

# Software engineering

## JavaScript Tutorial 2021.

reference: w3shools & tutorialspoint

### 01. What is JavaScript ?

**JavaScript** is a high-level, dynamic, untyped, and interpreted programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production; the majority of websites employ it, and all modern Web browsers support it without the need for plug-ins.

### 02. Syntax

JavaScript can be implemented using JavaScript statements that are placed within the **<script>...</script>** HTML tags in a web page. You can place the **<script>** tags, containing your JavaScript, anywhere within you web page, but it is normally recommended that you should keep it within the **<head>** tags. The **<script>** tag alerts the browser program to start interpreting all the text between these tags as a script.

```
<html>
  <body>
    <script language="javascript" type="text/javascript">

      document.write("Hello World!")

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

### 03. Enabling

All the modern browsers come with built-in support for JavaScript. Frequently, you may need to enable or disable this support manually. If you have to do something important using JavaScript,

then you can display a warning message to the user using `<noscript>` tags.

#### Example: noscript

```
<html>
  <body>
    <script language="javascript" type="text/javascript">

      document.write("Hello World!")

    </script>
    <noscript>
      Sorry...JavaScript is needed to go ahead.
    </noscript>
  </body>
</html>
```

## 04. Variable

Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container.

#### Example: var

```
<script type="text/javascript">

  var name = "Andy";
  var money;
  money = 2000.50;

</script>
```

## 05. Placement

You can include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTML file are as follows –

- Script in `<head>...</head>` section.

- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.

**Practice #1: Do the following and capture the result.**

```

<html>
  <head>
    <script type="text/javascript">

      function sayHello() {
        alert("Hello XXX") // put your name into XXX
      }

    </script>
  </head>

  <body>
    <script type="text/javascript">

      document.write("Hello XXX") // put your name into XXX

    </script>

    <input type="button" onclick="sayHello()" value="Say Hello" />

  </body>
</html>

```

## 06. Operator

A is 10, B is 20.

| No. | Operator and Description                        |
|-----|-------------------------------------------------|
| 1.  | <b>+</b> <b>(Addition)</b><br>Adds two operands |

|    |                                                                                                   |
|----|---------------------------------------------------------------------------------------------------|
|    | EX) A + B will give 30                                                                            |
| 2. | <b>- (Subtraction)</b><br>Subtracts the second operand from the first<br>EX) A – B will give – 10 |
| 3. | <b>* (Multiplication)</b><br>Multiply both operands<br>Ex) A * B will give 200                    |
| 4. | <b>/ (Division)</b><br>Divide the numerator by the denominator<br>Ex: B / A will give 2           |
| 5. | <b>% (Modulus)</b><br>Outputs the remainder of an integer division<br>Ex: B % A will give 0       |
| 6. | <b>++ (Increment)</b><br>Increases an integer value by one<br>Ex: A++ will give 11                |
| 7. | <b>-- (Decrement)</b><br>Decreases an integer value by one<br>Ex: A-- will give 9                 |

## 07. if-else

JavaScript supports conditional statements which are used to perform different actions based on different conditions.

**Example: if else if else**

```
<html>
  <body>

    <script type="text/javascript">

      var book = "maths";
      if( book == "history" ){
        document.write("<b>History Book</b>");
      }

      else if( book == "maths" ){
        document.write("<b>Maths Book</b>");
      }

      else if( book == "economics" ){
        document.write("<b>Economics Book</b>");
      }

      else{
        document.write("<b>XXX does not know it</b>");
      }

    </script>

    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

## 08. While Loop

The purpose of a while loop is to execute a statement or code block repeatedly as long as an expression is true. Once the expression becomes false, the loop terminates.

### Example: do while

```
<html>
  <body>

    <script type="text/javascript">

      var count = 0;

      document.write("Starting Loop" + "<br />");
      do{
        document.write("Current Count : " + count + "<br />");
        count++;
      }

      while (count < 5);
      document.write ("Loop stopped!");

    </script>

    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

## 09. For...in

The for...in loop is used to loop through an object's properties.

**Example: for...in**

```
<html>
  <body>

    <script type="text/javascript">
      <!--
        var aProperty;
        document.write ("XXX"); // XXX is your name
        document.write("<br />");
        document.write("Navigator Object Properties<br /> ");
        for (aProperty in navigator) {
          document.write(aProperty);
          document.write("<br />");
        }
        document.write ("Exiting from the loop!");
      //-->
    </script>
    <p>Set the variable to different object and then try...</p>
  </body>
</html>
```

Practice #2) Run the above code and capture it. Place your name into XXX.

## 10. Function

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. Functions allow a programmer to divide a big program into a number of small and manageable functions.

