4. JavaScript Objects

Object:

In JavaScript, almost "everything" is an object.

- Booleans can be objects (if defined with the new keyword)
- Numbers can be objects (if defined with the new keyword)
- Strings can be objects (if defined with the new keyword)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects

Objects are variables too. But objects can contain many values.
Object values are written as name:value pairs (name and value separated by a colon).
Example:

```
let person ={firstName:"John", lastName:"Doe", age:50, eColor:"blue"};
```

A JavaScript object is a collection of named values.

Object Properties:

The named values, in JavaScript objects, are called properties.

Property	Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

Object Methods:

An object method is an object property containing a function definition.

```
const person = {
firstName: "John",
lastName: "Doe",
id: 5566,
fullName: function( )
 {
return this.firstName + " " + this.lastName;
  }
};
```

Example:

```
const person =

{
  name: "John",
  age, : 30
  city: "New York"
};

let txt = "";

for (let x in person) {
txt += person[x] + " ";
};

document.getElementById("demo").innerHTML = txt;
```

Creating a JavaScript Object

With JavaScript, you can define and create your own objects.

There are different ways to create new objects:

1. Create a single object, using an object literal.
2. Create a single object, with the keyword new.
3. Define an object constructor, and then create objects of the constructed type.
4. Create an object using Object.create().

1. Using an Object Literal

- This is the easiest way to create a JavaScript Object.
- Using an object literal, you both define and create an object in one statement.
- An object literal is a list of name:value pairs (like age:50) inside curly braces {}.

The following example creates a new JavaScript object with four properties:

Example:

const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

Spaces and line breaks are not important. An object definition can span multiple lines:

Example:

```
const person = {
firstName: "John",
lastName: "Doe",
```

age: 50,
eyeColor: "blue"
};

This example creates an empty JavaScript object, and then adds 4 properties:
Example:

```
const person = {};
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

2. Using the JavaScript Keyword new

The following example create a new JavaScript object using new Object(), and then adds 4 properties:

Example:

```
const person = new Object();
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

3. Using Object Constructor:

The way to create an "object type", is to use an object constructor function.
In the example below, function Person() is an object constructor function.
Objects of the same type are created by calling the constructor function with the new keyword:

```
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eye;
this.nationality=”Indian”;
}
const myFather = new Person("John", "Doe", 50, "blue");
const myMother = new Person("Sally", "Rally", 48, "green");
```

Adding a Property to an Object

Adding a new property to an existing object is easy:

Example

```
myFather.nationality = "English";
```

The property will be added to myFather. Not to myMother. (Not to any other person objects).

Adding a Method to an Object

Adding a new method to an existing object is easy:

Example

```
myFather.name = function () {
  return this.firstName + " " + this.lastName;
};
```

The method will be added to myFather. Not to myMother. (Not to any other person objects).

Adding a Property to a Constructor

You cannot add a new property to an object constructor the same way you add a new property to an existing object:

Example

```
Person.nationality = "English";
```

To add a new property to a constructor, you must add it to the constructor function:

Example

```
function Person(first, last, age, eyecolor) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eyecolor;
  this.nationality = "English";
}
```

Date Object

We can create date object simply with the use of Date().

```
const d = new Date();
```

By default, JavaScript will use the browser's time zone and display a date as a full text string:

Fri Jul 02 2021 10:44:45 GMT+0530 (India Standard Time)

Creating Object:

Date objects are created with the new Date() constructor.

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

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)

Example

const d = new Date();

Date objects are static. The computer time is ticking, but date objects are not.

new Date(year, month, ...)

creates a new date object with a specified date and time.

7 numbers specify year, month, day, hour, minute, second, and millisecond (in that order):

Example

const d = new Date(2018, 11, 24, 10, 33, 30, 0);

Note: JavaScript counts months from 0 to 11.

January is 0. December is 11.

