Constructors

Class	js
Created	@Jan 16, 2021 7:01 PM
Materials	
Reviewed	☐
Type	

1-Constructors are functions.

2-We call them with the new keyword.

3-Return a newly created object

```
function Country(name) {
     this.name = name;
 }

 var germany = new Country('Germany');
```

4-When a constructor is called with new, **this in the body of the constructor function refers to the newly created object** that will be returned.

5- **Note that the Country function above does not have a return statement**.

6- **No return statement is necessary since the newly created object will be returned automatically.**

7-In fact, if a constructor called with new does have a `return` statement that returns a primitive value, the newly created object will still be returned and the primitive value will not be.

```
function Country(name) {
      this.name = name;
      return 10;
  }

  var germany = new Country('Germany'); //{ name: 'Germany' } (and not 10)
```

8-However, a `return` statement that returns a different object will be effective.

```
function Country(name) {
    this.name = name;
    return {};
}

var germany = new Country('Germany'); //{} (and not { name: 'Germany' })
```

Prototypes

1-Every function automatically has a prototype property

2-The prototype's value is and object.

3-If a function is not called with new, it's prototype property does not have any role to play.

4-Once we use the constructor function to create an object, the object created is automatically connected to the prototype object of the constructor by a prototype chain.

5-The connection between instances and their prototype is 'live' - changes to the properties of the prototype are immediately visible when those properties are accessed via the instances.

```
function GermanCity(name) {
    this.name = name;
}

GermanCity.prototype.country = 'Germany';

var berlin = new GermanCity('Berlin');
var hamburg = new GermanCity('Hamburg');

berlin.country; //'Germany'
hamburg.country; //'Germany'

GermanCity.prototype.country = 'Deutschland';

berlin.country; //'Deutschland'
hamburg.country; //'Deutschland'
```

The `instanceof` operator

The `instanceof` operator is used to test whether a given constructor exists as a constructor property of any of the prototypes in an object's prototype chain.

```
var date = new Date;

date instanceof Date; //true

date instanceof Object; //true

date instanceof Array; //false
```

If you would like to make a constructor that can be called without `new` and still return an instance, the `instanceof` operator can help with that.

```
function Country(name) {
    if (!(this instanceof Country)) {
        return new Country(name);
    }
    this.name = name;
}

var country = Country('Germany');

country instanceof Country; //true
```