Java Script

Introducing JavaScript

- JavaScript (JS) is a lightweight interpreted (or just-in-time compiled) programming language with first-class functions
- A programming language is said to have **First-class functions** when functions in that language are treated like any other variable. For example, in such a language, a function can be passed as an argument to other functions, can be returned by another function and can be assigned as a value to a variable
- JavaScript is a programming language that's used to turn web pages into applications
- Many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat

- JavaScript is used to manipulate the contents of a web page and to allow users to interact with web pages without reloading the page
- It’s deeply integrated with the browser
  - This integration allows programmers to manipulate various aspects of the browser behavior, as well as objects included on the page
- JavaScript uses what’s referred to as an event-driven model of execution
- When you embed JavaScript code in a web page, it isn’t run until the event it’s associated with is triggered
- JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles

Types of events

- Events that can call JavaScript include
  - loading the page, leaving the page,
  - interacting with a form element in some way,
    - clicking a link, or even just scrolling up or down
  - Often these events are utilized in what most users would consider to be annoying ways
- many sites open an additional window containing an advertisement when you navigate to one of their pages
  - This is accomplished using JavaScript and the page load event
- Other sites open additional windows when you leave them; this is also accomplished using JavaScript triggered by the page unload event

How it works

- JavaScript enables you to manipulate web pages without sending a request back to the server or to send a request to the server to retrieve information without leaving the page that the user is on
- Using these capabilities,
  - you can change the contents of a page,
  - change the style of elements on a page,
  - validate user input before a user submits a form,
  - and modify the behavior of the browser—
- all by using scripts embedded within your web pages

Setting Up

- You can run JavaScript in a web browser by creating an HTML file and embedding your JavaScript code within <script> tags
- Alternatively, you can use an integrated development environment (IDE) or a text editor and execute JavaScript using a runtime environment like Node.js
- The <script> tag is used to include a JavaScript script in a web page, in much the same way that the <style> tag is used to add a style sheet to a page
- For the best results across all browsers, you should include the type attribute in the script tag, which specifies the type of content that the tag contains. For JavaScript, use text/javascript

The Structure of a JavaScript

- When you include any JavaScript code in an HTML document (apart from using the <script> tag), you should also follow a few other conventions
- for performance reasons, it's almost always a better idea to put your <script> tags at the bottom of the page
- Unlike HTML, which uses the <!-- comment tag -->, comments inside JavaScript code use the // symbol at the start of a line

```
<html>
<head>
<title>Test script</title>
</head>
<body>
Your Web content goes here
<script>
// Your JavaScript code goes here
</script>
</body>
</html >
```

- Besides the language attribute, the <script> tag can also include an src attribute, which allows a JavaScript script stored in a separate file to be included as part of the current web page.

Basic JavaScript Concepts

- **Variables:** Variables are used to store and manage data in JavaScript. To declare a variable, you can use the **var**, **let**, or **const** keywords
- **var:** Historically used to declare variables, but it has some scoping issues. It's not recommended for modern JavaScript
- **let:** Introduced in ES6 (ECMAScript 2015), let allows you to declare block-scoped variables. It's a good choice for most use cases
- **const:** Also introduced in ES6, const declares block-scoped constants. You should use const when the value of the variable won't change.

Data Types

- JavaScript has several primitive data types, and it's a dynamically typed language, meaning you don't need to specify a data type explicitly; JavaScript infers it for you. Common data types include

Number

```
let count = 10;
let price = 19.99;
```

String

```
let name = "Alice";
let message = 'Hello, World!';
```

Boolean

```
let isOnline = true;
let isWorking = false;
```

Undefined

```
let notDefined;
```

Null

```
let emptyValue = null;
```

Object

```
let person = { firstName: "John",
lastName: "Doe" };
```

Array

```
let colors = ["red", "green", "blue"];
```

Function

```
function add(a, b) {
    return a + b;}
```

Global Variable

- A **JavaScript global variable** is declared outside the function or declared with window object
  - It can be accessed from any function
- Declaring JavaScript global variable within function
- To declare JavaScript global variables inside function, you need to use window object
  - For example window.value=90;
  - Now it can be declared inside any function and can be accessed from any function
  - function m(){   window.value=100;//declaring global variable by window object }
  - function n(){
  - alert(window.value);//accessing global variable from other function
  - }

