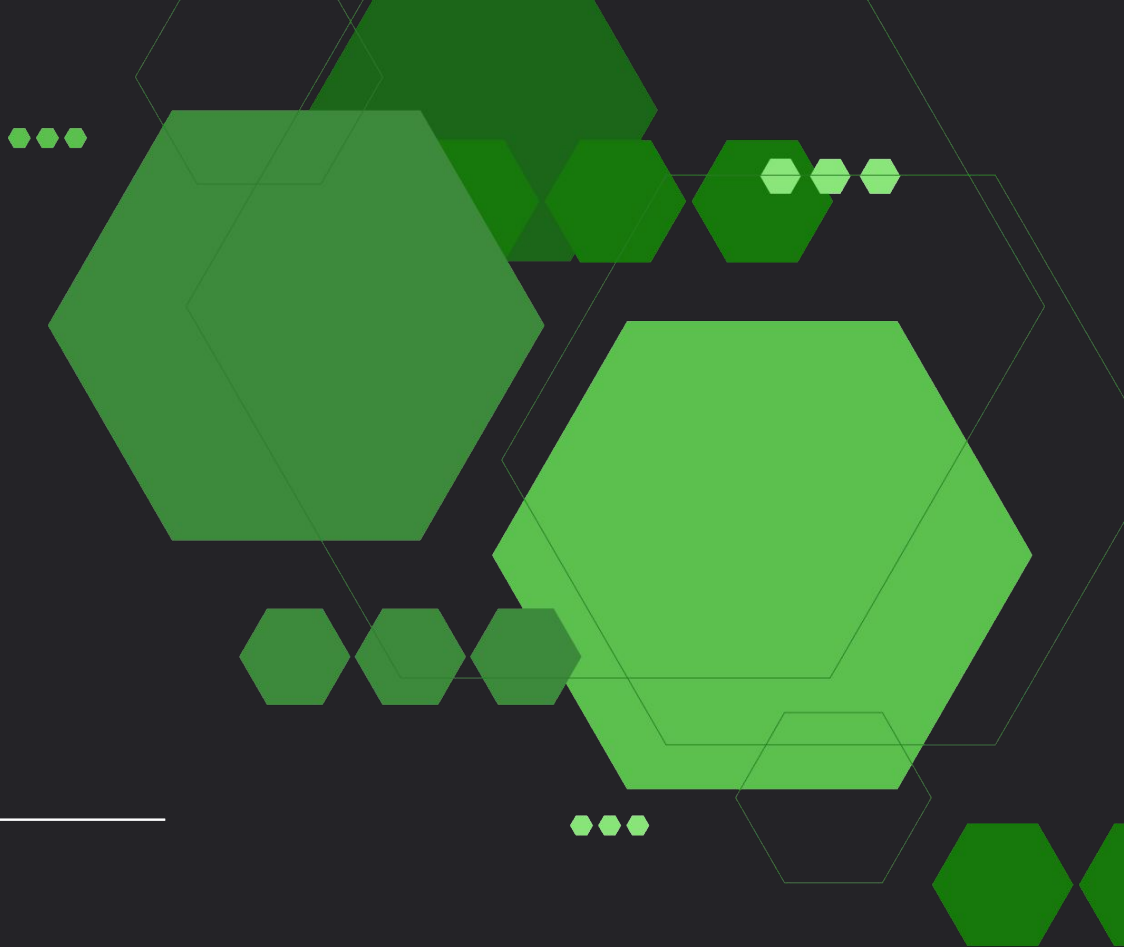


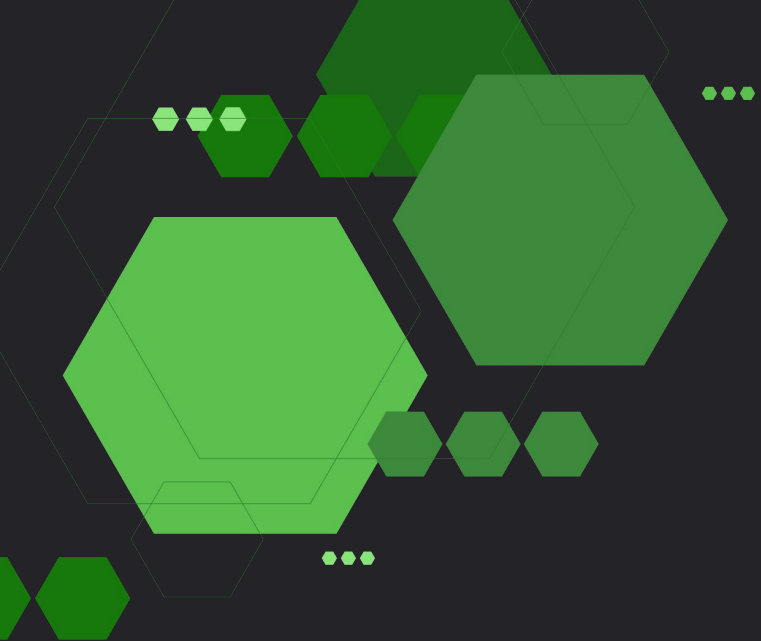
SKILLFACTORY

Функции и объекты

Юлия Токаревская

Frontend-разработчик в Socialbakers





Функции

Функции

Определение функции

Функция - фрагмент кода, который может вызываться многократно по ходу выполнения программы

