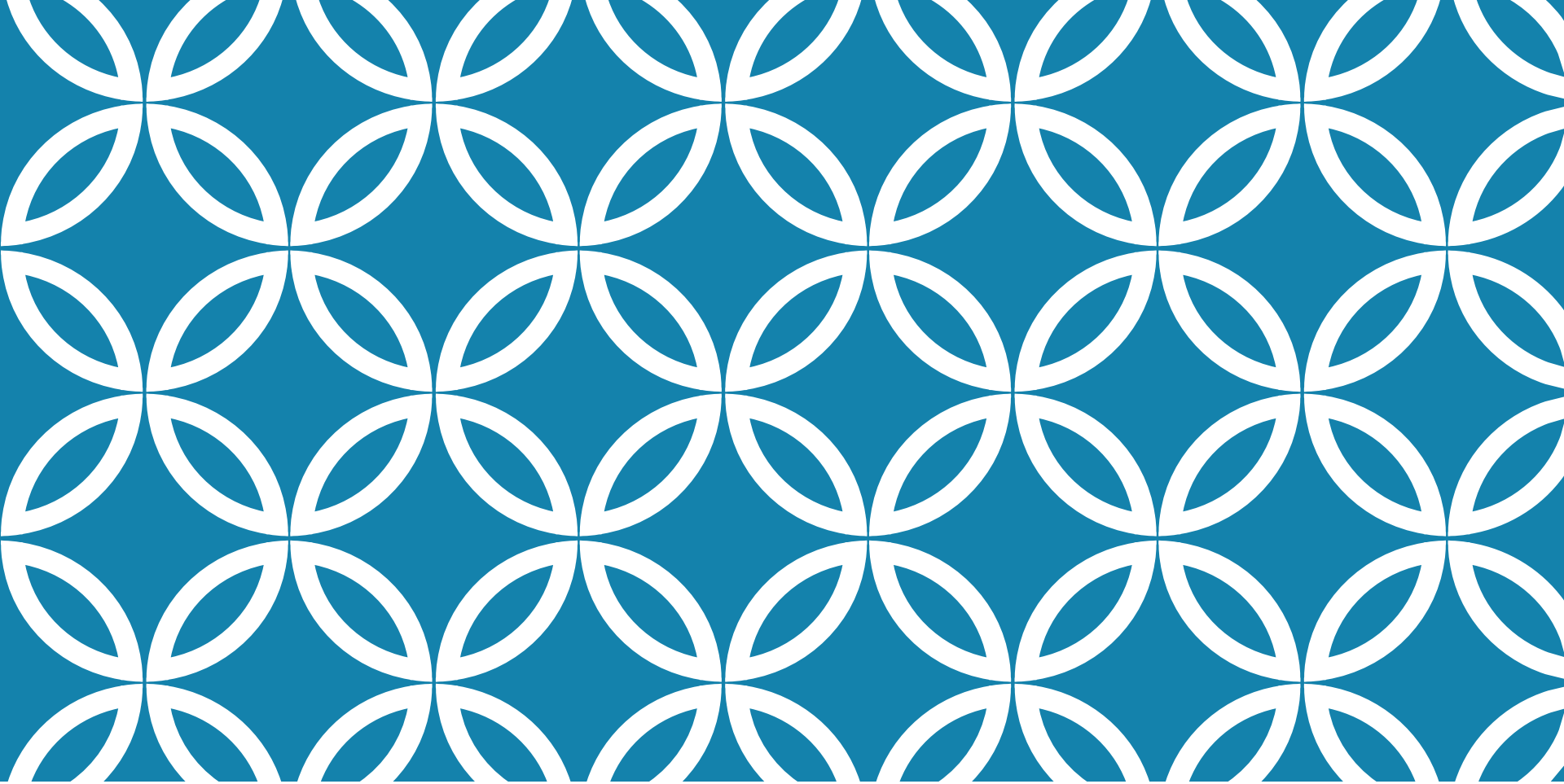


ОСНОВЫ WEB-ТЕХНОЛОГИЙ

Лекция 5

СТРУКТУРА ЗАНЯТИЯ

- Из истории языка JavaScript
- Особенности JS
- Основы синтаксиса JS

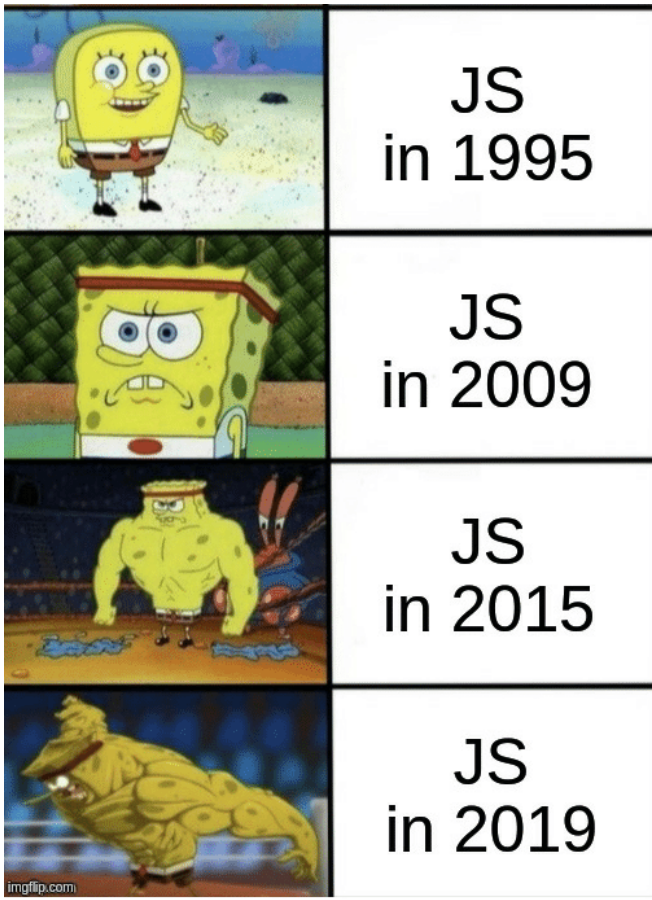
ИЗ ИЗТОРИИ…

Неистовый 1995 год, война браузеров

Брендан Эйх
Разработал язык JavaScript
за 10 дней
(по его словам)

ГОНКА НОВОВВЕДЕНИЙ

год	нововведение
1995	10 дней в мае 1995: Mocha
Сентябрь 1995	LiveScript
Декабрь 1995	JavaScript
1996-1997	ES1
1999	ES3: практически современный JS
2005	Революция AJAX
2009	ES5: ‘use strict’, JSON etc
2012	ES6: модули, let, proxies etc



Haters gonna hate

СКОРОСТЬ РАЗВИТИЯ → ∞

2009

2019

НУ КАК БЫ ДА…

JAVASCRIPT – ЭТО…

JavaScript – Язык сценариев для придания интерактивности web-страницам.

Язык программирования JavaScript (JS) придает веб-страницам возможность реагировать на действия пользователя и превращать статичные страницы в динамические, так, чтобы страницы буквально "оживали" на глазах.

Программы на этом языке называются ***скриптами***. В браузере они подключаются напрямую к HTML и, как только загружается страничка — тут же выполняются.

Особенности JS

Есть как минимум *три* замечательных особенности JavaScript:

- Полная интеграция с HTML/CSS.
- Простые вещи делаются просто.
- Поддерживается всеми распространёнными браузерами и включён по умолчанию.

ПРИМЕНЕНИЕ JAVASCRIPT

- Сайты (исполнение в браузере)
- Устанавливаемые приложения на основе браузера
- Игры на PC/xbox/psp/iOS/Android (unity3d)
- Приложения на телефон/планшет (React Native/Unity3d)
- Робототехника Iskra.js
- Сервера на Node.js
- etc

ПОДКЛЮЧЕНИЕ К ДОКУМЕНТУ

Для того, чтобы добавить сценарий на страницу нужно дописать следующий код между тегами

```
<script type="text/javascript" > </script>
```

Либо подключить внешний файл со скриптами:

```
<script src="/path/to/script.js"></script>
```

ВВОД И ВЫВОД ДАННЫХ

Метод – фрагмент кода многократного использования, предназначенный для решения общих задач.

