UNIT II

CHAPTER 2

Web Scripting Languages

University Prescribed Syllabus

JavaScript : Introduction to Scripting languages, Introduction to JavaScript (JS), JS Variables and Constants, JS Variable Scopes, JS Data Types, JS Functions, JS Array, JS Object, JS Events.

Advanced JavaScript : JSON - JSON Create, Key-Value Pair, JSON Access, JSON Array, JS Arrow Functions, JS Callback Functions, JS Promises, JS Async-Await Functions, JS Error Handling.

AJAX : Why AJAX, Call HTTP Methods Using AJAX, Data Sending, Data Receiving, AJAX Error Handling.

JQUERY : Why JQuery, How to Use, DOM Manipulation with JQuery, Dynamic Content Change with JQuery, UI Design Using JQuery.

2.1 Introduction to Scripting Languages 2-3
- 2.1.1 Advantages of Scripting Languages 2-3
- 2.1.2 Application of Scripting Languages 2-3

2.2 Introduction to JavaScript 2-3
- 2.2.1 Features of JavaScript 2-4
- 2.2.2 Applications of JavaScript 2-4
- 2.2.3 Characteristics of JavaScript 2-4
- 2.2.4 Advantages and Disadvantages of Client- Side Scripting 2-4
- 2.2.5 Differences between Client Side and Server Side Scripting Language 2-5
- 2.2.6 Using JavaScript in an HTML Document 2-5
- 2.2.7 JavaScript Example 2-5

2.3 JavaScript Variables and Constants 2-6
- 2.3.1 Variables 2-6
- 2.3.2 JavaScript Constants 2-7

2.4 JS Variable Scopes 2-7

2.5 JS Data Types 2-9

2.6 JS Functions 2-10
- 2.6.1 Function Definition 2-10
- 2.6.2 Defining Function Arguments 2-11
- 2.6.3 Defining a return statement 2-11

2.7 JS Array 2-12
2.7.1 Different ways to create an Array 2-12
2.7.2 Array Methods 2-13
2.7.3 Iterating Through an Array 2-13
2.7.4 Deleting Element from an Array 2-15
2.7.5 Array Method : Splice() 2-15
2.8 JavaScript Objects 2-15
2.8.1 Creating Objects in JavaScript 2-16
2.8.2 String Object 2-17
2.8.3 RegExp Object 2-18
2.8.4 Math Object 2-18
2.8.5 Date Object 2-19
2.8.6 Number Object 2-19
2.8.7 Array Object 2-20
2.9 JS Events 2-24
2.10 Advanced JavaScript: JSON 2-26
2.11 JSON Data Types 2-26
2.12 JSON Object 2-27
2.13 JS Arrow Functions 2-28
2.14 JS Callback Functions 2-30
2.15 JavaScript Promise 2-32
2.16 JS Async-Await Functions 2-34
2.17 JS Error Handling 2-35
2.18 AJAX 2-35
2.18.1 Asynchronous in AJAX 2-36
2.18.2 Benefits of Ajax 2-36
2.18.3 Working of AJAX 2-37
2.18.4 Advantages and Disadvantages of Ajax 2-37
2.19 AJAX Design Basics 2-37
2.19.1 AJAX Processing Steps and Ajax Script 2-39
2.20 Call HTTP Methods Using AJAX 2-41
2.21 AJAX Error Handling 2-42
2.22 JQUERY 2-42
2.22.1 Features of jQuery 2-42
2.22.2 Loading jQuery 2-43
2.22.3 Selecting Elements 2-43
2.23 DOM Manipulation with JQuery 2-45
2.23.1 Changing Styles 2-45
2.23.2 Creating and Appending Elements 2-46
2.23.3 Removing Elements 2-46
2.24 Dynamic Content Change with JQuery 2-46
2.25 UI Design Using Jquery 2-48
❑ **Chapter Ends** 2-50

2.1 INTRODUCTION TO SCRIPTING LANGUAGES

GQ. What do you mean by Scripting languages?

All scripting languages are programming languages. The scripting language is basically a language where instructions are written for a run time environment. They do not require the compilation step and are rather interpreted. It brings new functions to applications and glue complex system together. A scripting language is a programming language designed for integrating and communicating with other programming languages.

There are many scripting languages some of them are discussed below:

1. **Bash :** It is a scripting language to work in the Linux interface. It is a lot easier to use bash to create scripts than other programming languages. It describes the tools to use and code in the command line and create useful reusable scripts and conserve documentation for other people to work with.
2. **Node js :** It is a framework to write network applications using **JavaScript**. Corporate users of Node.js include IBM, LinkedIn, Microsoft, Netflix, PayPal, Yahoo for real-time web applications.
3. **Ruby :** There are a lot of reasons to learn Ruby programming language. Ruby's flexibility has allowed developers to create innovative software. It is a scripting language which is great for web development.
4. **Python :** It is easy, free and open source. It supports procedure-oriented programming and object-oriented programming. Python is an interpreted language with dynamic semantics and huge lines of code are scripted and is currently the most hyped language among developers.
5. **Perl :** A scripting language with innovative features to make it different and popular. Found on all windows and Linux servers. It helps in text manipulation tasks. High traffic websites that use Perl extensively include priceline.com, IMDB.

2.1.1 Advantages of Scripting Languages

(1) **Easy learning :** The user can learn to code in scripting languages quickly, not much knowledge of web technology is required.

(2) **Fast editing :** It is highly efficient with the limited number of data structures and variables to use.

(3) **Interactivity :** It helps in adding visualization interfaces and combinations in web pages. Modern web pages demand the use of scripting languages. To create enhanced web pages, fascinated visual description, which includes background and foreground colors and so on.

(4) **Functionality :** There are different libraries which are part of different scripting languages. They help in creating new applications in web browsers and are different from normal programming languages.

2.1.2 Application of Scripting Languages

Scripting languages are used in many areas:

1. Scripting languages are used in web applications. It is used in server side as well as client side. Server side scripting languages are: JavaScript, PHP, Perl etc. and client side scripting languages are: JavaScript, AJAX, jQuery etc.
2. Scripting languages are used in system administration. For example: Shell, Perl, Python scripts etc.
3. It is used in Games application and Multimedia.
4. It is used to create plugins and extensions for existing applications.

2.2 INTRODUCTION TO JAVASCRIPT

GQ. Write a short note on JavaScript.

- JavaScript is a light-weight object-oriented programming language which is used by several websites for scripting the webpages.
- JavaScript was developed by Mr. Brendan Eich in 1995 who was working in Netscape.
- JavaScript is not a compiled language, but it is a translated language.
- The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.
- It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document.
- JavaScript helps to make our webpage more lively and interactive. JavaScript is widely used in mobile application development as well as in game development.
- JavaScript was initially called as LiveScript and later on the name is changed to JavaScript.

- With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

2.2.1 Features of JavaScript

The features of JavaScript are as follows:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

2.2.2 Applications of JavaScript

JavaScript is used to create interactive websites. It is mainly used for:

1. Client-side validation,
2. Dynamic drop-down menus,
3. Displaying date and time,
4. Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
5. Displaying clocks, etc.

2.2.3 Characteristics of JavaScript

GQ. Explain characteristics of JavaScript.

1. JavaScript is a **lightweight, interpreted** client-side scripting language.
2. Designed for developing network-based applications.
3. JavaScript is complementary to Java.
4. JavaScript is complementary to and integrated with HTML.
5. It is an open source and cross-platform.
6. The user input is validated before sending the page to the server. This minimizes the server traffic, which tends to fewer loads on the server.
7. There is no need for the user to wait to see if something has been forgotten to enter.
8. Interactive interfaces can be created, which can give responses to end user actions like mouse or keyboard activities.
9. JavaScript can include elements like drag-drop components and sliders to provide a feel of rich interface to the users.

2.2.4 Advantages and Disadvantages of Client- Side Scripting

GQ. What are the advantages and disadvantages of client side scripting?

Advantages of Client- Side Scripting

1. Immediate response to a user's actions which enables more interactivity.
2. No need to go to the server; hence, execution is fast.
3. Improve the usability of web sites for users whose browsers support scripts.
4. Developers get more control over the look and behaviour of their web widgets.
5. Possible to substitute by HTML if a user's browsers do not support scripts.
6. Are reusable and obtainable from various types of free resources.

Disadvantages of Client- Side Scripting

1. Scripts are not supported by all of the browsers; hence, errors might occur if no alternatives have been provided.
2. There is need of more quality assurance testing as different browsers and browser versions support scripts differently.
3. May need more time and effort for development if the scripts are not already available through other resources.
4. Sometimes, the web widget looks like a standard control, but their behaviour may be different or vice-versa, which may lead to usability problems.

2.2.5 Differences between Client Side and Server Side Scripting Language

GQ. What are the difference between client side and server side scripting language?

Parameters	Client Side Scripting Language	Server Side Scripting Language
Execution	The client side script is executed by web browser which is located at the user's computer.	The server side script is executed by the Web Server that outputs the page which is to be sent to the browser.
Database	Client side scripting language cannot connect to the databases which is located on the web server.	Server side scripting language can connect and access to the databases which is located on the web server.
File System	Client side scripting language cannot have access to the file system which is located on the web server.	Server side scripting language has access to the file system which is located on the web server.
Access to Setting	Client side scripting language can access the files and settings that are local at the user's computer.	Server side scripting language cannot access the settings that belong to web server.
Block	User can block the Client side scripting language	User cannot block the Server side scripting language
Response	Response from a client-side script is quick since the scripts are processed on the local computer.	Response from a server-side script is slow since the scripts are processed on the remote computer.
Examples	JavaScript, VBScript, etc.	PHP, JSP, ASP, ASP.Net, Ruby, Perl, etc.

2.2.6 Using JavaScript in an HTML Document

- The code (script) of JavaScript is written in the script opening <script> and closing </script> HTML tags in a web page. Usually, the <script> tag is allowed anywhere in the html page, but the <head> section is normally recommended.
- The <script> tag basically notifies the browser that the code written is a script of JavaScript.

Syntax

- Syntax of JavaScript will appear as follows :

```
<script ...>
  JavaScript statements;
</script>
```

- The <script> tag has two important attributes which basically serves the same purpose.
- **Language and Type :** These attributes specify the name of scripting language used. Here its value will be "JavaScript" for attribute *Language* and "text/JavaScript" for attribute *Type*.
- JavaScript segment will look like :

```
<script language="JavaScript">
  JavaScript statements
</script>
```

OR

```
<script type="text/JavaScript">
  JavaScript statements
</script>
```

2.2.7 JavaScript Example

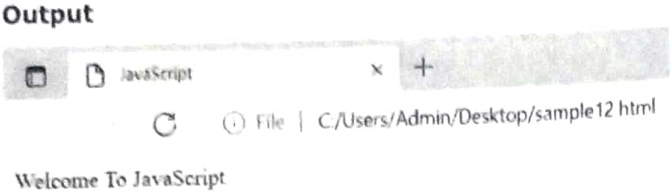
1. Embedded JavaScript

The JavaScript code can be embed into a HTML file using the <script> tag.

```
<!DOCTYPE html>
<html>
<head>
<title>
JavaScript
</title>
<script language="JavaScript">
document.write("Welcome To JavaScript");
</script>
</head>
<body>
</body>
</html>
```

document.write :
This method writes a string into HTML document.

Output

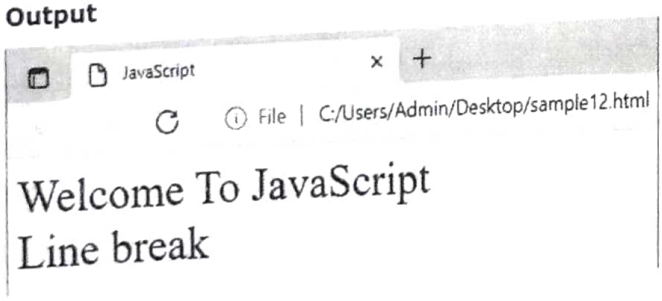


In the above program, outside the <script> element, the HTML tags can be used in their normal way, but if we want to use the HTML tags in <script> section, then document.write() method is used.

Program 2.2.1 : Write a program to demonstrate line break example.

```
<html>
<head>
<title>
JavaScript
</title>
</head>
<body>
<font size=6 color="blue">
<script type="text/JavaScript">
document.write("Welcome To JavaScript");
document.write("<br>Line break");
</script>
</font>
</body>
</html>
```

Output



2. External JavaScript

GQ. Write note on External JavaScript with an example.

- An external JavaScript file can be created to embed it in many html pages.
- It supports the concept of code reusability as single JavaScript file can be embed into several html pages. An external JavaScript file is saved by the extension ".js".

Example

External JavaScript code demonstrating welcome message.

MyFile.js

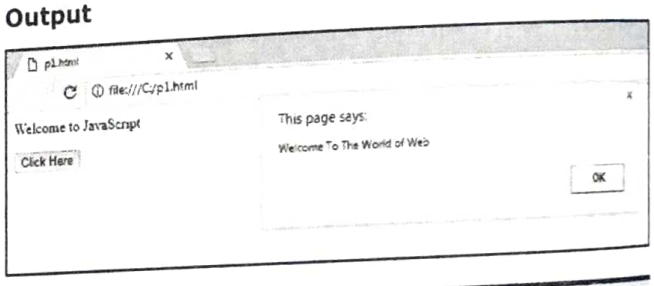
```
function msg()
{
    alert("Welcome To The World of Web");
}
```

HTML page

```
<html>
<head>
<script language="JavaScript" src="MyFile.js">
</script>
</head>

<body>
<p>Welcome to JavaScript</p>
<form>
<input type="button" value="Click Here"
onclick="msg()"/>
</form>
</body>
</html>
```

Output



2.3 JAVASCRIPT VARIABLES AND CONSTANTS

2.3.1 Variables

- Variable is a name given to memory location where we can store some value. The value depends upon the data type of the variable.
- In JavaScript there are number of **data types** used to store different types of values. These data types are primarily categorized as :

1. JavaScript primitive data types

In JavaScript, there are five types of primitive data types as follows :

Sr. No.	Data Type	Description
1.	String	Represents sequence of characters e.g. "Snehal"
2.	Number	Represents numeric values e.g. 13
3.	Boolean	Represents Boolean value either true or false
4.	Undefined	Represents undefined value
5.	Null	Represents null means no value at all

2. JavaScript non-primitive data types

Sr. No.	Data Type	Description
1.	Object	Represents instance which helps to access members
2.	Array	Represents set of same values
3.	RegExp	Represents regular expression

- JavaScript is considered as a **dynamic type language** that is there is no need to specify type of the variable. This type is dynamically decided by the JavaScript engine. While declaring a variable, "var" keyword is used on place of data type. Var means variant, that is the variable can store any type of value like numbers, strings, dates etc.

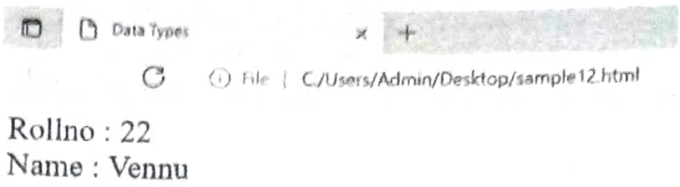
Examples

1. var rno = 101; //holding number
2. var sname="Shraddha"; //holding string

Program 2.3.1 : To display roll no. and name of student using "var" keyword.

```
<html>
<head>
<title>
Data Types
</title>
</head>
<body>
<scripi language="JavaScript">
document.write("<font size=5 color=blue>");
var rno = 22;
var sname = "Vennu";
document.write("Rollno : "+rno);
document.write("<br>Name : "+sname);
document.write("</font>");
</script>
</body>
</html>
```

Output



2.3.2 JavaScript Constants

- The const keyword was also introduced in the ES6(ES2015) version to create constants. For example,

```
const x = 5;
```

- Once a constant is initialized, we cannot change its value.

```
const x = 5;
x = 10; // Error! constant cannot be changed.
console.log(x)
```

- Simply, a constant is a type of variable whose value cannot be changed.
- Also, you cannot declare a constant without initializing it. For example,

```
const x; // Error! Missing initializer in const declaration.
x = 5;
console.log(x)
```

2.4 JS VARIABLE SCOPES

- Scope refers to the availability of variables and functions in certain parts of the code.
- In JavaScript, a variable has two types of scope:

(1) Global Scope	(2) Local Scope

1. Global Scope

- A variable declared at the top of a program or outside of a function is considered a global scope variable.

