Class declarations

One way to define a class is using a **class declaration**. To declare a class, you use the `class` keyword with the name of the class ("Rectangle" here).

```
class Rectangle {
  constructor(height, width) {
this.height = height;
this.width = width;
  }
}
```

Static methods

The `static` keyword defines a static method for a class. Static methods are called without instantiating their class and **cannot** be called through a class instance. Static methods are often used to create utility functions for an application.

```
class Point {
constructor(x, y) {
this.x = x;
this.y = y;
  }    static
distance(a, b) {
const dx = a.x - b.x;
const dy = a.y - b.y;

    return Math.hypot(dx, dy);
  }
}
 const p1 = new Point(5, 5);
const p2 = new Point(10,
10); p1.distance;
//undefined p2.distance;
//undefined

console.log(Point.distance(p1, p2)); // 7.0710678118654755
```

Private field declarations

Using private fields, the definition can be refined as below.

```
class Rectangle {
  #height = 0;
#width;
  constructor(height, width) {
this.#height = height;
this.#width = width;
  }
}
```

Class expressions

A **class expression** is another way to define a class. Class expressions can be named or unnamed. The name given to a named class expression is local to the class's body. (it can be retrieved through the class's (not an instance's) `name` property, though).

```
// unnamed
let Rectangle = class {
constructor(height, width) {
this.height = height;
this.width = width;
  }
};
console.log(Rectangle.name);
// output: "Rectangle"

// named
let Rectangle = class Rectangle2 {
constructor(height, width) {
this.height = height;
this.width = width;
  }
};
console.log(Rectangle.name);
// output: "Rectangle2"
```

```
const five = 5;
const ten = 10;
console.log(`Fifteen is ${five + ten} and not ${2 * five + ten}.`);
// "Fifteen is 15 and not 20."
```

```
var list = [1, 2, 3, 4];

function empty() {

//empty your array

list = [];

}
empty();
```

```
const  arrMax = arr => Math.max(...arr);
                                            // arrMax([20, 10, 5, 10]) -> 20

const  arrMin = arr => Math.min(...arr);
                                            // arrMin([20, 10, 5, 10]) -> 5

const  arrSum = arr => arr.reduce((a,b) => a + b, 0)
                                            // arrSum([20, 10, 5, 10]) -> 45
const arrMax = arr => Math.max(...arr);
```

// IS THE SAME AS

```
arrMax = function(arr){
return Math.max(...arr);
}
```

```
if(document.getElementById('button').clicked == true)
{
  alert("button was clicked");
}
```

```
<input id="button" type="submit" name="button" onclick="myFunction();" value="enter"/>

<script> function
myFunction(){
   alert("You button was pressed");
};
</script>
```

```
var paragraph = document.getElementById("p"); var text =
document.createTextNode("This just got added");
paragraph.appendChild(text);  <p id="p">This is some
text</p>

var paragraph = document.getElementById("p");
paragraph.textContent += "This just got added";
<p id="p">This is some text</p>
```

```
function addElement () {
// create a new div element
  var newDiv = document.createElement("div");
  // and give it some content
  var newContent = document.createTextNode("Hi there and
greetings!");
  // add the text node to the newly created div
newDiv.appendChild(newContent);

  // add the newly created element and its content into the
DOM    var currentDiv = document.getElementById("div1");
document.body.insertBefore(newDiv, currentDiv);  }
```

There are **4 ways** to create a new date object:

```
new Date()
new Date(year, month, day, hours, minutes, seconds, milliseconds)
new Date(milliseconds)
new Date(date string)
```

`new Date()` creates a new date object with the **current date and time**:

new Date(*milliseconds*)

`new Date(milliseconds)` creates a new date object as **zero time plus milliseconds**:

Example

```
var d = new Date(0);
```

Try it Yourself »

01 January 1970 **plus** 100 000 000 000 milliseconds is approximately 03 March 1973:

Example

```
var d = new Date(100000000000);
```

```
d = new Date();
document.getElementById("demo").innerHTML = d.toString();
```

```
var d = new Date();
document.getElementById("demo").innerHTML = d.toDateString();
```

Set Date Methods

Set Date methods are used for setting a part of a date:

Method	**Description**
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

Method	Description
getFullYear()	Get the year as a four digit number (yyyy)
getMonth()	Get the month as a number (0-11)
getDate()	Get the day as a number (1-31)
getHours()	Get the hour (0-23)
getMinutes()	Get the minute (0-59)
getSeconds()	Get the second (0-59)
getMilliseconds()	Get the millisecond (0-999)
getTime()	Get the time (milliseconds since January 1, 1970)
getDay()	Get the weekday as a number (0-6)
Date.now()	Get the time. ECMAScript 5.

```
var d = new Date("2015-03-25");
```

If you have a valid date string, you can use the `Date.parse()` method to convert it to milliseconds.

`Date.parse()` returns the number of milliseconds between the date and January 1, 1970:

```
var msec = Date.parse("March 21, 2012");
document.getElementById("demo").innerHTML = msec;
```

You can then use the number of milliseconds to **convert it to a date** object:

```
var msec = Date.parse("March 21, 2012");
var d = new Date(msec);
document.getElementById("demo").innerHTML = d;
```

```
Math.PI;            // returns 3.141592653589793
Math.round(4.7);    // returns 5
Math.pow(8, 2);      // returns 64
Math.sqrt(64);      // returns 8
Math.abs(-4.7);     // returns 4.7 (absolute positieve waarde)
Math.ceil(4.4);     // returns 5 (naar boven afronden)
Math.floor(4.7);    // returns 4 (naar onder afronden)
Math.sin(90 * Math.PI / 180);     // returns 1 (the sine of 90
degrees)
Math.cos(0 * Math.PI / 180);     // returns 1 (the cos of 0 degrees)
Math.min(0, 150, 30, 20, -8, -200);  // returns -200
Math.max(0, 150, 30, 20, -8, -200);  // returns 150
Math.random();     // returns a random number
Math.floor(Math.random() * 100);     // returns a random integer from
0 to 99
Math.floor(Math.random() * 10);     // returns a random integer from 0
to 9

function getRndInteger(min, max) {
  return Math.floor(Math.random() * (max - min) ) + min;
}
```