```
<html>
  <head>
    <script type="text/javascript">
      function concatenate(first, last)
      {
        var full;
        full = first + last;
        return full;
      }

      function secondFunction()
      {
        var result;
        result = concatenate('XXX', ' YYY');
        document.write (result );
      }
    </script>

  </head>

  <body>
    <p>Click the following button to call the function</p>

    <form>
      <input type="button" onclick="secondFunction()" value="Call
Function">
    </form>

    <p>Use different parameters inside the function and then try...</p>
  </body>
</html>
```

Practice #3) Run the above code and capture it. Place your first name into XXX and family name into YYY.



## 11. Event

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc. Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

### Example: onClick Event

```
<html>
  <head>

    <script type="text/javascript">
      <!--
        function sayHello() {
          alert("Hello World")
        }
      //-->
    </script>

  </head>

  <body>
    <p>Click the following button and see result</p>

    <form>
      <input type="button" onclick="sayHello()" value="Say Hello" />
    </form>

  </body>
</html>
```

### Example: onmouseover Event

The onmouseover event triggers when you bring your mouse over any element and the onmouseout triggers when you move your mouse out from that element.

```
<html>
  <head>

    <script type="text/javascript">
      <!--
        function over() {
          document.write ("Mouse Over");
        }

        function out() {
          document.write ("Mouse Out");
        }

      //-->
    </script>

  </head>
  <body>
    <p>Bring your mouse inside the division to see the result:</p>

    <div onmouseover="over()" onmouseout="out()">
      <h2> This is inside the division </h2>
    </div>

  </body>
</html>
```

Practice #4) Run the above code and capture it.

## 12. Page Redirection

You might have encountered a situation where you clicked a URL to reach a page X but internally you were directed to another page Y. It happens due to page redirection.

### Example: Page Redirection

You can show an appropriate message to your site visitors before redirecting them to a new page. This would need a bit time delay to load a new page.

Practice #5) Run the code below and capture it.

```
<html>
  <head>

    <script type="text/javascript">

      function Redirect() {
        window.location="http://www.dongguk.edu";
      }

    </script>

  </head>

  <body>
    <p>Click the following button, you will be redirected to home
page.</p>

    <form>
      <input type="button" value="Redirect Me" onclick="Redirect();" />
    </form>

  </body>
</html>
```

## 13. Object

JavaScript is an Object Oriented Programming (OOP) language. Objects are composed of attributes. If an attribute contains a function, it is considered to be a method of the object, otherwise the attribute is considered a property.

- **Object Properties**

Object properties are usually variables that are used internally in the object's methods.

*"title" property of the document object*

```
objectName.objectProperty = propertyValue;
```

```
var str = document.title;
```

- **Object Methods**

Methods are the functions that let the object do something or let something be done to it.

*Write() method*

```
document.write("This is test");
```

- **The 'with' Keyword**

The 'with' keyword is used as a kind of shorthand for referencing an object's properties or methods.

*The Syntax for with object*

```
with (object){  
    properties used without the object name and dot  
}
```

**Example: Object**

```

<html>
  <head>
    <title>User-defined objects</title>

    <script type="text/javascript">
      // Define a function which will work as a method
      function addPrice(amount){
        this.price = amount;
      }

      function book(title, author){
        this.title = title;
        this.author = author;
        this.addPrice = addPrice; // Assign that method as property.
      }
    </script>

  </head>
  <body>

    <script type="text/javascript">
      var myBook = new book("Harry Potter", "Joan K. Rowling");
      myBook.addPrice(100);

      document.write("Book title is : " + myBook.title + "<br>");
      document.write("Book author is : " + myBook.author + "<br>");
      document.write("Book price is : " + myBook.price + "<br>");
    </script>

  </body>
</html>

```

Practice #6) Run the code above and capture it. For book name and author, change it into your favorite book name and its author.

## 14. Number Object

The Number object represents numerical data, either integers or floating-point numbers. In general, the browser automatically converts number literals to instances of the number class.

#### *Number object*

