SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)
Approvedy by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chennai

Department of MCA

Advanced JavaScript Constructor

Course: 19CAT901 – Web Programming Essentials

Unit IV : Advanced Java Script

I Semester / I MCA

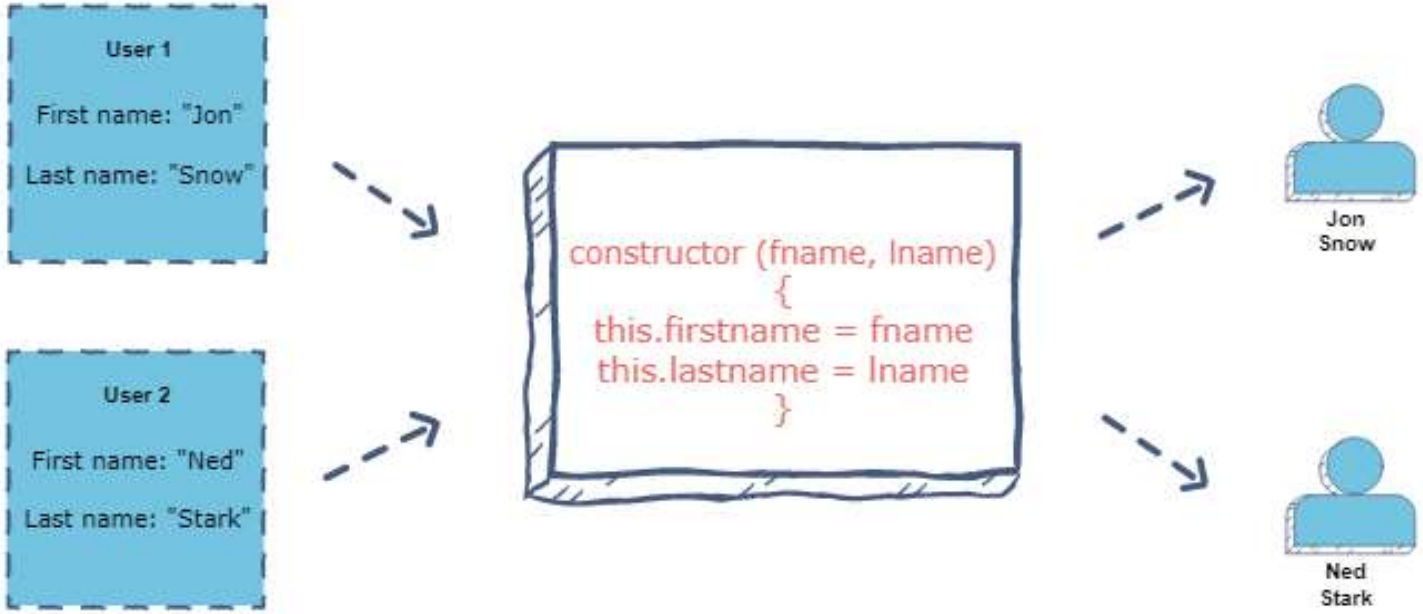
What is a constructor in JavaScript?

- A constructor is a function that creates an instance of a class which is typically called an “object”. In JavaScript, a constructor gets called when you declare an object using the new keyword.
- The purpose of a constructor is to create an object and set values if there are any object properties present. It’s a neat way to create an object because you do not need to explicitly state what to return as the constructor function, by default, returns the object that gets created within it.

What is a constructor in JavaScript?

- Example
- Let’s say there is an object, User, which has two properties: firstname and lastname.
- To initialize two instances with different names, you will use the same constructor function, as shown in the figure below:

What is a constructor in JavaScript?



What happens when a constructor gets called?

In JavaScript, here's what happens when a constructor is invoked:

- A new empty object is created
- this keyword starts referring to that newly created object and hence it becomes the *current instance* object
- The newly created object is then returned as the constructor's returned value
- Given below is the code to initialize two user instances just as shown in the above illustration:

What happens when a constructor gets called?

```
function User(first, last) {
  this.firstName = first
  this.lastName = last
}
var user1 = new User("Jon", "Snow")
console.log(user1)
var user2 = new User("Ned", "Stark")
console.log(user2)
```

Output

1.54s

User { firstName: 'Jon', lastName: 'Snow' }

User { firstName: 'Ned', lastName: 'Stark' }

Default constructors

In JavaScript, if you don’t specify any constructor, a default constructor is automatically created which has no parameters:

constructor(){ }

Note: Constructors have the same name as that of the object which is being created. As a convention, the first alphabet is kept capital in the constructor function.

