CLASS/OBJECT, EVENTS AND DOM

I4GIC
By Thavorac

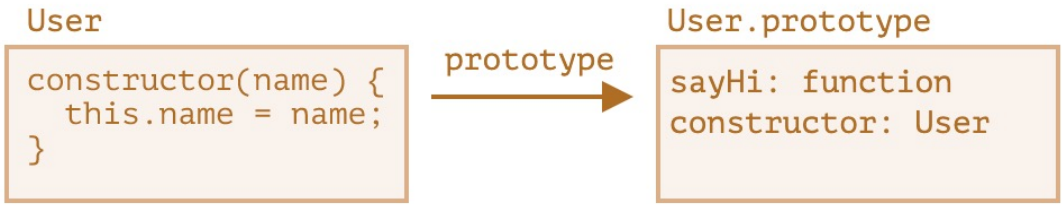
CLASSES AND OBJECTS

In object-oriented programming, a class is an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions or methods).

The basic syntax is:

```
class MyClass {
  // class methods
  constructor() { ... }
  method1() { ... }
  method2() { ... }
  method3() { ... }
  ...
}
```

Example




```
1  class User {
2
3    constructor(name) {
4      this.name = name;
5    }
6
7    sayHi() {
8      alert(this.name);
9    }
10
11 }
12
13 // Usage:
14 let user = new User("John");
15 user.sayHi();
```

Class's name

A function which is run by default at the time an object is initiated.

An action of the class

A new object is created.

The constructor runs with the given argument and assigns it to this.name

Gettters/Setters

- **Getter** — binds an object property to a function that will be called when that property is looked up.
- **Setter** — binds an object property to a function to be called when there is an attempt to set that property. It gives a simpler syntax for the properties and methods of an object.

```
1  class User {
2
3    constructor(name) {
4      // invokes the setter
5      this.name = name;
6    }
7
8    get name() {
9      return this._name;
10   }
11
12   set name(value) {
13     if (value.length < 4) {
14       alert("Name is too short.");
15       return;
16     }
17     this._name = value;
18   }
19
20 }
21
22 let user = new User("John");
23 alert(user.name); // John
24
25 user = new User(""); // Name is too short.
```

Class's properties/fields

The important difference of class fields is that they are set on individual objects, not User.prototype

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

  sayHi() {
    alert(`Hello, ${this.name}!`);
  }
}

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

CLASS INHERITANCE

Class inheritance is a way for one class to extend another class. So we can create new functionality on top of the existing.

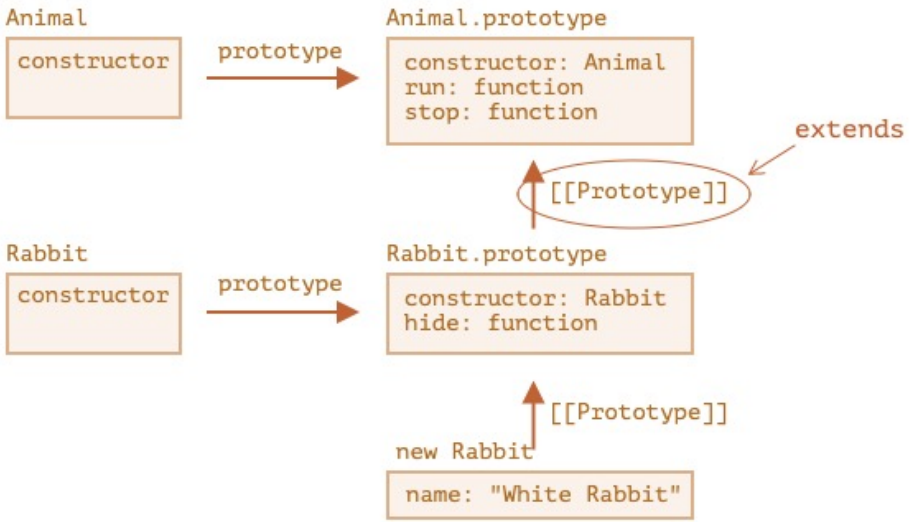
Syntax:

```
class B extends A {

    ..........

    ..........

}
```

```
1  class Animal {
2    constructor(name) {
3      this.speed = 0;
4      this.name = name;
5    }
6    run(speed) {
7      this.speed = speed;
8      alert(`${this.name} runs with speed ${this.speed}.`);
9    }
10   stop() {
11     this.speed = 0;
12     alert(`${this.name} stands still.`);
13   }
14 }
15
16 let animal = new Animal("My animal");
```