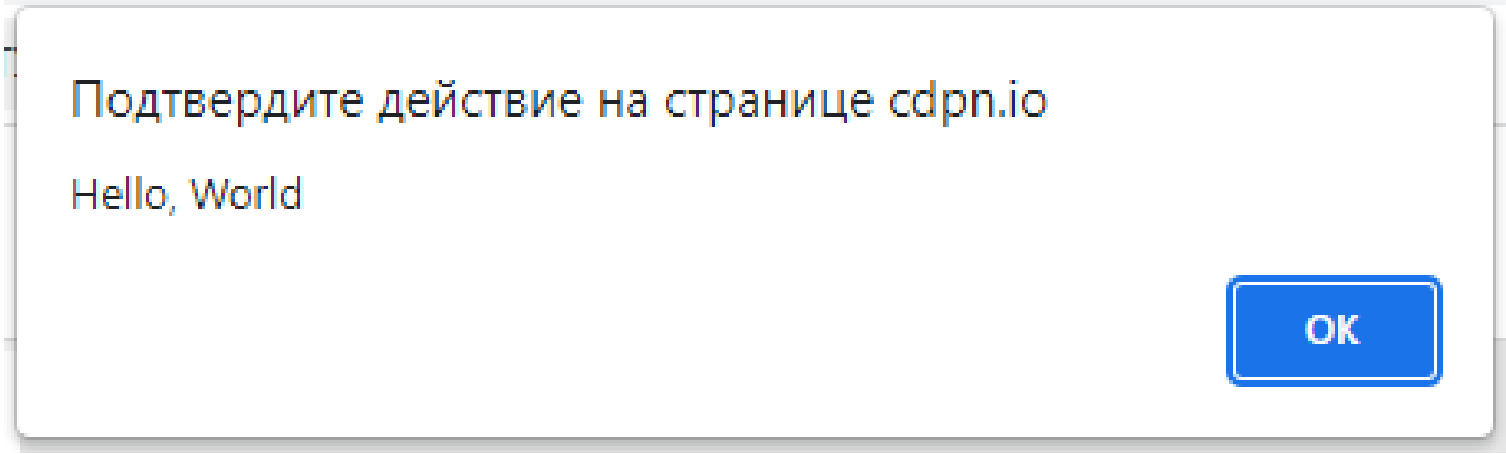
В JavaScript используются три стандартных метода для ввода и вывода данных:

- alert(),
- prompt()
- confirm()

ALERT

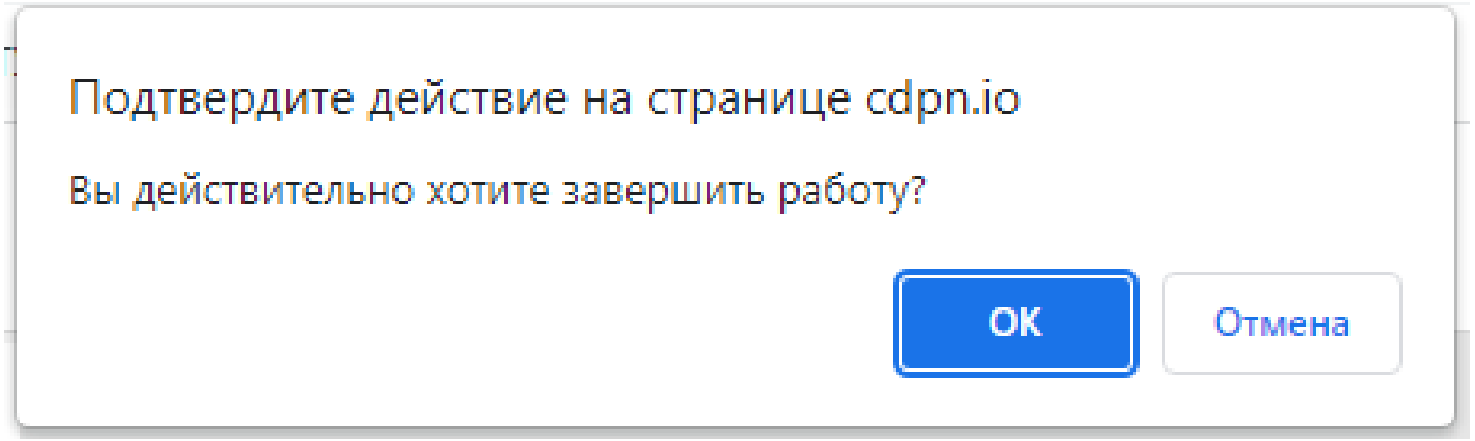
alert("Hello, World");