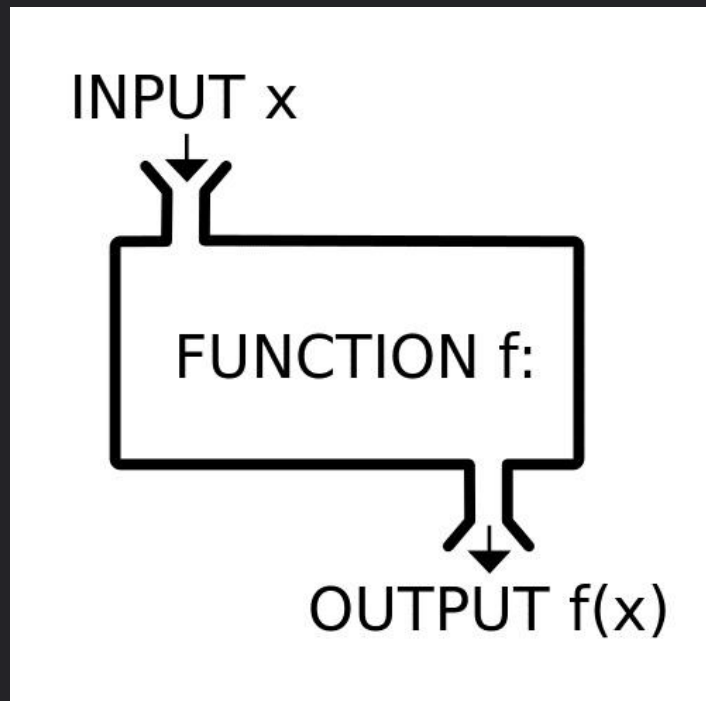
У функции должен быть идентификатор (имя), по которому к ней можно обратиться.

Определение функции

```
function имя_функции(список_параметров) {  
    блок кода  
}
```

Вызов функции

```
имя_функции(список_параметров);
```



Параметры функции. Ключевое слово return

Функция с аргументами:

