Java Script

Prepared By Deepali Sonawane, DoCSA, MIT-WPU

Introduction

- JavaScript is most **popular** language on the internet and works in all **major browsers** such as IE, Firefox, Chrome, Opera and Safari.
- It was written for the express purpose of **adding interactivity** to Web pages.
- It is a **dynamic** computer programming language.
- It is lightweight and most commonly used as a part of web pages, whose implementations allow **client side** script to interact with the user and make dynamic pages.
- It is an interpreted programming language with **object oriented** capabilities.
- It is open and cross-platform.

Advantages of JavaScript

- **Less Server Interaction:** Being client-side scripting, it reduces the demand on the website server.
- **Immediate feedback to the visitors:** They do not have to wait for a page reload to see if they have forgotten to enter something.
- **Increased Interactivity:** We can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Speed :** Client-side JavaScript is very fast because it can be run immediately within the client-side browser.
- **Simplicity:** JavaScript is relatively simple to learn and implement.
- **Popularity:** JavaScript is used everywhere in the web. The resources to learn JavaScript are numerous.

Limitations of JavaScript

- Client side JavaScript does not allow the **reading or writing of files**. This has been kept for security reason.
- JavaScript can not be used for **networking applications** because there is not such support available.
- JavaScript does not have any **multithreading or multiprocess** capabilities.
- JavaScript does not have **database connectivity**.

JavaScript Comments

- **Single Line Comments** : It can be added in javascript using double forward slash //
- **Multiline Comments**: To start multiline comment in javascript, a forward slash and an asterisk(/*) is used. To end a multi line comment an asterisk and a forward slash(*/) is used.

Identifiers

- An identifier is **simply a name**.
- In JavaScript, identifier are used to name **variables, functions, methods, objects events** etc.
- Rules to define identifier:

1. Identifiers can only contain **letters, numbers, underscore(_) and dollar sign($).**
2. Identifier can **not start with a number**.
3. Identifier are **case sensitive**.
4. Identifier can be **any length**.
5. Identifiers can not be the same as JavaScript reserved words i.e. **keywords**.

Variables

- Variables are **containers** for storing information.
- A variable's value can **change** during the script.
- We can refer to a variable by name to see its value or to change its value.
- **Variable declaration:**

  Syntax: var variable_name;

  Example: var name;
- **Variable Initialization:**

  Syntax: var variable_name=variable_value;

  Example: var n=10;

Variables Example

```
<html>
<body>
<h1>JavaScript Variables</h1>
<p id="demo"></p>
<script>
var x = 5;
var y = 6;
var z = x + y;
document.getElementById("demo").innerHTML = "The value of z is: " + z;
</script>
</body>
</html>
```

Accessing form elements

1. document.getElementById(): This method retrieves an element by its ID, allowing you to target specific input fields, text areas, select boxes, etc.

Example: usernameInput = document.getElementById("username").value;

2. document.getElementsByName(): This method retrieves all elements with a specific name attribute.

Example: radioButtons = document.getElementsByName("gender")[0].value;

Getting the Value of a Text Input Field using id

```
<html><head><title>Using Value Property</title></head>
<body>
   <input type="text" id="myInput">
   <button onclick="getValue()">Get Value</button>
   <script>
        function getValue()
       {
             var a = document.getElementById("myInput").value;
             alert("Input value: " + a);
        }
    </script></body></html>
```

Getting the Value of a Text Input Field using name

```
<html><head><title>Using Value Property</title></head>
<body>
 <input type="text" name="myTextBox" id="myTextBox">
<button onclick="getValue()">Get Value</button>
<script>
function getValue() {
  var textBox = document.getElementsByName("myTextBox")[0];
  if (textBox) {
    var value = textBox.value;
    alert("The value of the text box is: " + value);
  } else {
    alert("Text box with name 'myTextBox' not found.");
  }
}
</script> </body></html>
```

Variable Scope

The scope of a variable is the region of the program in which it is defined.

JavaScript variables will have two scopes:

1. Global Variables: A global variable has global scope which means it is defined everywhere in JavaScript code and all script and functions on a web page can access it. We use **var** keyword to declare global variable. Ex. var a=10;

2. Local Variables: A local variable will be visible only within a function where it is defined. Local variables have local scope and they can only be accessed within function. We use **let** keyword to declare local variable. Ex. let a=20;

Operators

An operator indicates an operation to be performed on data that yield a value.

An operand is a data item on which an operation is performed.

Types of operators:

1. **Arithmetic Operators**: Arithmetic operators are used to perform arithmetic operations. +, -, *, /, % are arithmetic operators.

2. Relational Operators: Relational operators are used for comparison. <, <=, >, >=, ==, != are relational operators.

3. Logical Operators: Logical operators are used for combining two conditions. &&, ||, ! are logical operators.

4. Increment and Decrement Operators: Increment operator (++) increase the value of variable by 1 and decrement operator (--) decrease the value of variable by 1.

5. Assignment Operator: Assignment operator (=) assign value to variable. +=, -=, *=, /=, %= are short-hand assignment operators.

6. Conditional Operators: ? : is a conditional operator. It is ternary operator also.

Synatx: (condition)? true part : false part;

7. Bitwise operators: Bitwise operators act upon the individual bits of their operands. &, |, ^, ~, << ,>> are bitwise operators.

8. Concatenation operator: Concatenation operator (+) concatenates two strings.

Selection / Decision Making Statements

1. If statement: The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

```
Syntax: if(condition)
            {
                        statements;
            }
```

2. If else statement: If else statement is use to execute some code if a condition is true and another code if the condition is false.

```
Syntax: if(condition)
            {
                        statement1;
            }
            else
            {
                        statement2;
            }
```

Selection / Decision Making Statements

3. Nested if statement: If condition within if is a nested if statement.

Syntax:
```
if(condition1)
        {
                if(condition2)
                statement1;
                else
                statement2;
        }
        else
        {
                statement3;
        }
```

Selection / Decision Making Statements

4. Else if ladder statement: else if ladder is the one level advance form of control statement that allows JavaScript to make correct decision out of several conditions.