Program 2.4.1 : Example of a global scope variable

```
// program to print a text
let a = "hello";

function greet () {
  console.log(a);
}

greet(); // hello
```

- In the above program, variable a is declared at the top of a program and is a global variable. It means the variable a can be used anywhere in the program.
- The value of a global variable can be changed inside a function. For example,

```
// program to show the change in global variable
let a = "hello";

function greet() {
  a = 3;
}

// before the function call
console.log(a);

//after the function call
greet();
console.log(a); // 3
```

- In the above program, variable a is a global variable. The value of a is hello. Then the variable a is accessed inside a function and the value changes to **3.**
- Hence, the value of a changes after changing it inside the function.
- In JavaScript, a variable can also be used without declaring it. If a variable is used without declaring it, that variable automatically becomes a global variable.
- For example,

```
function greet() {
  a = "hello"
}

greet();

console.log(a); // hello
```

- In the above program, variable a is a global variable.
- If the variable was declared using let a = "hello", the program would throw an error.

2. Local Scope

- A variable can also have a local scope, i.e it can only be accessed within a function.

Program 2.4.2 : Local Scope Variable

```
// program showing local scope of a variable
let a = "hello";

function greet() {
  let b = "World"
  console.log(a + b);
}

greet();
console.log(a + b); // error
```

Output

```
helloWorld
Uncaught ReferenceError: b is not defined
```

- In the above program, variable a is a global variable and variable b is a local variable. The variable b can be accessed only inside the function greet. Hence, when we try to access variable b outside of the function, an error occurs.

let is Block Scoped

- The let keyword is block-scoped (variable can be accessed only in the immediate block).

Program 2.4.3 : Block-scoped Variable

```
// program showing block-scoped concept
// global variable
let a = 'Hello';

function greet() {

  // local variable
  let b = 'World';

  console.log(a + ' ' + b);

  if (b == 'World') {

    // block-scoped variable
    let c = 'hello';

    console.log(a + ' ' + b + ' ' + c);
  }

  // variable c cannot be accessed here
  console.log(a + ' ' + b + ' ' + c);
}

greet();
```

Output

```
Hello World
Hello World hello
Uncaught ReferenceError: c is not defined
```

In the above program, variable

- a is a global variable. It can be accessed anywhere in the program.
- b is a local variable. It can be accessed only inside the function greet.
- c is a block-scoped variable. It can be accessed only inside the if statement block.

Hence, in the above program, the first two console.log() work without any issue.

However, we are trying to access the block-scoped variable c outside of the block in the third console.log(). This will throw an error.

2.5 JS DATA TYPES

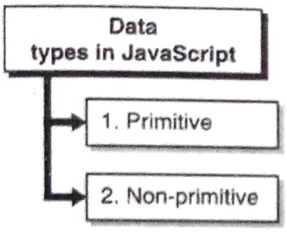


Fig. 2.5.1 : Data types in JavaScript

- Variable is a name given to memory location where we can store some value. The value depends upon the **data type** of variable.
- In JavaScript there are number of **data types** used to store different types of values. These data types are primarily categorized as :

1. JavaScript primitive data types

In JavaScript, there are five types of primitive data types as follows :

Sr. No.	Data Type	Description
1.	String	Represents sequence of characters e.g. "Ishita"
2.	Number	Represents numeric values e.g. 101
3.	Boolean	Represents Boolean value either true or false
4.	Undefined	Represents undefined value
5.	Null	Represents null means no value at all

2. JavaScript non-primitive data types

Sr. No.	Data Type	Description
1.	Object	Represents instance which helps to access members
2.	Array	Represents set of same values
3.	RegExp	Represents regular expression

- JavaScript is considered as a **dynamic type language** that is there is no need to specify type of the variable.
- This type is dynamically decided by the JavaScript engine. While declaring a variable, "var" keyword is used on place of data type. Var means variant, that is the variable can store any type of value like numbers, strings, dates etc.

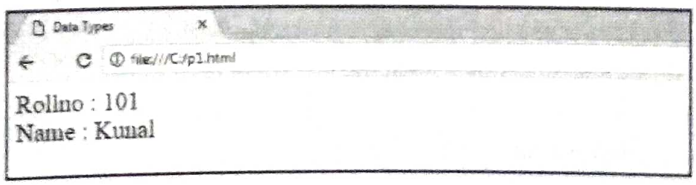
Examples

1. var rno = 101; //holding number
2. var sname="Kunal"; //holding string

Program 2.5.1 : Write a program to display roll no. and name of student using "var" keyword.

```
<html>
<head>
<title>
Data Types
</title>
</head>
<body>
<script language="JavaScript">
document.write("<font size=5 color=blue>");
var rno = 101;
var sname = "Kunal";
document.write("Rollno : "+rno);
document.write("<br>Name : "+sname);
document.write("</font>");
</script>
</body>
</html>
```

Output

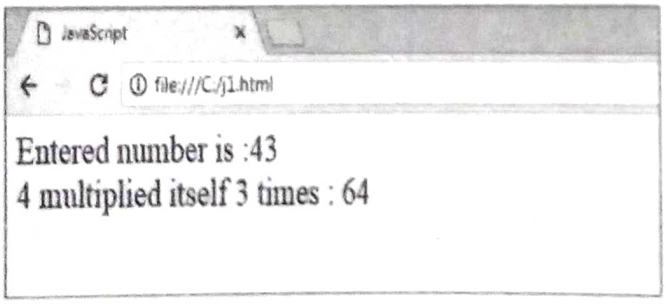


Program 2.5.2 : Write a JavaScript to take 2 digit number and then separate these 2 digits, then multiply first digit by itself for second digit times. (for example, 23 should be separated as 2 and 3. 2 should multiply with itself 3 times.

```
<html>
<head>
<title>JavaScript</title>
</head>

<body>
<font size=5>
<script type="text/javascript">
var num=parseInt(prompt(("enter no")));
var rem=num%10;
document.write("Entered number is :" + num +"<br/>")
var quat=parseInt(num/10);
var result=Math.pow(quat, rem);
document.write(quat+" multiplied itself "+rem+" times : "+
result);
</script>
</font>
</body>
</html>
```

Output



2.6 JS FUNCTIONS

GQ. Explain user defined function in JavaScript with suitable example.

- Function helps to avoid repetition of code and write modular codes. A program can be divided into small and manageable modules called as functions.
- Just like the other programming languages such as C, C++ and Java, the function concept with all its features is supported by JavaScript.
- We can create our own functions known as user defined functions.

2.6.1 Function Definition

In JavaScript, a function is defined by using the **function** keyword, followed by name of function, a parameters list [optional], and a statement block inside curly braces.

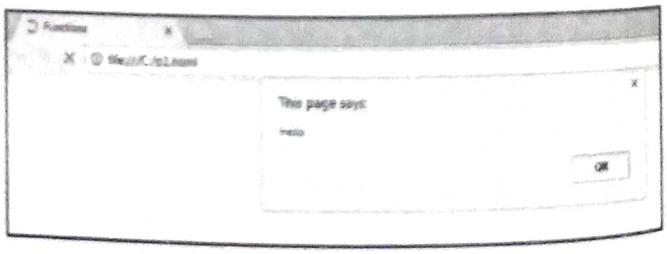
Syntax

```
<script type="text/javascript">
   function function_name([parameter-list])
   {
     Statements;
   }
</script>
```

Program to display hello message using function call

```
<html>
<head>
<title>
Functions
</title>
</head>
<body>
<script language="JavaScript">
function callMe()
{
   alert("Hello");
}
callMe();
</script>
</body>
</html>
```

Output

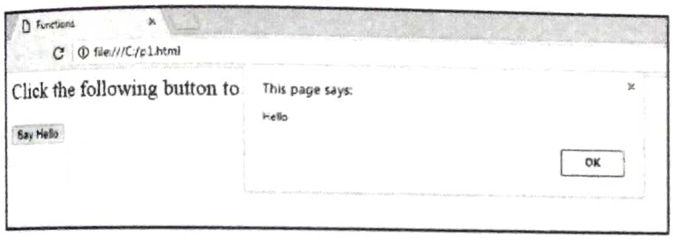


Calling a function on click event of a button

```
<html>
<head>
<title>
Functions
</title>
<script language="JavaScript">
function callMe()
```

```
{
   alert("Hello");
}
</script>
</head>
<body>
<font size=5>
<p>Click the following button to call the function</p>
<form>
<input type="button" onClick="callMe();" value="Say
Hello">
</form>
</font>
</body>
</html>
```

Output



2.6.2 Defining Function Arguments

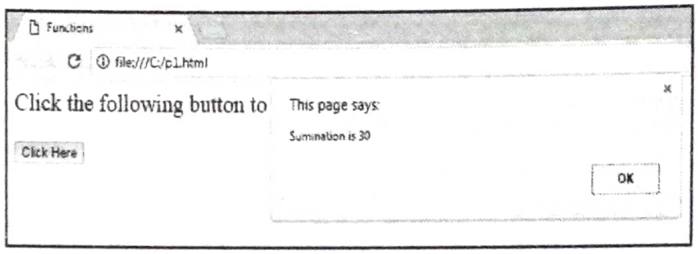
- While defining a function, we can declare variables in the header statement of the function. These variables are known as parameters or formal arguments.
- When this function gets called, we can pass values for these variables. These values are known as arguments or actual arguments.

GQ. Write a program to display the summation of two values using parameterized function.

```
<html>
<head>
<title>
Functions
</title>
<script language="JavaScript">
function add(a,b)
{
   var sum = a + b;
   alert("Summation is "+sum);
}
</script>
</head>
<body>
<font size=5>
<p>Click the following button to call the function</p>

   <form>
     <input type="button" onClick="add(10,20);"
   value="Click Here">
   </form>
</font>
</body>
</html>
```

Output



2.6.3 Defining a return statement

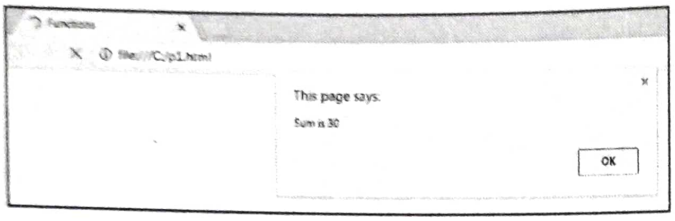
- In JavaScript function we can have return statement which is optional. This helps to return a value from a function. This statement is written at the last in a function.
- The returned value goes to the location where the function is called.

GQ. Write a program which will accept two numbers as arguments and returns their summation.

```
<html>
<head>
<title>
Functions
</title>
<script language="JavaScript">
function add(a,b)
{
   var sum = a + b;
   return(sum);
}
</script>
</head>
<body>
<script language="JavaScript">
```

```
var r = add(10,20);
alert("Sum is "+r);
</script>
</body>
</html>
```

Output



2.7 JS ARRAY

GQ. How to create arrays in JavaScript?

Definition : An array is a group of elements of same data type. All the elements in array have index numbers which starts from zero. These index numbers are used to access the specific array element.

Syntax

```
var students = new Array( "Kunal", "Ishita", "Shravi" ,
"Shrey");
```

- The array is created using new keyword. The maximum size allowed for an array is 4,294,967,295.
- You can create array by simply assigning values as follows –

```
var students = ["Kunal", "Ishita", "Shreya", "Shravi"];
```

- The index numbers are used to access the array elements

```
students[0] is the first student – Kunal
students[1] is the second student – Ishita
```

2.7.1 Different ways to create an Array

1. **Empty array without elements**

```
var empty = [];
```

2. **Array with 2 string elements**

```
var days = ["Sunday", "Monday"];
```

3. **Array with different types of elements**

```
var mixed = [true, 100, "Hello"];
```

4. **Two dimensional array with object literals**

```
var arr = [[1,{x:10, y:20}], [2, {x:30, y:40}]];
```

5. **The 3rd element is undefined**

```
var colors = ["Red", "Blue", undefined];
```

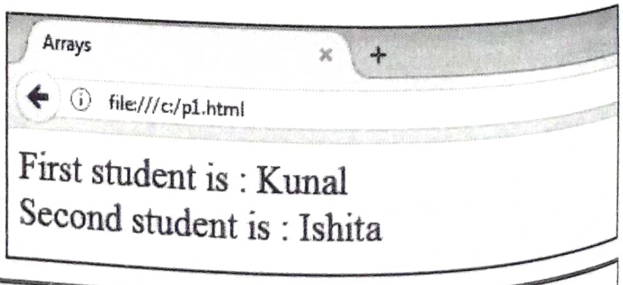
6. **No value in the 1st position, it is undefined**

```
var hobbies = [,"Sports"];
```

Program 2.7.1 : Write a simple program of array displaying student name.

```
<html>
<head>
<title>
Arrays
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var students = new Array( "Kunal", "Ishita", "Shravi" ,
"Shrey");
document.write("First student is : "+students[0]);
document.write("<br>Second student is : "+students[1]);
</script>
</font>
</body>
</html>
```

Output



NOTES

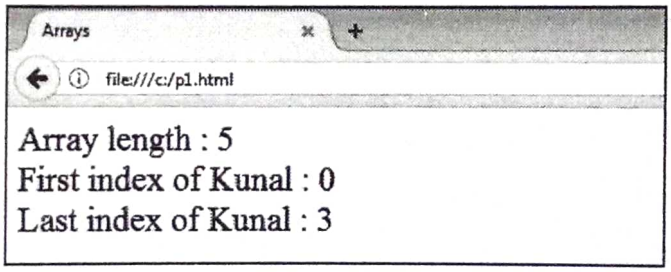
2.7.2 Array Methods

Method	Description
Length	Returns length of an array (It is a property of array)
concat()	Concatenates (merge) multiple arrays.
every()	If all the elements in the array satisfy the given condition of testing function, it returns true.
filter()	Returns array elements which satisfy the filter criteria of given function.
forEach()	Calls a specific function for all the elements in the array.
indexOf()	Return the index of first occurrence of given element. Returns -1 if element not found.
join()	Joins all the elements of an array and converts into a string.
lastIndexOf()	Return the index of last occurrence of given element. It returns 1, if the specified element not found.
pop()	Removes and returns the last element of an array.
push()	Adds one or more elements at the end of an array.
reverse()	Reverses the sequence of the elements of an array.
shift()	Removes and returns the last element of an array.
slice()	Extracts a portion of an array and returns it.
sort()	Sorts the elements of an array
splice()	Adds and/or removes elements from an array.
toString()	Returns a string representation of the given array
unshift()	Adds one or more elements at the beginning of an array and returns it.

Program 2.7.2 : Write a program to display the length of array and index of array element.

```
<html>
<head>
<title>
Arrays
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var students = ["Kunal", "Ishita", "Shravi" , "Kunal","Shrey"];
document.write("Array length : "+students.length);
document.write("<br>First index of Kunal :
"+students.indexOf("Kunal"));
document.write("<br>Last index of Kunal :
"+students.lastIndexOf("Kunal"));
</script>
</font>
</body>
</html>
```

Output



2.7.3 Iterating Through an Array

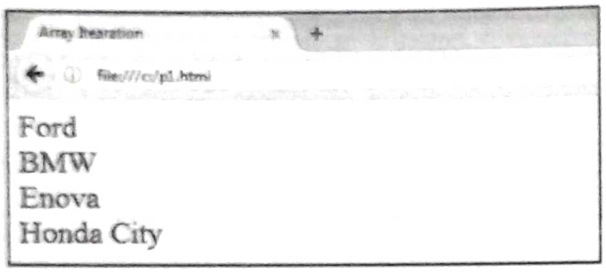
☞ **Use :** Iterating or traversing of an array means visiting each element at least once. For this purpose we can use the loops.

