

# Введение в JavaScript

Старков Дима

Какие языки вы знаете?

# Один язык чтобы править всеми

Веб-страницы



Front-End

Серверные приложения



Back-End

Настольные приложения



Desktop

Мобильные приложения



Mobile

Интернет вещей

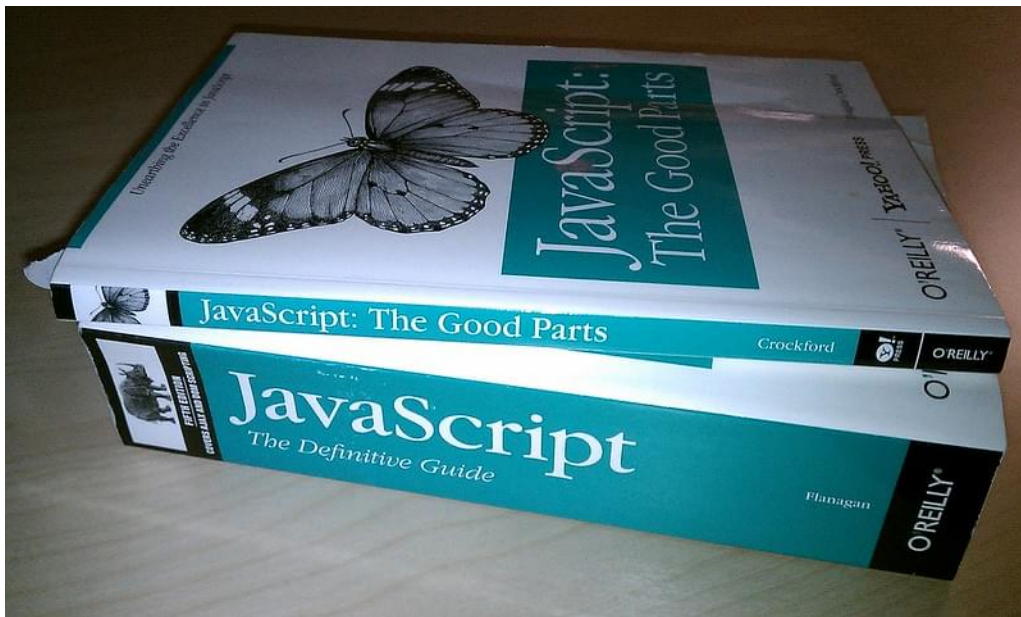


IoT

Машинное обучение



ML



Почему так произошло?

# История JavaScript



«Веб должен стать более динамичным»

Нужен легкий скриптовый язык

# Кандидаты

Oak

# Кандидаты

~~Oak~~ Java



Python



Tcl



Scheme





# Брендон Айк

Отец JavaScript



# Брендон Айк

Отец JavaScript

... он должен был быть написан  
за 10 дней, а иначе мы бы имели  
что-то похуже JavaScript ...

# Брендон Айк

Отец JavaScript

... мы должны были двигаться  
очень быстро, так как знали, что  
Microsoft идет за нами ...

# Брендон Айк

Отец JavaScript

... JavaScript должен «выглядеть как Java», только поменьше, быть братом-простаком Java ...

# Кандидаты

~~Oak~~ Java



Python



Tcl



Scheme



Mocha – LiveScript – JavaScript

# Почему JavaScript?

Простота и гибкость

Разработан с учетом требований

Сделка с Sun Microsystems


# Что в итоге?

Язык, созданный за 10 дней

Наследие и баги из Java

Свои реализации в других браузерах

# Движки JavaScript

SpiderMonkey –  FireFox

Chakra –  Internet Explorer

V8 –  Chromium

Node.JS – платформа на основе V8



# Установка Node.JS

Windows и macOS – <https://clck.ru/JQFV5>

Linux и другие – <https://clck.ru/JQFVR>

# Запускаем интерактивную оболочку (REPL)

```
node
```

# Запускаем файл с JS кодом

```
node index.js
```

Перерыв?

# Первая программа

```
console.log('Hello, Web!');
```

# Первая программа

```
console.log('Hello, Web!');
```

Глобальный объект.

Содержит функции вывода и отладки.

# Первая программа

```
console.log('Hello, Web!');
```

Метод объекта `console`.  
Выводит данные на консоль.

# Первая программа

```
console.log('Hello, Web!');
```

Строковый литерал.