Вызывает ***модальное*** окно



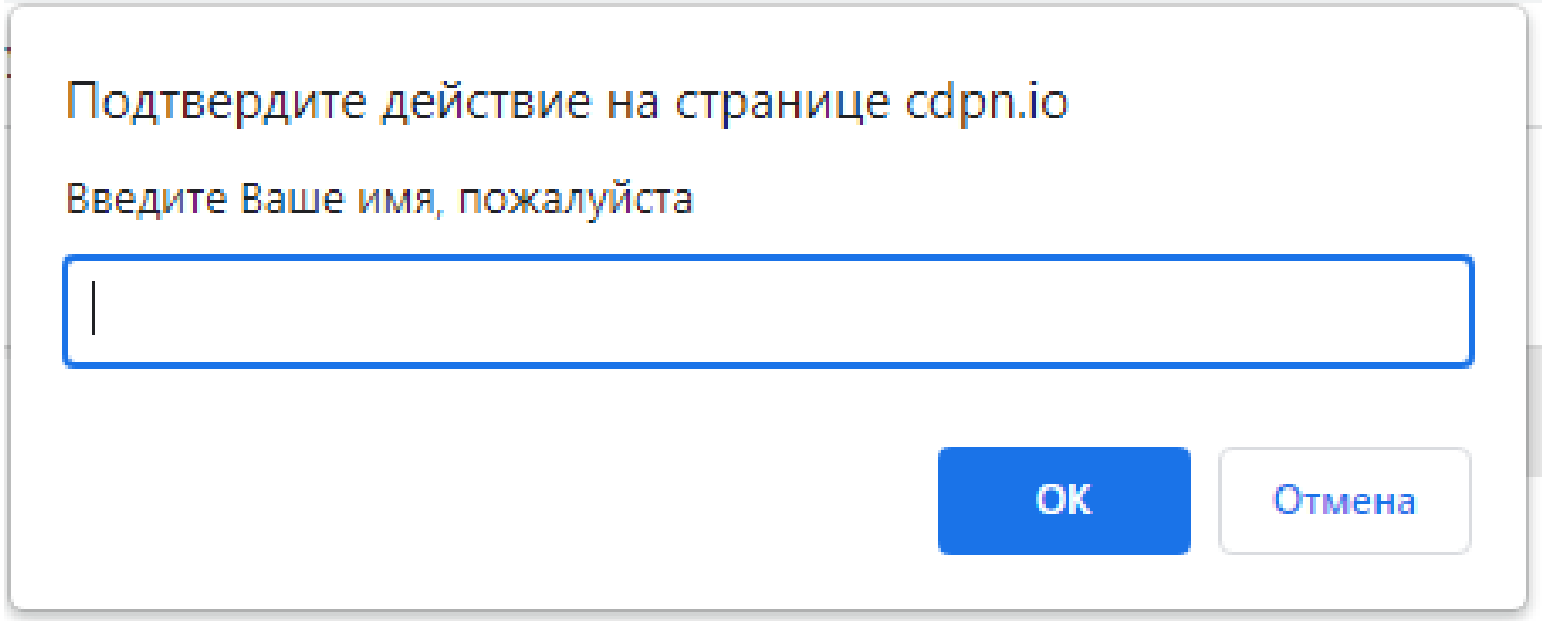
CONFIRM

confirm("Вы действительно хотите завершить работу?");



PROMPT

prompt("Введите Ваше имя, пожалуйста", "");



ВЫВОД В КОНСОЛЬ

```
console.log('Привет от JavaScript!');
```

ПЕРЕМЕННЫЕ

Переменная JS — это именованная область в памяти, которая хранит в себе данные (значение). К этим данным можно получить доступ, обратившись по имени переменной, в которой они хранятся.

ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ

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

```
var message;
```

ОСНОВНОЙ СИНТАКСИС И ФУНКЦИОНАЛ

```
var n = 123;
var str = "Привет мир";
var checked = true;
var obj = { key: "value" };
var arr = [0, 1, 2, 3, 4, 5];
var fn = function(argum1, argum2){ return; }
```

ПОЛУЧЕНИЕ ЗНАЧЕНИЙ

```
var n = 123;
n
var str = “Привет мир”;
str
var checked = true;
checked
var obj = { key: "value" };
obj.key
var arr = [0,1,2,3,4,5];
arr[0]
```

ВАЖНО

Всегда определяйте переменные через var.
Это хороший тон в программировании и помогает избежать ошибок.

CONST И LET

В ES-6 предусмотрены новые способы объявления переменных:

через let и const вместо var.

- Если вам нужно значение, которое не может измениться в ходе работы скрипта, используйте **const**
- Если вам нужна изменяемая величина, используйте **let**
- Если вам нужно, чтобы ваш код поддерживался старыми браузерами(до 2015 года), используйте **var**

РАЗНИЦА МЕЖДУ VAR И LET

Ключевое слово let лишено недостатков своего предшественника.

РАЗНИЦА МЕЖДУ VAR И LET

Переменные, объявленные с его помощью, нельзя объявить повторно — программа выдаст ошибку.

```
let a = 1;
let a = 2;

//выдаст ошибку, что "a" уже объявлена
```

РАЗНИЦА МЕЖДУ VAR И LET

Let-переменные тоже «всплывают», но при попытке обратиться к ним до инициализации вы получите ошибку ReferenceError.

```
function showAnswer() {
    //Вызовем переменную перед её объявлением
    console.log(answer);
    let answer = "I don't know";
}

showAnswer();
//Получим ошибку.
```

РАЗНИЦА МЕЖДУ VAR И LET

Переменные, объявленные с let, имеют блочную область видимости. А значит, они доступны только внутри того блока {}, в котором были созданы

```
let a = 5;
if(true)
{
    let b = 5;
}
//Будет выведено, потому что переменная a объявлена вне какого-либо блока и доступна глобально
console.log(a);
//Вызовет ошибку, потому что переменная b объявлена внутри блока if и доступна только в нём
console.log(b);
```

РАЗНИЦА МЕЖДУ VAR И LET

Вот как различается поведение счётчика цикла, если его создавать с помощью var и с помощью let:

var	let

for(var i = 0; i < 5; i++) {}
 	
//Переменная i доступна за пределами цикла
 	
console.log(i);
	for(let i = 0; i < 5; i++) {}
	
	//Переменная i доступна только внутри цикла
	//Попытка использовать её приведёт к ошибке
	
	console.log(i);

КОНСТАНТА

Константа — это постоянная величина, которая никогда не меняется. Как правило, их называют большими буквами, через подчёркивание.

```
const COLOR_RED = "#F00";
const COLOR_GREEN = "#0F0";
const COLOR_BLUE = "#00F";
const COLOR_ORANGE = "#FF7F00";
```

ИМЕНА ПЕРЕМЕННЫХ

На имя переменной в JavaScript наложено несколько ограничений.

1. Имя может состоять из: букв, цифр, символов $ и _
2. Первый символ не должен быть цифрой.
3. В качестве имён переменных нельзя использовать ключевые слова и зарезервированные слова.

ИМЕНА ПЕРЕМЕННЫХ

Ключевые слова в JavaScript — это слова, которые существуют в ядре языка JavaScript и встроены в его синтаксис, например слово var.

Зарезервированные слова в JavaScript — это слова, которые пока еще не существуют в ядре языка JavaScript и не встроены в его синтаксис, но в будущем, эти слова могут быть внедрены в ядро JavaScript, например слово abstract.

КЛЮЧЕВЫЕ СЛОВА В JAVASCRIPT

Keyword	Description
var	Объявляет переменную
let	Объявляет блочную переменную
const	Объявляет блочную константу
if	условие
switch	Множественное условие
for	цикл
function	Объявляет функцию
return	Выход из функции
try	Попытка исполнить код

ТИПЫ ДАННЫХ

- **number** ($\pm(2^{53}-1)$)
- **bigint** (для целых чисел произвольной длины)
- **string**
- **boolean**
- **null** для неизвестных значений
- **undefined** для неприсвоенных значений
- **symbol** для уникальных идентификаторов.
- **object** для более сложных структур данных

NUMBER

```
var n = 123; m = 12.345;
```

Единый тип **number** используется как для целых, так и для дробных чисел.