Program 2.7.3 : Write a simple program of array iteration.

```
<html>
<head>
<title>
Array Iteration
</title>
</head>
<body>
<font size=5>
```

```
<script language="JavaScript">
var cars = [];
cars[0] = "Ford";
cars[1] = "BMW";
cars[2] = "Enova";
cars[3] = "Honda City";
for (var i = 0; i < cars.length; i++)
{
    document.write(cars[i] + "<br>");
}
</script>
</font>
</body>
</html>
```

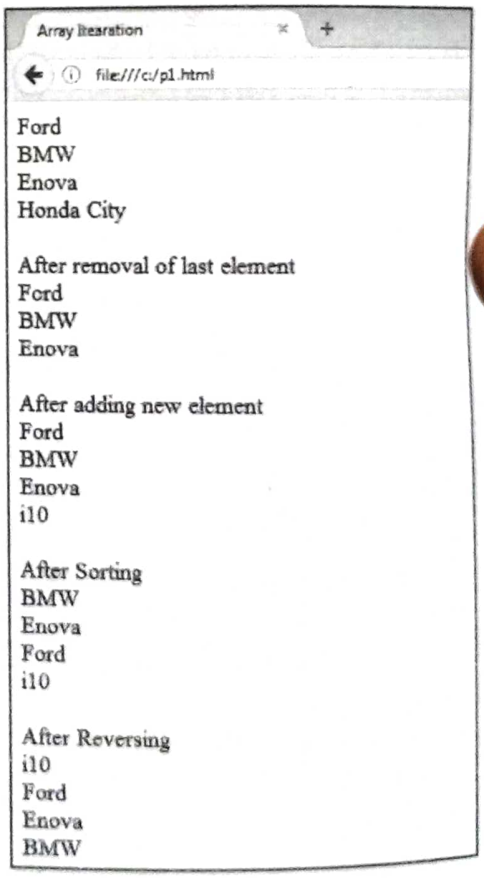
Output



Program 2.7.4 : Write a program to perform various methods like adding a new element, sorting, reversing, removal of last element etc.

```
<html>
<head>
<title>
Array Iteration
</title>
</head>
<body>
<font size=4>
<script language="JavaScript">
var cars = [];
cars[0] = "Ford";
cars[1] = "BMW";
cars[2] = "Enova";
cars[3] = "Honda City";
for (var i = 0; i < cars.length; i++)
{
document.write(cars[i] + "<br>");
}
cars.pop();
document.write("<br>After removal of last element<br>");
for (var i = 0; i < cars.length; i++)
{
document.write(cars[i] + "<br>");
}
cars.push("i10");
document.write("<br>After adding new element<br>");
for (var i = 0; i < cars.length; i++)
{
document.write(cars[i] + "<br>");
}
cars.sort();
document.write("<br>After Sorting<br>");
for (var i = 0; i < cars.length; i++)
{
document.write(cars[i] + "<br>");
}
cars.reverse();
document.write("<br>After Reversing<br>");
for (var i = 0; i < cars.length; i++) {
document.write(cars[i] + "<br>");}
</script>
</font>
</body>
</html>
```

Output



2.7.4 Deleting Element from an Array

GQ. Explain how to delete an element from array in JavaScript.

Use : The delete operator is used to remove an element from an array. Deleting an element from an array does not affect the length property and the array becomes sparse. Also the elements which are at the right of the deleted element do not get shifted to left to fill in the gap.

Example

```
var days = ["Sunday", "Monday", "Tuesday", "Wednesday"];
delete days[1];          // delete the element "Monday"
```

2.7.5 Array Method : Splice()

GQ. Explain array method splice() with suitable examples in JavaScript.

splice() method

Use : The splice() method is used to insert new, delete existing, and replace existing elements by new elements in the array.

- It moves the elements to higher or lower positions as per the requirement to avoid any gap.
- The first argument of splice() indicates the starting position and second argument indicates the number of elements to delete.

Examples

```
var letters = ["a","b","c","d","e","f","g"];
alert(letters.splice(5, 2));     // f, g (deleted elements)
alert(letters);                  // a, b, c, d, e
alert(letters.splice(2, 1));     // c  (the deleted element)
alert(letters);                  // a, b, d, e
```

- The third argument in the splice method is used to replace one or more elements with others.

```
var letters = ["a","b","c","d"];
alert(letters.splice(1, 2, "e", "f", "g"));        // b, c (deleted
ones)
alert(letters);                                    // a, e, f, g, d
```

- In this example, the splice starts at position 1 and removes two elements b and c. Then it fills the gap with the three elements e, f and g.

2.8 JAVASCRIPT OBJECTS

GQ. What are JavaScript objects? List the important built-in objects. How can you write your **own** object?

An object is nothing but an entity having its own state and behaviour (properties and methods).

- **For example :** A flower is an object having properties like color, fragrance etc. Other examples of objects are car, pen, bike, chair, glass, keyboard, monitor etc.
- JavaScript is an object-oriented language. Everything in JavaScript is considered as an object.

Examples of object

- Following are some of the examples of objects in JavaScript.
  - (i) Booleans (ii) Numbers
  - (iii) Strings (iv) Dates
  - (v) Regular expressions (vi) Arrays
  - (vii) Functions
- We can create our own user defined objects in JavaScript.
- JavaScript is primarily a template-based scripting language rather than a class-based language. Hence, we directly create the object without class.

2.8.1 Creating Objects in JavaScript

1. JavaScript Object by Object Literal

Syntax

- Following is the syntax of creating object using object literal :

```
object={property1:value1,property2:value2
propertyN:valueN}
```

- As we can observe, the property and value is separated by the separator : (colon).

Program 2.8.1 : Write a sample program of creating object using object literal.

```
<html>
<head>
<title>
Objects
</title>
</head>
<body>
<font size=5>
```

```
<script language="JavaScript">
employee = {id:101,name:" Prajakta Lonkar",salary:40000}
document.write(employee.id+"-"+employee.name+"-"
+employee .salary);
</script>
</font>
</body>
</html>
```

Output

29-Prajakta Lonkar-40000

2. By creating instance of object

Syntax

Following is the syntax of creating instance of object :

```
var objectname=new Object();
```

Here, **new keyword** is used to create object.

Program 2.8.2 : Write a program of creating the instance of object

```
<html>
<head>
<title>
Objects
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
var employee=new Object();
employee.id=18;
employee.name="Chetana Murudkar";
employee.salary=800000;
document.write(employee.id+"-"+employee.name+"-"
+employee.salary);
</script>
</font>
</body>
</html>
```

Output

18-Chetana Murudkar-800000

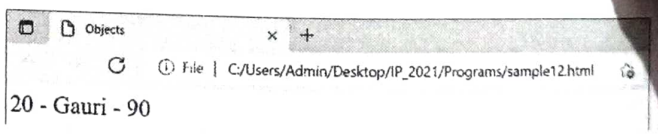
3. By using an object constructor

- Here, we have to create parameterized function. **"this"** keyword is used to assign each argument value in the current object.
- The "this" keyword refers to the current object.

Program 2.8.3 : Write a program using "this" keyword

```
<html>
<head>
<title>
Objects
</title>
</head>
<body>
<font size=5>
<script language="JavaScript">
function student(id,sname,marks)
{
   this.id=id;
   this.sname=sname;
   this.marks=marks;
}
s = new student(20,"Gauri",90);
document.write(s.id+" - "+s.sname+" - "+s.marks);
</script>
</font>
</body>
</html>
```

Output



20 - Gauri - 90

2.8.2 String Object

- The JavaScript String object is a global object that is used to store strings.
- A string is a sequence of letters, numbers, special characters and arithmetic values or combination of all.

Syntax

```
var val = new String(string);
```

- The String parameter is a series of characters that has been properly encoded.

String Methods

Here is a list of the methods available in String object along with their description.

Sr. No.	Method	Description
1.	charAt()	Returns the character at the specified index.
2.	charCodeAt()	Returns the Unicode of the character at the specified index.
3.	concat()	Joins two or more strings, and returns a new string.
4.	endsWith()	Checks whether a string ends with a specified substring.
5.	fromCharCode()	Converts Unicode values to characters.
6.	includes()	Checks whether a string contains the specified substring.
7.	indexOf()	Returns the index of the first occurrence of the specified value in a string.
8.	lastIndexOf()	Returns the index of the last occurrence of the specified value in a string.
9.	localeCompare()	Compares two strings in the current locale.
10.	match()	Matches a string against a regular expression, and returns an array of all matches.
11.	repeat()	Returns a new string which contains the specified number of copies of the original string.
12.	replace()	Replaces the occurrences of a string or pattern inside a string with another string, and return a new string without modifying the original string.
13.	search()	Searches a string against a regular expression, and returns the index of the first match.
14.	slice()	Extracts a portion of a string and returns it as a new string.
15.	split()	Splits a string into an array of substrings.
16.	startsWith()	Checks whether a string begins with a specified substring.
17.	substr()	Extracts the part of a string between the start index and a number of characters after it.
18.	substring()	Extracts the part of a string between the start and end indexes.
19.	toLocaleLowerCase()	Converts a string to lowercase letters, according to host machine's current locale.
20.	toLocaleUpperCase()	Converts a string to uppercase letters, according to host machine's current locale.
21.	toLowerCase()	Converts a string to lowercase letters.
22.	toString()	Returns a string representing the specified object.
23.	toUpperCase()	Converts a string to uppercase letters.
24.	trim()	Removes whitespace from both ends of a string.
25.	valueOf()	Returns the primitive value of a String object.

2.8.3 RegExp Object

- RegExp object used to validate the pattern of characters.
- RegExp define methods that use regular expressions to perform powerful pattern-matching and search and replace functions on text.
- Regular expressions can be defined by using following ways :
  - Using RegExp() Constructor: var RegularExpression = new RegExp("pattern","flag");
  - Using Literal: var RegularExpression = /pattern/flag;

- Pattern – A String that specifies the pattern of the regular expression or another regular expression.
- Flag – An optional string containing any of the "g", "I" and "m" attributes that specify global, case-insensitive and multiple matches respectively.

☞ Properties of RegExp object

1. **Constructor** - Returns the function that created the RegExp object.
2. **Global** - Checks whether the "g" modifier is set.
3. **ignoreCase** - Checks whether the "i" modifier is set.
4. **lastIndex** - Specifies the index at which to start the next match.
5. **multiline** - Checks whether the "m" modifier is set.
6. **source** - Returns the text of the RegExp pattern.

☞ Methods of RegExp object

1. **exec()** - Tests for a match in a string. Returns the first match.
2. **test()** - Tests for a match in a string. Returns true or false.
3. **toString()** - Returns the string value of the regular expression.
4. **toSource()** - Returns an object literal representing the specified object.

Following examples explain more about matching characters.

Expression	Description
[^a-zA-Z]	It matches any string not containing any of the characters ranging from a through z and A through Z.
p.p	It matches any string containing p, followed by any character, in turn followed by another p.
^.{2}$	It matches any string containing exactly two characters.
(.*)	It matches any string enclosed within and .
p(hp)*	It matches any string containing a p followed by zero or more instances of the sequence hp.

2.8.4 Math Object

- The Math object is used to perform simple and complex arithmetic operations.
- The Math object provides a number of properties and methods to work with number values
- The Math object does not have any constructors. All of its methods and properties are static; that is, they are member functions of the Math object itself. There is no way to create an instance of the Math object.

Syntax

```
var pi_val = Math.PI;
var sine_val = Math.sin(30);
```

☞ Math Methods

Following are some important methods of Math Object :

Method	Description
abs()	Returns the absolute value of a number.
ceil()	Returns the smallest integer greater than or equal to a number.
exp()	Returns E^N, where N is the argument, and E is Euler's constant, the base of the natural logarithm.
floor()	Returns the largest integer less than or equal to a number.
log()	Returns the natural logarithm (base E) of a number.
max()	Returns the largest of zero or more numbers.
min()	Returns the smallest of zero or more numbers.
pow()	Returns base to the exponent power, that is, base exponent.
random()	Returns a pseudo-random number between 0 and 1.
round()	Returns the value of a number rounded to the nearest integer.
sqrt()	Returns the square root of a number.

2.8.5 Date Object

- At times when user need to access the current date and time and also past and future date and times. JavaScript provides support for working with dates and time through the Date object.
- The Date object provides a system-independent abstraction of dates and times.
- Date object can be created as :

```
var today = new Date( );
```

- Dates may be constructed from a year, month, day of the month, hour, minute, and second, and those six components, as well as the day of the week, may be extracted from a date.
- Dates may also be compared and converted to a readable string form. A Date is represented to a precision of one millisecond.

Properties of Date object

1. **Constructor** - Returns the function that created the Date object.
2. **Prototype** - Add properties and methods to an object.

Methods of Date object

1. **Date()** - Returns today's date and time
2. **getDate()** - Returns the day of the month for the specified date
3. **getDay()** - Returns the day of the week for the specified date
4. **getFullYear()** - Returns the year of the specified date
5. **getHours()** - Returns the hour in the specified date according to local time.
6. **getMilliseconds()** - Returns the milliseconds in the specified date according to local time.
7. **getMinute(), getMonth(), getTime(), getTimezoneOffset(), setDate(), setFullYear(), setHours(), setMilliseconds(), setMinutes(), setMonth(), setSeconds(), setTime()** are some of the methods used in Date object.

2.8.6 Number Object

The Number objects represents numerical date, either integers or floating-point numbers.

A Number objects are created using the number() constructor **var num = new number(value);**

Properties of Number object

1. **Constructor** - Returns the function that created the Number object.
2. **Max Value** - Returns maximum numerical value possible in JavaScript.
3. **Min Value** - Returns minimum numerical value possible in JavaScript.
4. **Negative Infinity** - Represent the value of negative infinity.
5. **Positive Infinity** - Represent the value of infinity.
6. **Prototype** - Add properties and methods to an object.

Methods of Number object

1. **toExponential()** - Converts a number into exponential notation.
2. **toFixed()** - Formats a number with a specific number of digits to the right of the decimal.
3. **toLocaleString()** - Returns a string value version of the current number in a format that may vary according to a browser's locale settings.
4. **toPrecision()** - Defines how many total digits to display of a number.
5. **toString()** - Returns the string representation of the number's value.
6. **valueOf()** - Returns the number's value.

2.8.7 Array Object

- Multiple values are stored in a single variable using the Array object.
- In JavaScript, an array can hold different types of data types, which implies that an array can have a string, a number or an object in a single slot.
- An Array object can be created by using following ways:

Using the Array Constructor

To create empty array when don't know the exact number of elements to be inserted in an array

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

To create an array of given size

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

To create an array with given elements

```
var arrayname = new Array("element 1","element
2",.........,"element n");
```

Using the Array Literal Notation:

To create empty array

```
var arrayname =[ ];
```

To create an array when elements are given

```
var arrayname =["element 1","element 2",.........,"element
n"];
```

Properties of the Array object

1. **Length** - Returns the number of elements in the array.
2. **Constructor** - Returns the function that created the Array object.
3. **Prototype** - Add properties and methods to an object.

Methods of the Array object

1. reverse() - Reverses the array elements
2. concat() - Joins two or more arrays
3. sort() - Sort the elements of an array
4. push() - Appends one or more elements at the end of an array
5. pop() - Removes and returns the last element
6. shift() - Removes and returns the first element
7. **unshift(), join(), indexOf(), lastIndexOf(), slice(startindex, endindex)** are some of the methods used in Array object.