```
var val = new Number(number);
```

### Number Properties & Method

| Property  | Description                                                                        |
|-----------|------------------------------------------------------------------------------------|
| MAX_VALUE | The largest possible value a number in JavaScript can have 1.7976931348623157E+308 |
| MIN_VALUE | The smallest possible value a number in JavaScript can have 5E-324                 |
| NaN       | Equal to a value that is not a number.                                             |

| Method          | Description                                                                                                                                 |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| toExponential() | Forces a number to display in exponential notation, even if the number is in the range in which JavaScript normally uses standard notation. |
| toFixed()       | Formats a number with a specific number of digits to the right of the decimal.                                                              |
| toString()      | Returns the string representation of the number's value.                                                                                    |
| valueOf()       | Returns the number's value.                                                                                                                 |

### Example: Object

```
<html>
<body>

<p>The toString() method can output numbers as base 16 (hex), base 8
(octal), or base 2 (binary).</p>

<p id="demo"></p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
    var myNumber = 128;
    document.getElementById("demo").innerHTML = "128 = " +
    myNumber + " Decimal, " +
    myNumber.toString(16) + " Hexadecimal, " +
    myNumber.toString(8) + " Octal, " +
    myNumber.toString(2) + " Binary."
}
</script>

</body>
</html>
```

Practice #7) Run the code above and capture it. For book name and author, change it into your favorite book name and its author.

## 15. Array

The Array object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

*Array object*

```
var fruits = new Array( "apple", "orange", "mango" );
```

Array Elements Can Be Objects. JavaScript variables can be objects. Arrays are special kinds of objects. Because of this, you can have variables of different types in the same

Array. You can have objects in an Array. You can have functions in an Array. You can have arrays in an Array:

## Array Method

| Method   | Description                                                                                                        |
|----------|--------------------------------------------------------------------------------------------------------------------|
| concat() | Returns a new array comprised of this array joined with other array(s) and/or value(s).                            |
| filter() | Creates a new array with all of the elements of this array for which the provided filtering function returns true. |
| slice()  | Extracts a section of an array and returns a new array.                                                            |
| sort()   | Sorts the elements of an array                                                                                     |
| pop()    | Removes the last element from an array and returns that element.                                                   |
| push()   | Adds one or more elements to the end of an array and returns the new length of the array.                          |

## Example: Array

```
<html>
<body>

<p id="demo"></p>

<script>
var family = new Array("Kim", "Lee", "Park");
document.getElementById("demo").innerHTML = family;
document.write("slice method: "+family.slice(0,2));
</script>

</body>
</html>
```

Practice #8) Run the code above and capture it. You have to create your own array elements with size of 5.



## 16. Date

The Date object is a datatype built into the JavaScript language. Date objects are created with the new Date( ) as shown below.

Once a Date object is created, a number of methods allow you to operate on it. Most methods simply allow you to get and set the year, month, day, hour, minute, second, and millisecond fields of the object, using either local time or UTC (universal, or GMT) time.

### *Date object*

```
new Date( )  
new Date(milliseconds)  
new Date(datestring)  
new Date(year,month,date[,hour,minute,second,millisecond ])
```

### Date Method

| Method        | Description                                                                                                         |
|---------------|---------------------------------------------------------------------------------------------------------------------|
| Date()        | Returns today's date and time                                                                                       |
| getDate()     | Returns the day of the month for the specified date according to local time.                                        |
| getFullYear() | Returns the year of the specified date according to local time.                                                     |
| setDate()     | Sets the day of the month for a specified date according to local time.                                             |
| setTime()     | Sets the Date object to the time represented by a number of milliseconds since January 1, 1970, 00:00:00 UTC.       |
| toString()    | Returns a string representing the specified Date object.                                                            |
| valueOf()     | Returns the primitive value of a Date object.                                                                       |
| Date.parse()  | Parses a string representation of a date and time and returns the internal millisecond representation of that date. |
| Date.UTC()    | Returns the millisecond representation of the specified UTC date and time.                                          |

### Example: getYear

```
<html>
<body>

<p>The getFullYear() method returns the full year of a date:</p>

<p id="demo"></p>

<script>
var d = new Date();
document.getElementById("demo").innerHTML = d.getFullYear();
document.write(d.toString());
</script>

