

Unit-II Client Side Technologies: JavaScript & DOM

Subject: Web Technology(310252)

By: Prof. Kanchan A. Pekhale

Overview of JavaScript

- ▶ JavaScript is open source probably most popular client side scripting language
- ▶ Supported by all browsers
- ▶ This language is used for increasing the interaction of an end user with the webpage
- ▶ Helps to make our webpage more lively and interactive
- ▶ Widely used in mobile application development as well as in game development
- ▶ It supports dynamic scripting
- ▶ It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages
- ▶ It is an interpreted programming language with object-oriented capabilities

Cont...

- ▶ JavaScript was developed by Mr. Brendan Eich in 1995 who was working in Netscape
- ▶ JavaScript was initially called as LiveScript and later on the name is changed to JavaScript.
- ▶ The syntax of JavaScript is typically influenced by the C programming language
- ▶ **Client-Side JavaScript:**
 - ▶ Client-side JavaScript is the most common form of the language
 - ▶ The script should be included in or referenced by an HTML document for the code to be interpreted by the browser
 - ▶ It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content

Cont...

- ▶ The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts
- ▶ For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field
- ▶ The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server
- ▶ JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly

Advantages of JavaScript

- ▶ **Less server interaction** – You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server
- ▶ **Immediate feedback to the visitors** – They don't have to wait for a page reload to see if they have forgotten to enter something
- ▶ **Increased interactivity** – You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard
- ▶ **Richer interfaces** – You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors

Cont...

- ▶ Lightweight, interpreted
- ▶ Network based development
- ▶ Complementary to Java
- ▶ Integrated with HTML
- ▶ Open Source
- ▶ Cross-platform

Limitations of JavaScript

- ▶ Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- ▶ JavaScript cannot be used for networking applications because there is no such support available.
- ▶ JavaScript doesn't have any multi-threading or multiprocessor capabilities.

JavaScript Development Tools

- ▶ One of major strengths of JavaScript is that it does not require expensive development tools
- ▶ You can start with a simple text editor such as Notepad
- ▶ Since it is an interpreted language inside the context of a web browser, you don't even need to buy a compiler
- ▶ **JavaScript editing tools:**
 - ▶ Microsoft FrontPage
 - ▶ Macromedia Dreamweaver MX
 - ▶ Macromedia HomeSite 5

Cont...

- ▶ **Microsoft FrontPage** – Microsoft has developed a popular HTML editor called FrontPage. FrontPage also provides web developers with a number of JavaScript tools to assist in the creation of interactive websites.
- ▶ **Macromedia Dreamweaver MX** – Macromedia Dreamweaver MX is a very popular HTML and JavaScript editor in the professional web development crowd. It provides several handy prebuilt JavaScript components, integrates well with databases, and conforms to new standards such as XHTML and XML.
- ▶ **Macromedia HomeSite 5** – HomeSite 5 is a well-liked HTML and JavaScript editor from Macromedia that can be used to manage personal websites effectively.

Difference between Java and JavaScript

Sr. No.	Parameter	Java	JavaScript
1	Execution	Java creates application which can be executed on virtual machine or browser	JavaScript creates application which can be executed on browser only
2	Features	Java code allows programmer full functionality	JavaScript code contains limited number of commands and features
3	Naming	The first name of java was OAK and was developed by James Gosling, Sun Microsystems	JavaScript was earlier known as LiveScript and was developed by Brendan Eich, Netspace
4	Type Safety	Java is high level, compiled and strongly type language	JavaScript is next based and weakly typed language

Cont...

Sr. No.	Parameter	Java	JavaScript
5	Variables	Variables are created using the data type names like int, char double etc.	Variables are created using the var keyword
6	Extension	Java program has file extension .java and after compilation it creates .class file	JavaScript file has file extension .js or .html
7	Objects	Objects of java are class based	Objects of JavaScript are prototype based
8	Scope	Java has block based scope	JavaScript has function based scope and object based context

JavaScript Syntax

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

```
<script ...>
```

```
    JavaScript code;
```

```
</script>
```

Cont...

- ▶ The script tag takes two important attributes –
 - ▶ **Language** – This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
 - ▶ **Type** – This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".
 - ▶ So your JavaScript segment will look like –

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

```
    JavaScript code
```

```
</script>
```

Embedded JavaScript



- ▶ Program- Write an embedded JavaScript code displaying welcome message
- ▶ 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>
```

Cont...

► Output:

← → ↻ ⓘ File | D:/Kanchan/WT/sample.html

 Apps  Gmail  YouTube

Welcome to JavaScript!!!

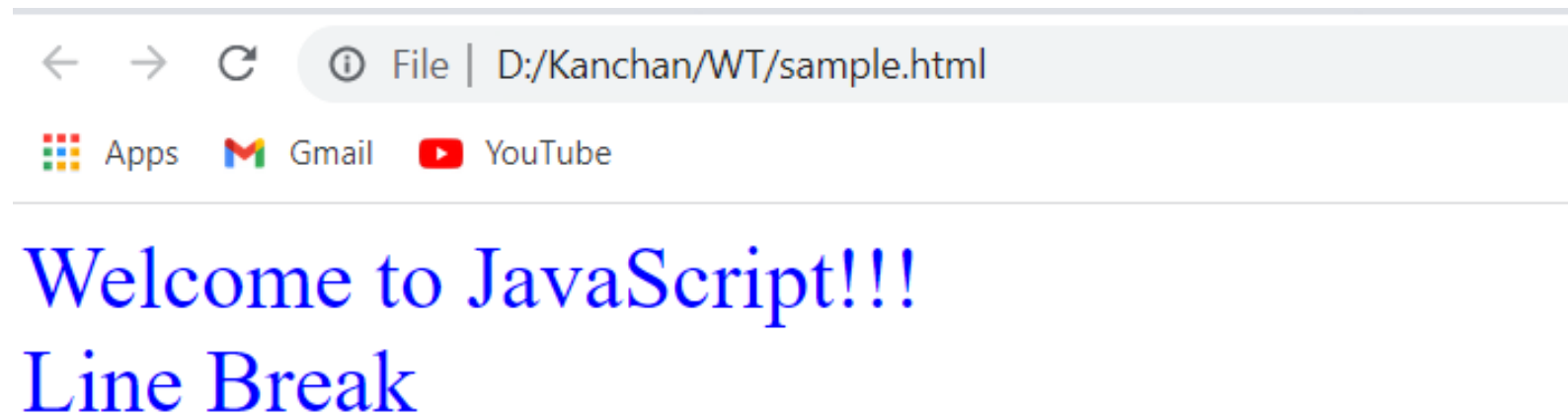
Cont...

- Program- Write a program to demonstrate line break example

```
<!DOCTYPE html>
<html>
  <head>
    <title> JavaScript </title>
  </head>
  <body>
    <font size=6 color="blue">
      <script language="JavaScript">
        document.write("Welcome to
JavaScript!!!");
        document.write("<br> Line Break");
      </script>
    </font>
  </body>
</html>
```


Cont...

► Output



External JavaScript

- ▶ An external JavaScript file can be created to embed it in many html pages
- ▶ It supports the concepts of **Code Reusability** as single JavaScript file can be embed into several html pages
- ▶ An external JavaScript file is saved by the extension “.js”
- ▶ **Program- Write an external JavaScript code demonstrating welcome message**
- ▶ **Myfile.js**

```
function sayHello()  
{  
    alert("Hello World")  
}
```

Cont...

► .html file

```
<!DOCTYPE html>
<html>
  <head>
    <title> JavaScript </title>
    <script type="text/javascript" src="myfile.js">
    </script>
  </head>
  <body>
    <p> Welcome to JavaScript </p>
    <form>
      <input type="button" value="Click Here"
        onclick="sayHello()"/>
    </form>
  </body>
</html>
```

Variables & Data Types

- ▶ 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
 - ▶ Primitive
 - ▶ Non-Primitive

Cont...

1. JavaScript Primitive Data Types:

Data Types	Description	Example
String	represents textual data	'hello', "hello world!" etc
Number	an integer or a floating-point number	3, 3.234, 3e-2 etc.
Boolean	Any of two values: true or false	true and false
undefined	a data type whose variable is not initialized	let a;
null	denotes a null value	let a = null;

Cont...

2. JavaScript Non-Primitive Data Types

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

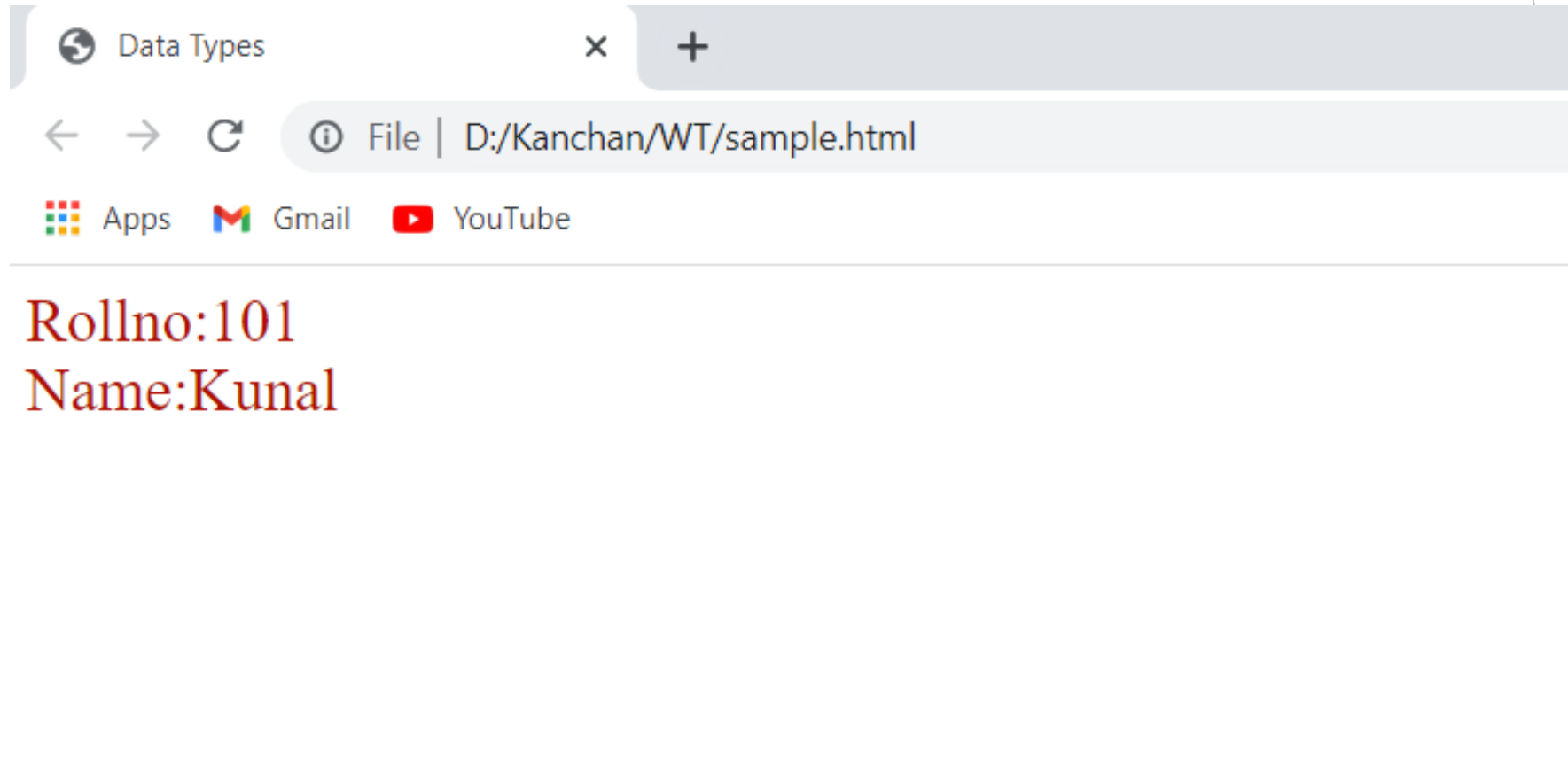
Cont...

- ▶ JavaScript is considered as dynamic type language that is there is no need to specify type of the variable
- ▶ 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.
- ▶ Example:
 - ▶ `Var rno=101; //holding number`
 - ▶ `Var sname=“Kunal” //holding string`

Program- 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=bule>");
      var rno=101;
      var sname="Kunal";
      document.write("Rollno:"+rno);
      document.write("<br> Name:"+sname);
      document.write("</font>");
    </script>
  </body>
</html>
```


Cont...



Prompt() and alert()

- ▶ **prompt():** This method is used to display an input box which can accept value from user.
- ▶ **alert():** This method is used to display a message box

Program- Write a program to accept two numbers from user and display their summation

```
<html>
  <head>
    <title> User Input </title>
  </head>
  <body>
    <script language="JavaScript">
      var n1,n2,sum;
      n1=parseInt(prompt("Enter First Number:"));
      n2=parseInt(prompt("Enter Second Number:"));
      sum=n1+n2;
      alert("Summation is:"+sum)
    </script>
  </body>
</html>
```

Statements

1. Conditional Statements

1. If
2. If else
3. Else if ladder
4. Switch case

2. Loop Statements

1. While
2. Do while
3. for

1. Conditional Statements

► If Statement:

- The if statement used to specify a block of statements to be executed, if the given condition is true

► Syntax:

```
► if (condition)
{
    statements;
}
```

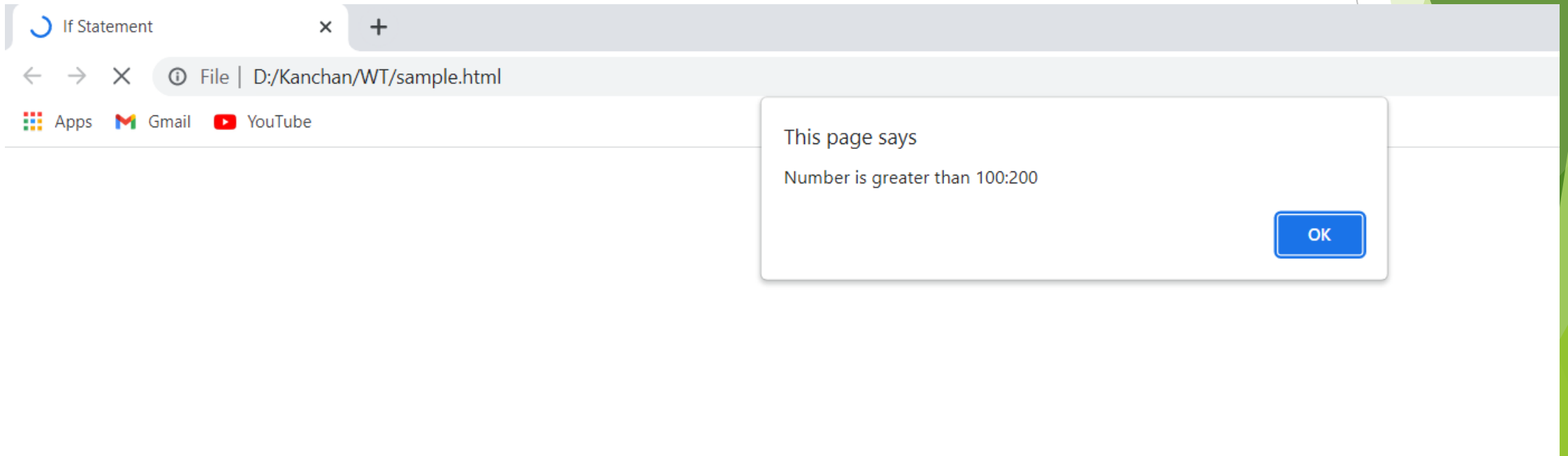
Cont...