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

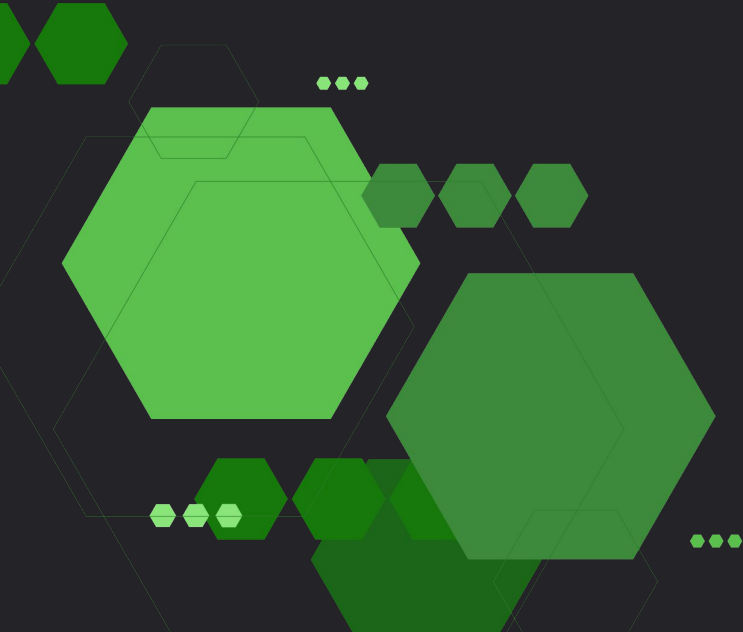
Функция без аргументов:

```
function sayHello() {  
    alert("Hello!");  
}
```

Если при определении функции у неё заданы аргументы, но при вызове аргумент не передан, он будет равен **undefined**.

Ключевое слово **return** останавливает выполнение функции и возвращает получившееся значение. Если значение не указано или слово return отсутствует, возвращается **undefined**.

Function Declaration и Function Expression



Разница между FD и FE

Function Declaration (FD)

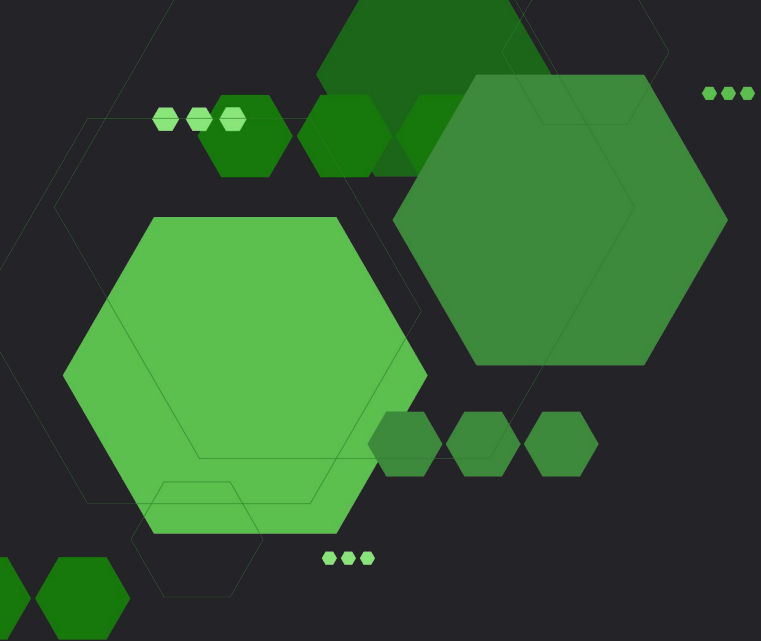
```
function square(a) {  
    return a * a;  
}
```

Function Expression (FE)

```
let square = function(a) {  
    return a * a;  
}
```

Функции, объявленные с помощью **FD**, можно вызвать в любом месте кода, в т. ч. до объявления.

Функции, объявленные как **FE**, ведут себя по-разному в зависимости от того, какой оператор был использован (var, let, const).



Объекты

Определение объекта

Объект — это набор **свойств**, и каждое свойство состоит из имени (ключа) и значения, ассоциированного с этим именем.