Operators

- There are following types of operators in JavaScript

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

Arithmetic Operators

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

Comparison Operators

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

Bitwise Operators

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

Logical Operators

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

Assignment Operators

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

Special Operators

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value

If Statement

```
1.if(expression){
2.//content to be evaluated
3.}
```

```
<script>
var a=20;
if(a>10){
console.log("value of a is greater than 10");
}
</script>
```

If...else Statement

```
if(expression){
//content to be evaluated if condition is true
}
else{
//content to be evaluated if condition is false
}
```

```
<script>
var a=20;
if(a%2==0){
console.log("a is even number");
}
else{
console.log("a is odd number");
}
</script>
```

If...else if statement

```
if(expression){
//content to be evaluated if condition is true
}
else{
//if(expression1){
//content to be evaluated if expression1 is true
}
else if(expression2){
//content to be evaluated if expression2 is true
}
else if(expression3){
//content to be evaluated if expression3 is true
}
else{
//content to be evaluated if no expression is true
}  to be evaluated if condition is false
}
```

```
<script>
var a=20;
if(a==10){
console.log("a is equal to 10");
}
else if(a==15){
console.log("a is equal to 15");
}
else if(a==20){
console.log("a is equal to 20");
}
else{
console.log("a is not equal to 10, 15 or 20");
}
</script>
```

Switch

```
<script>
let grade='B';
let result;
switch(grade){
case 'A':
result="A Grade";
break;
case 'B':
result="B Grade";
break;
 case 'C':
result="C Grade";
break;
default:
result="No Grade";
}
Console.log(result);
</script>
```

Loops

- loops in JavaScript

1. for loop
2. while loop
3. do-while loop

For loop

```
for (initialization; condition; increment)
{
    code to be executed
}
```

```
<script>
for (i=1; i<=5; i++)
{
console.log(i + "<br/>")
}
</script>
```

while loop

```
while (condition)
{
    code to be executed
}
```

```
<script>
var i=11;
while (i<=15)
{
console.log(i + "<br/>");
i++;
}
</script>
```

do while loop

```
do{
    code to be executed
}while (condition);
```

```
<script>
var i=21;
do{
console.log(i + "<br/>");
i++;
}while (i<=25);
</script>
```

JavaScript Function

- The syntax of declaring function is given below

1. function functionName([arg1, arg2, ...argN]){
2. //code to be executed
3. }

```
<script>
function msg(){
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()" value="call function"
/>
```

Function Arguments

```
<script>
function getcube(number){
alert(number*number*number);
}
</script>
<form>
<input type="button" value="click" onclick="getcube(4)"/>
</form>
```

Function with Return Value

```
<script>
function getInfo(){
return "hello! How r u?";
}
</script>
<script>
document.write(getInfo());
</script>
```

JavaScript Objects

- A javaScript object is an entity having state and behavior (properties and method)
  - For example: car, pen, bike, chair, glass, keyboard, monitor, etc.
- JavaScript is an object-based language
  - Everything is an object in JavaScript
- JavaScript is template based not class based
  - Here, we don't create class to get the object
  - But, we direct create objects

Creating Objects in JavaScript

- There are 3 ways to create objects
- by object literal:
- object={property1:value1,property2:value2.....propertyN:valueN}

```
1.<script>
2.emp={id:102,name:"Shyam Kumar",salary:40000}
3.document.write(emp.id+" "+emp.name+" "+emp.salary);
4.</script>
```

By creating instance of Object

- In this way you can create the object directly
- var objectname=new Object();
- Here, **new** keyword is used to create object

1.<script>

2.var emp=new Object();

3.emp.id=101;

4.emp.name="Ravi Malik";

5.emp.salary=50000;

6.document.write(emp.id+" "+emp.name+" "+emp.salary);

