

Основы ООП

Урок 2





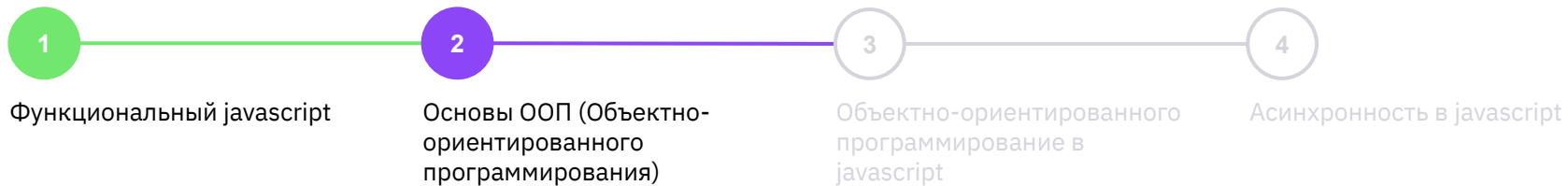
Кадочников Алексей

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

-  Объекты и их методы
-  this
-  Одалживание метода
-  Привязка контекста
-  Объект через class



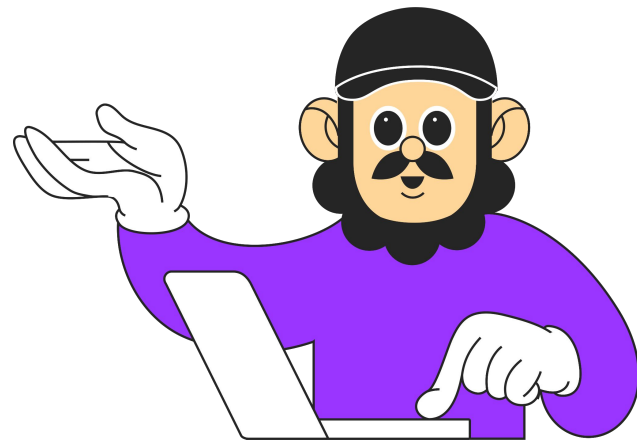
Объекты и их методы





Объекты и их методы

Программисты давно поняли, что проще всего создавать алгоритмы, оперируя в них объектами, которые отображают реальный мир. Так практически любую сущность реального мира можно представить в виде объекта с некоторыми свойствами и методами.





Объекты и их методы

Возьмем к примеру робот-пылесос, у него могут быть разные характеристики, такие как мощность двигателя, емкость аккумулятора, время работы без подзарядки, объем контейнера под мусор, также у него есть разные датчики, которые могут быть представлены булевыми переменными, например заполнен ли контейнер для мусора, или датчик препятствия. А также у робота-пылесоса есть методы, которые делают его не просто набором свойств, но и добавляют функциональности





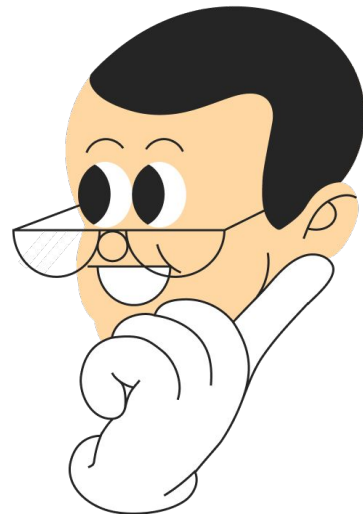
this





this

this - это ключевое слово в языке JavaScript, которое позволяет обратиться к свойствам и методам объекта внутри его методов. А также ключевое слово this доступно в любой функции, и либо принимает значение объекта, который являлся контекстом при вызове функции, либо undefined.





Одалживание метода





Одалживание метода

Что если мы хотим одолжить метод одного объекта и использовать его в другом объекте.





Привязка контекста





Привязка контекста

Когда мы вызывали методы объекта напрямую, после его создания, функция вызывалась имея возможность получить доступ к объекту, но когда функция вызывается внутри метода `setTimeout`, то эта функция теряет доступ к своему объекту, и ключевое слово `this` в такой функции получает значение `undefined`.

Вот тут и вступает в игру контекст вызова функции. Каждая функция вызывается в контексте некоторого объекта, если эта функция определена вне какого-то пользовательского объекта, то её контекстом будет глобальный объект (например `window` в браузере), а если определена в пользовательском объекте, и вызвана в нём, то контекстом для неё будет этот пользовательский объект.



Объект через class





Объект через class

Давайте рассмотрим вариант создания объекта через ключевое слово `class` и как осуществляется привязка контекста к методам в таком случае.











Объект через class

Подведем итог: любая функция имеет указатель `this` на свой контекст в момент вызова. Внутри объекта это обычно сам объект, но как мы могли видеть, если метод вызывается в отрыве от объекта, то этот указатель начинает ссылаться на глобальный объект, или принимает значение `undefined`, при использовании строгого режима в коде (“`use strict`”). Для решения таких ситуаций у каждой функции есть три метода `call`, `apply` и `bind`, которые позволяют привязать нужный нам контекст к функции во время её вызова (`call`, `apply`) или навсегда (`bind`).



Что будет на уроке сегодня

-  Ознакомились с объектами и их методами
-  Узнали что такое `this`
-  Узнали как работает одалживание метода
-  Освоили привязку контекста
-  Увидели как работает объект через `class`

Спасибо 
за внимание

