Inheritance: extends

The classes in JavaScript support single inheritance using the **`extends`** keyword.

In the expression `class Child extends Parent { }` the `Child` class inherits from Parent the constructor, fields, and methods.

If you'd like to call the parent constructor in a child class, you need to use the super() special function available in the child constructor.

For example, let's create a new child class `ContentWriter` that extends the parent class `User` and make `ContentWriter` constructor call the parent constructor of `User`, as well as initialize the posts field

```
class User {
  name;

  constructor(name) {
    this.name = name;
  }

  getName() {
    return this.name;
  }
}

class ContentWriter extends User {
  posts = [];

  constructor(name, posts) {
    super(name);    this.posts = posts;
  }
}

const writer = new ContentWriter("John Smith", ["Why I like JS"]);
console.log(writer.name);
console.log(writer.posts);
```

```
// Output
John Smith
["Why I like JS"]
```

`ContentWriter` inherits the constructor, the method `getName()` and the field `name` from `User` class. As well, the `ContentWriter` class declares a new field `posts`.

`Note that private members of a parent class are not inherited by the child class.`

`super(name)` inside the child class `ContentWriter` executes the constructor of the parent class `User`.

Note that inside the child constructor you must execute `super()` before using `this` keyword. Calling `super()` makes sure that the parent constructor initializes the instance.

```
class Child extends Parent {
  constructor(value1, value2) {
    // Does not work!
    this.prop2 = value2;
    super(value1);
  }
}
```

Parent instance: super in methods

If you'd like to access the parent method inside of a child method, you can use the special shortcut `super`

```
class User {
  name;

  constructor(name) {
    this.name = name;
  }

  getName() {
    return this.name;
  }
}

class ContentWriter extends User {
  posts = [];

  constructor(name, posts) {
    super(name);
    this.posts = posts;
  }

  getName() {
    const name = super.getName();
    if (name === '') {
      return "Unknwon";
    }
    return name;
  }
}

const writer = new ContentWriter("", ["Why I like JS"]);
console.log(writer.getName());
```

```
// Output
"Unknown"
```

`getName()` of the child class `ContentWriter` accesses the method `super.getName()` directly from the parent class `User`.

This feature is called method **overriding**.

Note that you can use super with static methods too, to access the parent's public static methods.