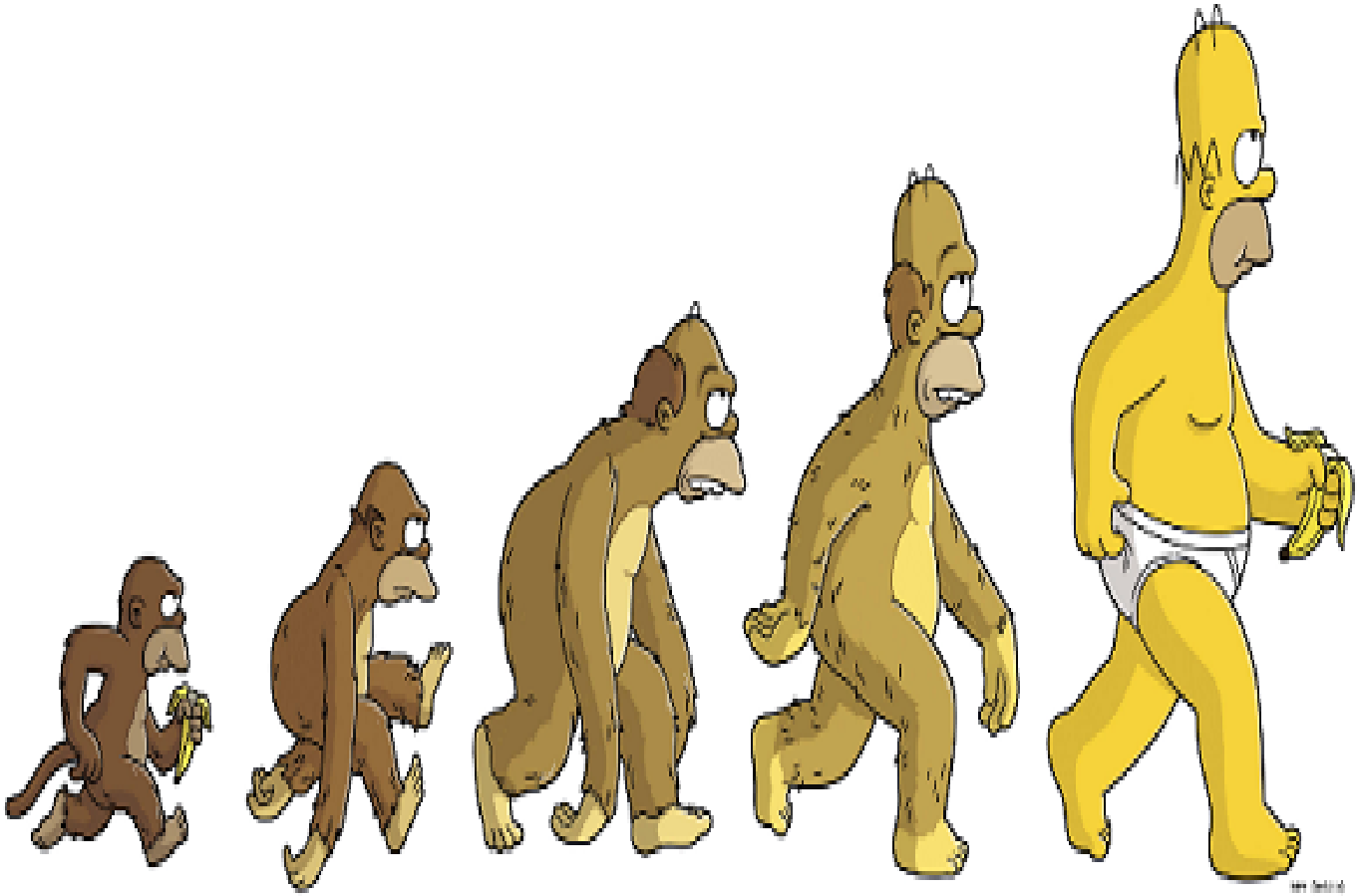
Хотя он и даёт возможность добавлять 3D-модели, его основной целью является упрощение сочетания 360-градусных панорам и двумерных пользовательских интерфейсов, текстов и изображений.


WebAssembly
and the faster web..
A Google, Microsoft, Apple and Mozilla initiative

WEB ASSEMBLY

WebAssembly, или wasm — это экспериментальный эффективный низкоуровневый язык программирования, выполняющийся в браузере, который в данный момент находится в разработке.

Первоначальной целью языка является поддержка C/C++, тем не менее, также предполагается поддержка других языков. WebAssembly представляет собой переносимое абстрактное синтаксическое дерево, обеспечивающее как более быстрый парсинг, так и более быстрое выполнение кода, чем JavaScript. Более подробно об этом языке можно почитать на hacks.mozilla.org и Хабре.