- Program: Write a program to accept a number and check whether it is greater than 100 or not

```
<html>
  <head>
    <title> If Statement </title>
  </head>
  <body>
    <script language="JavaScript">
      var n;
      n=parseInt(prompt("Enter a Number:"));
      if(n>100)
      {
        alert("Number is greater than 100 :"+n)
      }
    </script>
  </body>
</html>
```

Cont...

► Output:



Cont...

► Else Statement:

- The else statement is used to specify a block of statements to be executed, if the given condition is false

- Syntax:

- if(Condition)
 {
 Statements;
 }
 else
 {
 Statements;
 }

Cont...

- Program: Write a program to accept a number from user and check whether it is even or odd

```
<html>
  <head>
    <title> If Else Statement </title>
  </head>
  <body>
    <script language="JavaScript">
      var n;
      n=parseInt(prompt("Enter a Number:"));
      if(n%2==0)
      {
        alert("Number is Even: "+n)
      }
      else
      {
        alert("Number is Odd: "+n)
      }
    </script>
  </body>
</html>
```

Cont...

► Output:



Cont...

► Else if Ladder:

- It is used to specify another condition to test if the previous condition is false
- Syntax:

```
► if(Condition)
{
    Statements;
}
else if(Condition)
{
    Statements;
}
-----
else
{
    Statements;
}
```

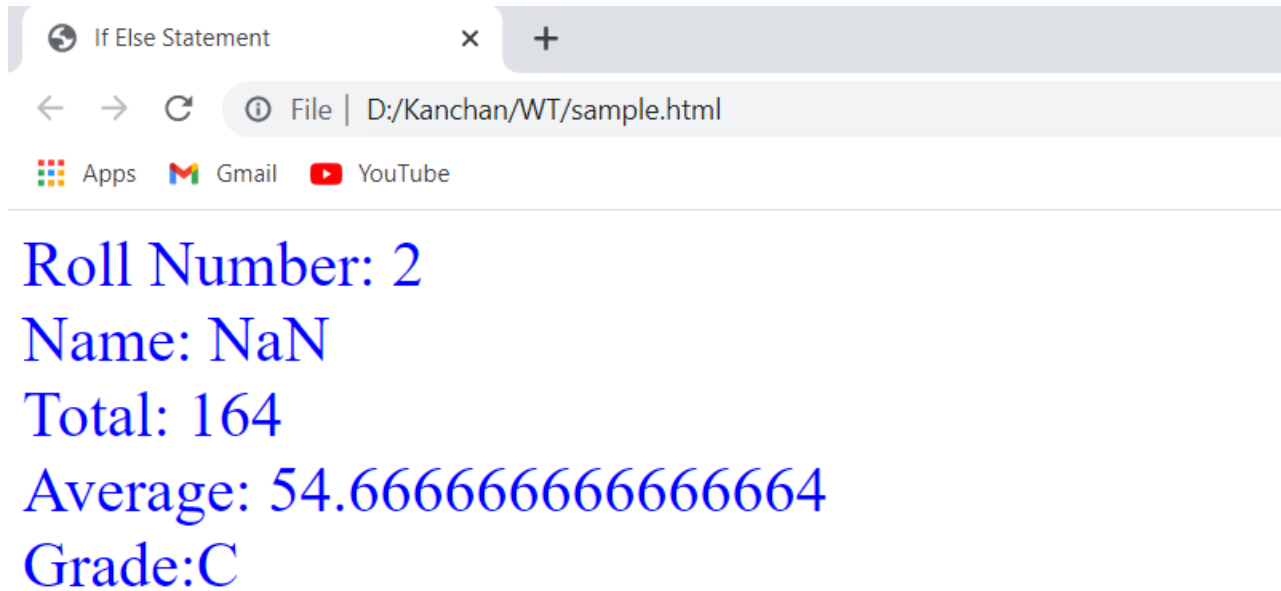
Cont...

- ▶ Program: Write a program which accepts roll no., name and marks of 3 subjects from student. Calculate the total and average of marks and print the grade (avg \geq 80; then grade-A, \geq 60 then grade-B, \geq 40 then grade-C else Fail)

```
<html>
  <head>
    <title> If Else Statement </title>
  </head>
  <body>
    <font size=6 color="blue">
      <script language="JavaScript">
        var rno,sname,WT,CC,Al,total,avg;
        rno=parseInt(prompt("Enter Roll Number:"));
        sname=parseInt(prompt("Enter Name:"));
        WT=parseInt(prompt("Enter WT Marks:"));
        CC=parseInt(prompt("Enter CC Marks:"));
        Al=parseInt(prompt("Enter Al Marks:"));
        total=WT+CC+Al;
        avg=total/3;
        document.write("Roll Number: "+rno);
        document.write("<br> Name: "+sname);
        document.write("<br> Total: "+total);
        document.write("<br> Average: "+avg);
        if(WT>=40 && CC>=40 && Al>=40)
        {
          if(avg>=80)
            document.write("<br> Grade:A");
          else if(avg>=60)
            document.write("<br> Grade:B");
          else if(avg>=40)
            document.write("<br> Grade:C");
        }
        else
        {
          document.write("<br> Fail...!!!");
        }
      </script>
    </font>
  </body>
</html>
```

Cont...

► Output:



Cont...

► Switch Case Statement:

- Switch statement is used to select one of many blocks of code to be executed
- Syntax:

```
Switch(expression)
{
    case constant_expression:
        statements;
        break;
    case constant_expression:
        statements;
        break;
    -----
    default:
        statements;
}
```

Cont...

- ▶ Switch expression is evaluated only once
- ▶ Value of expression is compared with the values of each constant expression
- ▶ If there is a match, then the related statements are executed

Cont...

- Program: Write a program to print the numbers between 1 to 5

```
<html>
  <head>
    <title> Switch Case </title>
  </head>
  <body>
    <font size=5 color="blue">
    <script language="JavaScript">
      var n;
      n=parseInt(prompt("Enter a number between 1 to 5:"));
      switch(n)
      {
        case 1:
          document.write("One");
          break;
        case 2:
          document.write("Two");
          break;
```

Cont...

```
case 3:
    document.write("Three");
    break;
case 4:
    document.write("Four");
    break;
case 5:
    document.write("Five");
    break;
default:
    document.write("Not in the Range");
}
</script>
</font>
</body>
</html>
```

Loop Statements

► While Loop:

- While is an entry controlled loop
- That means if the given condition is not satisfied then the loop statements will never get execute

► Syntax:

```
While(condition)
{
    statements;
}
```

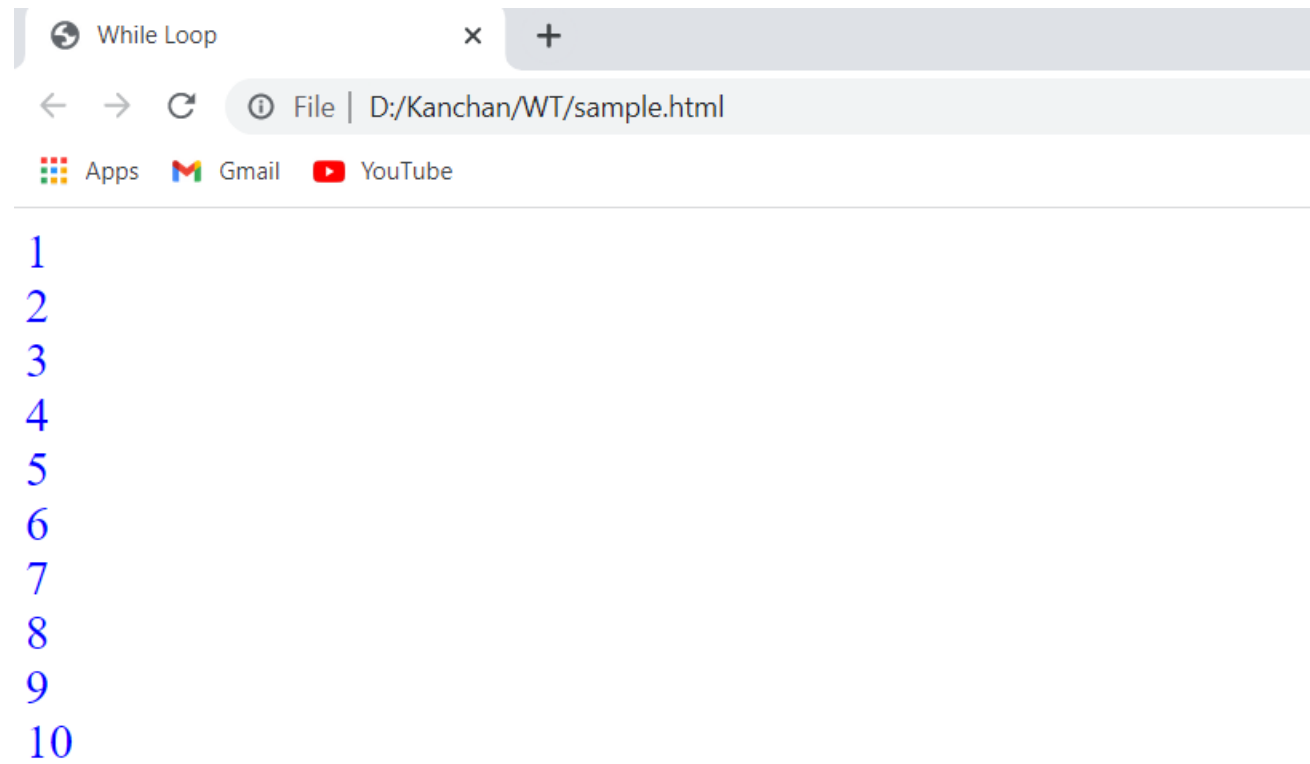
Cont...

- Program: Write a program to print 1 to 10 numbers using while loop

```
<html>
  <head>
    <title> While Loop </title>
  </head>
  <body>
    <font size=5 color="blue">
      <script language="JavaScript">
        var n=1;
        while(n<=10)
        {
          document.write(n+"<br>");
          n=n+1;
        }
      </script>
    </font>
  </body>
</html>
```

Cont...

► Output:



Cont...

► Do While Loop:

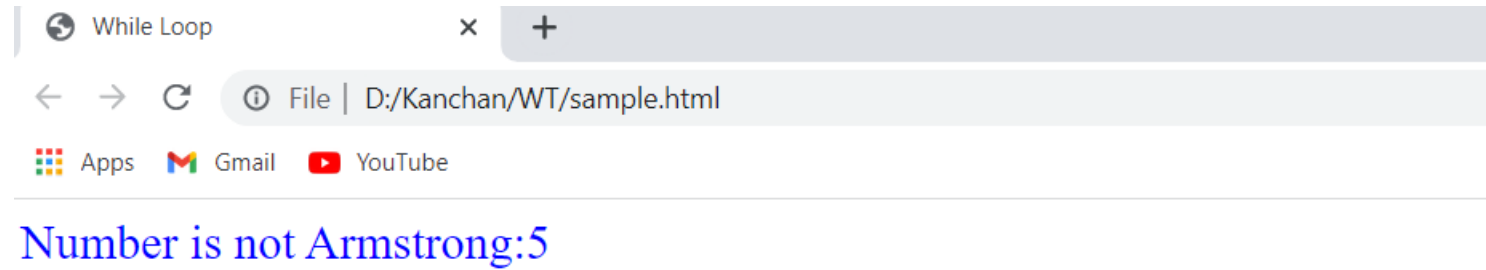
- Do while is an exit controlled loop
- The first execution of loop statement is done without checking any condition
- From second execution the condition get checked
- Hence even is the condition is not satisfying then also the loop statements get executed once.
- **Example: Write a program to accept a number from user and check whether it is Armstrong number or not.**

Cont...

```
<html>
  <head>
    <title> While Loop </title>
  </head>
  <body>
    <font size=5 color="blue">
      <script language="JavaScript">
        var n,n1,r,sum;
        sum=0;
        n=parseInt(prompt("Enter a Number:"));
        n1=n;
        do
        {
            r=n%10;
            sum=sum+(r*r*r);
            n=Math.floor(n/10);
        }while(n>0);
        if(n1==sum)
            document.write("Number is Armstrong:"+n1);
        else
            document.write("Number is not Armstrong:"+n1)
      </script>
    </font>
  </body>
</html>
```

Cont...

► Output:



Cont...

► For Loop:

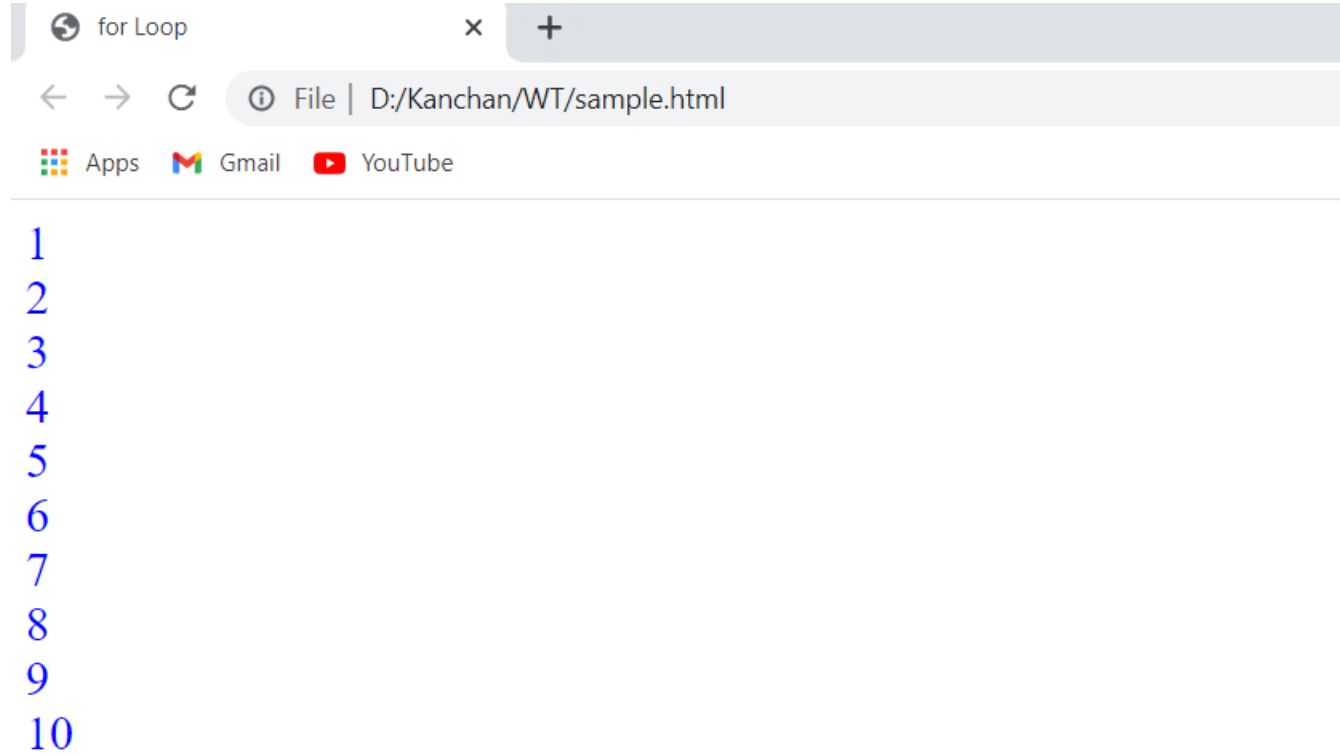
- For loop the initialization, condition and increment or decrement of loop variable is done in a single statement
- Helps to minimize the code
- **Example: Write a program to print 1 to 10 numbers using for loop**

Cont...

```
<html>
  <head>
    <title> for Loop </title>
  </head>
  <body>
    <font size=5 color="blue">
      <script language="JavaScript">
        var n=1;
        for(n=1;n<=10;n++)
        {
          document.write(n+"<br>");
        }
      </script>
    </font>
  </body>
</html>
```

Cont...