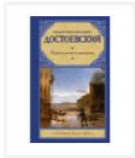


Значением свойства может быть также функция, которая в таком случае называется **методом** объекта.

```
let obj = {  
  key: "value",  
  method: function() { console.log("hello");  
}  
};
```

```
console.log(obj.key); // "value"  
obj.method(); // "hello"
```


Объекты в реальной жизни и в JS

Другие издания:



135 Р 145 Р **165 Р** 190 Р **231 Р** 273 Р **275 Р** 290 Р

Автор	Достоевский Федор Михайлович
Серия	Всемирная литература
Издательство	Эксмо
Год выпуска	2016
Тип обложки	Твердый переплет

```
let book = {  
  name: "Преступление и наказание",  
  author: "Достоевский Ф.М.",  
  year: 2016,  
  publisher: "Эксмо",  
  serie: "Всемирная литература",  
  cover: "Твердый переплет",  
  price: 135  
}
```

Ключевое слово **this**

Ключевое слово **this** содержит ссылку на текущий объект (иногда говорится “**содержит текущий контекст**”).

Как правило, **this** используется в методах объекта в случаях, когда необходим доступ к информации, которая хранится в объекте, чтобы выполнить с ней какие-либо действия. Кроме того, часто используется в **функциях-конструкторах**.

Создание объектов

Литеральная нотация

```
let book = {  
  name: "Преступление и наказание",  
  author: "Достоевский Ф.М.",  
  year: 2016,  
  price: 135  
};
```

Конструктор

```
function Book(name, author, year, price)  
{  
  this.name = name;  
  this.author = author;  
  this.year = year;  
  this.price = price;  
}  
  
let book = new Book("Преступление и  
наказание", "Достоевский Ф.М.", 2016,  
135);
```

Конструктор объекта

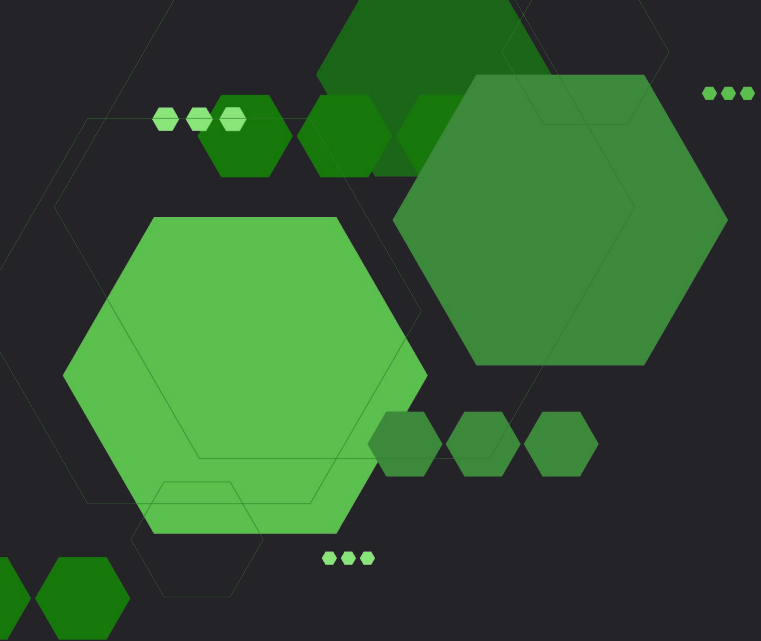
Конструктор объекта - функция, предназначенная для создания объектов.

Основная цель использования конструктора - удобное создание однотипных объектов.

- Имя функции-конструктора должно начинаться с большой буквы
- Функция конструктор вызывается с помощью ключевого слова **new**
- В качестве конструктора не может быть использована стрелочная функция

При создании объекта с помощью конструктора происходит следующее:

1. Создаётся новый пустой объект.
2. Ключевое слово **this** получает ссылку на этот объект.
3. Функция выполняется. Как правило, она модифицирует **this** (т.е. только что созданный объект), добавляет методы, свойства.
4. Возвращается **this**



Прототипное наследование

Прототипное наследование в JS

JavaScript — это **объектно-ориентированный** язык, основанный на **прототипировании**, а не на классах.

Прототип объекта — это объект, используемый в качестве шаблона для нового объекта. Новый объект получает все свойства и методы своего прототипа.