# Первая программа

```
console.log('こんにちは、ウェブ!');
```

Строковый литерал.

Может содержать любые символы Юникода.

# Переменные

```
const year = 2019;
```

```
year = 2020;
```

# Переменные

```
const year = 2019;
```

```
year = 2020; // TypeError: Assignment to constant variable.
```

Константная переменная.  
Перезапись невозможна.



# Переменные

```
let year = 2019;
```

```
year = 2020;  // OK
```

Обычная переменная.

Переменная может быть переприсвоена.

# Переменные

```
let year = 2019;
```

```
year = 2020;
```

Имя переменной.

Имя состоит из букв, цифр, `_` или `$`

Идентификатор не должен начинаться с цифры.

# Переменные

```
let camelCase;
```

В JavaScript принято использовать camelCase.

Слова пишутся слитно, без пробелов.

Первое – с маленькой буквы, следующие – с прописных.

# Переменные

```
let camelCase; // Декларация
```

```
camelCase = 'Hello, Web!'; // Присваивание
```

Задекларированные, но не присвоенные переменные установлены в `undefined`.

# Функции

```
function sum(a, b) {  
    return a + b;  
}
```

```
console.log(sum(42, 13));
```

# Функции

```
function sum(a, b) {  
    return a + b;  
}  
  
console.log(sum(42, 13));
```

Ключевое слово.

За ним должно следовать описание функции.

# Функции

```
function sum(a, b) {  
    return a + b;  
}  
  
console.log(sum(42, 13));
```

Имя функции.

Может быть любым правильным идентификатором.

# Функции

```
function sum(a, b) {  
    return a + b;  
}  
  
console.log(sum(42, 13));
```

Аргументы функции.

JavaScript – динамически-типизированный язык, поэтому типы аргументов не указываются.



# Функции

```
function sum(a, b) {  
    return a + b;  
}
```

```
console.log(sum(42, 13)); // ?
```

# Функции

```
function sum(a, b) {  
    return a + b;  
}
```

```
console.log(sum(42, 13)); // 55
```

# Основные типы данных

```
const integer = 1024;    // Целые числа
const float = 3.1415;    // Дробные числа
const boolean = true;    // Логический тип
const string = 'JS!';    // Строки
const array = [1, 2];    // Массивы
const object = {a: 1};   // Объекты
```

# Циклы

```
let result = 0;
```

```
for (let i = 1; i < 10; i++) {  
    result = result + i;  
}
```

```
console.log(result);
```

# Циклы

```
let result = 0;  
  
for (let i = 1; i < 10; i++) {  
    result = result + i;  
}  
  
console.log(result);
```

Блок инициализации.

Выполняется перед первой итерацией.

# Циклы

```
let result = 0;  
  
for (let i = 1; i < 10; i++) {  
    result = result + i;  
}  
  
console.log(result);
```

Условный блок.

Выполняется перед каждой итерацией.

# Циклы

```
let result = 0;

for (let i = 1; i < 10; i++) {
    result = result + i;
}

console.log(result);
```

Завершающий блок.

Выполняется в конце каждой итерации.

# Циклы

```
let result = 0;
```

```
for (let i = 1; i < 10; i++) {  
    result = result + i;  
}
```

```
console.log(result); // ?
```



# Циклы

```
let result = 0;

for (let i = 1; i < 10; i++) {
    result = result + i;
}

console.log(result); // 45
```

# Циклы

```
let result = 2;  
  
for (let i of [1, 2, 3]) {  
    result = result ** i;  
}  
  
console.log(result);
```

Цикл по итерируемому объекту.

# Циклы

```
let result = 2;
```

```
for (let i of [1, 2, 3]) {  
    result = result ** i;  // Возведение в степень  
}
```

```
console.log(result);  // ?
```

# Циклы

```
let result = 2;
```

```
for (let i of [1, 2, 3]) {  
    result = result ** i;  
}
```

```
console.log(result); // 64
```

# Циклы

```
let result = 0;  
  
for (let field in {1: 'a', 2: 'b', 3: 'c'}) {  
    result = result + field;  
}  
  
console.log(result);
```

Цикл по полям объекта.