► Output:



Operators

► JavaScript Arithmetic Operators:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

Cont...

► JavaScript Assignment Operators:

Operator	Example	Same As
=	<code>x = y</code>	<code>x = y</code>
+=	<code>x += y</code>	<code>x = x + y</code>
-=	<code>x -= y</code>	<code>x = x - y</code>
*=	<code>x *= y</code>	<code>x = x * y</code>
/=	<code>x /= y</code>	<code>x = x / y</code>
%=	<code>x %= y</code>	<code>x = x % y</code>
**=	<code>x **= y</code>	<code>x = x ** y</code>

Cont...

- ▶ JavaScript String Operators:

- ▶ The + operator can also be used to add (concatenate) strings.

- ▶ Example:

```
text1 = "John";
```

```
text2 = "Doe";
```

```
text3 = text1 + " " + text2;
```

- ▶ The += assignment operator can also be used to add (concatenate) strings

- ▶ Example:

```
text1 = "What a very ";
```

```
text1 += "nice day";
```

Cont...

► JavaScript Comparison Operators:

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Cont...

► JavaScript Logical Operators:

Operator	Description
&&	logical and
	logical or
!	logical not

Cont...

► JavaScript Type Operators:

Operator	Description
<code>typeof</code>	Returns the type of a variable
<code>instanceof</code>	Returns true if an object is an instance of an object type

Cont...

- ▶ JavaScript Bitwise Operators:
 - ▶ Bit operators work on 32 bits numbers.
 - ▶ Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

Cont...

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
	OR	5 1	0101 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	left shift	5 << 1	0101 << 1	1010	10
>>	right shift	5 >> 1	0101 >> 1	0010	2
>>>	unsigned right shift	5 >>> 1	0101 >>> 1	0010	2

Literals

- ▶ JavaScript Literals are constant values that can be assigned to the variables that are called literals or constants.
- ▶ JavaScript Literals are syntactic representations for different types of data like numeric, string, Boolean, array, etc. data.
- ▶ Literals in JavaScript provide a means of representing particular or some specific values in our program
- ▶ Example, `var name = "john"`, a string variable named `name` is declared and assigned a string value `"john"`
- ▶ The literal `"john"` represents, the value `john` for the variable `name`

Cont...

1. Integer Literals:

1. Integer literals are numbers, must have minimum one digit (0-9)
2. No blank or comma is allowed within an integer
3. It can store positive numbers or negative numbers
4. In integers, literals in JavaScript can be supported in three different bases.
5. The base 10 that is Decimal (Decimal numbers contain digits (0,9)) examples for Decimal numbers are 234, -56, 10060.
6. Second is base 8 that is Octal (Octal numbers contains digits (0,7) and leading 0 indicates the number is octal), 0X 073, -089, 02003.
7. Third is base 16 that is Hexadecimal numbers (Hexadecimal numbers contains (0,9) digits and (A,F) or (a, f) letters and leading 0x or 0X indicates the number is hexadecimal), examples for hexadecimal numbers are 0X8b, - 0X89, 0X2003.

Cont...

```
<!DOCTYPE html>
<html>
  <head>
    <title> This is an example for numeric literals </title>
  </head>
  <body>
    <h1>JavaScript Numbers </h1>
    <p> Number can be written of any base.</p>
    Decimal number : <b id="no1"></b><br>
    Octal number : <b id="no2"></b><br>
    Hexadecimal number : <b id="no3"></b><br>
    <script>
      document.getElementById("no1").innerHTML = 100.25;
    </script>
    <script>
      document.getElementById("no2").innerHTML = 073;
    </script>
    <script>
      document.getElementById("no3").innerHTML = 0X8b;
    </script>
  </body>
</html>
```

Cont...

► Output:

← → ↻ ⓘ File | D:/Kanchan/WT/sample.html

📱 Apps 📧 Gmail 📺 YouTube

JavaScript Numbers

Number can be written of any base.

Decimal number : **100.25**

Octal number : **59**

Hexadecimal number : **139**

Cont...

2. Floating Number Literals:

Floating numbers are decimal numbers or fraction numbers or even can have an exponent part as well.

Examples for hexadecimal numbers are 78.90, -234.90, 78.6e4 etc.

Cont...

```
<!DOCTYPE html>
<html>
  <head>
    <title> This is an example for float literals </title>
  </head>
  <body>
    <h1>JavaScript Float </h1>
    <p> Float Examples are : </p>
    1. <b id="no1"></b><br>
    2. <b id="no2"></b><br>
    3. <b id="no3"></b><br>
    <script>
      document.getElementById("no1").innerHTML = 100.25;
    </script>
    <script>
      document.getElementById("no2").innerHTML = -78.34;
    </script>
    <script>
      document.getElementById("no3").innerHTML = 56e4;
    </script>
  </body>
</html>
```

Cont...

► Output:



JavaScript Float

Float Examples are :

1. **100.25**
2. **-78.34**
3. **560000**

Cont...

3. String Literals:

1. A string literals are a sequence of zero or more characters.
2. A string literals are either enclosed in the single quotation or double quotation as (') and (") respectively and to concatenate two or more string we can use + operator.
3. Examples for string are "hello", "hello world", "123", "hello" + "world" etc.

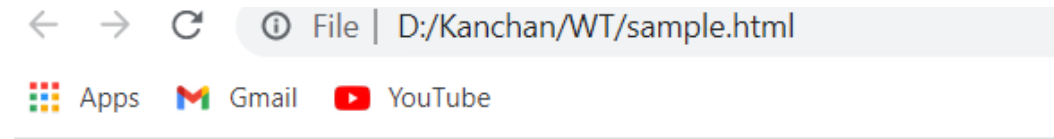
- \b: Backspace.
- \n: New Line
- \t: Tab
- \f: Form Feed
- \r: Carriage Return
- \\: Backslash Character (\)
- \' : Single Quote
- \" : Double Quote

Cont...

```
<!DOCTYPE html>
<html>
  <head>
    <title> This is an example for float literals </title>
  </head>
  <body>
    <h1>JavaScript String </h1>
    <p> String Examples are : </p>
    1. <b id="no1"></b><br>
    2. <b id="no2"></b><br>
    3. <b id="no3"></b><br>
    4. <b id="no4"></b><br>
    <script>
      var str = "This is first string";
      document.getElementById("no1").innerHTML = str;
    </script>
    <script>
      var strobj = new String("This is string store as object");
      document.getElementById("no2").innerHTML = strobj;
    </script>
    <script>
      var str = "This is first string";
      document.getElementById("no3").innerHTML = str.length;
    </script>
    <script>
      var str = "This is first string";
      document.getElementById("no4").innerHTML = str+" This is second string";
    </script>
  </body>
</html>
```

Cont...

► Output:



JavaScript String

String Examples are :

1. **This is first string**
2. **This is string store as object**
3. **20**
4. **This is first string This is second string**

Cont...

4. Array Literals:

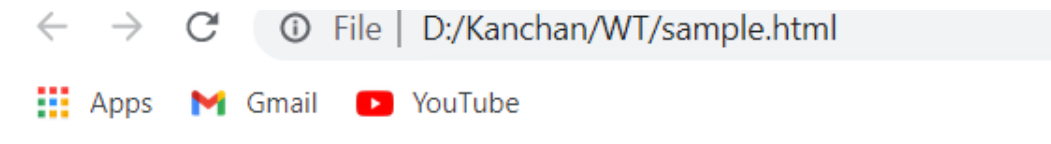
1. Array literals are a list of expressions or other constant values, each of which expression known as an array element
2. An array literal contains a list of element s within square brackets ‘ [] ‘
3. If no value is a pass when it creates an empty array with zero length
4. If elements are passed then its length is set to the number of elements passed
5. Examples for string are `var color = []`, `var fruits = [“Apple”, “Orange”, “Mango”, “Banana”]`

Cont...

```
<!DOCTYPE html>
<html>
  <head>
    <title> This is an example for float literals </title>
  </head>
  <body>
    <h1>JavaScript Array </h1>
    <p> Array Examples are : </p>
    1. <b id="no1"></b><br>
    2. <b id="no2"></b><br>
    3. <b id="no3"></b><br>
    4. <b id="no4"></b><br>
    <script>
      var fruits = ["Apple", "Orange", "Mango", "Banana"];
      document.getElementById("no1").innerHTML = fruits;
    </script>
    <script>
      document.getElementById("no2").innerHTML = fruits[0];
    </script>
    <script>
      document.getElementById("no3").innerHTML = fruits[fruits.length - 1];
    </script>
    <script>
      document.getElementById("no4").innerHTML = fruits.length;
    </script>
  </body>
</html>
```

Cont...

► Output:



JavaScript Array

Array Examples are :

1. **Apple,Orange,Mango,Banana**
2. **Apple**
3. **Banana**
4. **4**

Cont...

5. Boolean Literals:

1. Boolean literals in JavaScript have only two literal values that are true and false.

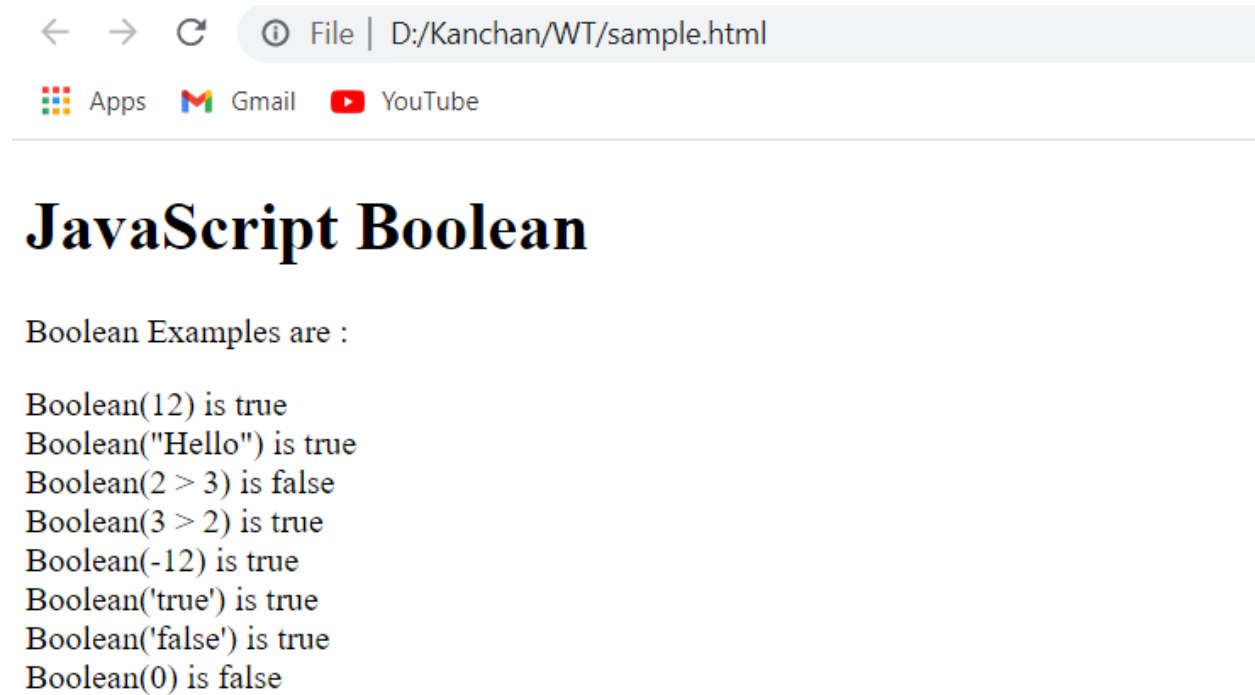
```
<!DOCTYPE html>
<html>
  <head>
    <title> This is an example for Boolean literals </title>
  </head>
  <body>
    <h1>JavaScript Boolean </h1>
    <p> Boolean Examples are : </p>
    <script>
      document.write('Boolean(12) is ' + Boolean(12));
      document.write('<br>');
```

Cont...

```
document.write('Boolean("Hello") is ' + Boolean("Hello"));
document.write('<br>');
document.write('Boolean(2 > 3) is ' + Boolean(2 > 3));
document.write('<br>');
document.write('Boolean(3 > 2) is ' + Boolean(3 > 2));
document.write('<br>');
document.write('Boolean(-12) is ' + Boolean(-12));
document.write('<br>');
document.write("Boolean('true') is " + Boolean('true'));
document.write('<br>');
document.write("Boolean('false') is " + Boolean('false'));
document.write('<br>');
document.write('Boolean(0) is ' + Boolean(0));
</script>
</body>
</html>
```

Cont...

► Output:



Cont...

6. Object Literals:

1. Object literals are collection zero or more key-value pairs of a comma-separated list, which are enclosed by a pair of curly braces ‘ { } ‘
2. Examples for object literal with declaration are

```
var userObject = { },
```

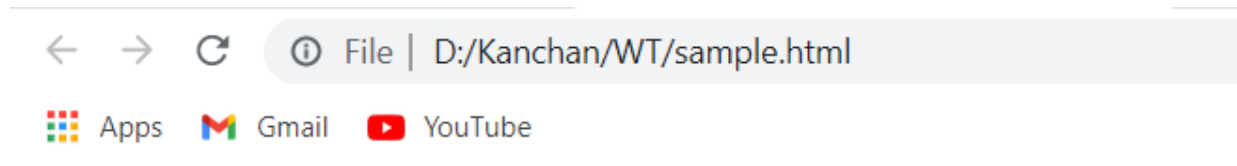
```
var student = { f-name : “John”, l-name : “D”, “rno” : 23, “marks” : 60 }
```

Cont...

```
<!DOCTYPE html>
<html>
  <head>
    <title> This is an example for Object literals </title>
  </head>
  <body>
    <h1>JavaScript Object </h1>
    <p> Object Examples are : </p>
    <p id= "no1"> </p>
    <script>
      // Create an object:
      var student = {firstName:"John", lastName:"D", "rno" : 23, "marks" : 60 };
      // Displaying some data from the object:
      document.getElementById("no1").innerHTML = student.firstName + " got " + student.marks + " marks.";
    </script>
  </body>
</html>
```

Cont...