Существуют специальные числовые значения **Infinity** (бесконечность) и **NaN** (ошибка вычислений). Они также принадлежат типу «число».

МЕТОДЫ NUMBER

Method	Description
toString()	Returns a number as a string
toExponetial()	Returns a number written in exponential notation
toFixed()	Returns a number written with a number of decimals
toPrecision()	Returns a number written with a specified length
ValueOf()	Returns a number as a number

STRING

```
var str = "Мама мыла раму";
str = 'Одинарные кавычки тоже подойдут';
```

В JavaScript одинарные и двойные кавычки равноправны. Можно использовать или те или другие.

STRING

String length
String slice()
String substring()
String substr()
String replace()
String replaceAll()
String toUpperCase()
String toLowerCase()
String concat()
String trim()
String trimStart()
String trimEnd()
String padStart()
String padEnd()
String charAt()
String charCodeAt()
String split()

BOOLEAN

true (истина) и false (ложь)

Как правило, такой тип используется для хранения значения типа да/нет, например:

```
var checked = true; // поле формы помечено галочкой
checked = false;    // поле формы не содержит галочки
```

NULL

Null — специальное значение. **Оно имеет смысл «ничего».**

Значение null не относится ни к одному из типов выше, а образует свой отдельный тип, состоящий из единственного значения null:

```
var age = null;
```

UNDEFINED

Если переменная объявлена, но в неё ничего не записано,

то ее значение как раз и есть undefined:

```
var u;
alert(u); // выведет "undefined"
```

В явном виде undefined обычно не присваивают, так как это противоречит его смыслу. Для записи в переменную «пустого значения» используется null.

OBJECT

Предыдущие типы называют *«примитивными»*.

Объект (т. е. член объектного типа данных) представляет собой коллекцию значений (либо элементарных, таких как числа и строки, либо сложных, например других объектов).

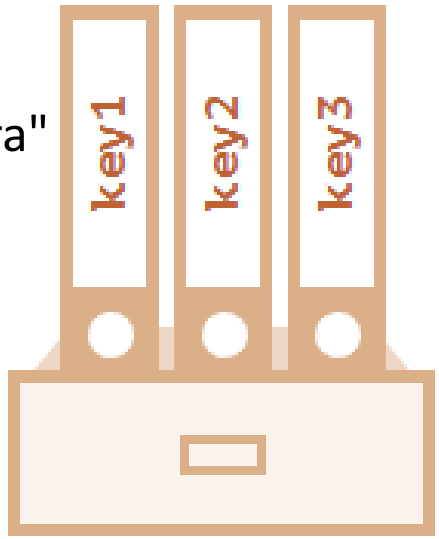
OBJECT

Объект может быть создан с помощью фигурных скобок {…} с необязательным списком свойств.

Свойство – это пара «ключ: значение», где ключ – это строка (также называемая «именем свойства»), а значение может быть чем угодно.

```
let user = new Object(); // синтаксис "конструктор объекта"

let user = {}; // синтаксис "литерал объекта"
```



OBJECT

Объект

Свойство

car.name = Fiat

car.model = 500

car.weight = 850kg

car.color = white

Метод

car.start()

car.drive()

car.brake()

car.stop()

OBJECT

```
const car = {
type:"Fiat",
model:"500",
color:"white“
};
```

Доступ:

car.type или car[‘type’]

OBJECT

Метод – это функция, хранящаяся внутри объекта

```
const person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

Вызов:

name = person.fullName();

THIS

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

«this» не является фиксированным

THIS

Значение this определяется во время исполнения кода.

- При объявлении любой функции в ней можно использовать this, но этот this не имеет значения до тех пор, пока функция не будет вызвана.
- Функция может быть скопирована между объектами (из одного объекта в другой).
- Когда функция вызывается синтаксисом «метода» – object.method(), значением this во время вызова является object.

ОПРЕДЕЛЕНИЕ ТИПА

Оператор typeof позволяет нам увидеть, какой тип данных сохранён в переменной.

МАССИВЫ

Для хранения упорядоченных коллекций существует особая структура данных, которая называется массив, Array.

