

Constructor and "new" operator

Constructor function

Function that is meant to be a constructor are named with capital letter first and be executed with the `new` operator.

We use a constructor because it creates a shorter and simpler syntax compared to creating object with `{...}` every time.

```
function User(name) {  
  this.name = name;  
  this.isAdmin = false;  
}  
  
// Invoking constructor  
let user = new User("Ricky");  
console.log(user.name); // Ricky  
console.log(user.isAdmin); // false
```

When a function is executed with the `new` operator here it what happens

1. A new empty object is created and assigned to `this` implicitly
2. The function body executes, usually modifying `this` by adding new properties or method to it
3. Then the value of `this` is returned implicitly

So step 1 and step 3 is done implicitly for you already, you just need to worry about modifying `this`.

`new function() {...}`

Using this syntax you can create a single complex object that you don't aim to create again in the future. The constructor isn't saved anywhere.

```
let user = new function() {  
  this.name = "John"  
  this.isAdmin = false;  
};
```

The constructor function is created and immediately called with the `new` operator.

Returning from constructor

Normally, a constructor function don't have a return statement. But if there is one the rule are as follows:

1. If `return` is called with an object, then the object is returned instead of `this`
2. If `return` is called with a primitive, it's ignored

Methods in constructor

You can add method in a constructor function as well, the same way you would add a property!

```
function User(name) {  
  this.name = name;  
  
  this.foo = function() {  
    console.log(`My name is ` + this.name);  
  };  
}
```

You are able to write arrow functions and refer to `this` in constructor function as the `this` is bounded to the constructor's `this`.

```
function User(name) {  
  this.name = name;  
  
  this.foo = () => {  
    console.log(this.name);  
  };  
  
  this.bar = function() {  
    console.log(this.name);  
  }  
}  
  
let u1 = new User("Ricky");  
u1.foo(); // Ricky  
u1.bar(); // Ricky
```

Revision #1

Created 20 December 2022 15:09:19 by Tamarine

Updated 20 December 2022 16:13:33 by Tamarine