► Output:



JavaScript Object

Object Examples are :

John got 60 marks.

Functions

- ▶ A JavaScript function is a block of code designed to perform a particular task.
- ▶ A JavaScript function is executed when "something" invokes it (calls it).
- ▶ **Function Syntax:**
 - ▶ A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses ().
 - ▶ Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
 - ▶ The parentheses may include parameter names separated by commas:
(*parameter1*, *parameter2*, ...)
 - ▶ The code to be executed, by the function, is placed inside curly brackets: {}

Cont...

```
function name(parameter1, parameter2, parameter3)  
{  
    // code to be executed  
}
```

- ▶ Function **parameters** are listed inside the parentheses () in the function definition.
- ▶ Function **arguments** are the **values** received by the function when it is invoked.
- ▶ Inside the function, the arguments (the parameters) behave as local variables.
- ▶ A Function is much the same as a Procedure or a Subroutine, in other programming languages.

Cont...

▶ **Function Invocation:**

- ▶ The code inside the function will execute when "something" **invokes** (calls) the function:
 - ▶ When an event occurs (when a user clicks a button)
 - ▶ When it is invoked (called) from JavaScript code
 - ▶ Automatically (self invoked)

▶ **Function Return:**

- ▶ When JavaScript reaches a return statement, the function will stop executing.
- ▶ If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

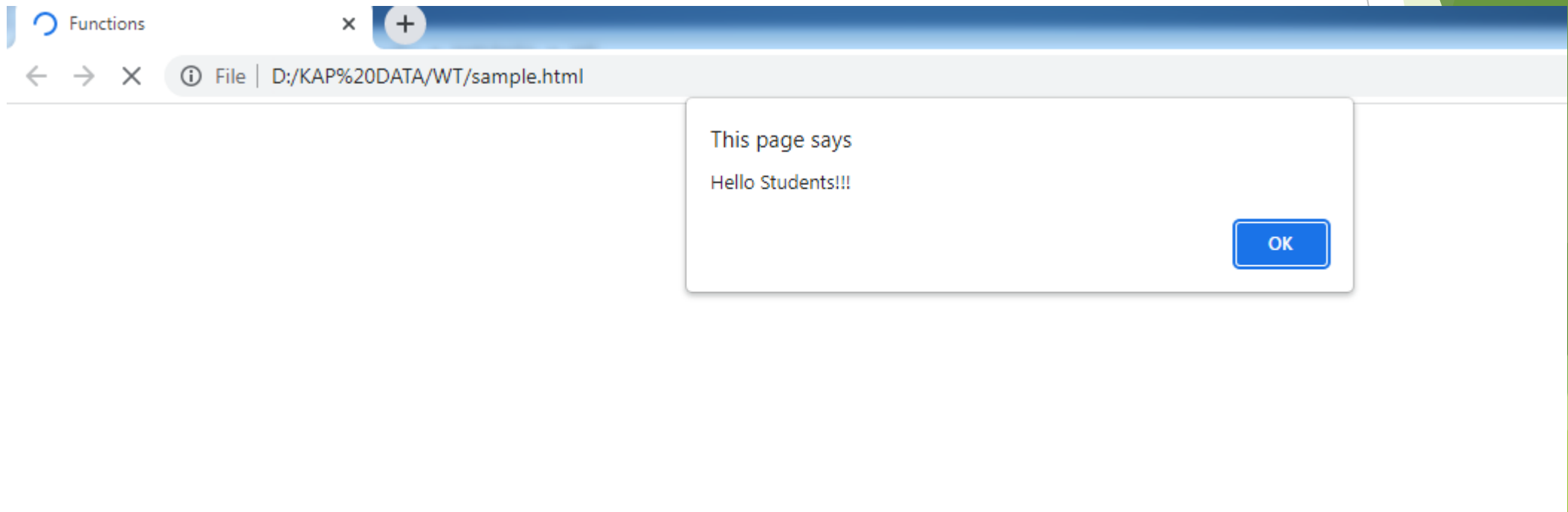
Cont...

- Program : Write a simple function calling program

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

Cont...

► Output:



- Program: Write a program to call a function on click event of button

```
<html>
  <head>
    <title> Functions </title>
    <script language="JavaScript">
      function hello()
      {
        alert("Hello Students!!!")
      }
    </script>
  </head>
  <body>
    <font size=5>
      <p> Click the following button to call the function </p>
      <form>
        <input type="button" onClick="hello()" value="Say Hello">
      </form>
    </font>
  </body>
</html>
```

Parameterized Function

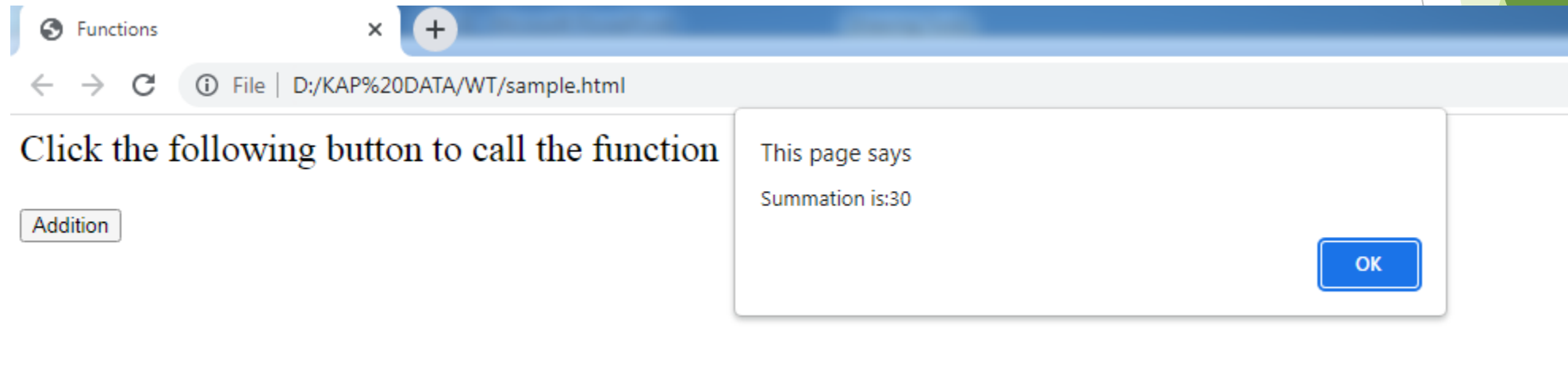
- ▶ While defining a function, we can declare variables in the header statement of the function
- ▶ These variables are known as parameters or formal arguments
- ▶ When this function gets called, we can pass values for these variables
- ▶ These values are known as arguments or actual arguments
- ▶ Program: Write a program to display the summation of two values using parameterized function

Cont...

```
<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="Addition">
      </form>
    </font>
  </body>
</html>
```

Cont...

► Output:



Return Statement

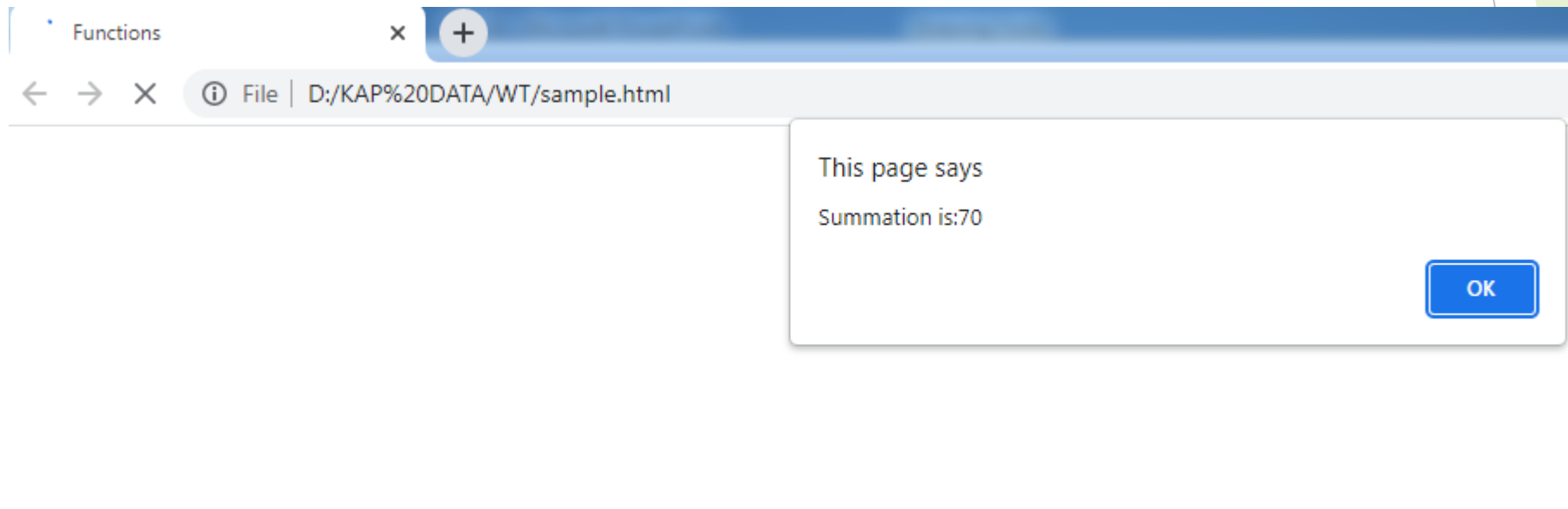
- ▶ In JavaScript function we can have return statement which is optional
- ▶ This helps to return a value from a function
- ▶ Program: Write a program which will accept two numbers as arguments and returns their summation

Cont...

```
<html>
  <head>
    <title> Functions </title>
    <script language="JavaScript">
      function add(a,b)
      {
        var sum=a+b;
        return(sum);
      }
    </script>
  </head>
  <body>
    <font size=5>
      <script language="JavaScript">
        var r=add(40,30);
        alert("Summation is:"+r)
      </script>
    </font>
  </body>
</html>
```

Cont...

► Output:



Objects in JavaScript

- ▶ Definition: An object is nothing but an entity having its own state and behavior
- ▶ Example: A flower is an object having properties like color, fragrance etc.
 - ▶ Booleans can be objects (if defined with the new keyword)
 - ▶ Numbers can be objects (if defined with the new keyword)
 - ▶ Strings can be objects (if defined with the new keyword)
 - ▶ Dates are always objects
 - ▶ Maths are always objects
 - ▶ Regular expressions are always objects
 - ▶ Arrays are always objects
 - ▶ Functions are always objects
 - ▶ Objects are always objects

Cont...

- ▶ We can create our own user defined objects in JavaScript
- ▶ JavaScript is basically a template based scripting language not class based
- ▶ Hence, we directly create the object without class
- ▶ **Creating objects on JavaScript:**
 - ▶ By Object Literal
 - ▶ By creating instance of Object directly
 - ▶ By using an object constructor

Cont...

1. JavaScript Object by Object Literal:

1. Syntax:

```
object={property1:value1, property2:value2,...,propertyn:valuen}
```

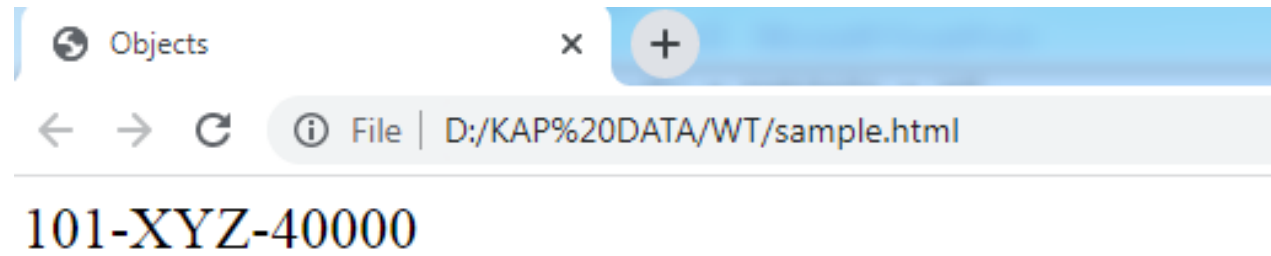
Program: Write a sample program of creating object using object literal

Cont...

```
<html>
  <head>
    <title> Objects </title>
  </head>
  <body>
    <font size=5>
      <script language="JavaScript">
        employee={id:101, name:"XYZ", salary:40000}
        document.write(employee.id+"-"+employee.name+"-"+employee.salary)
      </script>
    </font>
  </body>
</html>
```

Cont...

► Output:



Cont...

2. By Creating instance of Object:

Syntax:

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

new keyword is used to create new object.

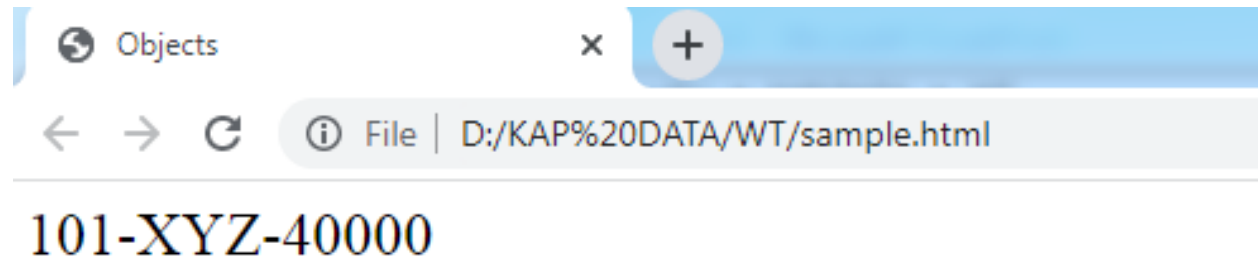
Program: Write a simple program of creating the instance of object

Cont...

```
<html>
  <head>
    <title> Objects </title>
  </head>
  <body>
    <font size=5>
      <script language="JavaScript">
        var employee=new Object();
        employee.id=101;
        employee.name="XYZ";
        employee.salary=40000;
        document.write(employee.id+"-"+employee.name+"-"+employee.salary)
      </script>
    </font>
  </body>
</html>
```

Cont...

► Output:



Cont...