```
let arr = new Array();
let arr = [];

let fruits = ["Яблоко", "Апельсин", "Слива"];

alert( fruits[0] ); // Яблоко
```

ОПЕРАЦИИ В JS

Мы рассмотрим 4 основных вида операций **в JS**:

1. Операция присваивания,
2. Арифметические операции,
3. Операции сравнения,
4. Логические операции.

ТЕРМИНЫ

Операнд — то, к чему применяется оператор. Например: 5 * 2 — оператор умножения с левым и правым операндами.

Унарным называется оператор, который применяется к одному выражению.

Бинарным называется оператор, который применяется к двум операндам.

АРИФМЕТИЧЕСКИЕ ОПЕРАТОРЫ

Операция	Значение
+	Сложение
-	Вычитание
*	Умножение
**	Возведение в степень (ES2016)
/	Деление
%	Деление по модулю
++	Increment
--	Decrement

ПРИСВАИВАНИЕ

Оператор	Пример	То же самое
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

СРАВНЕНИЕ

Оператор	Описание
==	равно
===	Одно значение и один тип
!=	Не равно
!==	Разные значения и тип
>	больше
<	меньше
>=	Больше или равно
<=	Меньше или равно
?	Тернарный оператор

ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Operator	Description
&&	Логическое И
||	Логическое ИЛИ
!	Логическое НЕ

УСЛОВИЕ

```
if(true) {
// делаем что-то
} else {
// иначе делаем другое
}
```

SWITCH

Конструкция switch заменяет собой сразу несколько if.

Она представляет собой более наглядный способ сравнить выражение сразу с несколькими вариантами.

```
switch(x) {
case 'value1': // if (x === 'value1')
  ...
  [break]
case 'value2': // if (x === 'value2')
  ...
  [break]
default:
  ...
  [break]
}
```

ЦИКЛЫ В JS

При написании скриптов зачастую встаёт задача сделать однотипное действие много раз.

Например, вывести товары из списка один за другим. Или просто перебрать все числа от 1 до 10 и для каждого выполнить одинаковый код.

Для многократного повторения одного участка кода предусмотрены циклы.

- for
- while

WHILE

```
while (condition) {
 // код
// также называемый "телом цикла"
}
Пример:
let i = 0;
while (i < 3) {
// выводит 0, затем 1, затем 2
alert( i );
 i++;
}
```

FOR

Более сложный, но при этом самый распространённый цикл — цикл for.

```
for (начало; условие; шаг) {
// ... тело цикла ...
}
```

```
for (let i = 0; i < 3; i++) {
// выведет 0, затем 1, затем 2
alert(i);
}
```

ФУНКЦИИ

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

```
function имя(параметры) {
        ...тело...
}
```

```
имя(); //вызов функции
```

ЛОКАЛЬНЫЕ ПЕРЕМЕННЫЕ

Переменные, объявленные внутри функции, видны только внутри этой функции.

```
function showMessage() {
  let message = "Привет, я JavaScript!"; // локальная переменная
  alert( message );
}

showMessage(); // Привет, я JavaScript!
alert( message ); // <-- будет ошибка, т.к. переменная видна только внутри функции
```

ВНЕШНИЕ ПЕРЕМЕННЫЕ

У функции есть доступ к внешним переменным, например:

```
let userName = 'Вася';
function showMessage() {
  let message = 'Привет, ' + userName;
  alert(message);
}

showMessage(); // Привет, Вася
```

ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ

Переменные, объявленные снаружи всех функций, такие как внешняя переменная userName в вышеприведённом коде – называются глобальными.

Глобальные переменные **видимы для любой функции** (если только их не перекрывают одноимённые локальные переменные).

Желательно сводить использование глобальных переменных к минимуму.

ВОЗВРАТ ЗНАЧЕНИЯ

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

Простейшим примером может служить функция сложения двух чисел:

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

let result = sum(1, 2);
alert( result ); // 3
```

ВЫБОР ИМЕНИ ФУНКЦИИ

Функция – это действие. Поэтому имя функции обычно является глаголом. Оно должно быть кратким, точным и описывать действие функции, чтобы программист, который будет читать код, получил верное представление о том, что делает функция.