Syntax: if(condition1)

```
        statement1;
        else if(condition2)
        statement2;
        else if(condition3)
        statement3;
        .
        .
        .
        else
        statement n;
```

Selection / Decision Making Statements

5. Switch statement: Switch statement is used to execute one of the statement from many blocks of statements.

Syntax:
```
switch(expression)
{
        case exp1 : statement1;
                    break;
        case exp2 : statement2;
                    break;
        .
        .
        default: statement;
}
```

Loop / Iterative Statements

1. while loop: The purpose of a while loop is to execute a statement or code block repeatedly as long as expression is true. Once, expression becomes false, the loop will be exited.

```
Syntax: initialization;
        while(condition)
        {
                statements;
                update statement;
        }
```

2. do_while loop: Sometimes, we want some statements to be executed at least once even if the condition is false for the first time. To do this we use a do_while loop.

```
Syntax: initialization;
        do
        {
                update statement;
        }while(condition);
```

Loop / Iterative Statements

3. for loop: The for loop is the most compact form of looping and includes initialization, condition, update statement.

Syntax: for(initialization ; condition ; update statement)

```
        {
                statement;
        }
```

Jump Statements

1. break statement: The break statement breaks the loop and continues executing the code after the loop if any.

Syntax: break;

2. Continue statement: The continue statement continues the loop.

Syntax: continue;

Example of Pattern

```
<html><head><title>Pattern Example</title></head>
<body>
<p id="abc"></p>
<script>
var n = parseInt(prompt("How many rows: "));
var s=‘ ‘,i,j;
for(i=1 ; i<=n; i++)
{
        for(j=1; j<=i; j++)
        {
                s=s+i;
        }
        s+="<br>";
}
document.getElementById("abc").innerHTML=s;
</script></body></html>
```

Array

Definition: Array is a collection of data items of same data type.

There are two types of array:

1. 1-Dimensional array 2. 2-Dimensional array

1. 1-Dimensional array

Syntax: var variable_name = new Array(size);

Example: var a = new Array(10);

2. 2-Dimensional array

```
Syntax: var variable_name = new Array(size1);
            for(var i=0;i<size1;i++)
            variable_name[i] = new Array(size2);
Example: var a = new Array(4);
              for(var i=0;i<4;i++)
              a[i] = new Array(3);
```

Functions

Functions are one of the fundamental building blocks in JavaScript. In JavaScript predefined functions are also called as built in functions.

String Functions:

1. charAt(index) : Returns character at the specified index.
2. charCodeAt(index) : Returns ASCII value of the character at the given index.
3. fromCharCode(Value) : Returns character for specified value.
4. concat() : Combines two strings and returns a new string.
5. indexOf(String): Returns the index of first occurrence of specified value or -1 if not found.
6. lastIndexOf(String) : Returns the index of last occurrence of specified value or -1 if not found.
7. replace(string1,string2): replace string1 to string2 and returns new string.

String Functions

8. search(String) : Search for a match between two strings and returns position.
9. toLowerCase() : Returns string in lowercase.
10. toUpperCase(): Returns string in uppercase.
11. substr(start,length): returns string beginning at the specified location through the specified number of characters.
12. substring(start index, end index): Returns string between two indexes.
13. length : To find length of a string.

Math Functions

1. abs(value) : Returns the absolute value of a specified number.
2. sin(value) : Returns the sine of number.
3. cos(value) : Returns the cosine of a number.
4. tan(value) : Returns the tangent of a number.
5. max(n1,n2,…) : Returns maximum number from specified list of numbers.
6. min(n1,n2,…) : Returns minimum number from specified list of numbers.
7. pow(x,y) : Returns base to exponent power.
8. random() : Returns random number between 0 to 1.
9. sqrt(value) : Returns square root of a number.
10. round(value) : Returns number rounded to nearest integer.
11. ceil(value) : Returns integer greater than or equal to a specified number.
12. floor(value) : Returns integer less than or equal to a specified number.

Date Functions

1. Date objects are created with **new Date().**

   Ex. const d = new Date();

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

Method	Description
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (yyyy)
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)

User Defined Functions

User defined functions are defined by user.

Syntax for function definition:

```
function function_name([parameter_list])
{
        block of statements;
}
```

Syntax for function call:

```
Function_name([parameter_list]);
```

Types of parameters

1. **Actual Parameters** : Parameters used in function call are called as actual parameter.
2. **Formal or dummy parameters** : Parameters used in function definition are called as formal or dummy parameters.

Recursive function: It is a function which call itself.

Recursion: It is a process in which function call itself.

- Do not use any iterative / Loop statements in recursive function

Example :

```
function display()
{
    display();
}
```

Event Handling in JavaScript

Common events used in JavaScript are as follows:

1. onclick event
2. onsubmit event
3. onmouseover event
4. onmouseout event
5. onmousedown event
6. onmouseup event
7. onload event
8. onunload event
9. onchange event

Form Validation

```
<html><head><script>
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
</script></head>
<body>
<form name="myForm" action="#" onsubmit="return validateForm()" method="post">
  Name: <input type="text" name="fname">
  <input type="submit" value="Submit">
</form></body></html>
```

```
<html>
<body>

<h2>JavaScript Validation</h2>

<form action="/action_page.php" method="post">
  <input type="text" name="fname" required>
  <input type="submit" value="Submit">
</form>

<p>If you click submit, without filling out the text field,
your browser will display an error message.</p>

</body>
</html>
```

Thank you