MACHINE
ASSEMBLY
PROCEDURAL
OBJECT ORIENTED
FUNCTIONAL

TypeScript и Elm

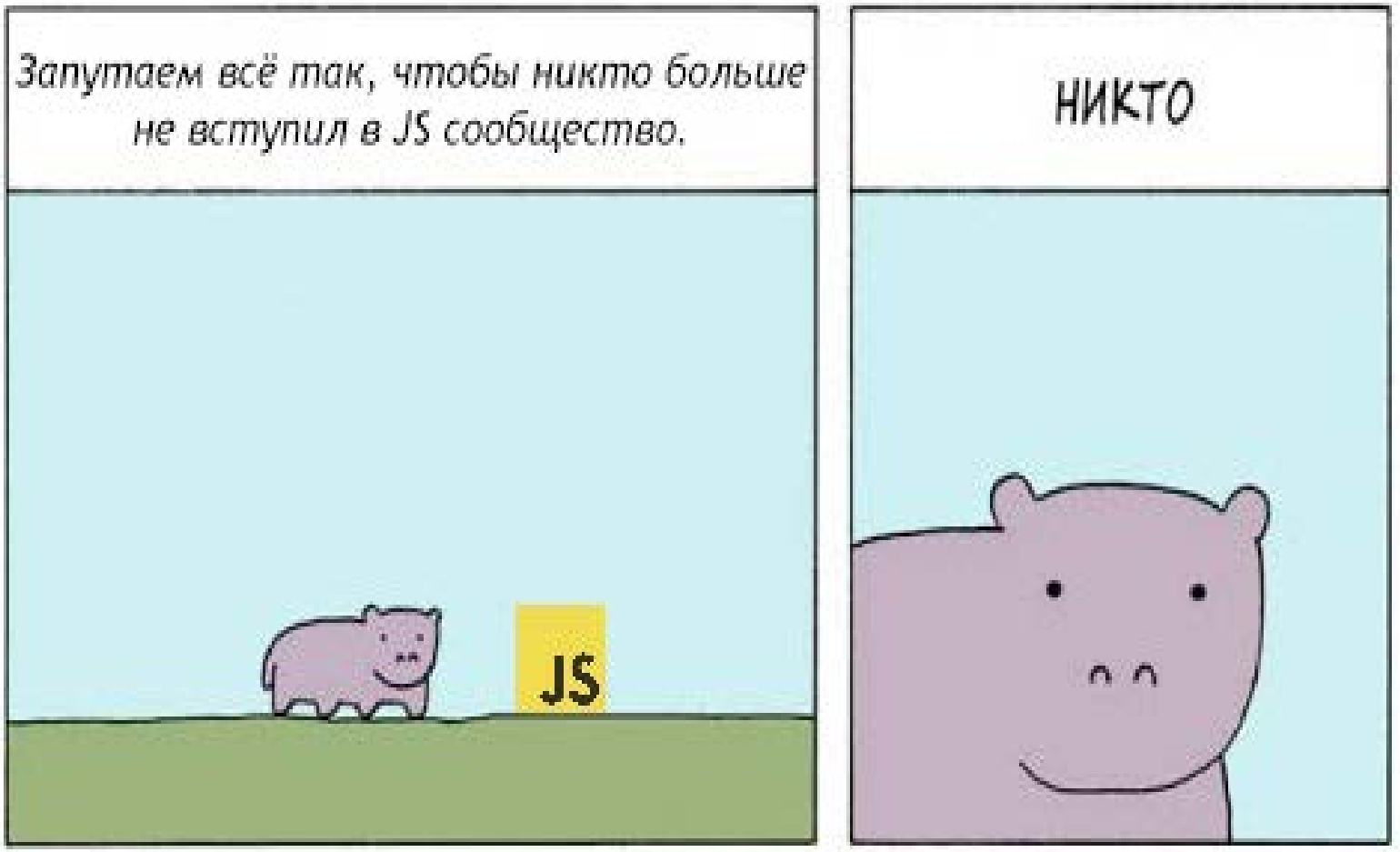
TypeScript является надмножеством JavaScript.