Любой объект может быть использован в качестве прототипа для другого объекта. Однако каждый объект может иметь **только один** прототип.

Примеры прототипов и объектов

Прототипы

Животное

Машина

Мебель



Объекты

Медведь

Audi

Стол

Реализация прототипирования в JS

__proto__ - системное свойство, которое содержит ссылку на прототип объекта или значение null, если прототип не задан.

При создании объекта через литерал {} его прототипом автоматически становится специальный встроенный объект, который содержит базовые методы для работы с объектом.

prototype - свойство, которое есть у всех функций. Это объект с единственным свойством constructor. Prototype используется при создании новых объектов оператором new

Свойство **__proto__** является скрытым и его использование в явном виде **некорректно!**

Прототипное наследование

Методы для работы с прототипами

Задать прототип:

```
Object.create(prototype) ;
```

```
Object.setPrototypeOf(obj, prototype) ;
```

Получить прототип:

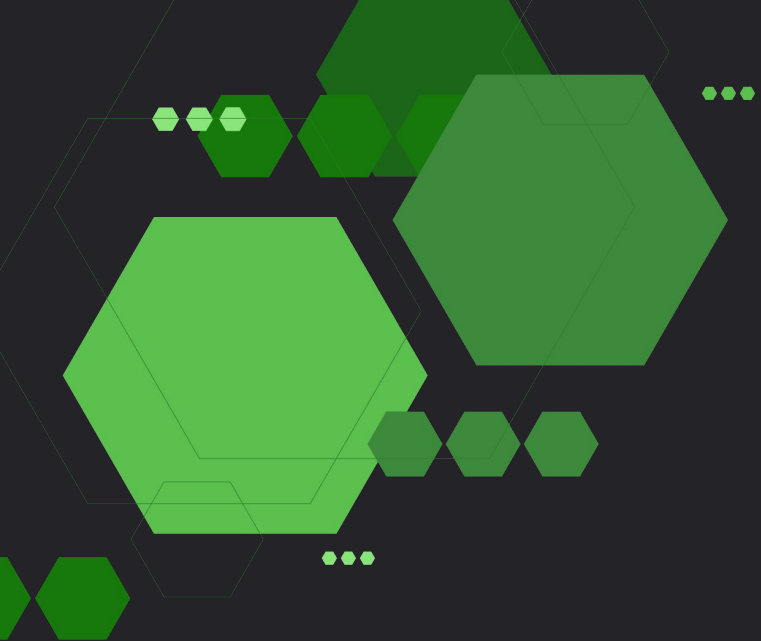
```
Object.getPrototypeOf(obj) ;
```

Проверить, является ли объект прототипом другого объекта:

```
prototype.isPrototypeOf(object) ;
```

Проверить, является ли свойство объекта собственным или унаследованным:

```
obj.hasOwnProperty(prop) ;
```



Классы в JS

Определение классов

Класс – это способ описания объекта, определяющий его состояние и поведение, а также правила взаимодействия с этим объектом.

Классы в Javascript отличаются от классов в других ООП-языках, т.к. в JS класс - разновидность функции-конструктора.

Класс

```
class Person {  
  constructor(name) {  
    this.name = name;  
  }  
}
```

Функция-конструктор

```
function Person (name) {  
  this.name = name;  
}
```

Отличия классов от функций-конструкторов

1. В отличие от обычных функций, конструктор класса не может быть вызван без ключевого слова `new`
2. Методы класса являются неперечисляемыми
3. Объявляемый класс не “поднимается”, в отличие от функций (т.е. его нельзя использовать до объявления)

Наследование классов

Для наследования классов используется ключевое слово **extends**

```
class Animal {  
    constructor(name, color) {  
        this.name = name;  
        this.color = color;  
    }  
  
    run() { ... }  
}  
  
class Cat extends Animal {  
    constructor(name, color, claws) {  
        super(name, color);  
        this.claws = claws;  
    }  
  
    sayMeow() { ... }  
}
```