</body>
</html>
```

Practice #9) Run the code above and capture it.

## 17. Error Handling

JavaScript implements the **try...catch...finally** construct as well as the throw operator to handle exceptions. You can catch programmer-generated and runtime exceptions, but you cannot catch JavaScript syntax errors.

The **try** statement lets you test a block of code for errors.

The **catch** statement lets you handle the error.

The **throw** statement lets you create custom errors.

The **finally** statement lets you execute code, after try and catch, regardless of the result.

### Example: try...catch.finally

```
<html>
<body>

<p>Please input a number between 5 and 10:</p>

<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>

<p id="message"> </p>

<script>
function myFunction() {
    var message, x;
    message = document.getElementById("message");
    message.innerHTML = "";
    x = document.getElementById("demo").value;
    try {
        if(x == "") throw "is empty";
        if(isNaN(x)) throw "is not a number";
        x = Number(x);
        if(x > 10) throw "is too high";
        if(x < 5) throw "is too low";
    }
    catch(err) {
        message.innerHTML = "Input " + err;
    }
    finally {
        document.getElementById("demo").value = "";
    }
}
</script>

</body>
</html>
```

Practice #10) Run the code above and capture it

### Example: onerror() method

```
<html>

<head>

<script type="text/javascript">

    window.onerror = function (msg, url, line) {
        alert("Message : " + msg );
        alert("url : " + url );
        alert("Line number : " + line );
    }

</script>

</head>

<body>
    <p>Click the following to see the result:</p>

    <form>
        <input type="button" value="Click Me" onclick="myFunc();" />
    </form>

</body>
</html>
```

## 18. Form Validation

Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button. If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information. This was really a lengthy process which used to put a lot of burden on the server.

JavaScript provides a way to validate form's data on the client's computer before sending it to the

web server. Form validation generally performs two functions.

Basic Validation : First of all, the form must be checked to make sure all the mandatory fields are filled in. It would require just a loop through each field in the form and check for data.

Data Format Validation : Secondly, the data that is entered must be checked for correct form and value. Your code must include appropriate logic to test correctness of data.

### Example: Form validation

```
<!DOCTYPE html>
<head><title>Understanding Form</title></head>
<html>
<body>

<form action="a1.asp" name="form1" method="get">
  <p>input your name</p>
  <input type="text" name="t1" value="name">
  <input type="button" value="result", onClick="p(this.form)">
</form>
<script language="JavaScript">
  function p(obj){
    document.write("Form name: "+ obj.t1.value);
  }
</script>
</body>
</html>
```

## 19. Animation

You can use JavaScript to create a complex animation having, but not limited to, the following elements –Fireworks, Fade Effect, Roll-in or Roll-out, Page-in or Page-out, Object movements

You might be interested in existing JavaScript based animation library: Script.Aculo.us.

This tutorial provides a basic understanding of how to use JavaScript to create an animation.

JavaScript can be used to move a number of DOM elements (<img />, <div> or any other HTML element) around the page according to some sort of pattern determined by a logical equation or function.

JavaScript provides the following two functions to be frequently used in animation programs.

1. `setTimeout(function, duration)`: This function calls function after duration milliseconds from now.
2. `setInterval(function, duration)` : This function calls function after every duration milliseconds.
3. `clearTimeout(setTimeout_variable)`: This function calls clears any timer set by the `setTimeout()` functions.

### **Example: Animation**

```
<html>

  <head>
    <title>JavaScript Animation</title>

    <script type="text/javascript">
      <!--
        var imgObj = null;
        var animate ;
```

```

function init(){
    imgObj = document.getElementById('myImage');
    imgObj.style.position= 'relative';
    imgObj.style.left = '0px';
}

function moveRight(){
    imgObj.style.left = parseInt(imgObj.style.left) + 10 + 'px';
    animate = setTimeout(moveRight,20); // call moveRight in
20msec
}

function stop(){
    clearTimeout(animate);
    imgObj.style.left = '0px';
}

window.onload =init;
//-->
</script>

</head>

<body>

<form>
    
    <p>Click the buttons below to handle animation</p>
    <input type="button" value="Start" onclick="moveRight();" />
    <input type="button" value="Stop" onclick="stop();" />
</form>

</body>
</html>

```