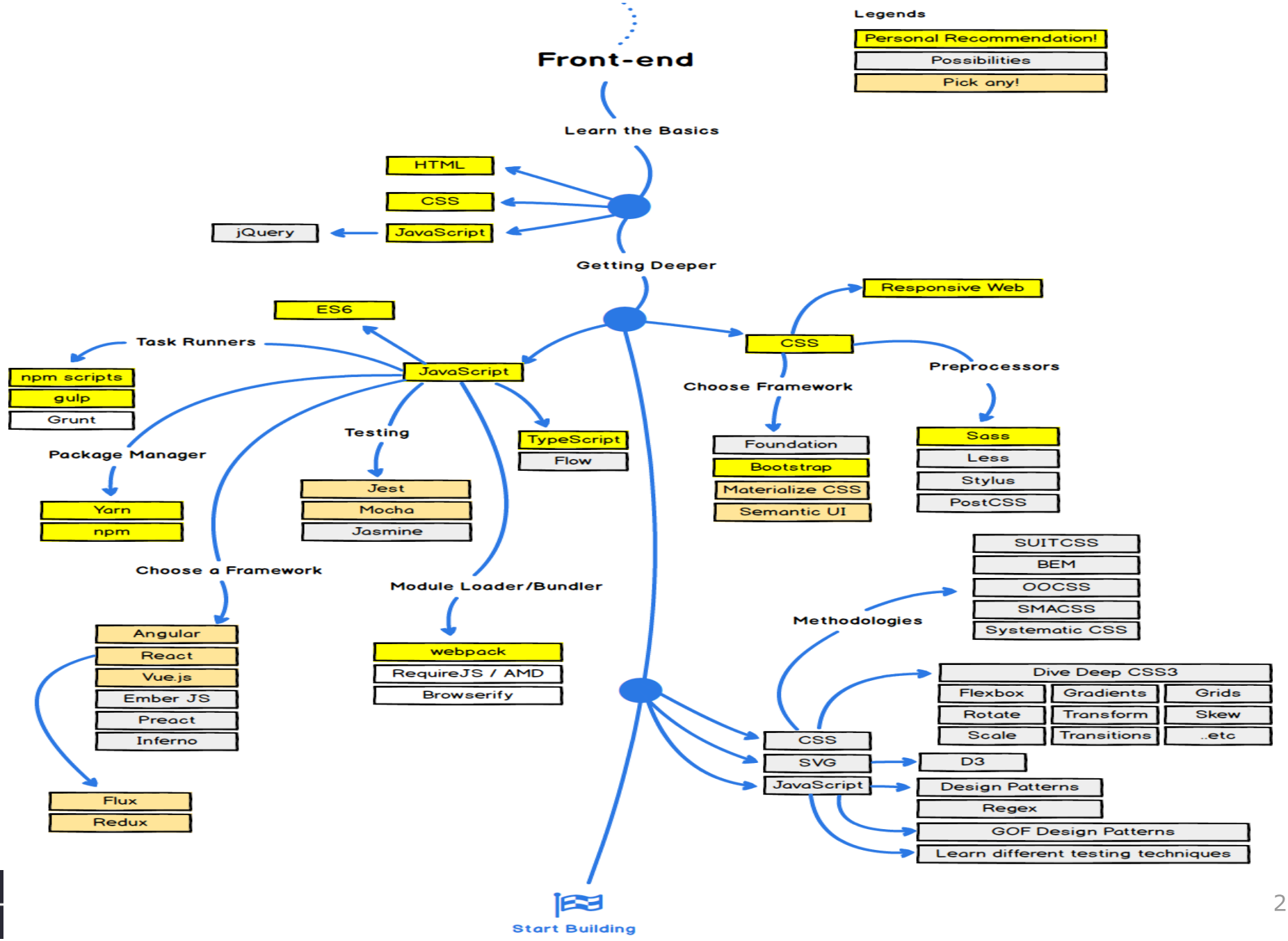
Отличается от JavaScript возможностью явного статического назначения типов, поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что призвано повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использования кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции, и, возможно, скорость выполнения программ.

Elm — функциональный язык, предназначенный для декларативного создания графических интерфейсов, основанных на браузере.

Elm предоставляет возможность описывать графические интерфейсы, не выходя за рамки функциональной парадигмы, используя функционально-реактивный стиль программирования.

САМООБУЧЕНИЕ




Front-end
Legends
Personal Recommendation!
Possibilities
Pick any!
Learn the Basics
HTML
CSS
JavaScript
jQuery
Getting Deeper
ES6
Task Runners
npm scripts
gulp
Grunt
JavaScript
Package Manager
Yarn
npm
Testing
Jest
Mocha
Jasmine
TypeScript
Flow
Choose a Framework
Angular
React
Vue.js
Ember JS
Preact
Inferno
Flux
Redux
Module Loader/Bundler
webpack
RequireJS / AMD
Browserify
CSS
Responsive Web
Choose Framework
Foundation
Bootstrap
Materialize CSS
Semantic UI
Preprocessors
Sass
Less
Stylus
PostCSS
Methodologies
SUITCSS
BEM
OOCSS
SMACSS
Systematic CSS
CSS
Dive Deep CSS3
Flexbox
Gradients
Grids
Rotate
Transform
Skew
Scale
Transitions
..etc
SVG
D3
JavaScript
Design Patterns
Regex
GOF Design Patterns
Learn different testing techniques
Start Building

ПОЛЕЗНЫЕ РЕСУРСЫ

https://javascript.info/

https://learn.javascript.ru/

https://developer.mozilla.org/en/docs/Web/JavaScript

И конечно же официальная документация для различных библиотек и фреймворков ☺

ВАШИ ВОПРОСЫ?

СПАСИБО ЗА ВНИМАНИЕ!