# JavaScript. Основы.

Сбербанк. Фронтенд школа.

# История. EcmaScript & JavaScript

| | |
|---|---|
| 1995 | **LiveScript, JavaScript, JScript...** |
| 1999 | **ECMAScript, ES3** |
| 2009 | **ECMAScript, ES5** |
| 2015 | **ES6, ES 2015** |
| 2016 | **ES7, ES 2016** |
| 2017 | **ES8, ES 2017** |
| 2018 | **ES9, ES 2018** |

# Дополнительное чтение

Краткая история JavaScript. Часть 1

Краткая история JavaScript. Часть 2

Краткая история JavaScript. Часть 3

# JavaScript. Введение. console.log()

**Выводит сообщение**

**в веб-консоль.**

**Инструменты разработчика - консоль**

**(F12, Ctrl-Shif-I, Cmd+Opt+I, ...)**

```
const coffeeCup = 'Latte';

console.log(coffeeCup);

// "Latte"

console.log('Americano');

// "Americano"

console.log(6);

// 6

console.log(5+6);

// 11
```

# Переменные.

«Именованные хранилища» для данных.

# JavaScript. Введение. Переменные

Жизнь до ES6:

**VAR**

```
var favCoffee; // undefined
favCoffee = 'Americano';
favCoffee = 'Latte'; // Latte

favCoffee = 0;
favCoffee = favCoffee + 2; // 2
```

# JavaScript. Введение. Переменные

Жизнь после ES6:

## LET, CONST

```
const favCoffee = 'Americano';
favCoffee = 'Latte'; // TypeError

let cupsOfCoffeeDaily = 0;
cupsOfCoffeeDaily = cupsOfCoffeeDaily + 2; // 2
```

# JavaScript. Введение. Переменные

Жизнь после ES6:

## LET, CONST

```
const favCoffee = 'Americano';
favCoffee = 'Latte'; // TypeError

let cupsOfCoffeeDaily = 0;
cupsOfCoffeeDaily = cupsOfCoffeeDaily + 2; // 2
```

## VAR

```
var favCoffee; // undefined
favCoffee = 'Americano';
favCoffee = 'Latte'; // Latte

favCoffee = 0;
favCoffee = favCoffee + 2; // 2
```

# JavaScript. Введение. Переменные

**LET, CONST**

БЛОЧНАЯ ОБЛАСТЬ ВИДИМОСТИ

**VAR**

ФУНКЦИОНАЛЬНАЯ ОБЛАСТЬ ВИДИМОСТИ

# JavaScript. Введение. Переменные

Что это означает?

```
function makeAnOrder(){

  var milk = 'almond';
  console.log('myOrder: '+milk);

  if (true) {
    var milk = 'cow';
    console.log('friendsOrder: '+milk);

  }
  console.log('confirmMyOrder? '+milk);
}
```

# JavaScript. Введение. Переменные

Что это означает?

```
function makeAnOrder(){
  var milk = 'almond';
  console.log('myOrder: '+milk);
  // "myOrder: almond"
  if (true) {
    var milk = 'cow';
    console.log('friendsOrder: '+milk);
    // "friendsOrder: cow"
  }
  console.log('confirmMyOrder? '+milk);
  // "confirmMyOrder? cow"
}
```

# JavaScript. Введение. Переменные

Что это означает? (Песочница: https://jsbin.com/jiqiwev/edit?js,console)

```
function makeAnOrder(){
  var milk = 'almond';
  console.log('myOrder: '+milk);
  // "myOrder: almond"
  if (true) {
    var milk = 'cow';
    console.log('friendsOrder: '+milk);
    // "friendsOrder: cow"
  }
  console.log('confirmMyOrder? '+milk);
  // "confirmMyOrder? cow"
}
```

```
function makeAnOrder2(){
  let milk = 'almond';
  console.log('myOrder: '+milk);
  // "myOrder: almond"
  if (true) {
    let milk = 'cow';
    console.log('friendsOrder: '+milk);
    // "friendsOrder: cow"
  }
  console.log('confirmMyOrder? '+milk);
  // "confirmMyOrder? almond"
}
```

