Classes in JavaScript

Class basic syntax

Besides using a constructor function there is a more advance object oriented programming construct called class in JavaScript. The basic syntax is as follows:

```
class MyClass {
constructor() {}
    method1() {}
    method2() {}
}
```

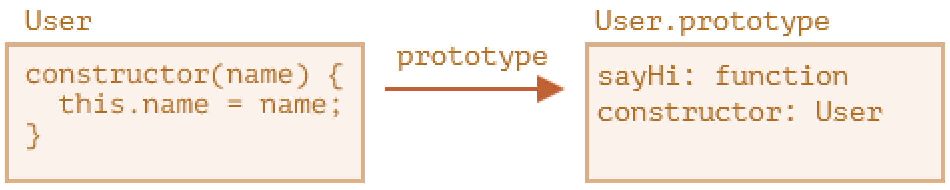
To utilize this class that you have just created you would use the same syntax as constructor functions: `let x = new MyClass()` to create a new object with all of the methods listed.

The `constructor()` method is automatically called by `new` and you would just do the same thing as a constructor function.

What happens underneath

When you write `class User {...}` what it happens underneath is that

1. Creates a function named `User`, and that is the result of the class declaration. The function code is taken from the `constructor` method, assume empty you didn't write such method
2. Stores class method that you wrote inside the class in `User.prototype` just like all the other native prototypes, i.e. `String.prototype, Array.prototype, Number.prototype`



So all the method that you call on the instantiated object will be on the taken from `YourClass.prototype.<methods>`

Not just a syntactic sugar

Many people say that the class declaration is a syntaxsyntactic sugar on top of constructor function. Yes, but there are still some differences:

1. The constructor created by class declaration must be invoked using `new` unlike constructor method where you can invoke it directly, even though it won't work properly
2. Class methods are non-enumerable, compared to constructor function's methods
3. Code inside class declaration are always `use strict`

Class expression

Like function expression you can also store class declaration into a variable or as something you would returned from a function, basically be part of another expression.

```
let User = class {
 sayHi() {
     console.log("Hi");
 }
};
```

Getters/setters

You can also provide getters and setters as well

Class fields

You can also add properties to the instances of class that you created.

These class fields are set on individual objects not under `Class.prototype`, so if you change one on one object, the other's class fields aren't affected.

```
class User {
 name = "John";

    sayHi() {
     console.log(`Hi I am ${this.name}`)
 }
}

new User().sayHi(); // Hello, John!
```

Revision #2
Created 27 December 2022 20:31:58 by Tamarine
Updated 27 July 2023 01:53:02 by Tamarine