6 numbers specify year, month, day, hour, minute, second:

Example

const d = new Date(2018, 11, 24, 10, 33, 30);

5 numbers specify year, month, day, hour, and minute:

Example

const d = new Date(2018, 11, 24, 10, 33);

4 numbers specify year, month, day, and hour:

Example

const d = new Date(2018, 11, 24, 10);

3 numbers specify year, month, and day:

Example

const d = new Date(2018, 11, 24);

2 numbers specify year and month:

Example

const d = new Date(2018, 11);

You cannot omit month. If you pass only one parameter it will be treated as milliseconds.

Example

const d = new Date(2018);

new Date(dateString)

new Date(dateString) creates a new date object from a date string:

Example

const d = new Date("October 13, 2014 11:13:00");

Date Methods

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.

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)

Example:

```
const d = new Date();
d.setFullYear(2020);
```

The setFullYear() method can optionally set month and day:

Example:

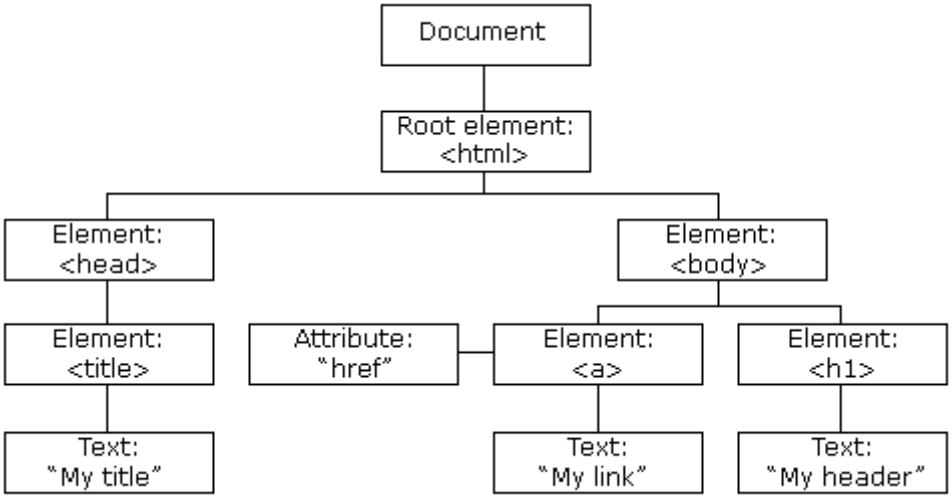
```
const d = new Date();
d.setFullYear(2020, 11, 3);
```

Document Object Model (DOM)

When a web page is loaded, the browser creates a Document Object Model of the page.

The HTML DOM model is constructed as a tree of Objects:

The HTML DOM Tree of Objects



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

"DOM is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

DOM Methods and Properties

In the DOM, all HTML elements are defined as objects.
The programming interface is the properties and methods of each object.
A property is a value that you can get or set (like changing the content of an HTML element).
A method is an action you can do (like add or deleting an HTML element).

DOM Methods for finding HTML Elements

Method	Description
document.getElementById(id)	Find an element by element id
document.getElementsByTagName(name)	Find elements by tag name
document.getElementsByClassName(name)	Find elements by class name

DOM Methods for Adding and Deleting HTML Elements

Method	Description
document.createElement(element)	Create an HTML element
document.removeChild(element)	Remove an HTML element
document.appendChild(element)	Add an HTML element
document.replaceChild(new, old)	Replace an HTML element
document.write(text)	Write into the HTML output stream

DOM Elements and Method for Changing HTML Elements

Property	Description
element.innerHTML= new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.property = new style	Change the style of an HTML element
Method	**Description**
element.setAttribute(attribute, value)	Change the attribute value of an HTML element

Example 1:

```
<html>
<head>
        <script>
                document.getElementById("demo").innerHTML = "Hello World!";
        </script>
</head>
<body>
        <p id="demo"></p>
```

```
</body>
</html>
```

Example 2:

```
const x = document.forms["frm1"];
let text = "";
for (let i = 0; i<x.length; i++)
 {
text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

This above example finds the form element with id="frm1", in the forms collection, and displays all element values.