# JavaScript. Введение. Переменные. ДопИнфо.

1) Жизненный цикл переменных (ENG)
2) Подробнее о переменных в JS
3) Var, let или const? Проблемы областей видимости переменных и ES6
4) Зарезервированные слова для имен

# Функции.

Фрагмент программного кода, к которому можно обратиться из другого места программы

# JavaScript. Введение. Функции

Обычная простая функция

```
function callMyName(){

  console.log(“ИВАААН!”);
  return “Ванька!”

}

callMyName()
//“ИВАААН!”

console.log(callMyName())
//“ИВАААН!”
//“Ванька!”
```

https://learn.javascript.ru/function-basics

# JavaScript. Введение. Функции

## Параметры, аргументы

```
function makeCoffee3(water, milk, coffee, syrup ='', keyword) {
  let price = 60

  if (keyword) {
    price = price*0.95
  }
  const cupOfCoffee = `${water} ${milk} ${coffee} ${syrup}`;
  const finalPhrase = `Вы заказали ${cupOfCoffee}. С Вас ${price} рублей`
  return finalPhrase;

}

console.log(makeCoffee3('ice', 'coconut', 'blond', 'banana'))
//"Вы заказали ice coconut blond banana. С Вас 60 рублей"
console.log(makeCoffee3('hot', 'coconut', 'blond', 'banana', 'уменякарта'))

//"Вы заказали ice coconut blond banana. С Вас 57 рублей"
```

# Типы данных.

# JavaScript. Введение. Типы данных

1. **Типизированный**
2. **Динамическая типизация** (*Все типы определяются уже во время выполнения программы. И если вы допустили ошибку, то узнаете об этом только при выполнении*)
3. **Нестрогая типизация (слабая)** (*Выполняет множество неявных преобразований типов автоматически. Даже если может произойти потеря точности или преобразование неоднозначно.*)

# JavaScript. Введение. Типы данных

# JavaScript. Введение. Типы данных

## Примитивы. Boolean.

```
const iLikeCoffee = true;
const iLikeColdBrew = false;

if (!iLikeCoffee) {
  console.log('Пей чай тогда!');
}
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Boolean

# JavaScript. Введение. Типы данных

## Примитивы. Null.

```
const cupOfCoffee = null;

if (cupOfCoffee === null) {
  console.log('Наполни мою кружку свежим
кофейком');
}
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/null

# JavaScript. Введение. Типы данных

## Примитивы. Undefined.

```
const cupOfCoffee;
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined

# Null vs Undefined

# JavaScript. Введение. Типы данных

## Number

```
let x = 13;
x = 3.14159265359; // все числа float
0xff; // 255
3e5; // 30000
2 + 3; // 5
2 - 3; // -1
5 * 5; // 25
12 / 4; // 3
11 % 3; // 2
```

http://learn.javascript.ru/number

# JavaScript. Введение. Типы данных

## Number

```
const coffeePrice = 120;
const coffeePrice2 = '120';
const coffeePrice3 = 'Сто двадцать';

console.log('Раф стоит '+ coffeePrice +' рублей')

// Раф стоит 120 рублей

console.log(typeof coffeePrice)

// number

console.log(typeof Number(coffeePrice2), Number(coffeePrice2))

// number 120

console.log('weird stuff: '+ typeof Number(coffeePrice3)+' '+Number(coffeePrice3)+' '+typeof NaN)

// weird stuff: number NaN number
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number

# JavaScript. Введение. Типы данных

## BigInt (candidate)

**`BigInt`** это встроенный объект, который предоставляет способ представлять целые числа больше $2^{53}$,это наибольшее число, которое JavaScript может надежно представить с `Number` примитивом.

```
const theBiggestInt = 9007199254740991n;
const alsoHuge = BigInt(9007199254740991);
// 9007199254740991n
const hugeString = BigInt("9007199254740991");
// 9007199254740991n
const hugeHex = BigInt("0x1fffffffffffff");
// 9007199254740991n
const hugeBin =
BigInt("0b11111111111111111111111111111111111111111111111111111");
// 9007199254740991n
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/BigInt

# JavaScript. Введение. Типы данных

## Symbol

