codecademy

Классы

Статические методы

В классе JavaScript `static` ключевое слово определяет статический метод для класса. Статические методы не вызываются для отдельных экземпляров класса, а вызываются для самого класса. Поэтому они, как правило, являются общими (полезными) методами.

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

  introduce() {
    console.log('This is '
+ this._name + ' !');
  }

  // A static method
  static bark() {
    console.log('Woof!');
  }
}

const myDog = new Dog('Buster');
myDog.introduce();

// Calling the static method
Dog.bark();
```

Сорт

JavaScript поддерживает концепцию *классов* как синтаксис для создания объектов. Классы определяют общие свойства и методы, которые будут иметь объекты, созданные из класса.
Когда объект создается на основе класса, новый объект упоминается как *экземпляр* класса. Новые экземпляры создаются с использованием `new` ключевого слова.
В примере кода показан класс, представляющий `Song` . Под ним создается новый вызываемый объект `mySong` , и вызывается `.play()` метод класса. Результатом будет текст, `Song playing!` напечатанный в консоли.

```
class Song {
  constructor() {
    this.title;
    this.author;
  }

  play() {
    console.log('Song playing!');
  }
}

const mySong = new Song();
mySong.play();
```

Конструктор класса

Классы могут иметь `constructor` метод. Это специальный метод, который вызывается при создании (экземпляре) объекта. Методы конструктора обычно используются для установки начальных значений объекта.

```
class Song {
  constructor(title, artist) {
    this.title = title;
    this.artist = artist;
  }
}

const mySong = new Song('Bohemian
Rhapsody', 'Queen');
console.log(mySong.title);
```

Методы класса

Свойства в объектах разделяются запятыми. При использовании `class` синтаксиса это не так. Между методами в классах нет разделителей.

```
class Song {
  play() {
    console.log('Playing!');
  }

  stop() {
    console.log('Stopping!');
  }
}
```

`extends`

Классы JavaScript поддерживают концепцию наследования — дочерний класс может *расширять* родительский класс. Это достигается использованием `extends` ключевого слова как части определения класса.

Дочерние классы имеют доступ ко всем свойствам экземпляра и методам родительского класса. Они могут добавлять свои собственные свойства и методы в дополнение к этим. Конструктор дочернего класса вызывает конструктор родительского класса, используя `super()` метод.

```
// Parent class
class Media {
  constructor(info) {
    this.publishDate
= info.publishDate;
    this.name = info.name;
  }
}

// Child class
class Song extends Media {
  constructor(songData) {
    super(songData);
    this.artist = songData.artist;
  }
}

const mySong = new Song({
  artist: 'Queen',
  name: 'Bohemian Rhapsody',
  publishDate: 1975
});
```