2.9 JS EVENTS

GQ. Explain the event handling in JavaScript with simple example.

- Events are the actions performed by the end users while browsing the website. For example mouse move or mouse click on the buttons.
- When an event is fired, objects are triggered which are associated with that specific event.
- The event is caught by the event handlers and in response the related code is executed.
- Events are basically classified in four categories :

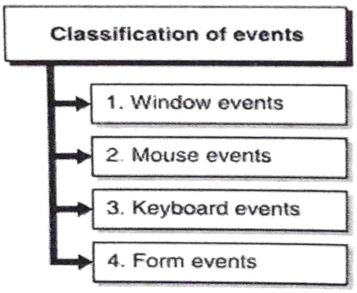


Fig. 2.9.1 : Classification of events

1. Window Events

There are various types of events associated with window :

(i) **onLoad** - triggered when a new page is starting up.
(ii) **onUnload** - triggers when a page is shutting down.
(iii) **onResize** - triggers when a page is resized.
(iv) **onMove** - triggers when a page is moved.
(v) **onAbort** - triggers when a page is cancelled.
(vi) **onError** - triggers when an error occurs.
(vii) **onFocus** - triggers when the window moves to foreground.
(viii) **onBlur** - triggers when window changes to background.

2. Mouse Events

There are various types of events associated with mouse :

(i) **onMousedown** - triggers when mouse button is pressed on an element.
(ii) **onMouseup** - triggers when mouse button is released.
(iii) **onMousemove** - triggers when mouse pointer is moved and the pointer is already over an element.
(iv) **onMouseout** - triggers when mouse pointer is moved out of an element.
(v) **onMouseover** - triggers when the pointer is over an element.
(vi) **onClick** - triggers when mouse button is clicked once.
(vii) **onDblclick** - triggers when mouse button is clicked twice.

3. Keyboard Events

There are various types of events associated with keyboard :

(i) **onKeydown** - Triggers when a key is pressed down.
(ii) **onKeyup** - Triggers when a key is released.
(iii) **onKeypress** - Triggers when complete key sequence, down press and up release happens.

4. Form events

There are following types of events associated with form :

(i) **onReset** - triggers when the reset button on the form is clicked.
(ii) **onSubmit** - triggers when the submit button is clicked.
(iii) **onSelect** - triggers when a content is selected on a page.

Using Events

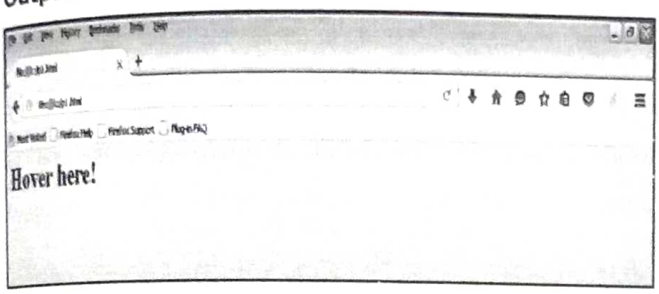
All these event can be used to execute some scripts. In the following programs we will use mouseover and mouseout events.

Program 2.9.1 : Program to illustrate the use of mouseover and mouseout events

```
<html>
<body bgcolor="lightgray">
```

```
<h1 onmouseover="style.color='blue'"
onmouseout="style.color='black'">
Hover here!
</h1>
</body>
</html>
```

Output

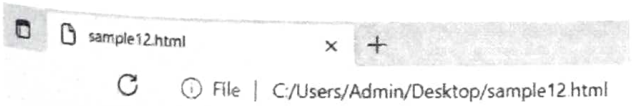


☞ Use of mouse down and mouse up events

Program 2.9.2 : Program illustrating mouse up and mouse down events.

```
<html>
<body>
<h2 id="myid1" onmousedown="fun1()"
onmouseup="fun2()">Click the text! </h2>
<script>
function fun1()
{
   document.getElementById("myid1").style.color = "blue";
}
function fun2()
{
   document.getElementById("myid1").style.color= "black";
}
</script>
</body>
</html>
```

Output

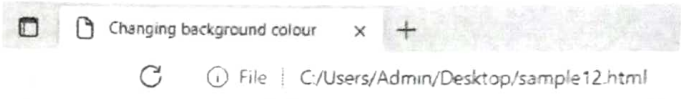


Click the text!

Program 2.9.3 : Write a program to show the use of events to change background color with DHTML. If mouse button is pressed, background color should be red. If mouse button is released up, background color should be yellow.

```
<html>
<head>
<title>Changing background colour </title>
<script type="text/javascript">
function red()
{
document.body.style.backgroundColor="red"
}
function yellow()
{
   document.body.style.backgroundColor="yellow"
}
</script>
</head>
<body onMouseDown="red();" onMouseUp="yellow();">
</body>
</html>
```

Output

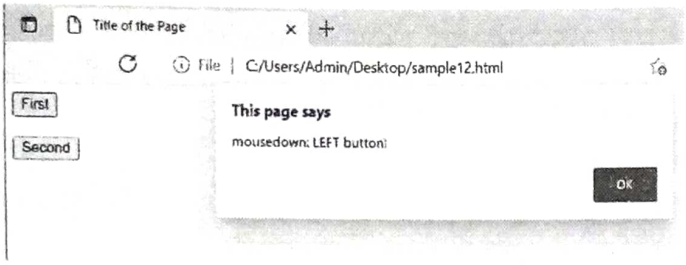


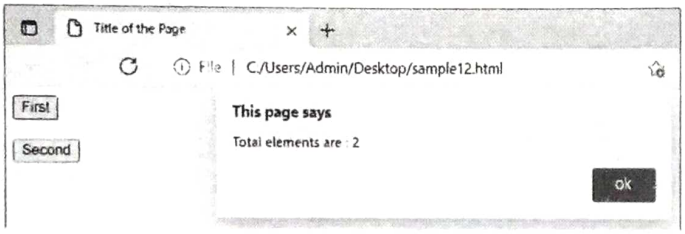
Program 2.9.4 : Write using JavaScript how to know which mouse button was clicked, number of elements in form, and write hello world.

```
<html>
<head>
<title>Title of the Page</title>
</head>
<body>
<script language="JavaScript">
var sTestEventType='mousedown';
function handleMouseEvent(e) {
 var evt = (e==null ? event:e);
 var clickType = 'LEFT';
 if (evt.type!=sTestEventType) return true;
 if (evt.which)
 {
    if (evt.which==3) clickType='RIGHT';
    if (evt.which==2) clickType='MIDDLE';
 }
    alert(evt.type+': '+clickType+' button!');
    alert("Total elements are : "+document.frm.length);
    alert("Hello World");
 return true;
}
```

```
    document.onmousedown = handleMouseEvent;
    document.onmouseup   = handleMouseEvent;
    document.onclick     = handleMouseEvent;
</script>
<form name="frm">
<input type="button" value="First"><br><br>
<input type="button" value="Second">
</form>
</body>
</html>
```

Output





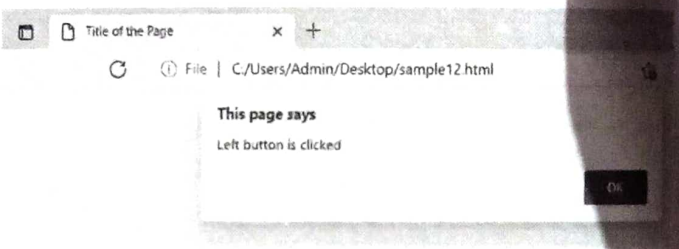
Program 2.9.5 : Write JavaScript that handles following mouse event.

(a) If mouse left button pressed on browser it displayed message "Left Clicked".

(b) If mouse right button pressed on browser it displayed message "Right Clicked".

```
<html>
<head>
<title>Title of the Page</title>
</head>
<body>
<script language="JavaScript">
var sTestEventType='mousedown';
function handleMouseEvent(e)
{
    var evt = (e==null ? event:e);
    var clickType = 'LEFT';
    if (evt.type!=sTestEventType) return true;
  if (evt.which)
  {
    if (evt.which==3) clickType='RIGHT';
  }
if(clickType == 'LEFT')
    alert("Left button is clicked");
    else if(clickType == 'RIGHT')
    alert("Right button is clicked");
    return true;
}
document.onmousedown = handleMouseEvent;
document.onmouseup   = handleMouseEvent;
document.onclick     = handleMouseEvent;
</script>
</body>
</html>
```

Output



Program 2.9.6 : If button named "red" is clicked, background should change to red and If button named "green" is clicked, background should change to green.

```
<html>
<head>
<title>Changing background colour </title>
<script type="text/javascript">
function red()
{
    document.body.style.backgroundColor="red"
}
function green()
{
    document.body.style.backgroundColor="green"
}
</script>
</head>
</head>
<body>
<form name="frm">
<input type="button" value="Red"
onClick="red();"><br><br>
<input type="button" value="Green" onClick="green();">
```

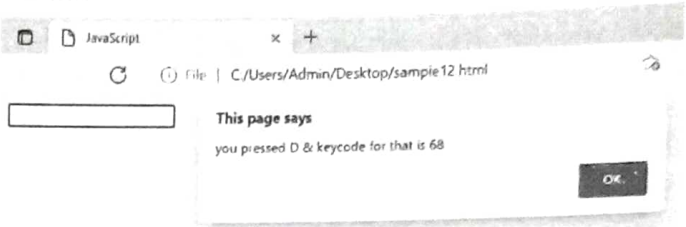
```
</form>
</body>
</html>
```

Output

Program 2.9.7 : Write a JavaScript that handles following mouse events. Add necessary elements. (i) JavaScript gives the key code for the key pressed. (ii) If the key pressed is "a","e","i","o","u", the script should announce that vowel is pressed. (iii) When the key is released background should change to blue.

```
<html>
<head>
<title>JavaScript</title>
<script type="text/javascript">
function myKeyPress(e)
{
var keynum;
if(window.event) // IE
{
   keynum = e.keyCode;
}
else if(e.which){ // Netscape/Firefox/Opera
keynum = e.which;
}
var char1=String.fromCharCode(keynum);
//for vovel
if(char1=='a' || char1=="e" || char1=='i' || char1=='o'
|| char1=='u')
{
   alert("you have pressed vovel");
}
alert("you pressed "+char1 +" & keycode for that is "+
keynum);
}
function changebgcolor()
{
document.getElementById("txt1").style.backgroundColor =
"blue";
}
</script>
</head>
<body>
<form>
<input type="text" id="txt1" onkeyup="changebgcolor()"
onkeypress="myKeyPress(event)"/>
</form>
</body>
</html>
```

Output



Program 2.9.8 : Design HTML form which include two fields username and password. Write JavaScript code to show and hide password.

```
<html>
<body>
<Script>
(function() {

   var PasswordToggler = function( element, field ) {
      this.element = element;
      this.field = field;

      this.toggle();
   };
   PasswordToggler.prototype = {
      toggle: function() {
         var self = this;
         self.element.addEventListener( "change", function()
{
            if( self.element.checked ) {
               self.field.setAttribute( "type", "text" );
            } else {
            self.field.setAttribute( "type", "password" );
            }
      }, false);
      }
   };
   document.addEventListener( "DOMContentLoaded",
function() {
   var checkbox = document.querySelector( "#show-hide" ),
   pwd = document.querySelector( "#pwd" ),
         form = document.querySelector( "#login" );

   form.addEventListener( "submit", function( e ) {
```

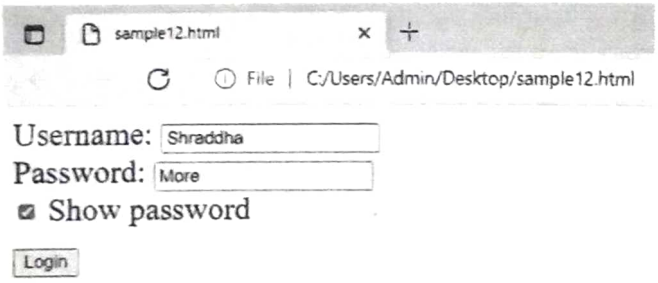
```
    e.preventDefault();
    }, false);

    var toggler = new PasswordToggler( checkbox, pwd );

    });

})();
</Script>
<form action="" method="post" id="login">
    <font size=5><div>
Username: <input type="text" id="name" name="name"
/><br>
Password: <input type="password" id="pwd" name="pwd"
/>
    </div>
    <div>
        <input type="checkbox" id="show-hide"
        name="show-hide" value="" />
        <label for="show-hide">Show password</label>
    </div>
</font>
    <p><input type="submit" value="Login" /></p>
</form>
</body>
</html>
```

Output



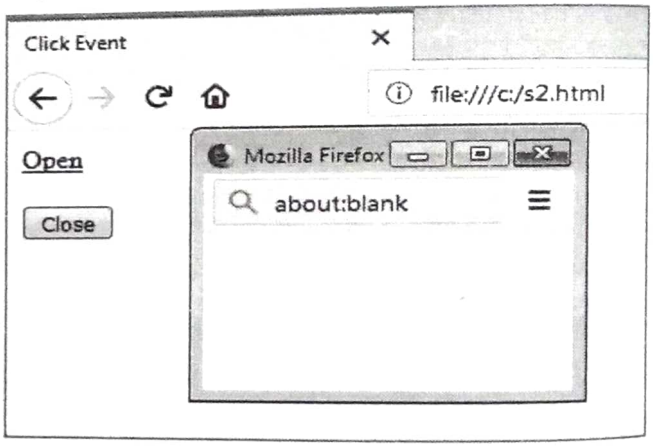
Program 2.9.9 : Write a code in JavaScript to open a window when a link on a page is clicked. The new window open is closed by placing a button on the window and writing JavaScript code on the onClick event of the button.

```
<html>
<head>
<title>
Click Event
</title>
</head>
<script language="JavaScript">
function openWin() {
   myWindow = window.open("", "myWindow", "width=200,
height=100");
}

function closeWin() {
   myWindow.close();
}
</script>

<form name="frm">
<a href="javascript:openWin()">Open</a>
<br><br>
<input type="button" value="Close"
onClick="closeWin();"></input>
</form>
</html>
```

Output



2.10 ADVANCED JAVASCRIPT: JSON

GQ. What is JSON?

- JSON stands for **J**avaScript **O**bject **N**otation.
- JSON is a text format which is used for storing and transporting data.
- JSON originated from JavaScript.
- JSON is a lightweight data-interchange format
- JSON is plain text written in JavaScript object notation
- JSON is "self-describing" and easy to understand.
- JSON is easy to read and write than XML.
- JSON is language independent.
- JSON supports array, object, string, number and values.

Why do we use JSON?

JSON is a lightweight and easy-to-use when compared to other open data interchange options. Following are the advantages of JSON:

(1) **Less Verbose :** In contrast to XML, JSON follows a compact style to improve its users' readability. While working with a complex system, JSON tends to make substantial enhancements.

(2) **Faster :** The JSON parsing process is faster than that of the XML because the DOM manipulation library in XML requires extra memory for handling large XML files. However, JSON requires less data that ultimately results in reducing the cost and increasing the parsing speed.

(3) **Readable :** The JSON structure is easily readable and straightforward. Regardless of the programming language that you are using, you can easily map the domain objects.

(4) **Structured Data :** In JSON, a map data structure is used, whereas XML follows a tree structure. The key-value pairs limit the task but facilitate the predictive and easily understandable model.