3. By Using an Object Constructor:

- We have to create parameterized function.
- “this” keyword is used to assign each argument value in the current object
- this: refers to current object

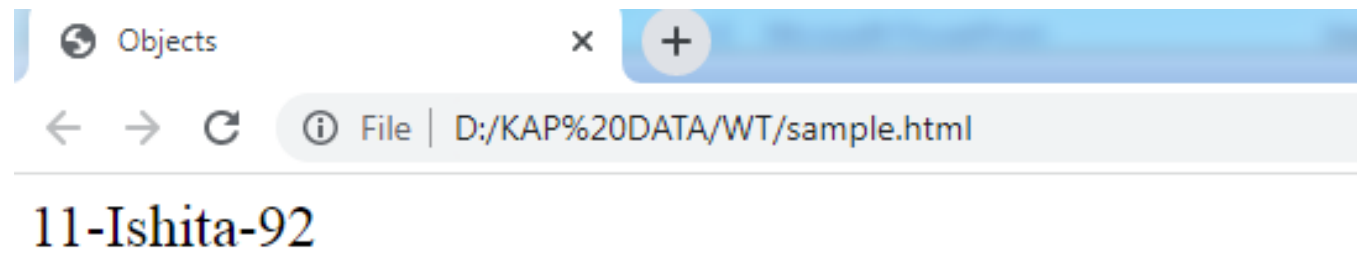
Program: Write a program with the use of this keyword

Cont...

```
<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(11,"Ishita",92)
        document.write(s.id+"-"+s.sname+"-"+s.marks);
      </script>
    </font>
  </body>
</html>
```

Cont...

► Output:



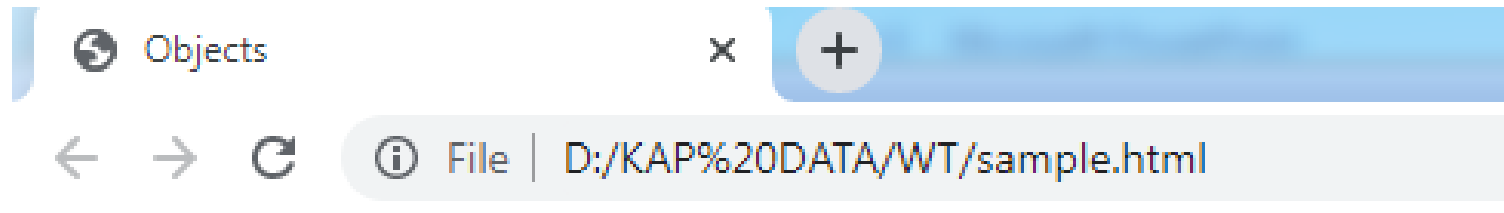
Defining Method in JavaScript Object

- ▶ Defining method in JavaScript object
- ▶ We have to add property in the method with same name as of the method name
- ▶ Program: Write a program to define the method in JavaScript

```
<html>
  <head>
    <title> Objects </title>
  </head>
  <body>
    <font size=5>
      <script language="JavaScript">
        function employee(id,sname,salary)
        {
          this.id=id;
          this.sname=sname;
          this.salary=salary;
          this.changeSalary=changeSalary;
          function changeSalary(otherSalary)
          {
            this.salary=otherSalary;
          }
        }
        e=new employee(101,"Ishita",50000);
        document.write(e.id+"-"+e.sname+"-"+e.salary);
        e.changeSalary(55000);
        document.write("<br>"+e.id+"-"+e.sname+"-"+e.salary);
      </script>
    </font>
  </body>
</html>
```

Cont...

► Output:



101-Ishita-50000

101-Ishita-55000

ARRAYS

- ▶ 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 number are used to access the specific array element
- ▶ Syntax:

```
var sname=new Array("Kunal","Ishita",Shrey","Shravani");
```

- You access an array element by referring to the **index number**:

```
const cars = ["Saab", "Volvo", "BMW"];  
car = cars[0];
```

Cont...

► 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 diff=[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"];
```

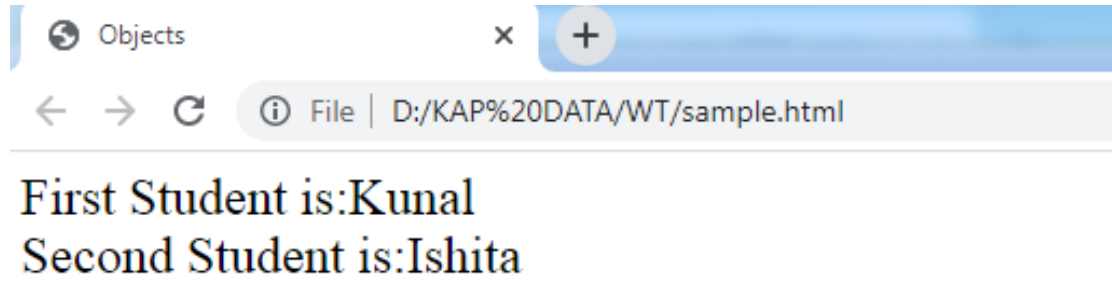
Cont...

- Program: Write a simple program of array displaying students name

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

Cont...

► Output:



Array Methods

Method	Description
<u>concat()</u>	Joins two or more arrays, and returns a copy of the joined arrays
<u>copyWithin()</u>	Copies array elements within the array, to and from specified positions
<u>entries()</u>	Returns a key/value pair Array Iteration Object
<u>every()</u>	Checks if every element in an array pass a test
<u>fill()</u>	Fill the elements in an array with a static value
<u>filter()</u>	Creates a new array with every element in an array that pass a test
<u>find()</u>	Returns the value of the first element in an array that pass a test

Cont...

[findIndex\(\)](#)

Returns the index of the first element in an array that pass a test

[forEach\(\)](#)

Calls a function for each array element

[from\(\)](#)

Creates an array from an object

[includes\(\)](#)

Check if an array contains the specified element

[indexOf\(\)](#)

Search the array for an element and returns its position

[isArray\(\)](#)

Checks whether an object is an array

[join\(\)](#)

Joins all elements of an array into a string

[keys\(\)](#)

Returns a Array Iteration Object, containing the keys of the original array

[lastIndexOf\(\)](#)

Search the array for an element, starting at the end, and returns its position

[map\(\)](#)

Creates a new array with the result of calling a function for each array element

[pop\(\)](#)

Removes the last element of an array, and returns that element

Cont...

[push\(\)](#)

Adds new elements to the end of an array, and returns the new length

[reduce\(\)](#)

Reduce the values of an array to a single value (going left-to-right)

[reduceRight\(\)](#)

Reduce the values of an array to a single value (going right-to-left)

[reverse\(\)](#)

Reverses the order of the elements in an array

[shift\(\)](#)

Removes the first element of an array, and returns that element

[slice\(\)](#)

Selects a part of an array, and returns the new array

[some\(\)](#)

Checks if any of the elements in an array pass a test

[sort\(\)](#)

Sorts the elements of an array

[splice\(\)](#)

Adds/Removes elements from an array

[toString\(\)](#)

Converts an array to a string, and returns the result

[unshift\(\)](#)

Adds new elements to the beginning of an array, and returns the new length

[valueOf\(\)](#)

Returns the primitive value of an array

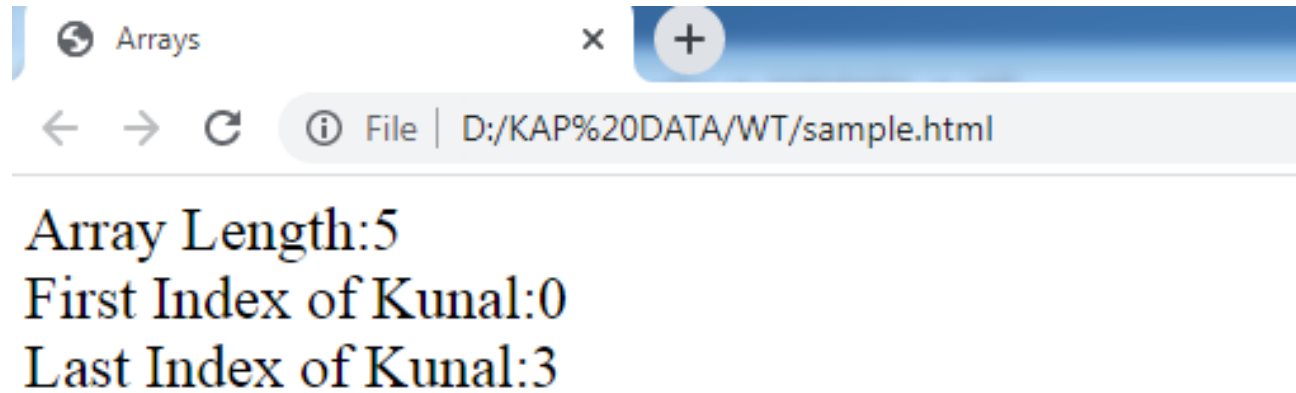
Cont...

- Program: 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 student=new Array("Kunal","Ishita","Shrey","Kunal","Shravi");
        document.write("Array Length:"+student.length);
        document.write("<br> First Index of Kunal:"+student.indexOf("Kunal"));
        document.write("<br> Last Index of Kunal:"+student.lastIndexOf("Kunal"));
      </script>
    </font>
  </body>
</html>
```


Cont...

► Output:



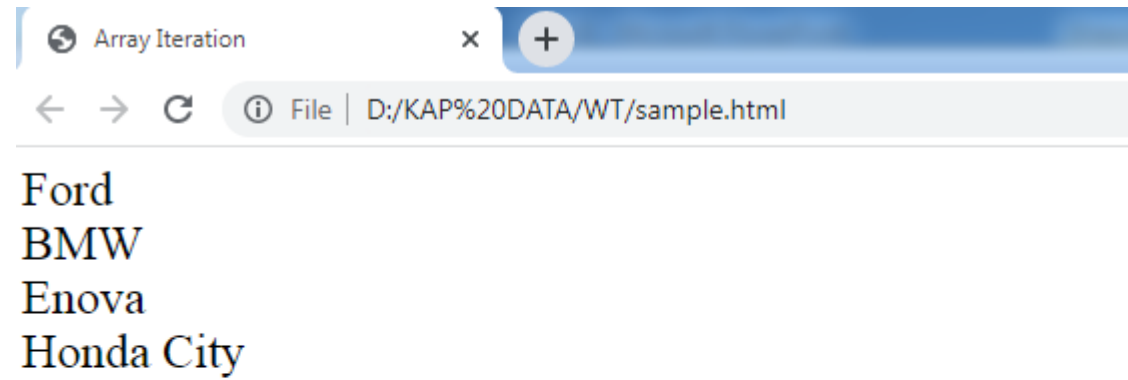
Iterating through an Array

- Program: 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>
```

Cont...

► Output:



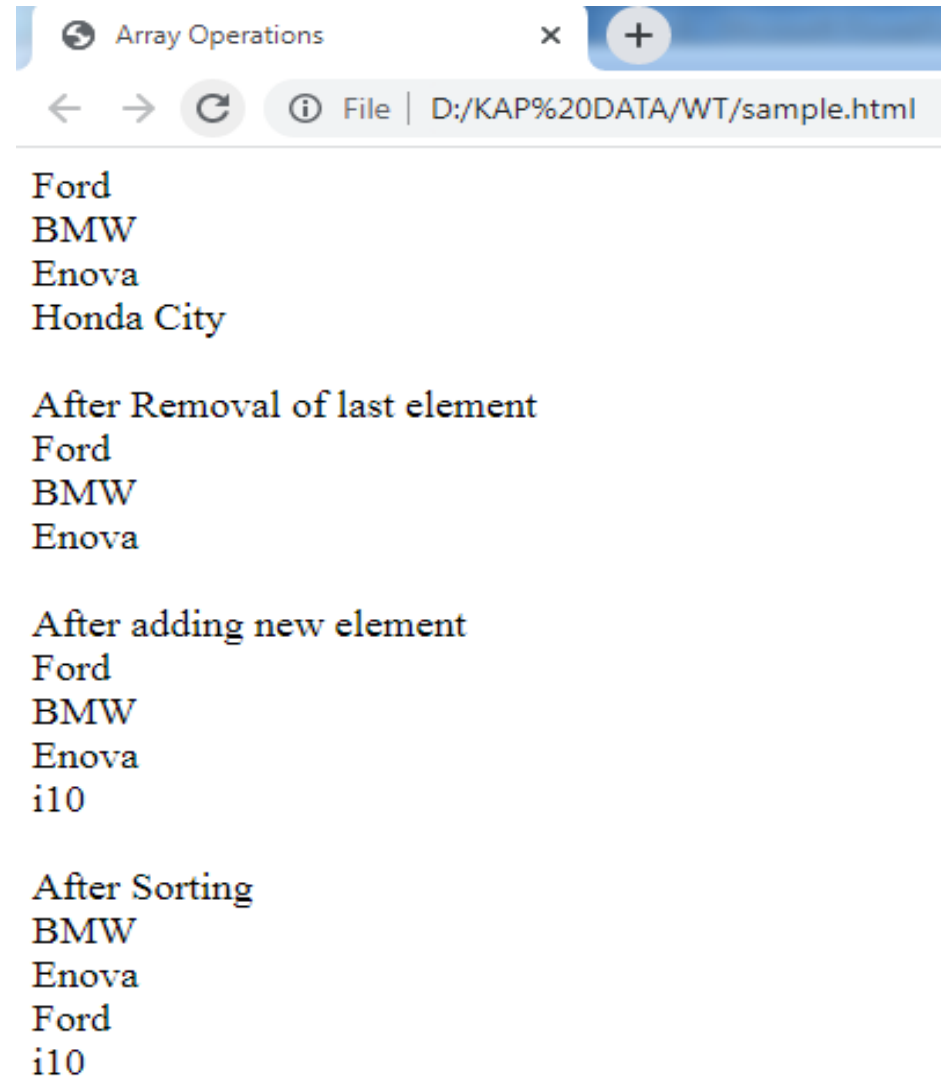
Program: Write a program to perform various methods like adding a new element, sorting, reversing, removal of last element etc.

```
<html>
  <head>
    <title> Array Operations</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>");
        }
      </script>
    </font>
  </body>
</html>
```

```
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>
```

Cont...

► Output:



The screenshot shows a web browser window with the title "Array Operations". The address bar displays the file path "D:/KAP%20DATA/WT/sample.html". The main content area of the browser shows the following text:

```
Ford
BMW
Enova
Honda City

After Removal of last element
Ford
BMW
Enova

After adding new element
Ford
BMW
Enova
i10

After Sorting
BMW
Enova
Ford
i10
```

Deleting Element from an Array

- ▶ 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
```

Array method: Splice()

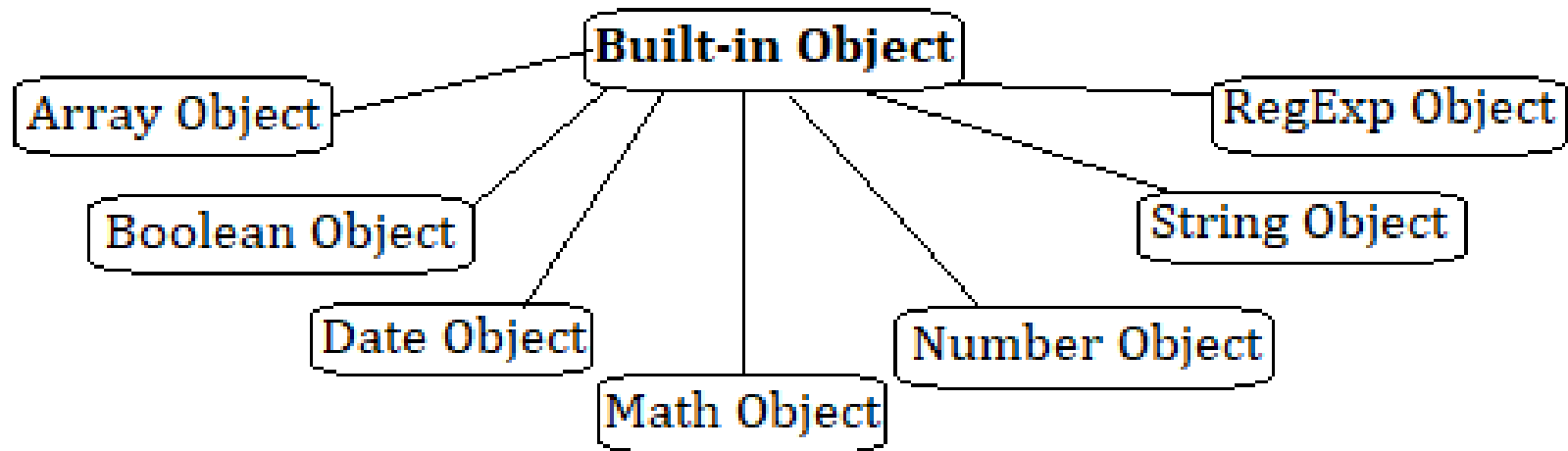
- ▶ The splice() method is used to insert new, delete existing and replace existing elements by new elements in the array
- ▶ It moves the elements to higher or lower positions as per the requirement to avoid any gap
- ▶ The first argument of splice() indicates the starting position and second argument indicates the number of elements to delete
- ▶ Example:

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


Cont...