Constructor Functions

In JavaScript, a **constructor function** is used to create objects.
It is considered good practice to name constructor functions with an **upper-case first letter**.

Example

```
const Phone = function (model, brand)
{
this.model = model,
this.brand = brand
}
phone.prototype.clickPicture = function()
{
console.log(`${this.brand} ${this.model} clicks picture!`)
}
const iphone = new Phone("Iphone 12", "Apple")
console.log(iphone.clickPicture())
```

Constructor Functions

In the above example, function Phone() is an object constructor function.
To create an object from a constructor function, we use the new keyword.

Output

"Apple Iphone 12 clicks picture!“

Built-in constructors

Example

```
let a = new Object(); // A new Object object
let b = new String(); // A new String object
let c = new Number(); // A new Number object
let d = new Boolean(); // A new Boolean object
```

All these are constructor functions provided to us by javascript.

Code-

```
typeof Object
```

Output-

'function’

Classes

Another way of creating templates for JavaScript Objects are **Classes**.
They are very similar to constructor functions. Classes have their own constructors that help initialize variables/values.

Constructor method-

The constructor method is a special method:
It has to have the exact name "constructor"
It is executed automatically when a new object is created
It is used to initialize object properties
If you do not define a constructor method, JavaScript will add an empty constructor method.
Just like constructor functions, we can use the new keyword to create new objects from this class.

Class Declarations Example: Re-declaring Class

```
class Phone
{
constructor(model, brand)
{
this.model = model
this.brand = brand
}
clickPicture()
{
console.log(`${this.brand} ${this.model} clicks picture!`)
}
}
const iphone = new Phone("Iphone 12", "Apple")
console.log(iphone.brand)
console.log(iphone.clickPicture())
```

Class

- Here iphone is an instance of the class Phone.
- There is a reason why we do not call the clickPicture function from inside the constructor because, the constructor is called every time we instantiate a new object and that would initialize the function every time instead of sharing it among objects thus taking up more space in memory.

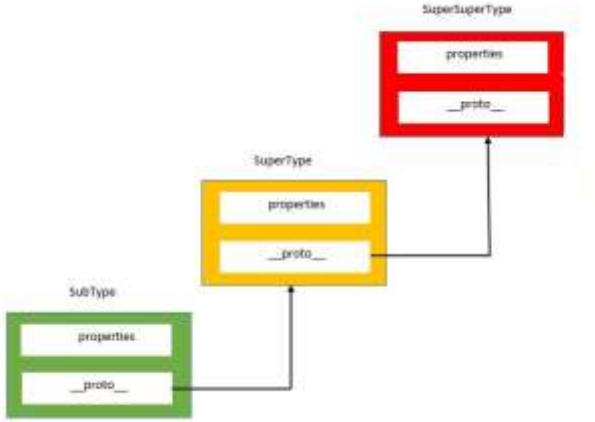
Output-

"Apple" "Apple Iphone 12 clicks picture!“

Classes in JavaScript is just syntactical sugar but under the hood it still uses **prototypal inheritance**.

Inheritance in Classes

- Inheritance enables you to define a class that takes all the functionality from a parent class and allows you to add more. Using class inheritance, a class can inherit all the methods and properties of another class.
- The super() method refers to the parent class.



Class expressions

```
class Phone
{
constructor(model, brand)
{
this.model = model
this.brand = brand
}
clickPicture()
{
console.log(`${this.brand} ${this.model} clicks picture!`)
}
}
```

Class expressions

```
class Iphone extends Phone
{
 constructor(model, brand, software)
{
super(model, brand) this.software = software
 }
}
const tenPro = new Iphone("10 pro", "Apple", "IOS")
 console.log(tenPro)
```

Output-

```
{ brand: "Apple",
model: "10 pro",
software: "IOS" }
```

new keyword in JS

The new operator lets developers create an instance of a user-defined object type or of one of the built-in object types that has a constructor function.

```
function Car(make, model, year, owner)
{
this.make = make;
this.model = model;
this.year = year;
this.owner = owner;
}
```

Output-

```
var car1 = new Car('Eagle', 'Talon TSi', 1993, rand);
var car2 = new Car('Nissan', '300ZX', 1992, ken);
```

References

- https://dev.to/pranav016/advanced-javascript-series-part-9-constructor-functions-object-oriented-new-keyword-1gg0
- https://www.educative.io/answers/what-is-a-constructor-in-javascript
- https://www.javatpoint.com/javascript-oops-constructor-method