```
const spicyMilkTeaPowder = Symbol('TeaPowder');
const milk = 'Коровье молоко';
const water = 'Горячая вода';
const makeSpicyMilkTea = {}

makeSpicyMilkTea[milk] = '100 ml';
makeSpicyMilkTea[water] = '100 ml';
makeSpicyMilkTea[spicyMilkTeaPowder] = '4 ст.л';

if (makeSpicyMilkTea[spicyMilkTeaPowder] !== null) {
    makeSpicyMilkTea['Основа'] = makeSpicyMilkTea[spicyMilkTeaPowder];
}

console.log('Каков состав чая? -'+Object.keys(makeSpicyMilkTea))
// "Каков состав чая? -Коровье молоко,Горячая вода,Основа"
console.log('А сколько основы? -'+makeSpicyMilkTea[spicyMilkTeaPowder])
// "А сколько основы? -4 ст.л"
```

https://habr.com/ru/company/ruvds/blog/444340/

# JavaScript. Введение. Типы данных

## String

```
const favCoffee = 'Макиато';
const favCoffeePrice = '120';
const friendFavCoffee = 'Латте';
const friendFavCoffeePrice = '140';
const totalPrice = (x,y) => x+y

console.log('Я возьму '+favCoffee+' для себя')
//"Я возьму Макиато для себя"
console.log('Еще возьму '+friendFavCoffee+' для друга')
//"Еще возьму Латте для друга"
console.log(`С Вас ${totalPrice(favCoffeePrice, friendFavCoffeePrice)} рублей`)
//"С Вас 120140 рублей"
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

# JavaScript. Введение. Типы данных

## String

```
const favCoffee = 'Макиато';
const favCoffeePrice = '120';
const friendFavCoffee = 'Латте';
const friendFavCoffeePrice = '140';
const totalPrice = (x,y) => x+y

console.log('Я возьму '+favCoffee+' для себя')
//"Я возьму Макиато для себя"
console.log('Еще возьму '+friendFavCoffee+' для друга')
//"Еще возьму Латте для друга"
console.log(`С Вас ${totalPrice(favCoffeePrice, friendFavCoffeePrice)} рублей`)
//"С Вас 120140 рублей"
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

# JavaScript. Введение. Типы данных

Все примитивы, кроме null и undefined, предоставляют множество полезных методов.

```
const str = 'Арабика';
str.length; // 7
str[0]; // 'A'

const num = 120.22856
num.toFixed(2) // 120.23
num.toString() // '120.22856'

parseInt(num.toString()) // 120
```

# JavaScript. Введение. Типы данных

## Object

```
const coffeePrices = {
    latte: 50,
    americano: 30,
    cappuccino: 40
  }

  console.log('Кофе стоит '+ coffeePrices.latte +' рублей')
  //"Кофе стоит 50 рублей"
```

https://learn.javascript.ru/object

# JavaScript. Введение. Типы данных

## Object (Array)

```
const coffeeTypes = ['Латте', 'Американо', 'Капучино']

console.log('Хочу '+ coffeeTypes[0])
//"Хочу Латте"
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array

В переменных сохраняются:
для примитивов — значения
для объектов — ссылки

# JavaScript. Введение. Типы данных. Доп.Инфо.

1) Песочница
2) Более подробная информация по типам данных
3) Более подробная информация по типам данных 2
4) Больше практических примеров

JavaScript. Введение. Типы данных. Практика

https://jsbin.com/giluqis/edit?js,console

JavaScript. Введение. Типы данных. Практика

https://jsbin.com/giluqis/edit?js,console

**Что примерно должно получиться:**

https://jsbin.com/fifiho/edit?js,console

# Порядок выполнения, инструкции

# JavaScript. Введение. Инструкции.

Инструкция block

Инструкция block является фундаментальной и используется для группировки других инструкций. Блок ограничивается фигурными скобками.

Блок обычно используется с управляющими инструкциями (например, if, for, while).

```
while (x < 10) { x++; }
```

# JavaScript. Введение. Инструкции.

Условные инструкции

Условная инструкция — это набор команд, которые выполняются, если указанное условие является истинным.