Features of JSON

1. Simplicity
2. Openness
3. Self-Describing
4. Internationalization
5. Extensibility
6. Interoperability

JSON Vs XML

Both XML and JSON are widely used today. They are used as data interchange formats and both have been adopted by applications as a way to store structured data. The following are the differences between the JSON and XML:

Sr. No.	JSON	XML
1.	JSON stands for javascript object notation.	XML stands for an extensible markup language.
2.	The extension of json file is .json.	The extension of xml file is .xml.
3.	The internet media type is application/json.	The internet media type is application/xml or text/xml.
4.	The type of format in JSON is data interchange.	The type of format in XML is a markup language.
5.	It is extended from JavaScript.	It is extended from SGML.
6.	It is open source means that we do not have to pay anything to use JSON.	It is also open source.
7.	The object created in JSON has some type.	XML data does not have any type.
8.	The data types supported by JSON are strings, numbers, Booleans, null, array.	XML data is in a string format.
9.	It does not have any capacity to display the data.	XML is a markup language, so it has the capacity to display the content.
10.	JSON has no tags.	XML data is represented in tags, i.e., start tag and end tag.
11.	JSON is quicker to read and write.	XML file takes time to read and write because the learning curve is higher.
12.	JSON can use arrays to represent the data.	XML does not contain the concept of arrays.
13.	It can be parsed by a standard javascript function. It has to be parsed before use.	XML data which is used to interchange the data, must be parsed with respective to their programming language to use that.
14.	It can be easily parsed and little bit code is required to parse the data.	It is difficult to parse.
15.	File size is smaller as compared to XML.	File size is larger.
16.	JSON is data-oriented.	XML is document-oriented.
17.	It is less secure than XML.	It is more secure than JSON.

Syntax Rules

- The JSON syntax is a subset of the JavaScript syntax. The syntax is derived from JavaScript object notation syntax and consist of the following rules:
  - Data is written in name or value pairs.
  - Data is separated by commas.
  - Curly braces hold the objects.
  - Square brackets hold the arrays.
- The JSON format is very similar to JavaScript objects. Here, keys must be strings, written with double quotes such as:

```
{ "name": "abc" }
```

JSON Example

```
employees.json
{"employees":[
  {"name":"Vivian", "email":"vivianl@gmail.com"},
  {"name":"Shraddha", "email":"shraddham@gmail.com"},
{"name":"Ronald", "email":"ronaldl@gmail.com"}
]}
```

The XML representation of above JSON example is given below.

```
employees.xml
<employees>
  <employee>
    <name> Vivian </name>
    <email> vivianl@gmail.com </email>
  </employee>
  <employee>
    <name> Shraddha </name>
    <email> shraddham@gmail.com </email>
  </employee>
  <employee>
    <name> Ronald </name>
    <email> ronaldl@gmail.com </email>
  </employee>
</employees>
```

2.11 JSON DATA TYPES

In JSON, values must be one of the following data types:

1.	String	2.	Number
3.	Object	4.	Arrays
5.	Boolean	6.	Null

1. JSON Strings

Strings in JSON must be written in double quotes.

Example

```
{"name":"Vidya"}
```

2. JSON Numbers

Numbers in JSON must be an integer or a floating point.

Example

```
{"age":22}
```

3. JSON Objects

Values in JSON can be objects.

Example

```
{
"employee":{"name":"Prajakta", "age":30, "city":"Thane}
}
```

4. JSON Arrays

Values in JSON can be arrays.

Example

```
{
"employees":["Piyu", "Khyati", "Inu", "Praju", "Sanyu",
"More",]
}
```

5. JSON Booleans

Values in JSON can be true/false.

Example

```
{"sale":true}
```

6. JSON null

Values in JSON can be null.

Example

```
{"middlename":null}
```

2.12 JSON OBJECT

- JSON object holds key/value pair. Each key is represented as a string in JSON and value can be of any type.
- The keys and values are separated by colon. Each key/value pair is separated by comma.

  The curly brace { } represents JSON object.

Let's see an example of JSON object.

```
{
  "employee": {
    "name":     "Sai",
    "salary":   56000,
    "married":  true    }
}
```

In the above example, employee is an object in which "name", "salary" and "married" are the key. In this example, there are string, number and boolean value for the keys.

☞ JSON Object with Strings

The string value must be enclosed within double quote.

```
{
  "name":     "Vennu",
  "email":    "vennul@gmail.com"
}
```

☞ JSON Object with Numbers

JSON supports numbers in double precision floating-point format. The number can be digits (0-9), fractions (.33, .532 etc) and exponents (e, e+, e-,E, E+, E-).

```
{
"integer": 34,
"fraction": .2145,
"exponent": 6.61789e+0
}
```

☞ JSON Object with Booleans

JSON also supports boolean values true or false.

```
{
"first": true,
"second": false
}
```

☞ JSON Nested Object Example

A JSON object can have another object also. Let's see a simple example of JSON object having another object.

```
{
  "firstName": "Chetana",
  "lastName": "Murudkar",
  "age": 30,
  "address" : {
    "Address": "Virar",
    "city": "Mumbai",
    "state": "Maharashtra",
    "postalCode": "400140"      }
}
```

☞ JSON Array

JSON array represents ordered list of values. JSON array can store multiple values. It can store string, number, boolean or object in JSON array.

In JSON array, values must be separated by comma.

The [ ](square bracket) represents JSON array.

☞ JSON Array of Strings

Let's see an example of JSON arrays storing string values.

```
["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday"]
```

☞ JSON Array of Numbers

Let's see an example of JSON arrays storing number values.

```
[12, 34, 56, 43, 95]
```

☞ JSON Array of Booleans

Let's see an example of JSON arrays storing boolean values.

```
[true, true, false, false, true]
```

☞ JSON Array of Objects

Let's see a simple JSON array example having 4 objects.

```
{"employees":[
  {"name":"Chetana", "email":"gaurim@gmail.com",
"age":31},
 {"name":"Gauri", "email":"gaurim@gmail.com", "age":31},
  {"name":"Prajakta", "email":"prajul@gmail.com",
"age":32},
  {"name":"Vidya", "email":"vidyam@gmail.com", "age":32}
  {"name":"Shreedevi", "email":"shreek@gmail.com",
"age":31},
  {"name":"Yogita", "email":"yogin@gmail.com", "age":32},
  {"name":"Shraddha", "email":"shraddham@gmail.com",
"age":32},
]}
```

⏭ 2.13 JS ARROW FUNCTIONS

- Arrow function is one of the features introduced in the ES6 version of JavaScript.
- It allows you to create functions in a cleaner way compared to regular functions.

For example,

```
// function expression
let x = function(x, y) {
  return x * y;}
```

Above function can be written as using an arrow function.

```
// using arrow functions
let x = (x, y) => x * y;
```

The syntax of the arrow function is:

```
let myFunction = (arg1, arg2, ...argN) => {
   statement(s)
}
```

Here,

myFunction is the name of the function

arg1, arg2, ...argN are the function arguments

statement(s) is the function body

If the body has single statement or expression, you can write arrow function as:

```
let myFunction = (arg1, arg2, ...argN) => expression
```

Example 1: Arrow Function with No Argument

If a function doesn't take any argument, then you should use empty parentheses. For example,

```
let greet = () => console.log('Hello');
greet(); // Hello
```

Example 2 : Arrow Function with One Argument

If a function has only one argument, you can omit the parentheses. For example,

```
let greet = x => console.log(x);
greet('Hello'); // Hello
```

Example 3 : Arrow Function as an Expression

You can also dynamically create a function and use it as an expression. For example,

```
let age = 5;

let welcome = (age < 18) ?
  () => console.log('Baby') :
  () => console.log('Adult');

welcome(); // Baby
```

Example 4 : Multiline Arrow Functions

If a function body has multiple statements, you need to put them inside curly brackets { }. For example,

```
let sum = (a, b) => {
   let result = a + b;
   return result;
}

let result1 = sum(5,7);
console.log(result1); // 12
```

☞ this with Arrow Function

- Inside a regular function, this keyword refers to the function where it is called.
- However, this is not associated with arrow functions. Arrow function does not have its own this. So whenever you call this, it refers to its parent scope. For example,
- Inside a regular function

```
function Person() {
   this.name = 'Jack',
   this.age = 25,
   this.sayName = function () {

      // this is accessible
      console.log(this.age);

      function innerFunc() {

         // this refers to the global object
         console.log(this.age);
         console.log(this);
      }
      innerFunc();

   }
}
let x = new Person();
x.sayName();
Output
25
undefined
Window {}
```

⏭ 2.14 JS CALLBACK FUNCTIONS

In JavaScript, you can also pass a function as an argument to a function. This function that is passed as an argument inside of another function is called a callback function.

For example,

```
// function
function greet(name, callback) {
  console.log('Hi' + ' ' + name);
  callback();
}

// callback function
function callMe() {
  console.log('I am callback function');
}

// passing function as an argument
greet('Peter', callMe);
```

- In the above program, a string value is passed as an argument to the greet() function.
- In JavaScript, you can also pass a function as an argument to a function. This function that is passed as an argument inside of another function is called a callback function. For example,

```
// function
function greet(name, callback) {
  console.log('Hi' + ' ' + name);
  callback();
}

// callback function
function callMe() {
  console.log('I am callback function');
}

// passing function as an argument
greet('Peter', callMe);
```

Output

```
Hi Peter
I am callback function
```

- In the above program, there are two functions. While calling the greet() function, two arguments (a string value and a function) are passed.
- The callMe() function is a callback function.

☞ Benefit of Callback Function

- The benefit of using a callback function is that you can wait for the result of a previous function call and then execute another function call.
- In this example, we are going to use the setTimeout() method to mimic the program that takes time to execute, such as data coming from the server.

Example: Program with setTimeout()

```
// program that shows the delay in execution

function greet() {
  console.log('Hello world');
}

function sayName(name) {
  console.log('Hello' + ' ' + name);
}

// calling the function
setTimeout(greet, 2000);
sayName('John');
```

Output

```
Hello John
Hello world
```

As you know, the setTimeout() method executes a block of code after the specified time.

- Here, the greet() function is called after 2000 milliseconds (2 seconds). During this wait, the sayName('John'); is executed. That is why HelloJohn is printed before Hello world.
- The above code is executed asynchronously (the second function; sayName() does not wait for the first function; greet() to complete).

Example : Using a Callback Function

In the above example, the second function does not wait for the first function to be complete. However, if you want to wait for the result of the previous function call before the next statement is executed, you can use a callback function. For example,

```
// Callback Function Example
function greet(name, myFunction) {
  console.log('Hello world');

  // callback function
  // executed only after the greet() is executed
  myFunction(name);
}

// callback function
function sayName(name) {
  console.log('Hello' + ' ' + name);
}

// calling the function after 2 seconds
setTimeout(greet, 2000, 'John', sayName);
```

Output

```
Hello world
Hello John
```

- In the above program, the code is executed synchronously. The sayName() function is passed as an argument to the greet() function.
- The setTimeout() method executes the greet()function only after 2 seconds. However, the sayName() function waits for the execution of the greet() function.

2.15 JAVASCRIPT PROMISE

- In JavaScript, a promise is a good way to handle **asynchronous** operations. It is used to find out if the asynchronous operation is successfully completed or not.
- A promise may have one of three states.
  - Pending
  - Fulfilled
  - Rejected
- A promise starts in a pending state. That means the process is not complete. If the operation is successful, the process ends in a fulfilled state. And, if an error occurs, the process ends in a rejected state.
- For example, when you request data from the server by using a promise, it will be in a pending state. When the data arrives successfully, it will be in a fulfilled state. If an error occurs, then it will be in a rejected state.

Create a Promise

- To create a promise object, we use the Promise() constructor.

```
let promise = new Promise(function(resolve, reject){
    //do something
});
```

- The Promise() constructor takes a function as an argument. The function also accepts two functions resolve() and reject().
- If the promise returns successfully, the resolve() function is called. And, if an error occurs, the reject() function is called.
- Let's suppose that the program below is an asynchronous program. Then the program can be handled by using a promise.

Example 1: Program with a Promise

```
const count = true;

let countValue = new Promise(function (resolve, reject) {
  if (count) {
    resolve("There is a count value.");
  } else {
    reject("There is no count value");
  }
});

console.log(countValue);
```

Output

```
Promise {<resolved>: "There is a count value."}
```

- In the above program, a Promise object is created that takes two functions: resolve() and reject(). resolve() is used if the process is successful and reject() is used when an error occurs in the promise.
- The promise is resolved if the value of count is true.

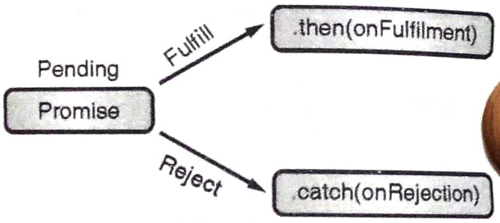


Fig. 2.15.1 : Working of JavaScript Promises

JavaScript Promise Chaining

- Promises are useful when you have to handle more than one asynchronous task, one after another. For that, we use promise chaining.
- You can perform an operation after a promise is resolved using methods then(), catch() and finally().

JavaScript then() method

- The then() method is used with the callback when the promise is successfully fulfilled or resolved.
- The syntax of then() method is:

  promiseObject.then(onFulfilled, onRejected);

Example 2 : Chaining the Promise with then()

```
// returns a promise

let countValue = new Promise(function (resolve, reject) {
 resolve("Promise resolved");
});

// executes when promise is resolved successfully

countValue
 .then(function successValue(result) {
  console.log(result);
 })

 .then(function successValue1() {
  console.log("You can call multiple functions this way.");
 });
```

Output

```
Promise resolved
You can call multiple functions this way.
```

- In the above program, the then() method is used to chain the functions to the promise. The then() method is called when the promise is resolved successfully.
- You can chain multiple then() methods with the promise.

☞ JavaScript catch() method

- The catch() method is used with the callback when the promise is rejected or if an error occurs. For example,

```
// returns a promise
let countValue = new Promise(function (resolve, reject) {
  reject('Promise rejected');
});

// executes when promise is resolved successfully
countValue.then(
  function successValue(result) {
    console.log(result);
  },
)

// executes if there is an error
.catch(
  function errorValue(result) {
    console.log(result);
  }
);
```

Output

```
Promise rejected
```

- In the above program, the promise is rejected. And the catch() method is used with a promise to handle the error.

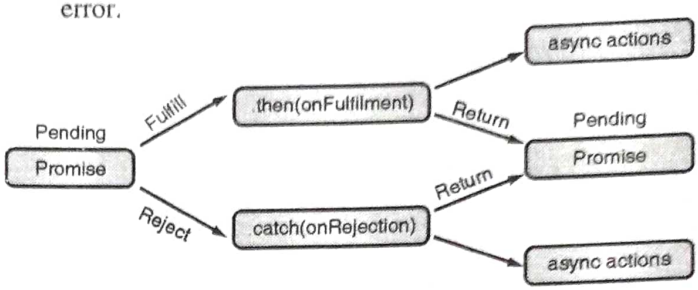


Fig. 2.15.2

☞ Working of JavaScript promise chaining

JavaScript Promise Versus Callback

- Promises are similar to callback functions in a sense that they both can be used to handle asynchronous tasks.
- JavaScript callback functions can also be used to perform synchronous tasks.
- Their differences can be summarized in the following points:

☞ JavaScript Promise

- The syntax is user-friendly and easy to read.
- Error handling is easier to manage.

Example

```
api().then(function(result) {
   return api2() ;
}).then(function(result2) {
   return api3();
}).then(function(result3) {
   // do work
}).catch(function(error) {
   //handle any error that may occur before this point
});
```

☞ JavaScript Callback

- The syntax is difficult to understand.
- Error handling may be hard to manage.

Example

```
api(function(result){
  api2(function(result2){
```

```
    api3(function(result3){
        // do work
        if(error) {
            // do something
        }
        else {
            // do something
        }
    });
  });
});
```

JavaScript finally() method

- You can also use the finally() method with promises. The finally() method gets executed when the promise is either resolved successfully or rejected. For example,

```
// returns a promise
let countValue = new Promise(function (resolve, reject) {
  // could be resolved or rejected
  resolve('Promise resolved');
});
// add other blocks of code
countValue.finally(
  function greet() {
    console.log('This code is executed.');
  }
);
```

Output

```
This code is executed.
```

JavaScript Promise Methods

There are various methods available to the Promise object.