- ▶ The third argument in the splice method is used to replace one or more elements with others
- ▶ `var letters=["a","b","c","d","e","f","g"];`
- ▶ `Alert(letters.splice(1,2,,"e","f","g"));` //b,c (deleted ones)
- ▶ `Alert(letters);` //a,e,f,g,d
- ▶ The splice start at position 1 and removes two elements b and c
- ▶ Then it fills the gap with the three elements e,f, and g

Built in Objects in JavaScript



JavaScript Debuggers

- ▶ Debugging is not easy. But fortunately, all modern browsers have a built-in JavaScript debugger.
- ▶ Built-in debuggers can be turned on and off, forcing errors to be reported to the user.
- ▶ With a debugger, you can also set breakpoints (places where code execution can be stopped), and examine variables while the code is executing.
- ▶ Normally, otherwise follow the steps at the bottom of this page, you activate debugging in your browser with the F12 key, and select "Console" in the debugger menu.
- ▶ The `log()` method writes (logs) a message to the console.
- ▶ The `log()` method is useful for testing purposes.

The console.log() Method

- If your browser supports debugging, you can use console.log() to display JavaScript values in the debugger window:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1>My First Web Page</h1>
```

```
<script>
```

```
a = 5;
```

```
b = 6;
```

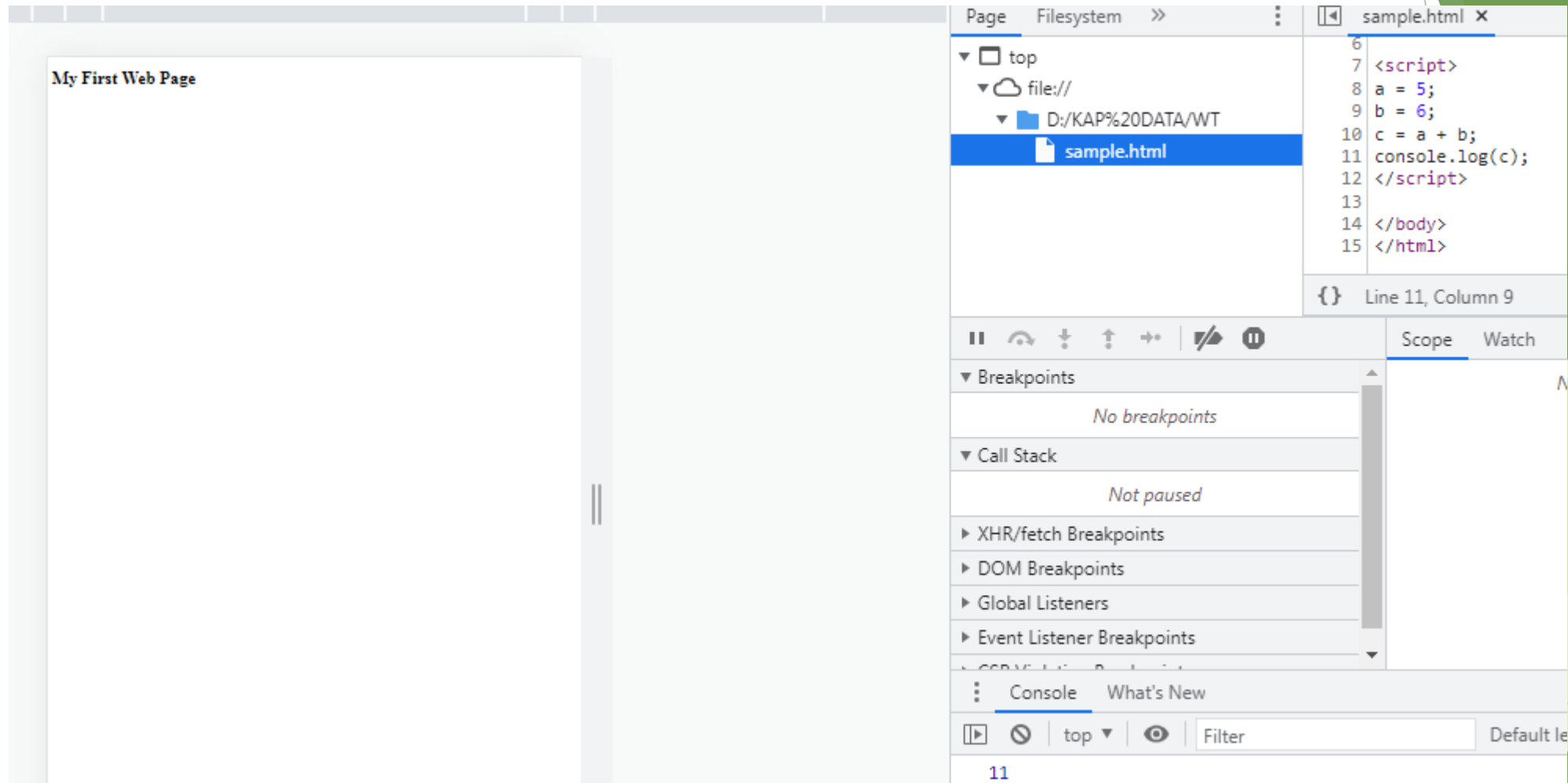
```
c = a + b;
```

```
console.log(c);
```

```
</script>
```

```
</body>
```

```
</html>
```



Setting Breakpoints

- ▶ In the debugger window, you can set breakpoints in the JavaScript code.
- ▶ At each breakpoint, JavaScript will stop executing, and let you examine JavaScript values.
- ▶ After examining values, you can resume the execution of code (typically with a play button).

The debugger Keyword

- ▶ The debugger keyword stops the execution of JavaScript, and calls (if available) the debugging function.
- ▶ This has the same function as setting a breakpoint in the debugger.
- ▶ If no debugging is available, the debugger statement has no effect.
- ▶ With the debugger turned on, this code will stop executing before it executes the third line.
- ▶

```
let x = 15 * 5;  
debugger;  
document.getElementById("demo").innerHTML = x;
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>JavaScript Debugger</h2>
```

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

<p>With the debugger turned on, the code below should stop executing before it executes the third line.</p>

```
<script>
```

```
let x = 15 * 5;
```

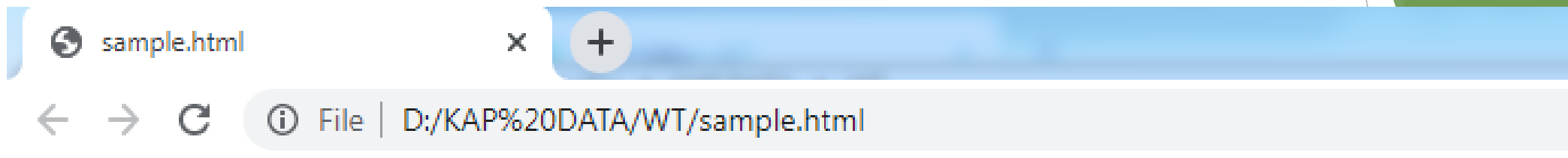
```
debugger;
```

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

```
</script>
```

```
</body>
```

```
</html>
```

JavaScript Debugger

75

With the debugger turned on, the code below should stop executing before it executes the third line.

Cont...

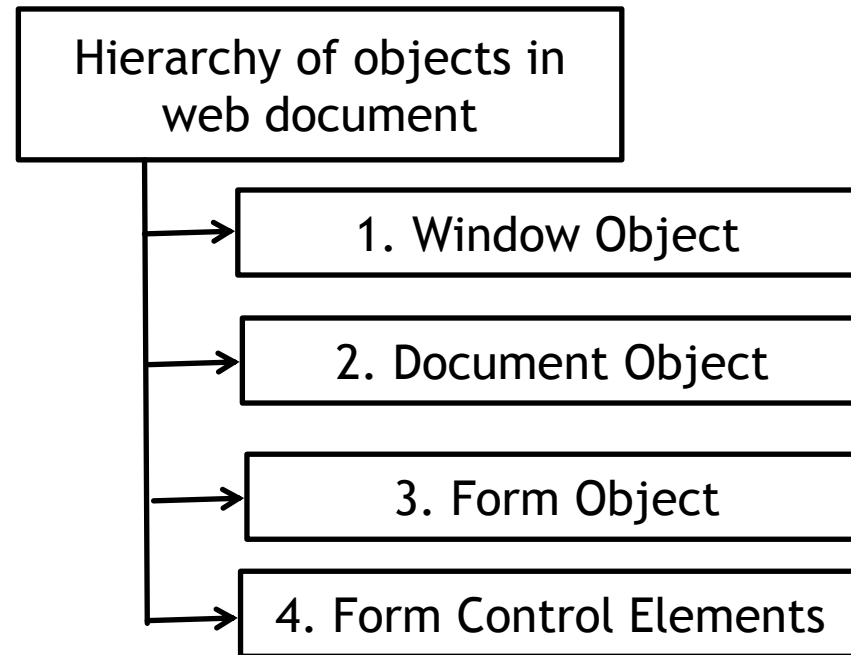
- ▶ Debugging is the process of testing, finding, and reducing bugs (errors) in computer programs.
- ▶ The first known computer bug was a real bug (an insect) stuck in the electronics.

DOM

- ▶ **DOM- Document Object Model**
- ▶ The entire html document is represented by the document object
- ▶ The html document becomes document object when it is loaded in the browser
- ▶ The **root element** represents the html document
- ▶ the document object helps to add content dynamically in the web page
- ▶ Any element of HTML page can be accessed by using the document object
- ▶ According to W3C(World Wide Web Consortium)- the W3C Document Object Model(DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of a document

Hierarchy of Objects in Web Document

- DOM is the method by which the content of document is accessed and modified.



Cont...

1. Window Object:

It resides at top of the hierarchy. It is the topmost element of the object hierarchy

2. Document Object:

All the html documents which get loaded into browser are considered as document objects. The contents of the page are stored in the document object

3. Form Object:

Everything which is contained in the opening `<form>` and closing `</form>` tag sets the form object

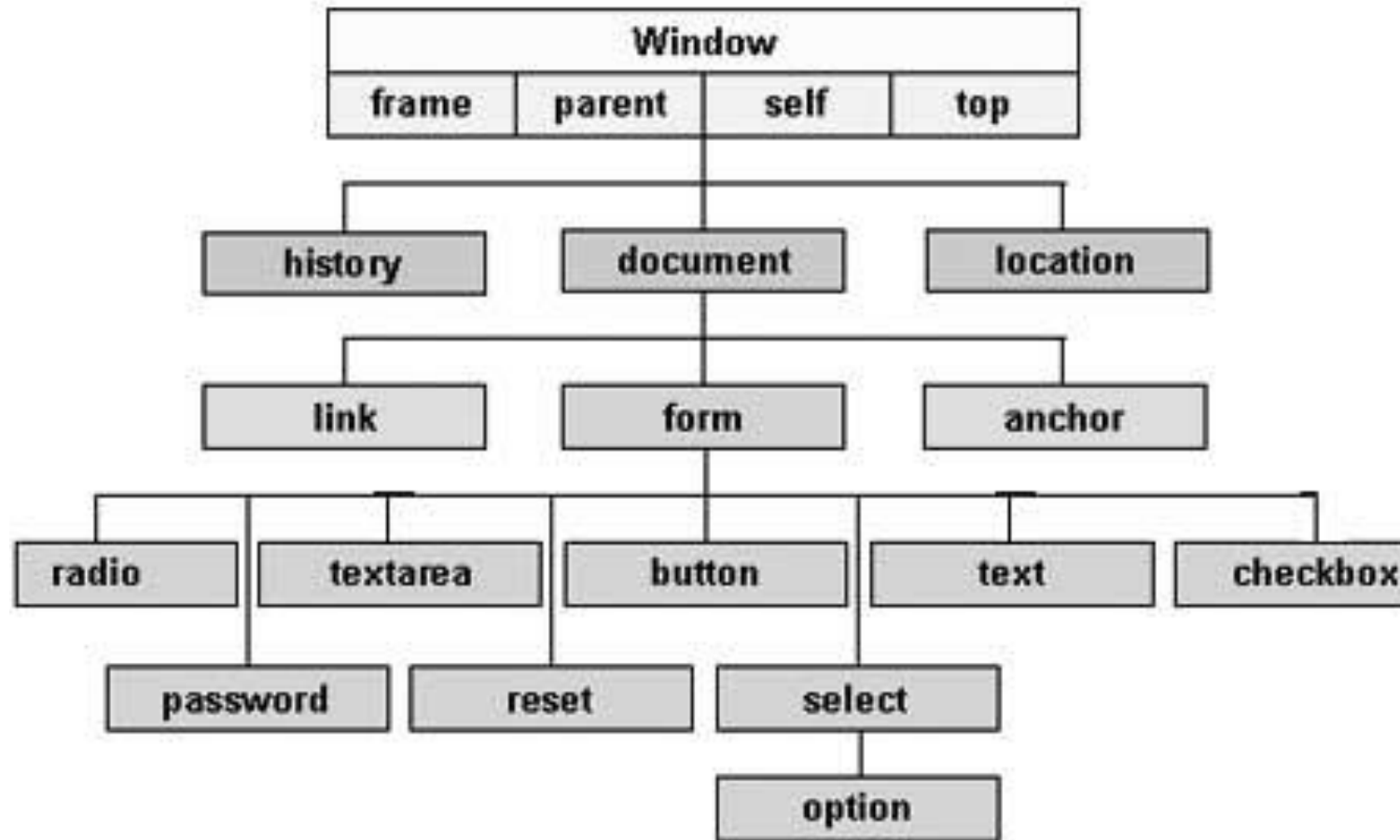
4. Form Control Elements:

The form object has all the elements which are defined for the objects like text fields, buttons, checkboxes, radio buttons etc.

DOM Level

- ▶ The DOM provides all the features to JavaScript to create dynamic HTML:
 - ▶ Changes can be made in all HTML elements
 - ▶ Changes can be made in attributes of HTML elements
 - ▶ Changes can be made in all CSS styles in the page
 - ▶ Existing HTML elements and attributes can be deleted
 - ▶ New HTML elements and attributes can be added
 - ▶ Responses can be given to HTML events
 - ▶ New HTML events can be created in HTML page

Cont...



Properties and Methods of Document Object

Sr. No.	Method/Properties	Description
1	Value	Returns value of specified text
2	Write("String")	Writes the specified string in the document
3	WriteLn("string")	Writes the specified string in the document with newline character at the end
4	getElementById()	Returns the element having the specified id value
5	getElementsByName()	Returns all the elements having the specified name value
6	getElementsByTagName()	Returns all the elements having the specified tag name
7	getElementsByClassName()	Returns all the elements having the specified class name

Accessing field value by document object

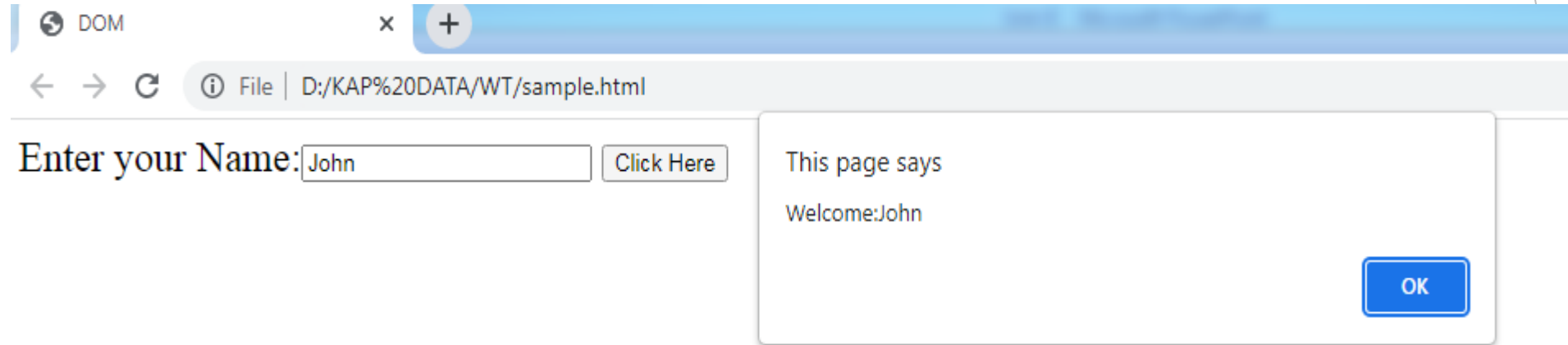
- Write a program to accept value from user with the help of text field and access it using the document.from1.name.value