JavaScript поддерживает две условные инструкции: **if...else и switch.**

# JavaScript. Введение. Инструкции.

**if...else**

Условная инструкция — это набор команд, которые выполняются, если указанное условие является истинным.

```
if (condition) {
    statement_1;
} else {
    statement_2;
}
```

# JavaScript. Введение. Инструкции.

### switch

Инструкция switch позволяет сравнить значение выражения с различными вариантами и при совпадении выполнить соответствующий код.

```
switch (fruittype) {
  case "Oranges":
    console.log("Oranges are $0.59 a pound.");
    break;
  case "Mangoes":
    console.log("Mangoes are $0.56 a pound.");
    break;
  default:
   console.log("Sorry, we are out of " + fruittype + ".");
}
console.log("Is there anything else you'd like?");
```

# JavaScript. Введение. Инструкции.

**Обработка исключений**

Try - catch, Error, etc...

https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Control_flow_and_error_handling

# Функции. Часть 2.

# JavaScript. Введение. Функции 2

Вспомним, что такое функция:

```
function drinkCoffee(coffee) {
    console.log('Мило пьем кофеек')
}

drinkCoffee('americano')
// 'Мило пьем кофеек'
```

# JavaScript. Введение. Функции 2

Вспомним, что такое функция:

```
function drinkCoffee(coffee) {
    console.log('Мило пьем кофеек')
}

drinkCoffee('americano')
// 'Мило пьем кофеек'
```

## Function declaration

Объявление Функции

# JavaScript. Введение. Функции 2

А можно так:

```
const drinkCoffee = function (coffee) {
    console.log('Мило пьем кофеек')
}

drinkCoffee('americano')
// 'Мило пьем кофеек'
```

## Function Expression

Функциональное Выражение

# JavaScript. Введение. Функции 2

В чем разница?

```
drinkCoffee('americano')
// ReferenceError

const drinkCoffee = function (coffee) {
    console.log('Мило пьем кофеек')
}
```

```
drinkCoffee('americano')
// 'Мило пьем кофеек'

function drinkCoffee (coffee) {
    console.log('Мило пьем кофеек')
}
```

**Основное отличие между ними: функции, объявленные как Function Declaration, создаются интерпретатором до выполнения кода.**

# JavaScript. Введение. Функции 2

Стрелочные функции (как Function Expression, только лучше) **ES6**

```
const drinkCoffee = (coffee) => console.log('Мило пьем кофеек')

drinkCoffee('americano')
// 'Мило пьем кофеек'
```

# JavaScript. Введение. Функции 2

## Анонимные функции

```
(coffee) =>{ console.log('У функции нет имени') }

const alala = function(pew){
console.log('Я тоже все еще анонимная')
}
```

# JavaScript. Введение. Функции 2

Тернарный if

```
function willTravel(spacesuit) {
  return spacesuit ? 'Have SpaceSuit Will Travel'
: 'Will stay at home';
}

console.log(willTravel(true))
// "Have SpaceSuit Will Travel"

console.log(willTravel(false))
// "Will stay at home"
```

# JavaScript. Введение. Функции 2.

1) Функции
2) Функции 2
3) Анонимные функции
4) MDN

JavaScript. Введение.Функции / CF. Практика

https://jsbin.com/letaxus/edit?js,console

JavaScript. Введение.Функции / CF. Практика

https://jsbin.com/letaxus/edit?js,console

**Что примерно должно получиться:**

https://jsbin.com/valibad/edit?js,console

# Замыкания, области видимости

# JavaScript. Замыкания, области видимости.

Локальные переменные.

```
function iDrinkCoffee() {
   const myCup = 'Iced Malinovii Coffee'
   console.log(`Я пью ${myCup}.`)
}

console.log(iDrinkCoffee())
// "Я пью Iced Malinovii Coffee."

console.log(myCup)
// ReferenceError: myCup is not defined
```

# JavaScript. Замыкания, области видимости.

## Внешние переменные

```
var coffee = 'Три в одном'

function ibuyCoffee() {
   console.log(`Мне стаканчик ${coffee}, пожалуйста!`)
}

ibuyCoffee()
// "Мне стаканчик Три в одном, пожалуйста!"

console.log(coffee)
// "Три в одном"
```

# JavaScript. Замыкания, области видимости.

Области видимости

# JavaScript. Замыкания, области видимости.

## Замыкания

Функции в JavaScript формируют так называемые замыкания.

