**Методы объекта, "this". Конструкторы, создание объектов через "new"**

**Слайд 2**

# Методы объекта, "this"

Объекты обычно создаются, чтобы представлять сущности реального мира, будь то пользователи, заказы и так далее:

// Объект пользователя

let user = {

name: "Джон",

age: 30

};

И так же, как и в реальном мире, пользователь может *совершать действия*: выбирать что-то из корзины покупок, авторизовываться, выходить из системы, оплачивать и т.п.

Такие действия в JavaScript представлены свойствами-функциями объекта.

**Слайд 3**

**[Примеры методов](https://learn.javascript.ru/object-methods" \l "primery-metodov)**

Для начала давайте научим нашего пользователя user здороваться:

let user = {

name: "Джон",

age: 30

};

user.sayHi = function() {

alert("Привет!");

};

user.sayHi(); // Привет!

Здесь мы просто использовали Function Expression (функциональное выражение), чтобы создать функцию для приветствия, и присвоили её свойству user.sayHi нашего объекта.

Затем мы вызвали её. Теперь пользователь может говорить!

Функцию, которая является свойством объекта, называют методом этого объекта.

Итак, мы получили метод sayHi объекта user.

**Слайд 4**

## [Ключевое слово «this» в методах](https://learn.javascript.ru/object-methods" \l "klyuchevoe-slovo-this-v-metodah)

Как правило, методу объекта необходим доступ к информации, которая хранится в объекте, чтобы выполнить с ней какие-либо действия (в соответствии с назначением метода).

Например, коду внутри user.sayHi() может понадобиться имя пользователя, которое хранится в объекте user.

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

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

**Слайд 5**

Например:

let user = {

name: "Джон",

age: 30,

sayHi() {

// this - это "текущий объект"

alert(this.name);

}

};

user.sayHi(); // Джон

Здесь во время выполнения кода user.sayHi() значением this будет являться user (ссылка на объект user).

**Слайд 6**

Технически также возможно получить доступ к объекту без ключевого слова this, ссылаясь на него через внешнюю переменную (в которой хранится ссылка на этот объект):

let user = {

name: "Джон",

age: 30,

sayHi() {

alert(user.name); // используем переменную "user" вместо ключевого слова "this"

}

};

**Слайд 7**

…Но такой код будет ненадёжным. Если мы решим скопировать ссылку на объект user в другую переменную, например, admin = user, и перезапишем переменную user чем-то другим, тогда будет осуществлён доступ к неправильному объекту при вызове метода из admin.

Это показано ниже:

let user = {

name: "Джон",

age: 30,

sayHi() {

alert( user.name ); // приведёт к ошибке

}

};

let admin = user;

user = null; // обнулим переменную для наглядности, теперь она не хранит ссылку на объект.

admin.sayHi(); // Ошибка! Внутри sayHi() используется user, которая больше не ссылается на объект!

Если мы используем this.name вместо user.name внутри alert, тогда этот код будет работать.

**Слайд 8**

# Конструкторы, создание объектов через "new"

Обычный синтаксис {...} позволяет создать только один объект. Но зачастую нам нужно создать множество однотипных объектов, таких как пользователи, элементы меню и т.д.

Это можно сделать при помощи функции-конструктора и оператора "new".

**Слайд 9**

[**Функция-конструктор**](https://learn.javascript.ru/constructor-new#funktsiya-konstruktor)

Функции-конструкторы являются обычными функциями. Но есть два соглашения:

1. Имя функции-конструктора должно начинаться с большой буквы.
2. Функция-конструктор должна вызываться при помощи оператора "new".

Например:

function User(name) {

this.name = name;

this.isAdmin = false;

}

let user = new User("Вася");

alert(user.name); // Вася

alert(user.isAdmin); // false

**Слайд 10**

Когда функция вызывается как new User(...), происходит следующее:

1. Создаётся новый пустой объект, и он присваивается this.
2. Выполняется код функции. Обычно он модифицирует this, добавляет туда новые свойства.
3. Возвращается значение this.

Другими словами, вызов new User(...) делает примерно вот что:

function User(name) {

// this = {}; (неявно)

// добавляет свойства к this

this.name = name;

this.isAdmin = false;

// return this; (неявно)

}

**Слайд 11**

То есть, результат вызова new User("Вася") – это тот же объект, что и:

let user = {

name: "Вася",

isAdmin: false

};