7.</script

By using an Object constructor

- Here, you need to create a function with arguments
- Each argument value can be assigned in the current object by using **this** keyword

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal",30000);
document.write(e.id+" "+e.name+" "+e.salary);
```

Defining method in JavaScript Object

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
this.changeSalary=changeSalary;
function changeSalary(otherSalary){
this.salary=otherSalary;
}  }
e=new emp(103,"Sonoo Jaiswal",30000);
document.write(e.id+" "+e.name+" "+e.salary);
e.changeSalary(45000);
```

JavaScript Array

- JavaScript array is an object that represents a collection of similar type of elements
- There are 3 ways to construct array in JavaScript

1. By array literal: var arrayname=[value1,value2.....valueN];

```
<script>
var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
```

Array directly (new keyword)

2. By creating instance of Array directly (using new keyword)

var arrayname=new Array();

Here, **new keyword** is used to create instance of array

```
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
```

Array constructor (using new keyword)

3. By using an Array constructor (using new keyword): Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly

```
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

JavaScript Date Object

- You can use 4 variant of Date constructor to create date object

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)

JavaScript string

- The JavaScript string is an object that represents a sequence of characters
- There are 2 ways to create string in JavaScript

1. By string literal: var stringname="string value";

```
1.<script>
2.var str="This is string literal";
3.document.write(str);
4.</script>
```

string object (using new keyword)

2. By string object (using new keyword):
var stringname=new String("string literal");

Here, **new keyword** is used to create instance of string.

```
1.<script>
2.var stringname=new String("hello javascript string");
3.document.write(stringname);
4.</script>
```

JavaScript Objects

- A javaScript object is an entity having state and behavior (properties and method)
  - For example: car, pen, bike, chair, glass, keyboard, monitor, etc.
- JavaScript is an object-based language
  - Everything is an object in JavaScript
- JavaScript is template based not class based
  - Here, we don't create class to get the object
  - But, we direct create objects

Creating Objects in JavaScript

- There are 3 ways to create objects
- by object literal:
- object={property1:value1,property2:value2.....propertyN:valueN}

```
1.<script>
2.emp={id:102,name:"Shyam Kumar",salary:40000}
3.document.write(emp.id+" "+emp.name+" "+emp.salary);
4.</script>
```

By creating instance of Object

- In this way you can create the object directly
- var objectname=new Object();
- Here, **new** keyword is used to create object

```
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script
```

By using an Object constructor

- Here, you need to create a function with arguments
- Each argument value can be assigned in the current object by using **this** keyword

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal",30000);
document.write(e.id+" "+e.name+" "+e.salary);
```

Defining method in JavaScript Object

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
this.changeSalary=changeSalary;
function changeSalary(otherSalary){
this.salary=otherSalary;
} }
e=new emp(103,"Sonoo Jaiswal",30000);
document.write(e.id+" "+e.name+" "+e.salary);
e.changeSalary(45000);
```

JavaScript Array

- JavaScript array is an object that represents a collection of similar type of elements
- There are 3 ways to construct array in JavaScript

1. By array literal: var arrayname=[value1,value2.....valueN];

```
<script>
var emp=["Sonoo","Vimal","Ratan"];
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
```

Array directly (new keyword)

2. By creating instance of Array directly (using new keyword)

```
var arrayname=new Array();
```

Here, **new keyword** is used to create instance of array

```
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
```

Array constructor (using new keyword)

3. By using an Array constructor (using new keyword): Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly

```
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

JavaScript Date Object

- You can use 4 variant of Date constructor to create date object

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds)

JavaScript string

- The JavaScript string is an object that represents a sequence of characters
- There are 2 ways to create string in JavaScript

1. By string literal: var stringname="string value";

```
<script>
var str="This is string literal";
document.write(str);
</script>
```

string object (using new keyword)

2. By string object (using new keyword):
var stringname=new String("string literal");

Here, **new keyword** is used to create instance of string.

```
1.<script>
2.var stringname=new String("hello javascript string");
3.document.write(stringname);
4.</script>
```