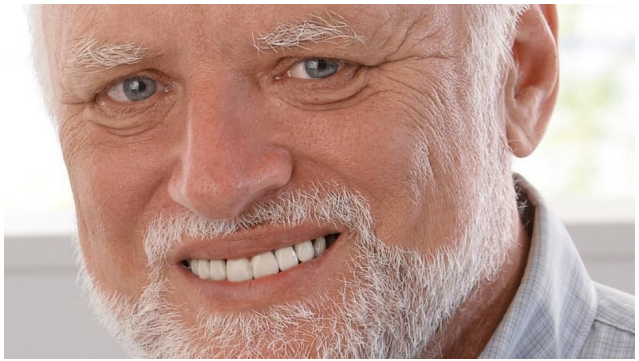
**Замыкание** — это комбинация функции и лексического окружения, в котором эта функция была объявлена. Это окружение состоит из произвольного количества локальных переменных, которые были в области действия функции во время создания замыкания.

# JavaScript. Замыкания, области видимости.

## Замыкания

Функции в JavaScript формируют так называемые замыкания.

**Замыкание** — это комбинация функции и лексического окружения, в котором эта функция была объявлена. Это окружение состоит из произвольного количества локальных переменных, которые были в области действия функции во время создания замыкания.



# JavaScript. Замыкания, области видимости.

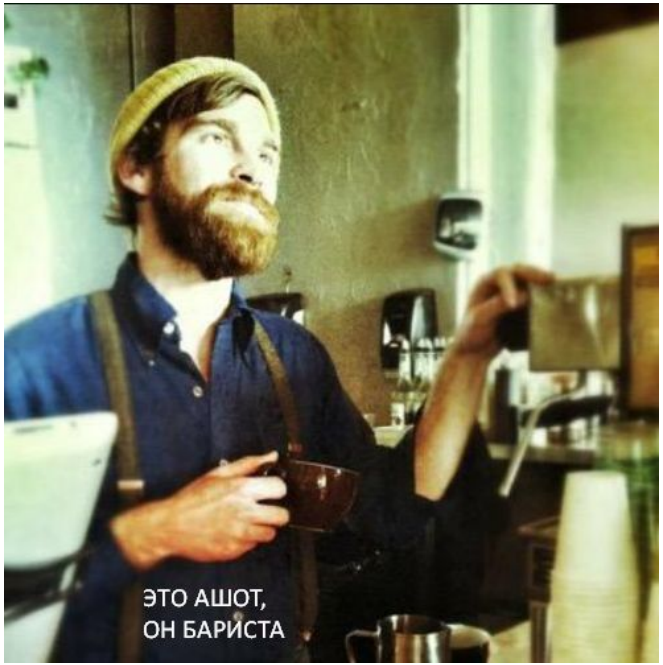
## Замыкания на примере заказа кофе

```
function makeMeACoffee(order) {
    const ingredients = order.join(' ');
    function ashot() {
        return ingredients.concat(' кофе');
    }

    return ashot()
}
```

# JavaScript. Замыкания, области видимости.

## Замыкания на примере заказа кофе

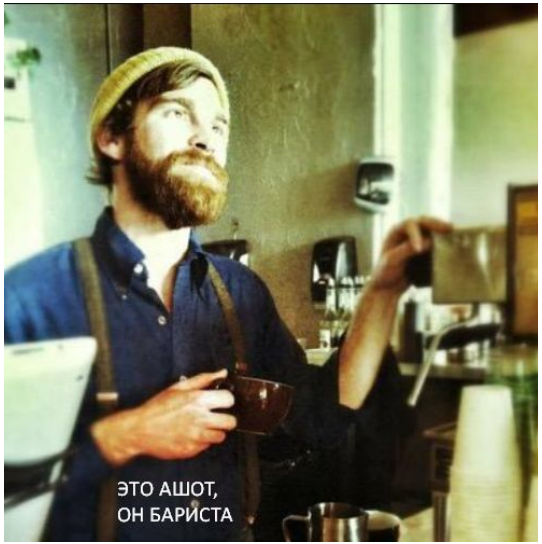
```
function makeMeACoffee(order) {
    const ingredients = order.join(' ');
    function ashot() {
        return ingredients.concat(' кофе');
    }

    return ashot()
}
```