# Циклы

```
let result = 0;
```

```
for (let field in {1: 'a', 2: 'b', 3: 'c'}) {  
    result = result + field;  
}
```

```
console.log(result); // ?
```

# Циклы

```
let result = 0;

for (let field in {1: 'a', 2: 'b', 3: 'c'}) {
    result = result + field;
}

console.log(result); // 0123
```

# Циклы

```
let result = 2;  
  
while (result < 100) {  
    result = result * result;  
}  
  
console.log(result);
```

Цикл, который выполняется до тех пор, пока выполнено условие.



# Циклы

```
let result = 2;
```

```
while (result < 100) {  
    result = result * result;  
}
```

```
console.log(result); // ?
```

# Циклы

```
let result = 2;
```

```
while (result < 100) {  
    result = result * result;  
}
```

```
console.log(result); // 256
```

# Условия

```
let type = '...';  
  
if (type === 'string') {  
  
} else if (type === 'number') {  
  
} else {  
  
}
```

# Условия

```
let type = '...';
```

```
if (type === 'string') {  
    // Эта ветка будет выполнена  
    // Если type равен 'string'  
} else if (type === 'number') {  
  
} else {  
  
}
```

# Условия

```
let type = '...';  
  
if (type === 'string') {  
  
} else if (type === 'number') {  
    // Иначе эта ветка  
    // Если type равен 'number'  
} else {  
  
}
```

# Условия

```
let type = '...';  
  
if (type === 'string') {  
  
} else if (type === 'number') {  
  
} else {  
    // Эта ветка выполнится только  
    // Если не выполнены все предыдущие  
}
```

# Switch

```
let type = '...';

switch (type) {
  case 'string':
    break;

  case 'number':
    break;

  default:
    break;
}
```

# Switch

```
let type = '...';

switch (type) {
  case 'string':
    // Эта ветка выполнится
    // Если type равен 'string'
    break;
  case 'number':
    break;

  default:
    break;
}
```



# Switch

```
let type = '...';

switch (type) {
  case 'string':
    // Если убрать break
    // То код продолжит выполняться

  case 'number':
    break;

  default:
    break;
}
```

# Switch

```
let type = '...';

switch (type) {
  case 'string':
    break;

  case 'number':
    break;

  default:
    // Выполниться
    // Если не было break
    break;
```

# Throw / Try / Catch

```
try {  
    throw new Error('ALARM!');  
  
    console.log('Hello, Web!'); // Никогда не выполнится  
} catch (error) {  
    console.log(error.message); // ALARM!  
}
```

# Утверждения

Описывают некоторую выполняемую операцию.  
Программа – последовательность утверждений.

```
let x;           // declaration statement
```

```
x = 2 * 2;       // assignment statement
```

```
if (x === 0) {   // conditional statement  
    x = 123;  
}
```

# Выражения

Выражения производят значения.

```
let x;
```

```
x = 2 * 2;
```

```
if (x === 0) {  
    x = 123;  
}
```

# Выражения

Выражения производят значения.

```
let x;
```

```
x = 2 * 2;
```

```
if (x === 0) {  
    x = 123;  
}
```

# Выражения

Выражения производят значения.

```
let x;
```

```
x = 2 * 2;
```

```
if (x === 0) {  
    x = 123;  
}
```

# Выражения

Выражения производят значения.

```
let x;
```

```
x = 2 * 2;
```

```
if (x === 0) {  
    x = 123;  
}
```



Перерыв?

# Редакторы кода

Atom

Sublime Text

Notepad++

# Редакторы кода

~~Atom~~

~~Sublime Text~~

~~Notepad++~~

Visual Studio Code

WebStorm

# WebStorm

Более мощные инструменты разработки

Все настроено и готово к работе

Много плагинов из коробки

Требователен к ресурсам компьютера

Трудно писать плагины

Платный \$

 WebStorm	5,44 ГБ	86	2 203	1096	dimastark
--	---------	----	-------	------	-----------

# VS Code

Не требователен к ресурсам компьютера

Официальные плагины ко всем языкам

Очень легко писать плагины

Нужно настраивать под себя

Менее качественные инструменты

# Лицензия для WebStorm

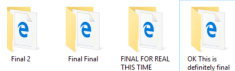
Завести [JetBrains Account](#)

[Подтвердить статус студента](#)

PROFIT!!!

# Система контроля версий





# Ссылки

[Гайд по git прошлых годов](#)

[Интерактивный веб-гайд про git](#)

[JavaScript weekly](#)

Вопросы?

Спасибо!