Method	Description
all(iterable)	Waits for all promises to be resolved or anyone to be rejected
allSettled(iterable)	Waits until all promises are either resolved or rejected
any(iterable)	Returns the promise value as soon as any one of the promises is fulfilled
race(iterable)	Wait until any of the promises is resolved or rejected
reject(reason)	Returns a new Promise object that is rejected for the given reason
resolve(value)	Returns a new Promise object that is resolved with the given value
catch()	Appends the rejection handler callback
then()	Appends the resolved handler callback
finally()	Appends a handler to the promise

2.16 JS ASYNC-AWAIT FUNCTIONS

- We use the async keyword with a function to represent that the function is an asynchronous function. The async function returns a promise.
- The syntax of async function is:

```
async function name(parameter1, parameter2, ...paramaterN)
{
  // statements
}
```

Here,

- name - name of the function
- parameters - parameters that are passed to the function

Example of async Function

```
// async function example

async function f() {
  console.log('Async function.');
  return Promise.resolve(1);
}

f();
```

Output

```
Async function.
```

- In the above program, the async keyword is used before the function to represent that the function is asynchronous.
- Since this function returns a promise, you can use the chaining method then() like this:

```
async function f() {
  console.log('Async function.');
  return Promise.resolve(1);
}

f().then(function(result) {
  console.log(result)
});
```

Output

```
Async function
1
```

- In the above program, the f() function is resolved and the then() method gets executed.

☞ JavaScript await Keyword

- The await keyword is used inside the async function to wait for the asynchronous operation.
- The syntax to use await is:

```
let result = await promise;
```

- The use of await pauses the async function until the promise returns a result (resolve or reject) value. For example,

```
// a promise
let promise = new Promise(function (resolve, reject) {
  setTimeout(function () {
  resolve('Promise resolved')}, 4000);
});

// async function
async function asyncFunc() {

  // wait until the promise resolves
  let result = await promise;

  console.log(result);
  console.log('hello');
}

// calling the async function
asyncFunc();
```

Output

```
Promise resolved
hello
```

- In the above program, a Promise object is created and it gets resolved after 4000 milliseconds. Here, the asyncFunc() function is written using the async function.
- The await keyword waits for the promise to be complete (resolve or reject).

```
let result = await promise;
```

- Hence, hello is displayed only after promise value is available to the result variable.
- In the above program, if await is not used, hello is displayed before Promise resolved.

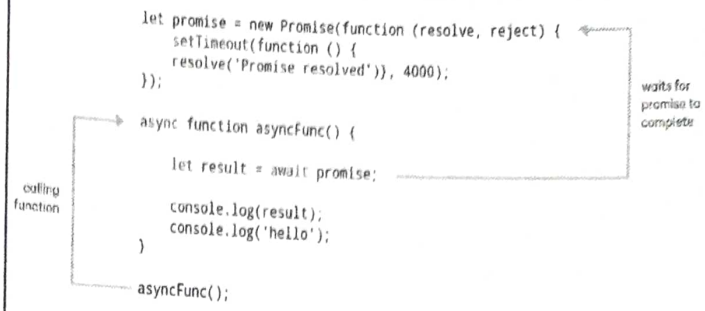


Fig. 2.16.1

☞ Working of async/await function

- The async function allows the asynchronous method to be executed in a seemingly synchronous way.
- Though the operation is asynchronous, it seems that the operation is executed in synchronous manner.
- This can be useful if there are multiple promises in the program. For example,

```
let promise1;
let promise2;
let promise3;

async function asyncFunc() {
  let result1 = await promise1;
  let result2 = await promise2;
  let result3 = await promise3;

  console.log(result1);
  console.log(result1);
  console.log(result1);
}
```

- In the above program, await waits for each promise to be complete.

☞ Error Handling

- While using the async function, you write the code in a synchronous manner. And you can also use the catch() method to catch the error. For example,

```
asyncFunc().catch(
  // catch error and do something
)
```

- The other way you can handle an error is by using try/catch block. For example,

```
// a promise
let promise = new Promise(function (resolve, reject) {
  setTimeout(function () {
```

```
  resolve('Promise resolved')}, 4000);
});

// async function
async function asyncFunc() {
  try {
    // wait until the promise resolves
    let result = await promise;

    console.log(result);
  }
  catch(error) {
    console.log(error);
  }
}

// calling the async function
asyncFunc(); // Promise resolved
```

- In the above program, we have used try/catch block to handle the errors. If the program runs successfully, it will go to the try block. And if the program throws an error, it will go to the catch block.

☞ Benefits of Using async Function

1. The code is more readable than using a callback or a promise.
2. Error handling is simpler.
3. Debugging is easier.

⏭ 2.17 JS ERROR HANDLING

- JavaScript is a loosely-typed language. It does not give compile-time errors. So sometimes, you will get a runtime error for accessing an undefined variable or calling undefined function etc.
- JavaScript provides error-handling mechanism to catch runtime errors using try-catch-finally block, similar to other languages like Java or C#.

Syntax

```
try
{
  // code that may throw an error
}
catch(ex)
{
  // code to be executed if an error occurs
}
finally{
  // code to be executed regardless of an error occurs or not
}
```

- **try:** wrap suspicious code that may throw an error in try block.
- **catch:** write code to do something in catch block when an error occurs. The catch block can have parameters that will give you error information. Generally catch block is used to log an error or display specific messages to the user.
- **finally:** code in the finally block will always be executed regardless of the occurrence of an error. The finally block can be used to complete the remaining task or reset variables that might have changed before error occurred in try block.

Example of Error Handling in JS

```
try
{
  var result = Sum(10, 20); // Sum is not defined yet
}
catch(ex)
{
  document.getElementById("errorMessage").innerHTML =
ex;
}
```

Output

Demo: Error Handling

ReferenceError: Sum is not defined

- In the above example, we are calling function Sum, which is not defined yet. So, try block will throw an error which will be handled by catch block. Ex includes error message that can be displayed.
- The finally block executes regardless of whatever happens.

Example of finally Block

```
try
{
  var result = Sum(10, 20); // Sum is not defined yet
}
catch(ex)
{
  document.getElementById("errorMessage").innerHTML =
ex;
```

```
}
finally{
   document.getElementById("message").innerHTML =
"finally block executed";
}
```

Output

Demo: Error Handling

Error: ReferenceError: Sum is not defined

finally block executed

- **throw :** Use throw keyword to raise a custom error.

Example of throw Error

```
try
{
  throw "Error occurred";
}
catch(ex)
{
  alert(ex);
}
```

Output

Demo: throw

Error occurred

- You can use JavaScript object for more information about an error.

Example of throw error with error info

```
try
{
  throw {
    number: 101,
    message: "Error occurred"
  };
}
catch (ex) {
  alert(ex.number + "- " + ex.message);
}
```

Output

Demo: throw

101- Error occurred

2.18 AJAX

GQ. What is AJAX?

- AJAX is an acronym for Asynchronous JavaScript and XML.
- It is not a programming language. It is a technology for developing better, faster and interactive web applications using HTML, CSS, JavaScript and XML.
  - **HTML :** Hypertext Markup Language (HTML) is used for defining the structure of a web application.
  - **CSS :** Cascading Style Sheet (CSS) is used to provide look and style to a web application.
  - **JavaScript :** JavaScript is used for making a web application interactive, interesting and user friendly.
  - **XML :** Extensible Markup Language (XML) is a format to store and transport data from the web server.
- It is a group of inter-related technologies like JavaScript, DOM, XML, HTML/XHTML, CSS, XMLHttpRequest etc.
- AJAX allows you to send and receive data asynchronously without reloading the web page.
- AJAX allows you to send only important information to the server not the entire page. So only valuable data from the client side is routed to the server side. It makes your application interactive and faster.

2.18.1 Asynchronous in AJAX

GQ. What is the meaning of Asynchronous in AJAX?

- Asynchronous means that the web application could send and receive data from the web server without refreshing the page.
- This background process of sending and receiving data from the server along with updating different sections of a web page defines Asynchronous property/feature of AJAX.

2.18.2 Benefits of Ajax

1. Using AJAX you can create better, faster, and more user-friendly web applications.
2. Ajax is based on JavaScript, CSS, HTML and XML etc. So, you can easily learn.
3. Ajax behaviour and works is like a desktop application. So, Ajax use for creating a rich web application.

2.18.3 Working of AJAX

It creates more interactive techniques for faster and more efficient web applications by using JavaScript, XML, CSS, and HTML. For various web applications, Ajax uses various techniques like :

- In Ajax, when the user needs to create content, XHTML is used while CSS is used for presenting the user request purpose, the document object model (DOM) and javascript both will be used to display the content dynamically.
- By using synchronous methods in web applications, information can be transmitted and received effectively, for example, when you fill a form and submit it. You will be automatically directed to the new server with that page information.
- After hitting the submit button in the background, javascript sends a request, and with the response generated, it will update to the current screen. In this process, the user will be unaware of the background XML code requests.
- XML is used as a format to generate and receive the server data in any format.
- While most of the web browsers are dependent on web server technology, it is independent of web server software.

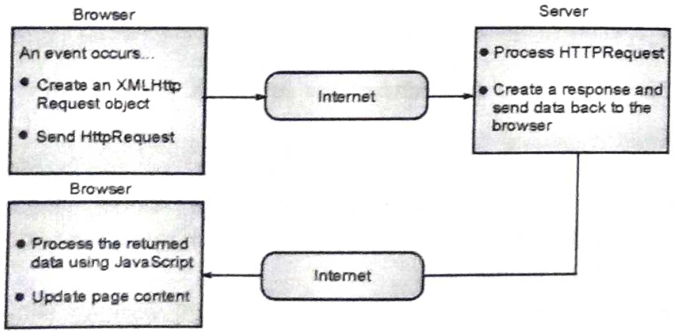


Fig. 2.18.1 : AJAX Working

Where to use AJAX?

Below mentioned are the places where Ajax is used.

- **Login forms :** Eg: user can type their login credentials in the original page form, their software will send a request to the server to logged in, and the page will be updated as needed.
- **Auto-complete :** When you run a query in the Google search engine with the help of auto fill settings, suggestions will be shown in drop down below.
- **Rating and Voting :** The voting can decide the site's main content in web pages like Digg and Reddit by bookmarking them.
- **Updating with user content :** When a user posts a tweet, it will be added to their feed, and everything is updated.
- Now it is used by tweeter to run their trending topics page
  - Form submission and validation.
  - It makes web applications quicker, and the numbers of responses are also reduced.
  - Light-boxes are used nowadays instead of pop-ups.
  - Using Ajax with flash application.

Examples of Ajax Application

Given below are the lists of web applications that commonly use Ajax

(1) Google suggests that auto-complete options will be offered while typing when a user enters the search query in the Google search engine. Suggestions given by Google can be navigated by using operational keys.

(2) Yahoo maps are easier while operating, and user experiences more fun. This Map uses Ajax to drag the entire map with the mouse without using buttons that will be at ease to the user.

(3) Google maps are general applications that use Ajax. This is a real-time application in which the user can manipulate the data and change the view settings. Ajax directly works on a web browser without any plugin installations. Firstly, only Microsoft internet explorer used Ajax, but due to its reliability, more web applications like chrome, Mozilla... etc. using this.

2.18.4 Advantages and Disadvantages of Ajax

Advantages of Ajax

1. Reduces the server traffic and increases the speed
2. It is responsive, and the time taken is also less
3. Form validation
4. Bandwidth usage can be reduced
5. Asynchronous calls can be made; this reduces the time for data arrival.

Disadvantages of Ajax

1. Open-source
2. ActiveX request is created only in internet explorer and a newly created web browser.
3. For security reason, you can only access information from the web host that serves pages. Fetching information from other servers is not possible with Ajax.

2.19 AJAX DESIGN BASICS

- AJAX uses the **XMLHttpRequest** object to communicate with the server. The Fig. 4.2.1 shows the flow of AJAX.
- An important role is played by the XMLHttpRequest object in AJAX processing. Request is sent by the user through UI (User interface) and a JavaScript call goes to XMLHttpRequest object.
- Using the XMLHttpRequest object, the HTTP Request sent to the server.
- Now the server interacts with the database using any of the server side scripting language like JSP, PHP, Servlet, or ASP.net.

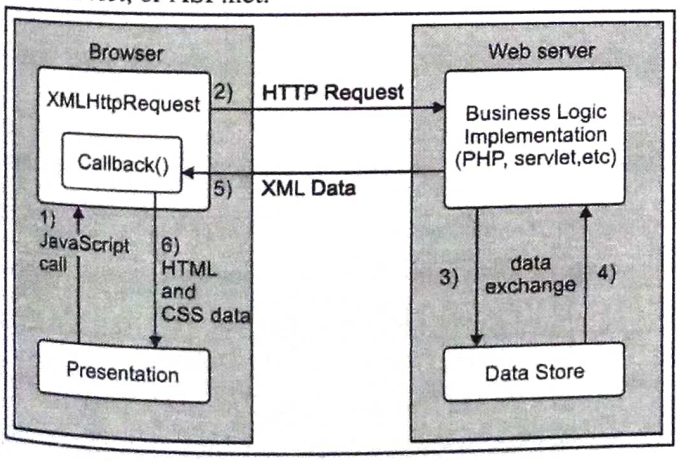


Fig. 2.19.1 : AJAX Communication

- Data is retrieved from the database as per the requirement. Server sends the data in the form of XML or JSON through the XMLHttpRequest callback function.
- Then the data is displayed using HTML and CSS on the browser.

2.19.1 AJAX Processing Steps and Ajax Script

GQ. Write steps to process AJAX.

Steps to process AJAX

▶ **Step 1 :** An event by the client is fired.

When an event is fired, a JavaScript method gets called.

Example

validateID() JavaScript method is mapped as an event handler to an *onKeyPress* event on input form field. The id here is *"myid"*

```
<input type="text" size="25" id="myid" name="id"
onkeyPress="validateID();">.
```

▶ **Step 2 :** An XMLHttpRequest object is created.

```
var ajaxReqObj;
function ajaxFun(){
  try{
    ajaxReqObj = new XMLHttpRequest();
  }
catch (e)
{
    try{

      ajaxReqObj = new
activeXObject("Msxml2.XMLHTTP");
    }
catch (e)
{
      try{
        ajaxReqObj = new
ActiveXObject("Microsoft.XMLHTTP");
      }catch (e){
        // Exception occurs
        alert("Your browser broke!");
        return false;
      }
    }
```

This variable makes AJAX

Opera 8.0+, Firefox, Safari

Internet Explorer

```
  }
}
```

▶ **Step 3 :** The XMLHttpRequest object is configured.

In this step, we will write a function that will be triggered by the client event and a callback function processReqFun() will be registered.

```
function validateID() {
  ajaxFun();

  ajaxReqObj.onreadystatechange = processReqFun;
                                  processReqFun() is callback
  if (!target) target = document.getElementById("myid");
  var url = "validate?id=" + escape(target.value);
  ajaxReqObj.open("GET", url, true);
  ajaxReqObj.send(null);
}
```

▶ **Step 4 :** An asynchronous request is made by the XMLHttpRequest to the Webserver.

- Here we will make request to webserver. This request can be made using XMLHttpRequest object *ajaxRequest*.

```
function validateID() {
  ajaxFun();

  ajaxReqObj.onreadystatechange = processReqfun;
                                  processReqFun() is callback function
  if (!target) target = document.getElementById("myid");
  var url = "validate?id=" + escape(target.value);

  ajaxReqObj.open("GET", url, true);
  ajaxReqObj.send(null);          Internet Explorer
}
```

- Now consider in the textfield, Subhash as userid, then in the request, the url will be "validate?id=Subhash".

▶ **Step 5 :** The Webserver returns the result in XML format.

