

# Объектно -ориентированное программирование и наследование

Урок 3





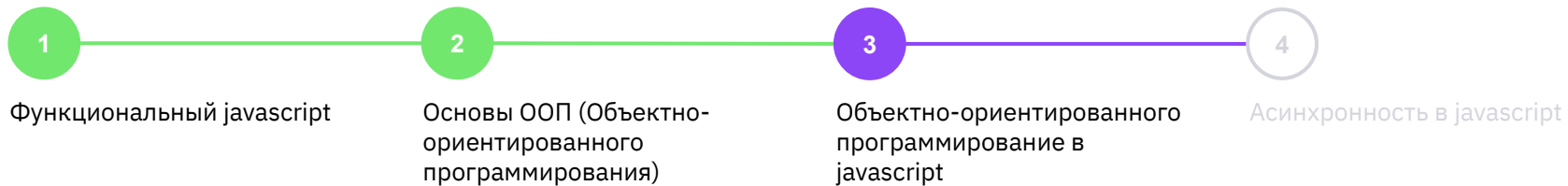
## Кадочников Алексей

Frontend-разработчик

- ☀ Веб-разработчик со стажем более 9 лет
- ☀ Преподаватель GeekBrains с 2015 года
- ☀ Автор курсов по Frontend на портале Geekbrains
- ☀ Работал в таких компаниях как VK и Wizard-C









# План курса





## Что будет на уроке сегодня

-  Прототип
-  Методы для установки прототипа.
-  Конструктор объекта
-  new
-  Object.create
-  Создание объектов и наследование с использованием class и extends



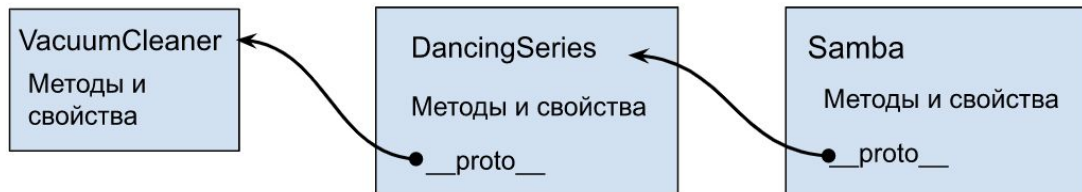
Прототип





## Прототип

код конкретного объекта стал очень маленьким, и мы получили цепочку объектов, связанных через прототипы благодаря свойству `__proto__`.





Методы для установки  
прототипа.





## Методы для установки прототипа.

Устанавливать прототип объекта можно используя свойство `__proto__`, но также в языке есть два метода для чтения и установки прототипа объекта - это **`getPrototypeOf`** и **`setPrototypeOf`**. Эти методы не доступны в браузере Internet Explorer версии ниже 10.





## **getPrototypeOf**

Метод `getPrototypeOf` позволяет получить ссылку на объект прототип.



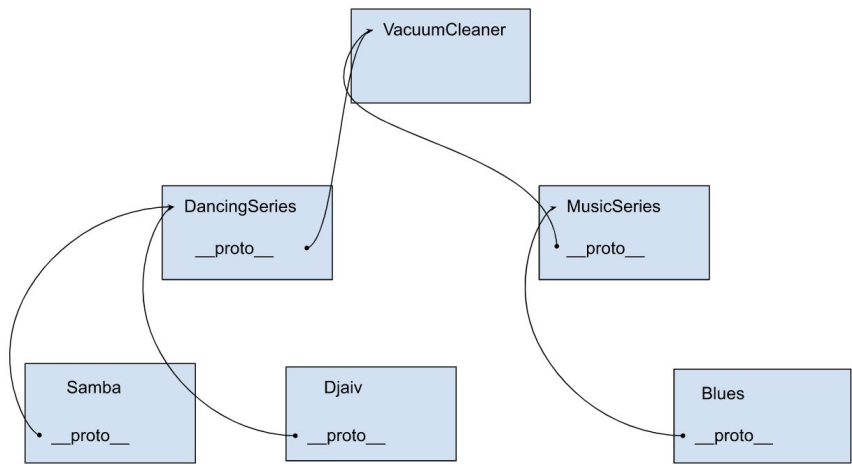
## setPrototypeOf

Для установки прототипа используем метод `setPrototypeOf` - он принимает два аргумента, первый это объект для которого устанавливается прототип, второй - это объект который будет прототипом для первого.



## setPrototypeOf

Давайте посмотрим на структуру созданных нами базовых моделей и всех роботов что мы уже создали, чтобы лучше увидеть наследование:





# Конструктор объекта





## Конструктор объекта

До этого момента мы создавали объекты и устанавливали прототип для одного конкретного объекта. А что если мы хотим создать сразу пять объектов одного типа



## new

Оператор new позволяет создавать новые объекты, используя для этого функцию-конструктор. Работает он следующим образом:

1. Создает пустой объект, который наполнит всем необходимым.
2. Устанавливает этот объект как `this` для функции конструктора, чтобы можно было использовать `this` внутри функции и добавлять свойства и методы в этот объект.
3. Вызывает функцию конструктор для инициализации объекта.
4. Если у функции конструктора есть свойство `prototype`, устанавливает значение этого свойства как прототип для нового объекта (свойство `__proto__`).
5. Устанавливает свойство `constructor` объекта ссылкой на функцию конструктор.
6. Если функция конструктор не возвращает ничего или возвращает какое-то примитивное значение, то оператор `new` вернет новый созданный и наполненный объект, если конструктор возвращает объект, то оператор `new` вернет этот объект.



Object.create





## Object.create

Метод `Object.create` позволяет создавать новые объекты, принимая в качестве аргументов объект прототип для создаваемого объекта, и вторым аргументом (необязательным) свойства для нового объекта в формате объект с ключами и значениями дескрипторов для свойств.





Создание объектов  
и наследование  
с использованием class  
и extends





## Создание объектов и наследование с использованием `class` и `extends`

Мы научились создавать объекты и устанавливать прототипы, создавать конструкторы и разобрались как работает оператор `new`. С приходом ES2015 в язык был добавлен синтаксис классов, чтобы все эти операции можно было делать удобнее и в более привычном синтаксисе для тех, кто уже программировал с использованием классов в других языках программирования.



## Создание объектов и наследование с использованием `class` и `extends`

Давайте теперь разберем что же делают классы. Работа классов достаточно проста и прозрачна. Объявляя класс, движок JavaScript создает функцию конструктор по имени класса и берет её код из метода `constructor` класса, если такого метода нет, то функция будет пустой. Если класс расширяет другой класс, то для этой функции указывается свойство `prototype`. После чего находит все остальные методы объекта и прописывает их в свойство `prototype` для новой функции.










## Создание объектов и наследование с использованием `class` и `extends`

Синтаксис классов намного проще и лучше читается, а для многих программистов, кто пришел к изучению JavaScript после других объектно-ориентированных языков он намного привычнее. К сожалению еще не все браузеры умеют работать с синтаксисом классов напрямую, поэтому чаще всего в проектах необходимо использовать дополнительные инструменты для конвертации кода ES2015 в более привычный для браузеров ES5.



## Итоги

-  Узнали что такое прототип
-  Ознакомились с методами для установки прототипа.
-  Разобрали что такое конструктор объекта
-  Изучили `new`
-  Ознакомились подробнее с `Object.create`
-  Узнали подробнее создание объектов и наследование с использованием `class` и `extends`

**Спасибо**   
**за внимание**