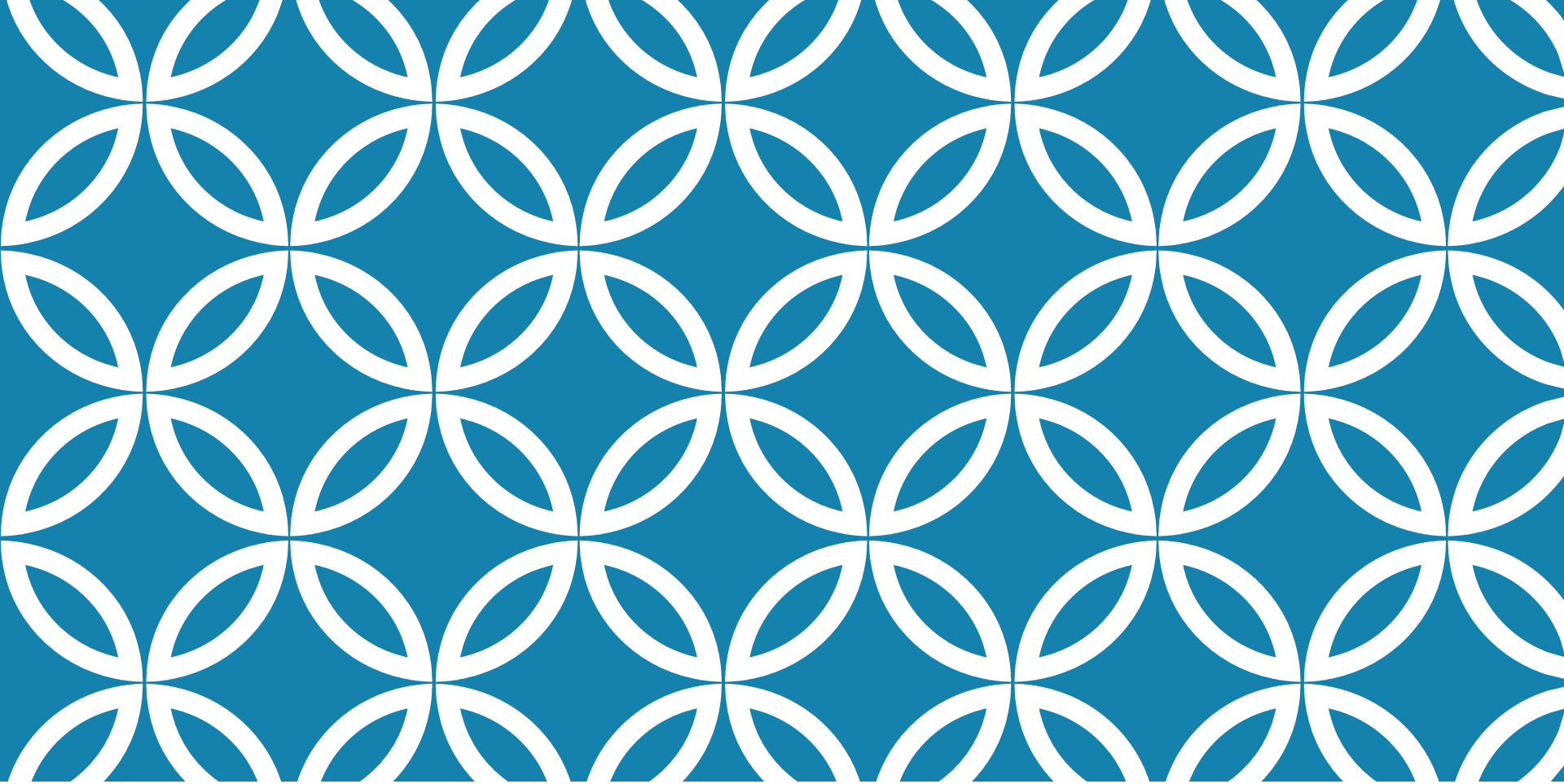
Функции, начинающиеся с…

"get…" – возвращают значение,

"calc…" – что-то вычисляют,

"create…" – что-то создают,

"check…" – что-то проверяют и возвращают логическое значение, и т.д.



СПАСИБО ЗА ВНИМАНИЕ!

конец лекции 5