- The server side script can be implemented in any server side language using the following logic.
- Accept request from the client side.
- Parse the input given by the client.
- Execute necessary processing.
- Send the output to the client in response.
- We will use the Servlet for this purpose.

```
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException,
ServletException
{
  String ID = request.getParameter("id");

  if ((ID != null) && !accounts containsKey(ID.trim()))
  {
    response.setContentType("text/xml");
    response.setHeader("Cache Control", "no cache");
    response.getWriter().write("true");
  }
  else
  {
    response.setContentType("text/xml");
    response.setHeader("Cache Control", "no cache");
    response.getWriter().write("false");
  }
}
```

▶ **Step 6 :** The callback() function is called by the XMLHttpRequest object and processes the result.

- When the state of *XMLHttpRequest* object is changed to the *readyState*, then the XMLHttpRequest object was configured to invoke (call) the processReqFun() method.
- The result returned by the server is received by this function and required processing will be done. Now see in the given example, this function will set message depending upon the webserver's returned value.

```
functionprocessReqFun()
{
if(req.readyState==4)
{
if(req.status==200)
{
varmsg=...;
...
}
```

▶ **Step 7 :** The HTML DOM is updated.

- Now this is the last step. Here the HTML document will be updated. Process will be as follows:
- UsingDOMAPI, the JavaScript receives a reference to any element in the web page.

```
document.getElementById("userIdMsg"),
// "userIdMsg " is the ID attribute of an element present in the
HTML document
```

- Using JavaScript, the attributes of element can be modified. The style properties can be modified or child elements can be added, modified or removed.

```
<scripttype="text/javascript">
<!--
functionsetMsgDOM(msg){
varuserMessageElement=document.getElementById("userIdM
sg");
varmsgText;
if(msg==="false")
{
userMessageElement.style.color="red";
msgText="Wrong UserId";
}
else
{
userMessageElement.style.color="green";
msgText="ValidUserId";
}
varmsgBody=document.createTextNode(msgText);

if(userMessageElement.childNodes[0])
{
```

If the messageBody element is already created then just replace it otherwise append the new element

```
userMessageElement.replaceChild(msgBody,userMessageEle
ment.childNodes[0]);
}
else
{
userMessageElement.appendChild(msgBody);
}
}
-->
</script>
<bodybgcolor="gray">
<divid="userIdMsg"><div>
</body>
```

2.20 CALL HTTP METHODS USING AJAX

- To make an HTTP call in Ajax, you need to initialize a new XMLHttpRequest() method, specify the URL endpoint and HTTP method (in this case GET).
- Finally, we use the open() method to tie the HTTP method and URL endpoint together and call the send() method to fire off the request.
- We log the HTTP response to the console by using the XMLHTTPRequest.onreadystatechange property which contains the event handler to be called when the readystatechanged event is fired.

☞ AJAX Request

- AJAX sends a request to a server by using open() and send() methods of the XMLHttpRequest object. Following code snippet illustrates how an AJAX request can be sent to the server.

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

- In the above code snippet, the **GET** is the request type, **ajax_info** is the location of the server type and **true** shows that the request is asynchronous.
- Following are the methods used to send a request in AJAX.

1. **open(method, url, async):** Specifies the type of request
   - method: the type of request: GET or POST
   - url: the server (file) location
   - async: true (asynchronous) or false (synchronous)
2. **send() :**Sends the request to the server (used for GET)
3. **send(string) :** Sends the request to the server (used for POST)

☞ AJAX Response

- The readyState property holds the status of the XMLHttpRequest.
- The onreadystatechange property defines a function to be executed when the readyState changes as shown in the following code snippet.

```
var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
```

- In the above code snippet, the response is decided by two properties of XMLHttpRequest: readyState and status. When the status is 200 and readyState is 4, the response is ready.
- The status property and the statusText property holds the status of the XMLHttpRequest object.

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found"
statusText	Returns the status-text (e.g. "OK" or "Not Found")

☞ Server Response Properties

There are two server response properties, which are as follows:

1. **responseText :** Returns the response from the server as a JavaScript string.
2. **responseXML:** Returns the response from the server as an XML DOM object.

☞ Server response Methods

There are two server response methods, which are as follows:

1. **getResponseHeader()** : Returns specific header information from the server resource.
2. **getAllResponseHeaders()** : Returns all the header information from the server resource.

The AJX request and response functions can be used with XMLHttpRequest API.

☞ XMLHttpRequest API

- AJAX comprises important API as XMLHttpRequest, which transfers XML data using HTTP to and from the web server. XMLHttpRequest is an API, which was initially built by Microsoft. It is used by various scripting languages such as JavaScript, VBScript, etc.
- The XMLHttpRequest object is used to exchange data with a web server in the background. This means that it is possible to update parts of a web page, without reloading the whole page.

☞ XMLHttpRequest Object Methods

Methods	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(method, url, async, user, psw)	Specifies the request method: the request type GET or POST url: the file location async: true (asynchronous) or false (synchronous) user: optional user name psw: optional password
send()	Sends the request to the server Used for GET requests
send(string)	Sends the request to the server.
setRequestHeader()	Adds a label/value pair to the header to be sent

Program 2.20.1 : Create AJAX Application Create a simple XMLHttpRequest, and retrieve data from a txt file.

☞ Create a file Ajax.html

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h1>The XMLHttpRequest Object</h1>
<button type="button" onclick="loadDoc()">Change
Content</button>
</div>

<script>
function loadDoc() {
 var xhttp = new XMLHttpRequest();
 xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
   document.getElementById("demo").innerHTML =
   this.responseText;
  }
```

```
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
}
</script>

</body>
</html>
```

☞ **Create a text file ajax_info.txt**

- AJAX is not a programming language.
- AJAX is a technique for accessing web servers from a web page.
- AJAX stands for Asynchronous JavaScript and XML.
- View Ajax.html on the browser as an output

Output

The XMLHttpRequest Object

Change Content

- Click the Change Content button. As soon as, the Change Content button is clicked, the text appears from the ajax_info.txt on the same page as shown in below figure.

AJAX is not a programming language.

AJAX is a technique for accessing web servers from a web page.

AJAX stands for Asynchronous JavaScript And XML.

- Thus the content is displayed on the same page, with the help of the XMLHttpRequest object. The contents are updated partially on the webpage instead of refreshing the whole page.

2.21 AJAX ERROR HANDLING

- Some times when we request to server we didn't get proper response (HTTP Status 200) as per our expectation. For ex. we can get
  (1) Internal Server Error
  (2) Network Connection Error
  (3) Page Not Found Error,
  (4) Access Denied Error.
- So if we are making call to server via http post request/simple http request directly then user will see these error in browser. But when we use AJAX service calls and if we don't properly handle the inevitable errors (like mentioned above) then our code fails and it didn't display anything and user get stucked. AJAX service calls are especially problematic. So to show proper error message to user we need to use proper error handling.
- Even when a client-side error is thrown, most users won't notice it and the ones who do notice won't know what the error means or what to do next. In fact, many developers don't notice client-side scripting errors that occur while they're debugging their own applications!
- For example : After logined in the system user open site link in new tab. Now user has opened two tab and if user logout from first tab and switch to second tab then in second tab whenever user will not refresh that page he will not be logged out.
- And without refresh the page if user fire an event that is AJAX based and if we didn't used proper error handling user will get stucked because he will not get proper message, but we must show him a proper message that "Sorry!! Your session has expired. Please login again.". Same may happen when session expired automatically after not using the system for few minutes.
- So to come over this problem we will use AJAX error function.

```
$.post(url,
        data,
function (response) {

}
).error(function (jqXHR, textStatus, errorThrown) {

            alert(formatErrorMessage(jqXHR, errorThrown));

    }

);
```

- So whenever we didn't get "HTTP Status 200 OK" from server and get any error this response will automatically will go to error function.
  (1) jqXHR : This variable will contain status code and we can access by "jqXHR.status".
  (2) textStatus : Response in text like Success.

(3) errorThrown : Some time error may be related to parse error, timeout error so these error message will come under this variable.

```
alert(formatErrorMessage(jqXHR, errorThrown));
```

(1) **alert** : Simple. using to show error message

(2) **formatErrorMessage** : We are using this function to parse this variable and get exact error and to show a proper message to user.

- Its defined below.

```
function formatErrorMessage(jqXHR, exception)
{
  if (jqXHR.status === 0) {
    return ('Not connected.\nPlease verify your network
connection.');
  } else if (jqXHR.status == 404) {
    return ('The requested page not found.');
  } else if (jqXHR.status == 401) {
    return ('Sorry!! You session has expired. Please login to
continue access.');
  } else if (jqXHR.status == 500) {
    return ('Internal Server Error.');
  } else if (exception === 'parsererror') {
    return ('Requested JSON parse failed.');
  } else if (exception === 'timeout') {
    return ('Time out error.');
  } else if (exception === 'abort') {
    return ('Ajax request aborted.');
  } else {
    return ('Unknown error occured. Please try again.');
  }}
```

2.22 JQUERY

Introduction to jQuery

- jQuery is basically a JavaScript library which is cross-platform and used to support the client side scripting of HTML. jQuery is open source and free application available on internet.
- jQuery provides various functionalities like navigation in web site, selecting DOM elements, managing events, creating different types of animations etc. jQuery provides a modular approach which simplifies the creation of dynamic content.
- jQuery was initially released by John Resig at BarCamp NYC in January 2006. Now a day jQuery is maintained by Timmy Willison and his team.

2.22.1 Features of jQuery

GQ. Write features of jQuery.

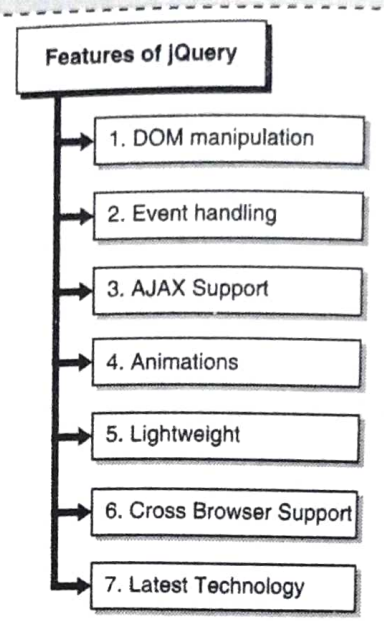


Fig. 2.22.1 : Features of jQuery

1. DOM manipulation

The DOM manipulation which includes selection of elements, traversing and modification in contents is easy with jQuery. jQuery make it easy with the help of cross-browser open source selector engine called **Sizzle**.

2. Event handling

Handling a wide variety of events like link click, mouse over etc. are effectively handled in the jQuery using event handlers.

3. AJAX Support

AJAX technology can be used with jQuery to develop responsive websites having a rich set of advanced features.

4. Animations

Number of built-in animation effects are provided by the jQuery.

5. Lightweight

The jQuery is a lightweight library which takes minimum time to load. It is about 19KB in size only.

6. Cross Browser Support

jQuery is supported by various latest browsers like Chrome, Mozila, IE, Opera, Safari erc.

7. Latest Technology

The advanced technologies like CSS3 selectors and basic XPath syntax are supported by jQuery.

2.22.2 Loading jQuery

1. **Local Installation :** The jQuery library can be loaded on our local machine. This library can be used in the HTML document.
2. **CDN Based Version :** It is also possible to add jQuery library in the HTML document directly from Content Delivery Network (CDN).

1. Local Installation

- URL to download jQuery :

https://jQuery.com/download/

- Now save this downloaded **jQuery-3.2.1.min.js** file in a directory of your website, e.g. /jQuery.
- Now you can include *jQuery* library in your HTML file as follows.

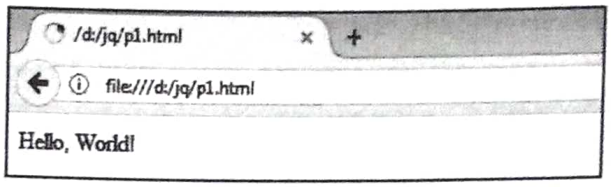
  As the webpage and jQuery library are in the same directory, there is no need to specify whole path in "src" attribute.

Program 2.22.1 : Write a simple program displaying hello world message using jQuery library.

Program displaying hello world message using jQuery library

```
<html>
 <head>
  <title>jQuery</title>
  <script type = "text/javascript" src = "jQuery-
3.2.1.min.js"></script>
  <script type = "text/javascript">
   $(document).ready(function()        {
    document.write("Hello, World!");        });
  </script>
 </head>
 <body>
 </body>
</html>
```

Output



2. CDN Based Version

- It is also possible to include jQuery library in the HTML document directly from Content Delivery Network (CDN). Google and Microsoft are the providers for the latest version. Here we have to set the value of *src* attribute as follows :

src : https://ajax.googleapis.com/ajax/libs/jQuery/3.2.1/jQuery.min.js

2.22.3 Selecting Elements

GQ. Explain different selectors in jQuery.

- jQuery provides selectors to select and make changes in HTML elements.
- jQuery selectors use the name, id, classes, types, attributes, values of attributes etc to select HTML elements.
- In jQuery, the selectors start with the symbol $ (dollar sign) and parentheses: $()
- Now we will discuss different type of selectors available in jQuery.

Types of selectors

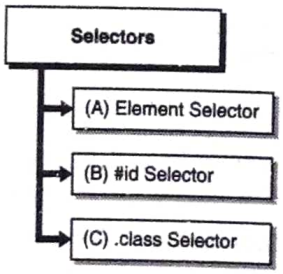


Fig. 2.22.2 : Different types of selectors

(A)The Element Selector

The jQuery element selector selects the HTML elements depending upon the element name.

<p> elements can be selected as follows :

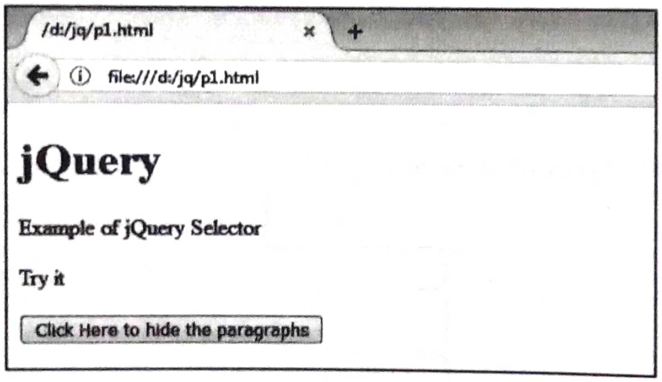
```
$("p")
```

Program 2.22.2 : Write a program to demonstrate the use of Element Selector.

Program to demonstrate the use of Element Selector

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
</script>
</head>
<body>
<h1>jQuery</h1>
<p>Example of jQuery Selector</p>
<p>Try it</p>
<button>Click Here to hide the paragraphs</button>
</body>
</html>
```

Output



(B) The #id Selector

The jQuery #id selector selects the HTML id attribute of the HTML tag to find the specific element. All the elements should have unique id.

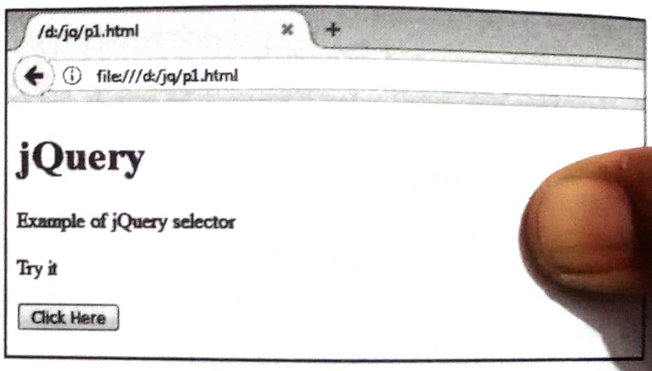
To search an element with a given id, hash character is written which is followed by the HTML element id.

```
$("#id1")
```

Program 2.22.3 : Write a program to demonstrate the use of #id Selector.