```
<!DOCTYPE html>
<html>
  <head>
    <title> DOM </title>
    <script language="JavaScript">
      function callme()
      {
        var sname=document.form1.txtname.value;
        alert("Welcome:"+sname)
      }
    </script>
  </head>
```

Cont...

```
<body>
  <font size=5>
    <form name="form1">
      Enter your Name:<input type="text" name="txtname"/>
      <input type="button" onClick="callme();" value="Click Here"/>
    </form>
  </font>
</body>
</html>
```

Output

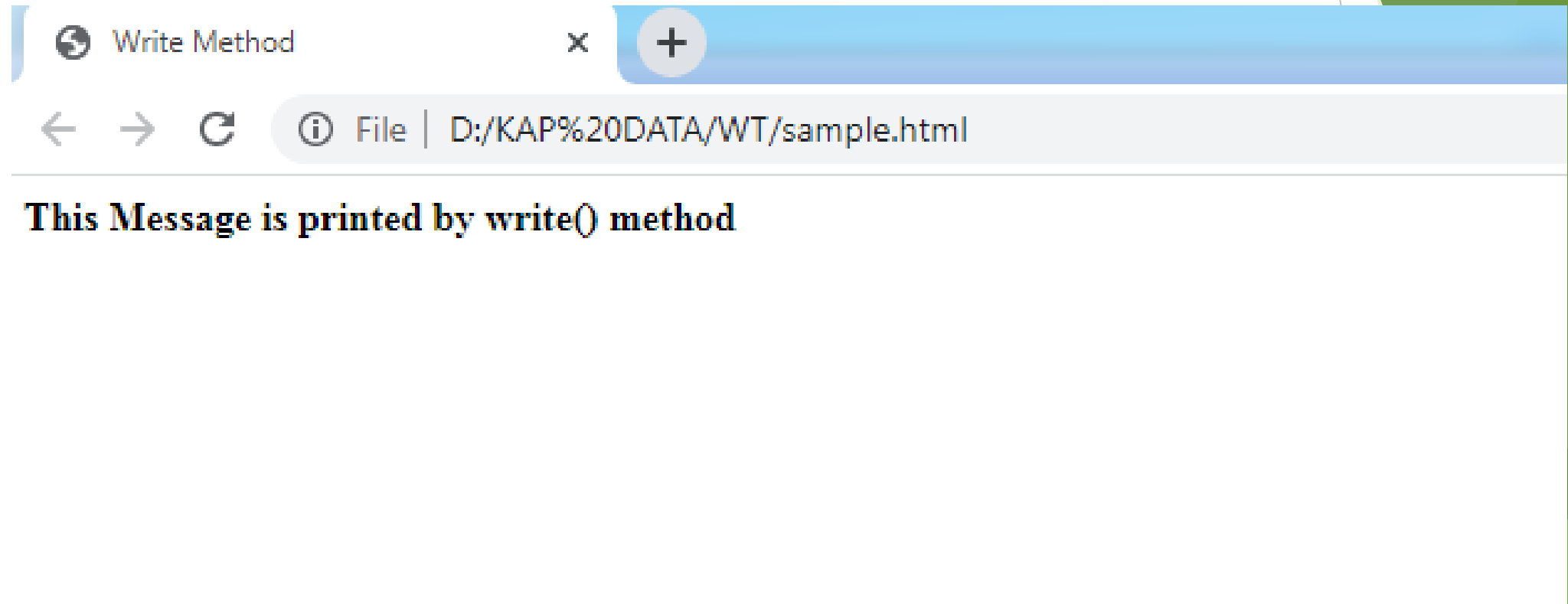


Write("string") method

- Write a program to display the message using write("string") method

```
<!DOCTYPE html>
<html>
  <head>
    <title> Write Method </title>
  </head>
  <body>
    <script language="JavaScript">
      document.write("<b> This Message is printed by write() method </b>");
    </script>
  </body>
</html>
```

Output



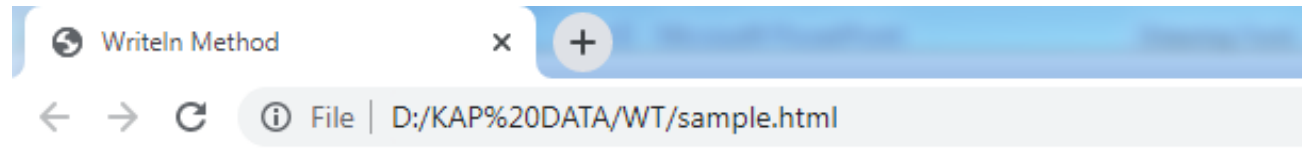
Writeln(“string”) method

- ▶ This method is used to write output stream
- ▶ Writeln() add a new line after each statement

```
<!DOCTYPE html>
<html>
  <head>
    <title> WriteLn Method </title>
  </head>
  <body>
    <font size=5>
      <p> Note: write() does not add a new line after each statement </p>
    <pre>
    <script language="JavaScript">
      document.write("First Line");
      document.write("... Also on the First Line");
    </script>
    </pre>
      <p> Note: writeln() add a new line after each statement </p>
    <pre>
    <script>
      document.writeln("First Line");
      document.writeln("Second Line");
    </script>
    </pre>
    </font>
  </body>
</html>
```

Cont...

► Output:



Note: write() does not add a new line after each statement

First Line... Also on the First Line

Note: writeln() add a new line after each statement

First Line

Second Line

document.getElementById() method

- ▶ This method returns the element of specified id
- ▶ Syntax:

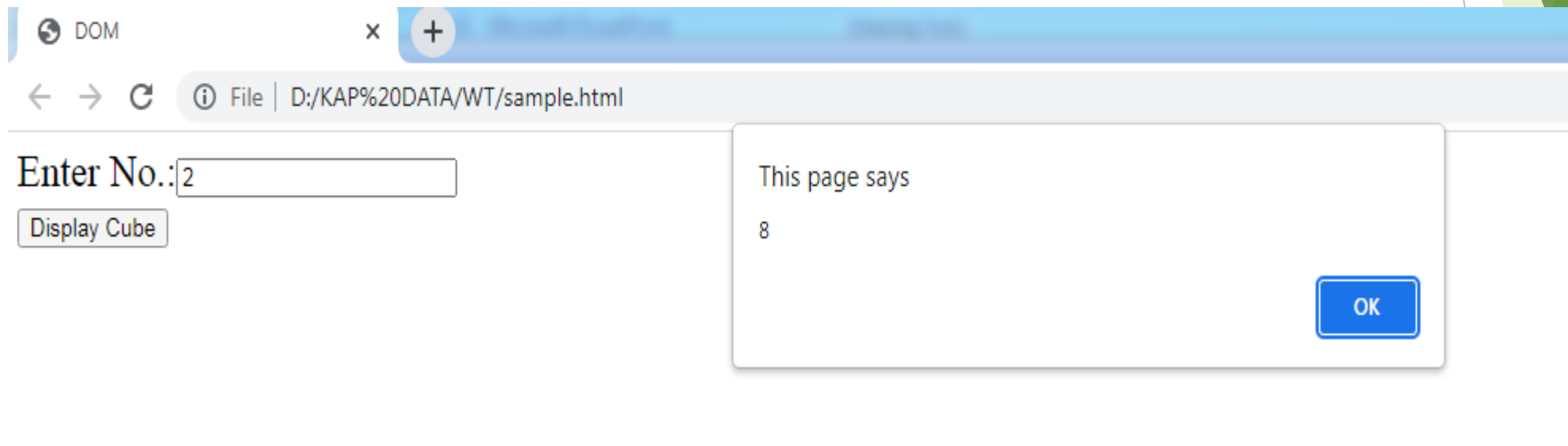
```
document.getElementById("id");
```

Program: Display the cube of number using document.getElementById() method

```
<!DOCTYPE html>
<html>
  <head>
    <title> DOM </title>
    <script language="JavaScript">
      function cube()
      {
        var no=document.getElementById("num").value;
        alert(no*no*no);
      }
    </script>
  </head>
  <body>
    <font size=5>
      <form>
        Enter No.:<input type="text" id="num" name="num"/><br/>
        <input type="button" value="Display Cube" onClick="cube()"/>
      </form>
    </font>
  </body>
</html>
```

Cont...

► Output:



document.getElementsByName() method

- ▶ This method returns all the elements of given name

- ▶ Syntax:

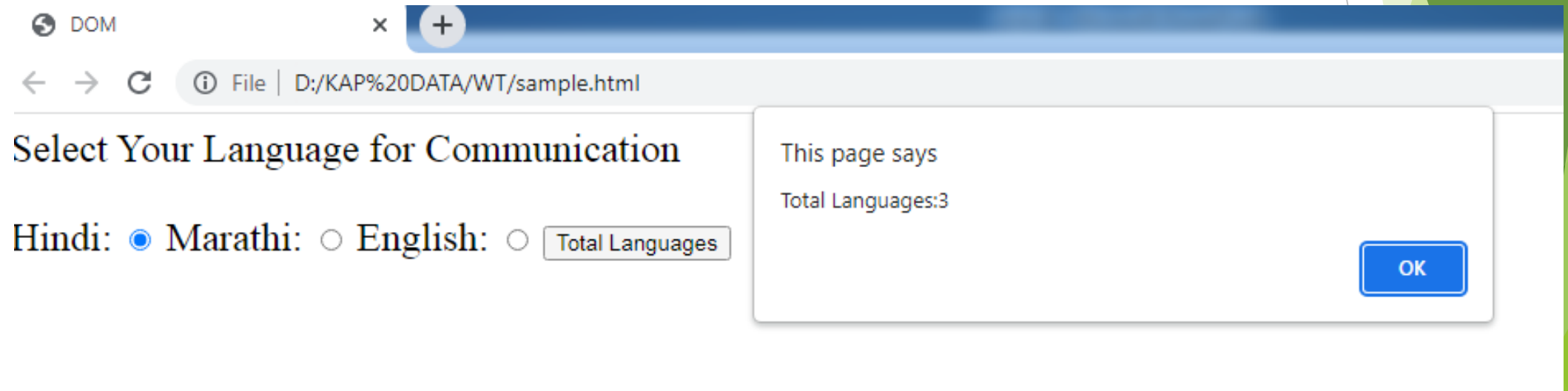
```
document.getElementsByName("name") //count total number of languages  
//get all the languages
```

Program: Write a program to display number of languages available using radio buttons. Display the count of languages on click of a button

```
<!DOCTYPE html>
<html>
  <head>
    <title> DOM </title>
    <script type="text/javascript">
      function languages()
      {
        var lan=document.getElementsByName("lang");
        alert("Total Languages:"+lan.length);
      }
    </script>
  </head>
  <body>
    <font size=5>
      <form>
        Select Your Language for Communication</br></br>
        Hindi: <input type="radio" name="lang" value="Hindi">
        Marathi: <input type="radio" name="lang" value="Marathi">
        English: <input type="radio" name="lang" value="English">
        <input type="button" onClick="languages()" value="Total Languages">
      </form>
    </font>
  </body>
</html>
```

Cont...

► Output:



document.getElementsByTagName() method

- ▶ All the elements of specified tag are returned by this method
- ▶ Syntax:

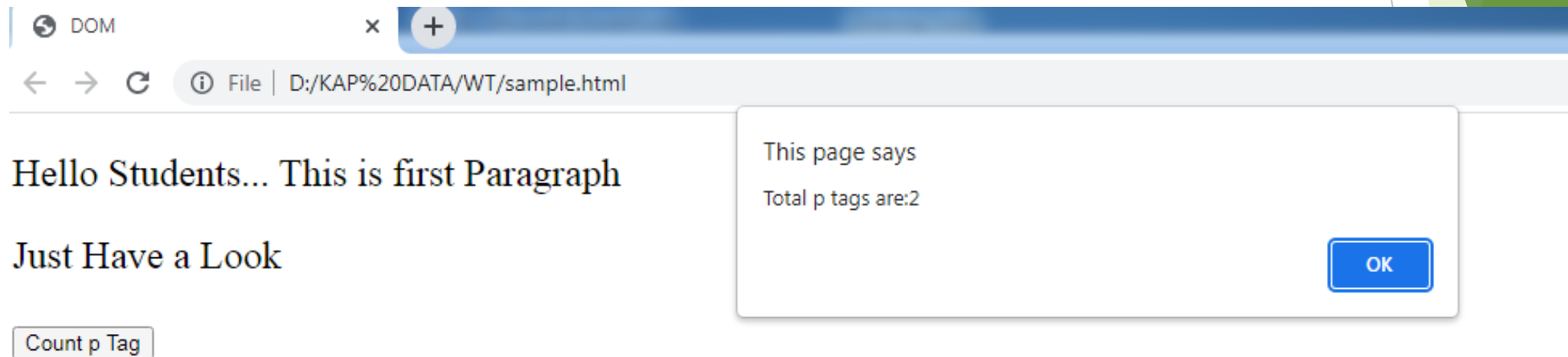
```
document.getElementsByTagName("tag_name")
```

Program: Write a program to count total number of tags used by
document.getElementsByTagName() method