Example

```
1  class Rabbit extends Animal {
2    hide() {
3      alert(`${this.name} hides!`);
4    }
5  }
6
7  let rabbit = new Rabbit("White Rabbit");
8
9  rabbit.run(5); // White Rabbit runs with speed 5.
10 rabbit.hide(); // White Rabbit hides!
```



Overriding a method

An inherited class can override functions from parent class. Classes provide “super” keyword for that:

- super.method(...) to call a parent method.
- super(...) to call a parent constructor (inside our constructor only).

```
class Rabbit extends Animal {
  hide() {
    alert(`${this.name} hides!`);
  }

  stop() {
    super.stop(); // call parent stop
    this.hide(); // and then hide
  }
}

let rabbit = new Rabbit("White Rabbit");

rabbit.run(5); // White Rabbit runs with speed 5.
rabbit.stop(); // White Rabbit stands still. White Rabbit hides!
```

PRACTICE

STEP 1:

Create a class call Vehicle. The vehical has brand, type, color and price. Brand and type are important so it requires at the time an object is initialized from this class.

Vehicle can move forward, turn left, turn right and stop. Except stop, speeed can be provided as a parameter.

STEP 2:

Now, instead of providing type at the time of intialized object from Vehicle class. We create different sub-classes:

- Bicycle
- Motorbike
- Car

STEP 3:

To make our classes more interesting, we implement more sub-classes:

- Mountain Bike
- Honda Car
- Mercedes Car

You can provide more detail to these class to make it unique from its parent.

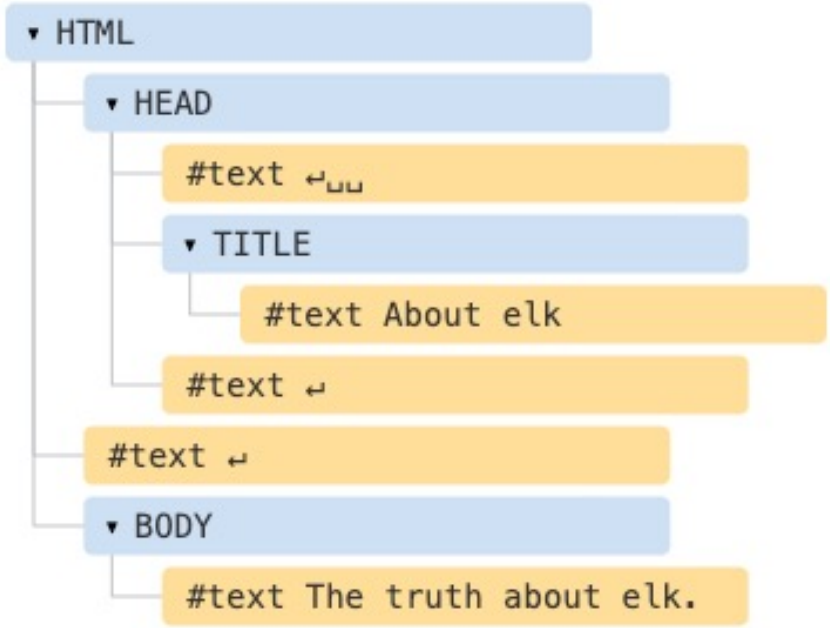
STEP 4:

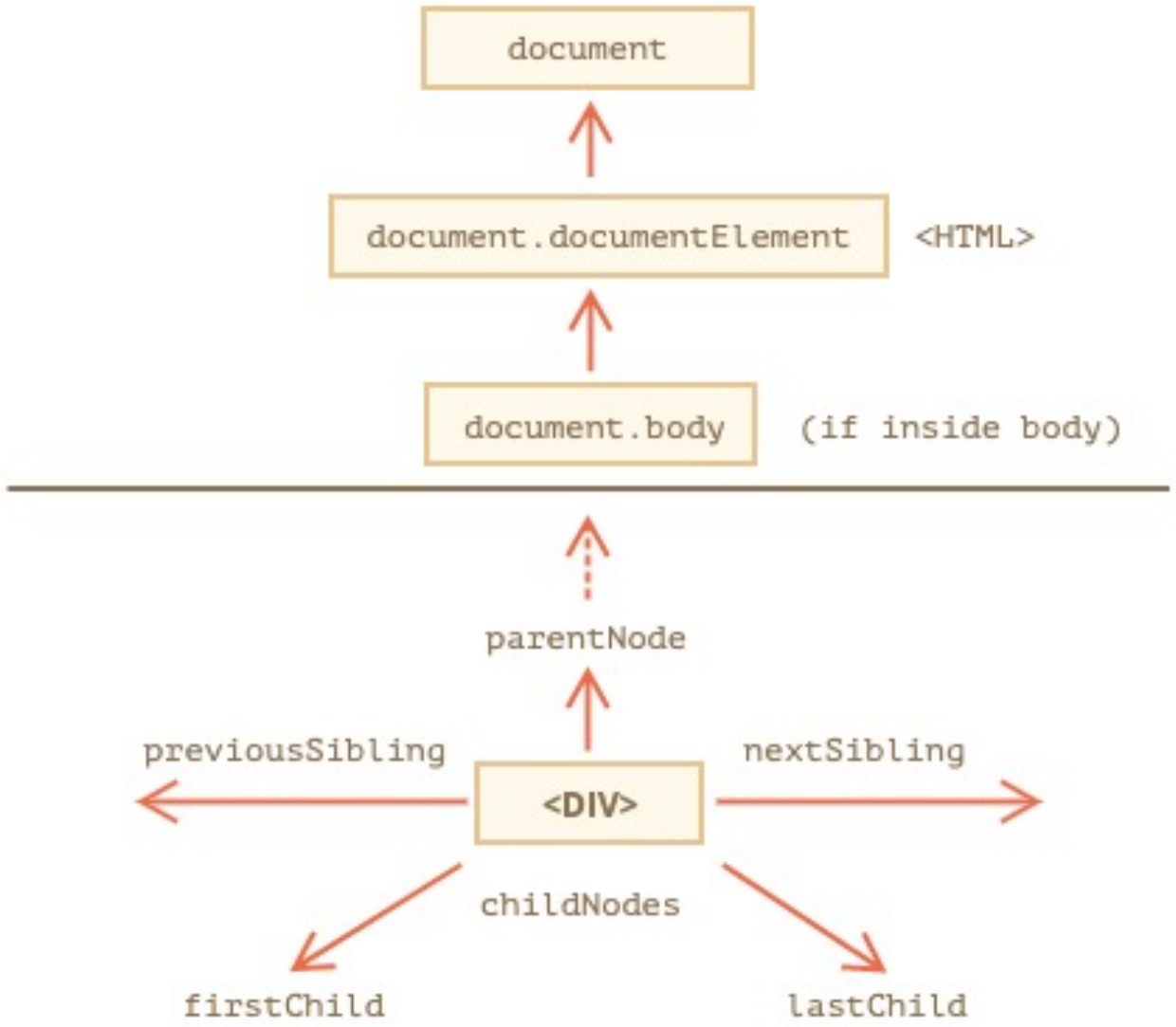
Write your class and appy to Canvas.

DOCUMENT OBJECT MODEL

Document Object Model (DOM) represent all the element inside HTML.

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4    <title>About elk</title>
5  </head>
6  <body>
7    The truth about elk.
8  </body>
9  </html>
```





<html> = document.documentElement

<body> = document.body

<head> = document.head

Child nodes (or children) – elements that are direct children. In other words, they are nested exactly in the given one.

Descendants – all elements that are nested in the given one, including children, their children and so on.

EVENTS LISTENER

Mouse events:

- click – when the mouse clicks on an element (touchscreen devices generate it on a tap).
- contextmenu – when the mouse right-clicks on an element.
- mouseover / mouseout – when the mouse cursor comes over / leaves an element.
- mousedown / mouseup – when the mouse button is pressed / released over an element.
- mousemove – when the mouse is moved.

Keyboard events:

- keydown and keyup – when a keyboard key is pressed and released.

Form element events:

- submit – when the visitor submits a <form>.
- focus – when the visitor focuses on an element, e.g. on an <input>.