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#id1").hide();
  });
});
</script>
</head>
<body>
<h1>jQuery</h2>
<p>Example of jQuery selector</p>
<p id="id1">Try it</p>
<button>Click Here</button>
</body>
</html>
```

Output



(C) The .class Selector

The jQuery class selector finds elements with a specific class.

To search an element with a specific class, period character is written which is followed by the name of the class.

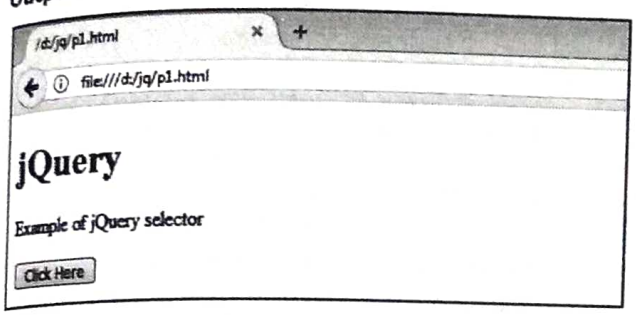
```
$(".myclass")
```

Program 2.22.4 : Write a program to demonstrate the use of .class Selector.

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $(".myclass").hide();
  });
```

```
});
</script>
</head>
<body>
<h1 class="myclass">jQuery</h2>
<p class="myclass">Example of jQuery selector</p>
<button>Click Here</button>
</body>
</html>
```

Output



2.23 DOM MANIPULATION WITH JQUERY

2.23.1 Changing Styles

GQ. How to change style in jQuery? Give example.

jQuery css() Method is used to set or return one or more style properties for the selected elements.

Syntax

To set a specified CSS property, use the following syntax :

```
css("propertyname","value");
```

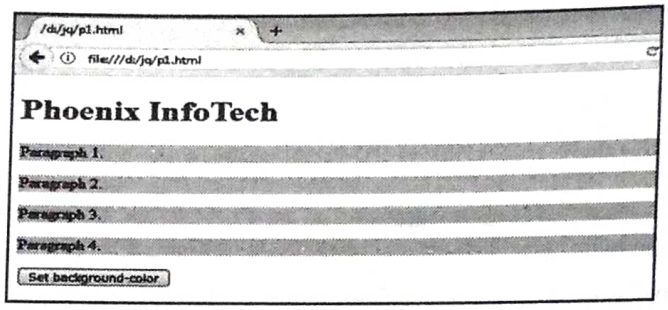
Example

```
$("p").css("background-color", "skyblue");
```

Program 2.23.1 : Write a program to change style in jQuery.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.
min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").css("background-color", "skyblue");
  });
});
</script>
</head>
<body>
<h1>Phoenix InfoTech</h1>
<p style="background-color:red">Paragraph 1.</p>
<p style="background-color:green">Paragraph 2.</p>
<p style="background-color:blue">Paragraph 3.</p>
<p>Paragraph 4.</p>
<button>Set background-color</button>
</body>
</html>
```

Output



2.23.2 Creating and Appending Elements

GQ. Write an example (program) to create and append HTML element in jQuery.

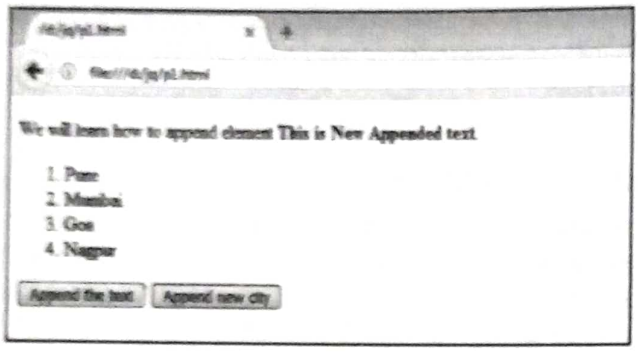
jQuery provides **append()** method to add an element.

Program 2.23.2 : Write a program to create and append HTML element in jQuery.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.
min.js"></script>
<script>
$(document).ready(function(){
  $("#btnA").click(function(){
    $("p").append(" <b>This is New Appended
text</b>.");
  });
  $("#btnB").click(function(){
    $("ol").append("<li>Nagpur</li>");
  });
});
</script>
</head>
<body>
```

```
<p>We will learn how to append element</p>
<ol>
 <li>Pune</li>
 <li>Mumbai</li>
 <li>Goa</li>
</ol>
<button id="btnA">Append the text</button>
<button id="btnB">Append new city</button>
</body>
</html>
```

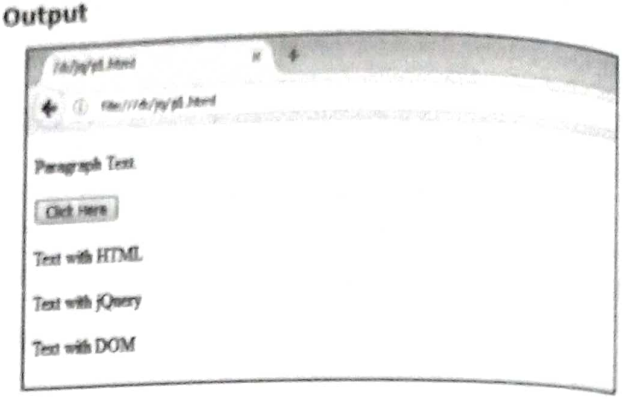
Output



Program 2.23.3 : Write a program to add multiple elements at a time in jQuery.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.
min.js"></script>
<script>
function appText()
{
  var t1 = "<p> Text with HTML</p>";
  var t2 = $("<p></p>").text("Text with jQuery");
  var t3 = document.createElement("p");
  t3.innerHTML = "Text with DOM";
  $("body").append(t1, t2, t3);    // Append new elements
}
</script>
</head>
<body>
<p>Paragraph Text.</p>
<button onclick="appText()">Click Here</button>
</body>
</html>
```

Output



2.23.3 Removing Elements

GQ. Write an example (program) to remove HTML element in jQuery.

- Removing existing HTML elements is very easy in jQuery.
- To remove elements and its content, jQuery provides two methods :
  - (i) **remove() :** Removes the selected element with its child elements.
  - (ii) **empty() :** Removes the child elements from the selected element.

jQuery remove() Method

This method removes the selected element with its child elements.

Syntax

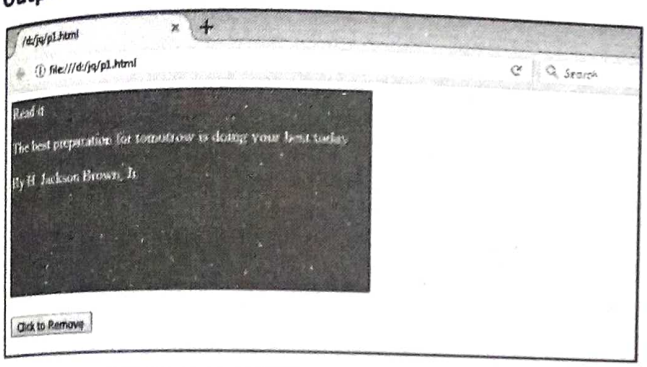
```
$("#div1").remove();
```

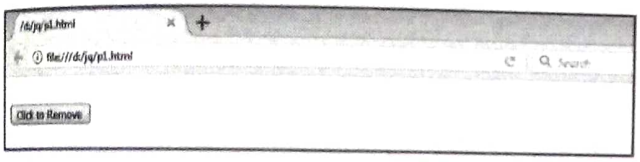
Program 2.23.4 : Write a program to demonstrate the use of remove method.

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.
min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#mydiv").remove();
  });
});
</script>
</head>
<body>
```

```
<div id="mydiv"
style="height:200px;width:400px;color:white;border:2px
solid black;background-
color:gray;">
Read it
<p>The best preparation for tomorrow is doing your best
today.</p>
<p>By H. Jackson Brown, Jr.</p>
</div>
<br>
<button>Click to Remove</button>
</body>
</html>
```

Output





☞ DOM Manipulation Methods

Following table lists down all the methods which you can use to manipulate DOM elements :

- **after(content):** Insert content after each of the matched elements.
- **append(content):** Append content to the inside of every matched element.
- **appendTo(selector):** Append all of the matched elements to another, specified, set of elements.
- **before(content):** Insert content before each of the matched elements.
- clone(bool): Clone matched DOM Elements, and all their event handlers, and select the clones.
- **clone( ):** Clone matched DOM Elements and select the clones.
- **empty( ):** Remove all child nodes from the set of matched elements.
- **html(val):** Set the html contents of every matched element.
- **html( ):** Get the html contents (innerHTML) of the first matched element.
- **insertAfter(selector):** Insert all of the matched elements after another, specified, set of elements.
- **insertBefore(selector):** Insert all of the matched elements before another, specified, set of elements.
- **prepend(content):** Prepend content to the inside of every matched element.
- **prependTo(selector):** Prepend all of the matched elements to another, specified, set of elements.
- **remove(expr):** Removes all matched elements from the DOM.
- **replaceAll(selector):** Replaces the elements matched by the specified selector with the matched elements.
- **replaceWith(content) :** Replaces all matched elements with the specified HTML or DOM elements.
- **text(val) :** Set the text contents of all matched elements.
- **text( ) :** Get the combined text contents of all matched elements.
- **wrap(elem) :** Wrap each matched element with the specified element.
- **wrap(html) :** Wrap each matched element with the specified HTML content.
- **wrapAll(elem) :** Wrap all the elements in the matched set into a single wrapper element.
- **wrapAll(html) :** Wrap all the elements in the matched set into a single wrapper element.
- **wrapInner(elem) :** Wrap the inner child contents of each matched element (including text nodes) with a DOM element.
- **wrapInner(html) :** Wrap the inner child contents of each matched element (including text nodes) with an HTML structure.

⏭ 2.24 DYNAMIC CONTENT CHANGE WITH JQUERY

GQ. How to handle events in dynamically created elements in jQuery ?

When we want to bind any event to an element, normally we could directly bind to any event of each element using the on() method.

Program 2.24.1 : Write a program using jQuery on() method to add paragraph element dynamically.

```
<!DOCTYPE html>
<html>

<head>
    <title>
        How to handle events in dynamically
        created elements in jQuery?
    </title>

    <script src=
"https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.j
s">
    </script>

    <script>
        $(document).ready(function () {
            $("#list li").on("click", function (event) {
                $('#list').append('<li>New Paragraph</li>');
            });
        });
    </script>

    <style>
        li {
            font-size: 30px;
            width: 400px;
            padding: 20px;
            color: green;
        }
    </style>
</head>

<body>
    <!-- Click on this paragraph -->
    <ul id="list">
        <li>Click here to append !!!</li>
    </ul>
</body>

</html>
```

When we add a new list item and click it, nothing happens. This is because of the event handler attached before which is executed when the document is loaded. At that time only the first list item existed and not the new ones. Hence the .on() method was applied only for the first list item and not the rest.

Program 2.24.2 : Implement program using on() method

```
<!DOCTYPE html>
<html>

<head>
    <title>
        How to handle events in dynamically
        created elements in jQuery?
    </title>
    <script src=
"https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.j
s">
    </script>

    <script>
        $(document).ready(function () {
            $("#list").on("click",
                "li", function (event) {
                $('#list').append(
                    '<li>New Paragraph</li>');
            });
        });
    </script>

    <style>
        li {
            font-size: 30px;
            width: 400px;
            padding: 20px;
            color: green;
        }
    </style>
</head>

<body>
    <ul id="list">

        <!-- Click on this item -->
        <li>Click here to check on()!!!</li>
    </ul>
</body>

</html>
```

Program 2.24.3 : Implement program using delegate() function

The following example is implemented using **delegate()** function. To bind the event handler to dynamically created elements, we will be using Event

Delegation. On clicking the new list items, the same action is performed.

Event delegation is the process that allows us to attach a single event listener, to the parent element and it will fire for all the children elements that exist now or will be added in the future. Both on() and delegate() functions allow us to do event delegation.

```
<!DOCTYPE html>
<html>
<head>
  <title>
    How to handle events in dynamically
    created elements in jQuery?
  </title>

  <script src=
"https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.j
s">
  </script>
  <script>
    $(document).ready(function () {
      $("#list").delegate("li",
        "click", function (event) {

        $('#list').append(
          '<li>New Paragraph</li>');
      });
    });
  </script>

  <style>
    li {
      font-size: 30px;
      width: 400px;
      padding: 20px;
      color: green;
    }
  </style>
</head>

<body>
  <ul id="list">

    <!-- Click on this item -->
    <li>Click to check delegate !!!</li>
  </ul>
</body>

</html>
```

2.25 UI DESIGN USING JQUERY

GQ. What is jQuery UI ?

- Good for highly interactive web applications
- Open source and free to use
- Powerful theme mechanism
- Stable and maintenance friendly
- Extensive browser support

Example of jQuery UI

- jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library.
- Whether you're building highly interactive web applications, or you just need to add a date picker to a form control, jQuery UI is a perfect choice.

Why jQuery UI ?

- jQuery UI contains a set of plugins that provide new features to the core jQuery library. It is categorized into four groups i.e. interactions, widgets, effects, utilities.
- It is an open-source library that contains different animation effects and widgets that help in the development of highly interactive user-friendly web applications. This library needs very less maintenance. It can be used with almost all browsers.

Download & Installation

- Go to the official documentation (*https://jqueryui.com/*) of the jQueryUI and click on the Custom Download button to download a customized version of the library. After downloading the zip file, unzip the files and save them in a folder.
- Create an HTML file like index.html and add file links inside head section of code.

```
<link rel="stylesheet" href="jqueryui/jquery-ui.min.css">
<script src="jqueryui/external/jquery/jquery.js"></script>
<script src="jqueryui/jquery-ui.min.js"></script>
```

- Using CDN Link: Without downloading the jQuery UI files, you can include CDN links inside the head section to run your code.

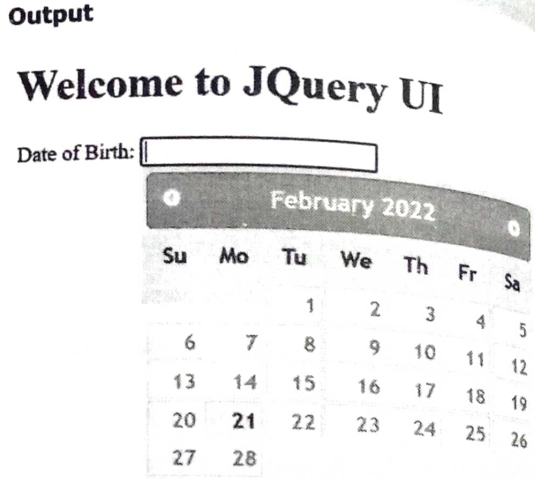
```
<link href="https://code.jquery.com/ui/1.10.4/themes/ui-
lightness/jquery-ui.css" rel ="stylesheet">
<script src="https://code.jquery.com/jquery-
1.10.2.js"></script>
<script src="https://code.jquery.com/ui/1.10.4/jquery-
ui.js"></script>
```

- **Example:** In this example, we will display the date picker on the page using jQuery UI.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content=
    "width=device-width, initial-scale=1">
  <link href=
"https://code.jquery.com/ui/1.10.4/themes/ui-
lightness/jquery-ui.css"
    rel="stylesheet">
  <script src=
    "https://code.jquery.com/jquery-1.10.2.js">
  </script>
  <script src=
    "https://code.jquery.com/ui/1.10.4/jquery-ui.js">
  </script>
</head>

<body>
  <h1>Welcome to JQuery UI</h1>
  <p>Date of Birth:
    <input type="text" id="dob">
  </p>
  <script>
    $("#dob").datepicker();
  </script>
</body>

</html>
```

Output