# JavaScript. Замыкания, области видимости.

## Замыкания на примере заказа кофе

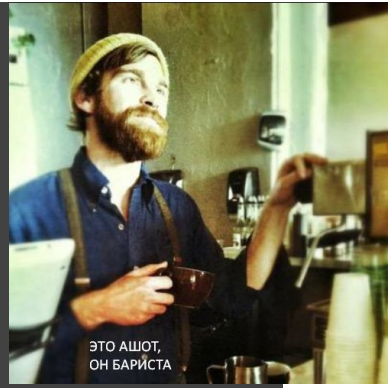
https://jsbin.com/hiqeket/edit?js,console



# JavaScript. Замыкания, области видимости.

## Замыкания на примере заказа кофе

```
function makeMeACoffeeVer2(order){
const ingredients = order.join(' ');
  function ashot() {
    return ingredients.concat(' кофе');
  }
  let pshik = 0
  function ashotAddSyrup() {
        pshik += 2;
        return ingredients.concat(' кофе с ', pshik, ' порциями ванильного сиропа');
  }
  return { noSyrup: () => ashot(),
           addSyrup: () => ashotAddSyrup()
    }
}
```



# JavaScript. Замыкания, области видимости.

## Замыкания на примере заказа кофе

```
const icedark2 = makeMeACoffeeVer2(['лед', 'темная обжарка'])
const icedark3 = makeMeACoffeeVer2(['лед', 'темная обжарка'])
icedark2.addSyrup()
icedark2.addSyrup()
icedark2.addSyrup()
icedark2.addSyrup()
// "лед темная обжарка кофе с 8 порциями ванильного сиропа"
console.log(icedark3.addSyrup())
// "лед темная обжарка кофе с 2 порциями ванильного сиропа"
```

# JavaScript. Замыкания, области видимости.

**Прочитать к сл.занятию**

1) https://learn.javascript.ru/closure
2) Замыкания в подробностях

# JavaScript. Замыкания, области видимости.

## THIS

**Для доступа к информации внутри объекта метод может использовать ключевое слово this.**

**Значение this – это объект «перед точкой», который использовался для вызова метода.**

*this.cupOfCoffee*

# JavaScript. Замыкания, области видимости.

## THIS и наш вагончик с кофе

```
const van= {
  name: "КофийНаКолесах",
  makeaDrink:function() {
      console.log(this.name+' уже приступил к
выполнению заказа!');
  }
}
```

https://jsbin.com/qeyigaq/edit?js,console

# JavaScript. Замыкания, области видимости.

1) https://learn.javascript.ru/object-methods
2) https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/this

# Объекты, массивы и прочие коллекции

# JavaScript. Объекты, массивы и прочие коллекции

## ES6: ...spread

Оператор spread позволяет расширять выражения в тех местах, где предусмотрено использование нескольких аргументов.

```
function sum(x, y, z) {
  return x + y + z;
}

const numbers = [1, 2, 3];

console.log(sum(...numbers));
// expected output: 6
```

```
const mid = [3, 4];
const = [1, 2, ...mid, 5, 6];
console.log(arr);
// 1, 2, 3, 4, 5, 6
```

```
const obj1 = { foo: 'bar', x: 42 };
const obj2 = { foo: 'baz', y: 13 };

const clonedObj = { ...obj1 };
// Object { foo: "bar", x: 42 }

const mergedObj = { ...obj1, ...obj2 };
// Object { foo: "baz", x: 42, y: 13 }
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Spread_syntax

# JavaScript. Объекты, массивы и прочие коллекции

## ES6: деструктуризация аргументов

Позволяет извлекать данные из массивов или объектов при помощи синтаксиса, подобного объявлению массива или литералов в объекте.

