academeez

Prototype

How inheritance works in JavaScript

© Copyright Nerdeez Ltd

Prototype

- JavaScript in it’s own way is an Object Oriented programming language
- JavaScript OOP works very differently that other programming languages
- Before we can really understand how OOP works in JavaScript, create classes, instances, inheritance, etc. We have to first understand the basic OOP engine in Javascript - Prototype

© Copyright Nerdeez Ltd

The purpose of Prototype

academeez

- When we create an object in Javascript, that Object will contain methods
  - An Object can be any type we are using like: String, Number, Function, etc.
- For example creating a simple variable containing a String, that variable will contain methods and properties we can use
- The Prototype mechanism determines which properties and methods are available in our variable

```
// define string variable
var hello = 'world';

// our variable contain properties and
// methods we can use
hello.length;
hello.toString();
```

© Copyright Nerdeez Ltd

Finding our Prototype

- When you define a variable you can find that variable prototype using: **Object.getPrototypeOf**
- For example to find the prototype of a String variable

```
// define string variable
var hello = 'world';

// get the prototype of hello variable
Object.getPrototypeOf(hello) === String.prototype // true
```

© Copyright Nerdeez Ltd

Inheriting from prototype

academeez

- The variable we define will get methods and properties defined in it’s prototype.
- In the following example our variable with contain the methods and properties defined inside **String.prototype**

```
// define string variable
var hello = 'world';

// get the prototype of hello variable
Object.getPrototypeOf(hello) === String.prototype // true
```

© Copyright Nerdeez Ltd

String.prototype - example of prototype

- We can examine the **String.prototype** to see what methods and properties our variable will contain:

```
// String.prototype
{
    ...
    trimLeft: ƒ trimStart()
    trimRight: ƒ trimEnd()
    trimStart: ƒ trimStart()
    trimStart: ƒ trimStart(),
    valueOf: ƒ valueOf(),
    __proto__: Object
}
```

© Copyright Nerdeez Ltd

Other prototypes examples

- When we define a number variable, that variable will get the **Number.prototype**
- When we define a function variable, that variable will get the **Function.prototype**
- When we define an array variable, that variable will get the **Array.prototype**

```
const stringExample = 'hello world';
const numberExample = 42;
const arrayExample = [1,2,3,4];

Object.getPrototypeOf(stringExample) === String.prototype;
Object.getPrototypeOf(numberExample) === Number.prototype;
Object.getPrototypeOf(arrayExample) === Array.prototype;
```

© Copyright Nerdeez Ltd

Inheriting method / properties in prototype

academeez

- An array variable is attached to an Array.prototype
- When looking for methods and properties Javascript is looking for them in the prototype as well
- This means the prototype can contain methods and properties common for every array
- We can use the prototype to create a different methods and properties for a certain type

```
const arrayExample = [1,2,3,4];

arrayExample.sort()
```

© Copyright Nerdeez Ltd

Prototypes also have prototypes

academeez

- A Prototype also may have a prototype
- Which means that a property and method will be looked on the prototype of the prototype, and so on...
- Example:

```
// a prototype is also connected to a prototype
Object.getPrototypeOf(String.prototype) === Object.prototype; // true
```

© Copyright Nerdeez Ltd

Prototype chain

- Property methods will be looked in the prototype and in the prototype of the prototype and so on
- The prototypes will form a chain called Prototype chain
- The chain is connected by a the __proto__ property

```
const helloString = 'hello world';
helloString.__proto__ === String.prototype;
helloString.__proto__.__proto__ === Object.prototype;
```

© Copyright Nerdeez Ltd

Prototype chain - last elements

- Usually the last element in the chain will be **Object.prototype**
- After that the **__proto__** will be set to **null**

```
const helloString = 'hello world';
helloString.__proto__ === String.prototype;
helloString.__proto__.__proto__ === Object.prototype;
helloString.__proto__.__proto__.__proto__ === null;
```

© Copyright Nerdeez Ltd

new

- Creates an object with a prototype
- For example:
  - The following code will create a **person** variable with it's prototype set to the function **Person** prototype

```
function Person() {

}

const person = new Person();
person.__proto__ === Person.prototoype;
person.__proto__.__proto__ === Object.prototype;
```

© Copyright Nerdeez Ltd

Search property method algorithm

- First it will look for the property in the object
- It will then look for the property in the **__proto__** of the object
- It will then look for the property in the **__proto__.__proto__** and so on

```
function Person(name) {
    if (name) {
        this.name = name;
    }
}

Person.prototype.name = 'default name'

const yariv = new Person('Yariv');
const anonymous = new Person();
yariv.name === 'Yariv';
anonymous.name === 'default name';
```

© Copyright Nerdeez Ltd

Summary

academeez

- The entire OOP system in Javascript is built around prototypes
- When you create an object or an instance of a class you are creating a chain of prototypes
- Javascript will first look in the object than will start looking in the prototype chain

© Copyright Nerdeez Ltd

academeez

Thank You

Next Lesson: Class

Create class and instances

© Copyright Nerdeez Ltd