```
const veganLatte  = {milk:"almond", beans:
'roasted', syrup: 'vanilla'}

const { milk, syrup } = veganLatte;
console.log(milk, syrup)
// "almond", "vanilla"
```

```
const array = [1, 2, 3];
const [first, second, third] = array;
console.log(first, second, third);
// 1 2 3
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment

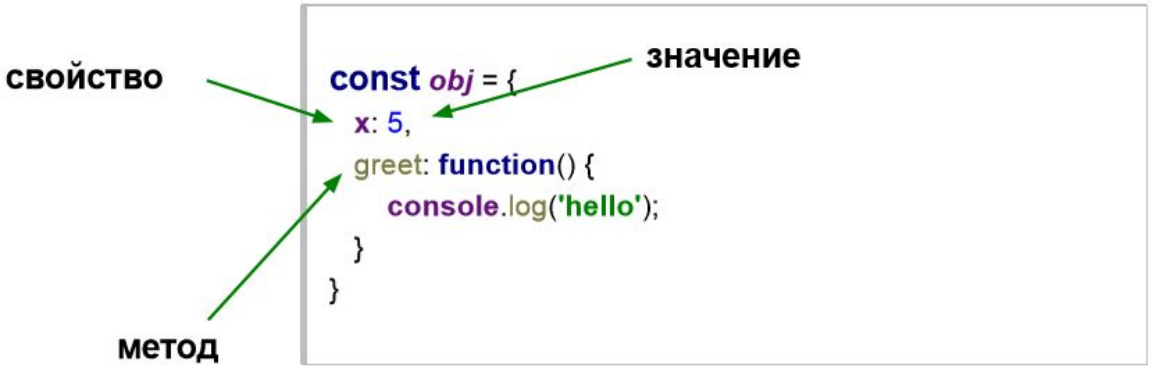
# JavaScript. Объекты, массивы и прочие коллекции

## Создание объектов

```
const obj1 = new Object();
const obj2 = {};
const obj3 = { x: 5 };
```

# JavaScript. Объекты, массивы и прочие коллекции

## Объекты



# JavaScript. Объекты, массивы и прочие коллекции

## Чтение свойств

```
const starbucks = {
  address: 'Кутузовский проспект, 32',
  menu: {
    latte: 'Pumpkin Latte',
    tea: 'Spicy Milk Tea'
  }
}

starbucks.address;
// 'Кутузовский проспект, 32'
starbucks.menu.latte;
// 'Pumpkin Latte'
starbucks['menu']['tea'];
// 'Spicy Milk Tea'
```

# JavaScript. Объекты, массивы и прочие коллекции

## Доступ к несуществующему ключу

```
const starbucks = {
  address: 'Кутузовский проспект, 32',
  menu: {
    latte: 'Pumpkin Latte',
    tea: 'Spicy Milk Tea'
  }
}

starbucks.address;
// 'Кутузовский проспект, 32'
starbucks.menu.latte;
// 'Pumpkin Latte'
starbucks['menu']['tea'];
// 'Spicy Milk Tea'

starbucks.soup;
// undefined
```

# JavaScript. Объекты, массивы и прочие коллекции

## Запись свойств

```
const starbucks = {
  address: 'Кутузовский проспект, 32',
  menu: {
    latte: 'Pumpkin Latte',
    tea: 'Spicy Milk Tea'
  }
}
starbucks.soup='Tom Yam';
// а вот так удаляем:
delete starbucks.soup;
```

# JavaScript. Объекты, массивы и прочие коллекции

1) https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Working_with_Objects (особое внимание на for...in, Object.keys(o), Object.getOwnPropertyNames(o))
2) http://jsraccoon.ru/oop-object-base
3) https://medium.com/nuances-of-programming/rest-и-spread-в-javascript-возможности-о-которых-вы-не-знали-3371dc86b788
4) Разбор массивов. https://javascript.info/array
5) Методы массивов.

# JavaScript. Объекты, массивы и прочие коллекции

https://jsbin.com/yumazeb/edit?js,console

# JavaScript. Объекты, массивы и прочие коллекции

**https://jsbin.com/yumazeb/edit?js,console**

Что должно примерно получиться:

**https://jsbin.com/hewavun/edit?js,console**