```
<!DOCTYPE html>
<html>
  <head>
    <title> DOM </title>
    <script type="text/javascript">
      function cntp()
      {
        var cnt=document.getElementsByTagName("p");
        alert("Total p tags are:"+cnt.length);
      }
    </script>
  </head>
  <body>
    <font size=5>
      <form>
        <p> Hello Students... This is first Paragraph </p>
        <p> Just Have a Look </p>
        <button onClick="cntp()"> Count p Tag </button>
      </form>
    </font>
  </body>
</html>
```


Cont...

► Output:



document.getElementsByClassName() method

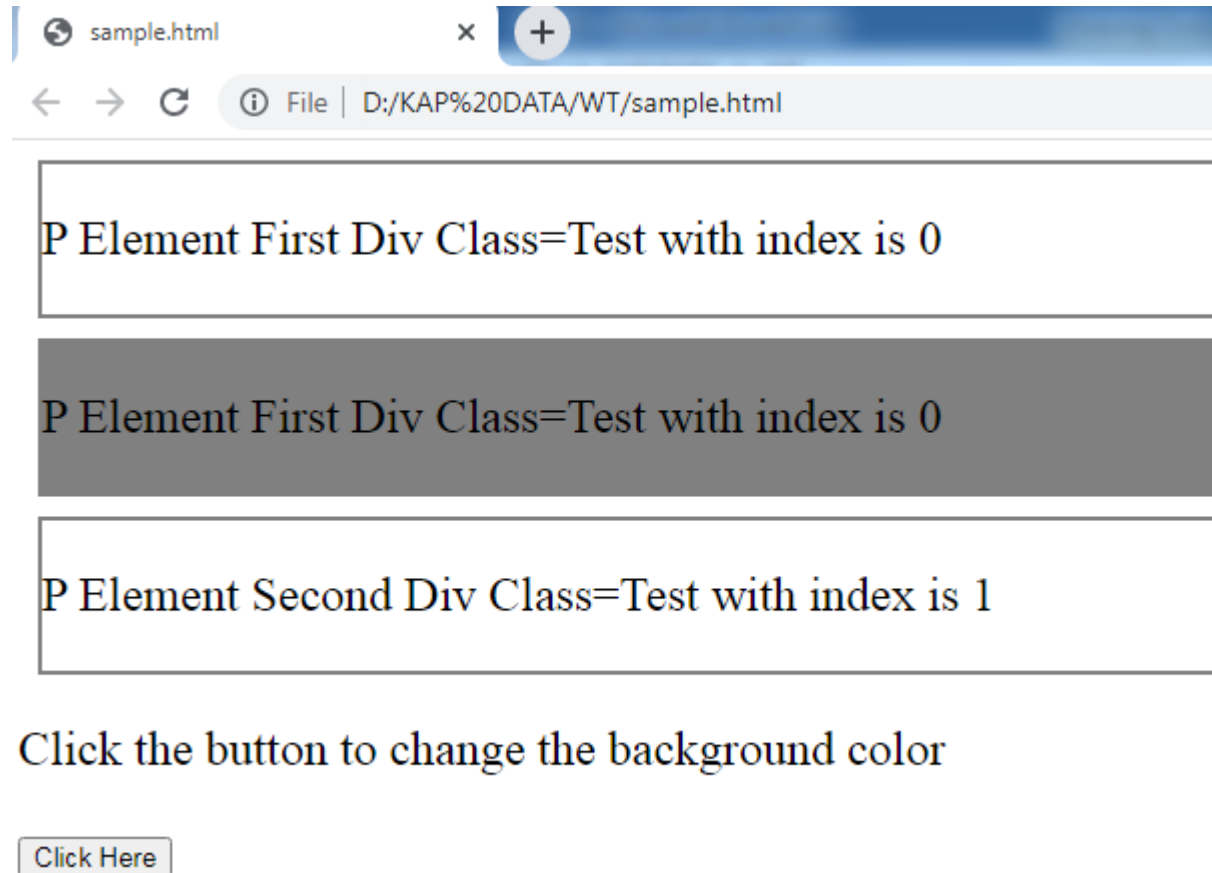
- ▶ This method returns a set of all the elements of the document with the given class name, as a NodeList object
- ▶ The NodeList object issued to represent a set of nodes
- ▶ The nodes are accessed using index numbers which starts from 0
- ▶ Program: Write a program to demonstrate the use of `getElementsByClassName()` method

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      div{
        border:2px solid gray;
        margin:10px;
      }
    </style>
  </head>
  <body>
    <font size=5>
```

```
<div class="Test">
    <p> P Element First Div Class=Test with index is 0 </p>
</div>
<div class="Test Color">
    <p> P Element First Div Class=Test with index is 0 </p>
</div>
<div class="Test Color">
    <p> P Element Second Div Class=Test with index is 1 </p>
</div>
<p> Click the button to change the background color </p>
<button onClick="changecolor()"> Click Here </button>
<script language="JavaScript">
    function changecolor()
    {
        var x=document.getElementsByClassName("Test Color");
        x[0].style.backgroundColor="gray";
    }
</script>
</font>
</body>
</html>
```

Cont...

► Output:



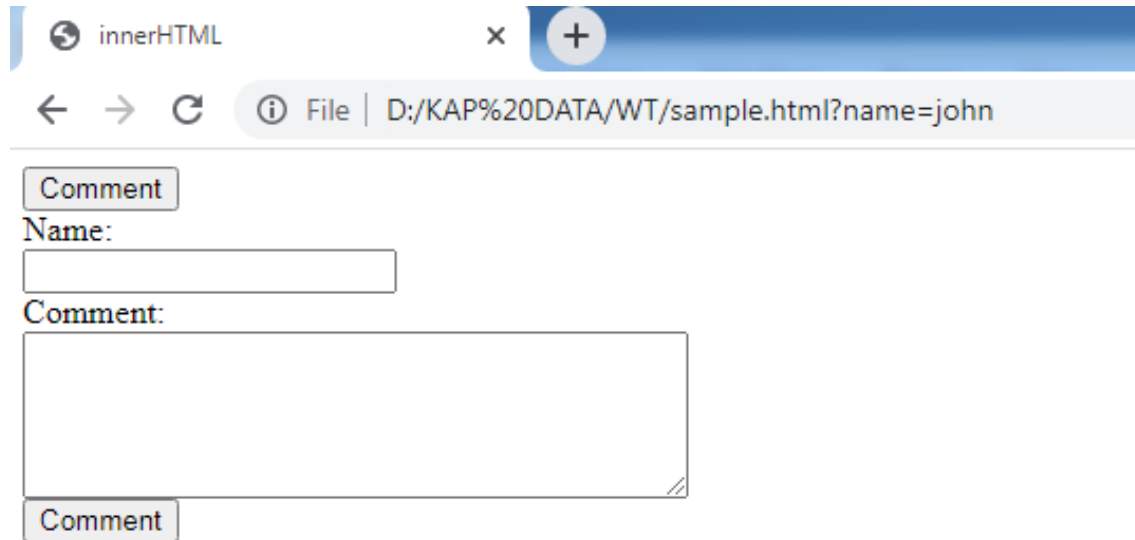
javaScript-innerHTML

- ▶ Dynamic html contents can be written in html document with the help of innerHTML property
- ▶ Used in the html documents to generate the dynamic contents like registration form, comment form, links etc.
- ▶ Program: Write a program to generate comment form using innerHTML property

```
<html>
  <head>
    <title> innerHTML </title>
  </head>
  <body>
    <script type="text/javascript">
      function disp()
      {
var data="Name:<br><input type='text' name='name'><br> Comment:<br><textarea rows='5'
cols='40'></textarea><br> <input type='Submit' value='Comment'>";
        document.getElementById('myloc').innerHTML=data;
      }
    </script>
    <form name="myform">
      <input type="button" value="Comment" onclick="disp()">
      <div id="myloc"> </div>
    </form>
  </body>
</html>
```

Cont...

► Output:



The screenshot shows a web browser window with a single tab titled 'innerHTML'. The address bar displays the file path 'D:/KAP%20DATA/WT/sample.html?name=john'. The page content includes a 'Comment' button at the top, followed by a 'Name:' label and a single-line text input field. Below this is a 'Comment:' label and a multi-line text area. At the bottom of the form is another 'Comment' button.

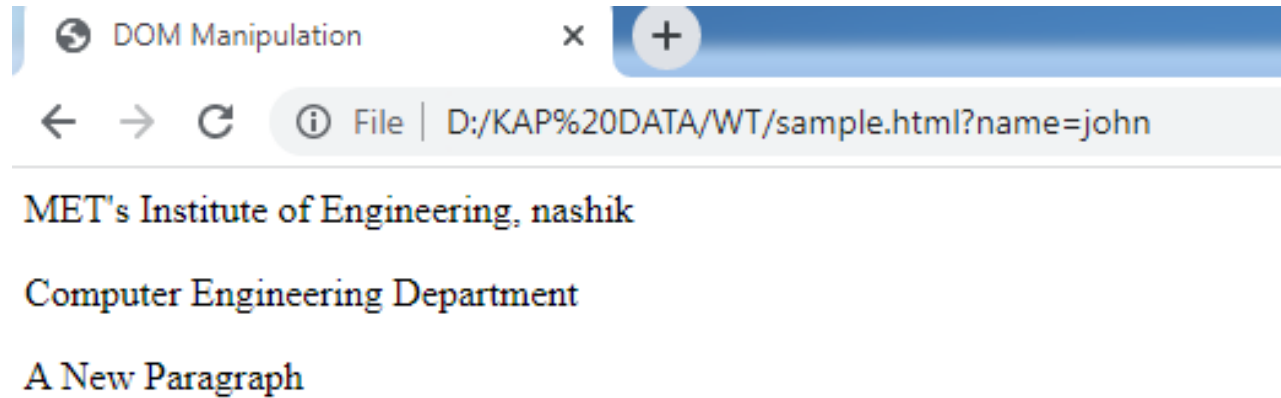
Manipulating DOM

- ▶ Using JavaScript we can add or remove nodes (HTML elements) in the document
- 1. Creating New HTML Elements (Nodes):
 - 1. Initially we have to create the element (element node) which can be then appended to the existing element
 - 2. Program: Write a program to append the new element to an existing element node

```
<html>
  <head>
    <title> DOM Manipulation </title>
    <div id="div1">
      <p id="p1"> MET's Institute of Engineering, nashik</p>
      <p id="p2"> Computer Engineering Department </p>
    </div>
  </head>
  <body>
    <script>
      var prg=document.createElement("p");
      var nd=document.createTextNode("A New Paragraph");
      prg.appendChild(nd);
      var element=document.getElementById("div1");
      element.appendChild(prg);
    </script>
  </body>
</html>
```

Cont...

► Output:



Cont...

2. Creating New HTML Elements- insertBefore():

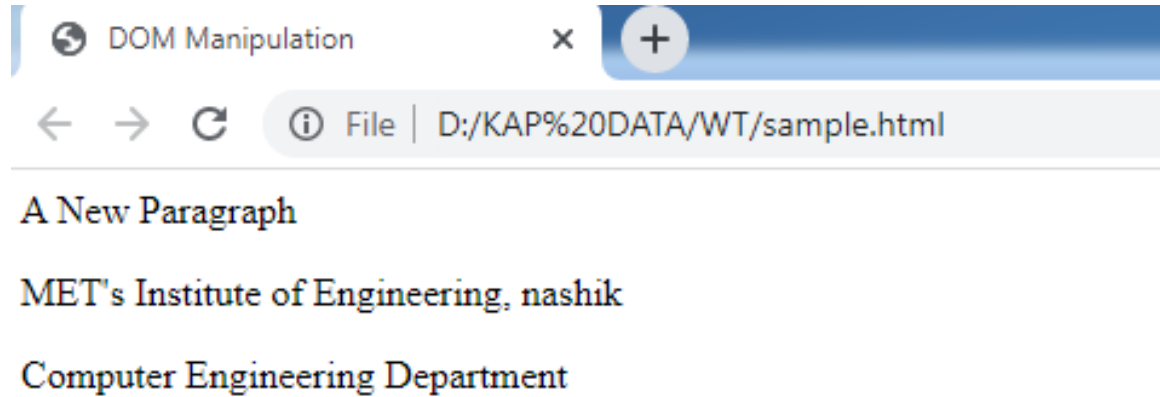
- used to add the new element at the beginning

Program: Write a program to add new element at the beginning using insertBefore() method

```
<html>
  <head>
    <title> DOM Manipulation </title>
    <div id="div1">
      <p id="p1"> MET's Institute of Engineering, nashik</p>
      <p id="p2"> Computer Engineering Department </p>
    </div>
  </head>
  <body>
    <script>
      var prg=document.createElement("p");
      var nd=document.createTextNode("A New Paragraph");
      prg.appendChild(nd);
      var element=document.getElementById("div1");
      var child=document.getElementById("p1");
      element.insertBefore(prg,child);
    </script>
  </body>
</html>
```

Cont...

► Output:



Cont...

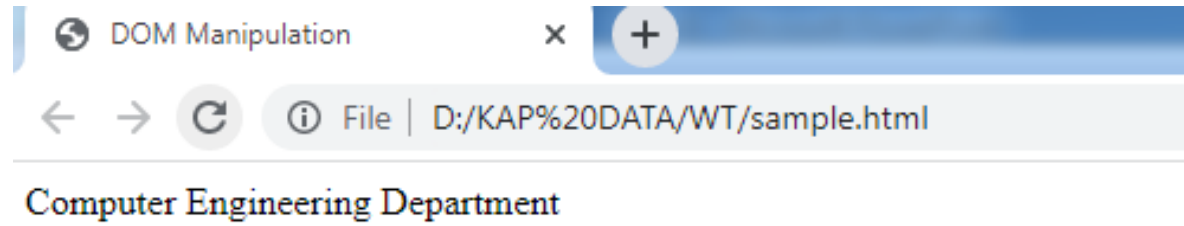
3. Removing Existing HTML Elements:

- to remove html element we should have the information about the parent element
- the `removeChild()` method is used to remove the html element
- Program: Write a program to remove an existing element using `removeChild()` method

```
<html>
  <head>
    <title> DOM Manipulation </title>
    <div id="div1">
      <p id="p1"> MET's Institute of Engineering, nashik</p>
      <p id="p2"> Computer Engineering Department </p>
    </div>
  </head>
  <body>
    <script>
      var parent=document.getElementById("div1");
      var child=document.getElementById("p1");
      parent.removeChild(child);
    </script>
  </body>
</html>
```


Cont...

► Output:



Cont...

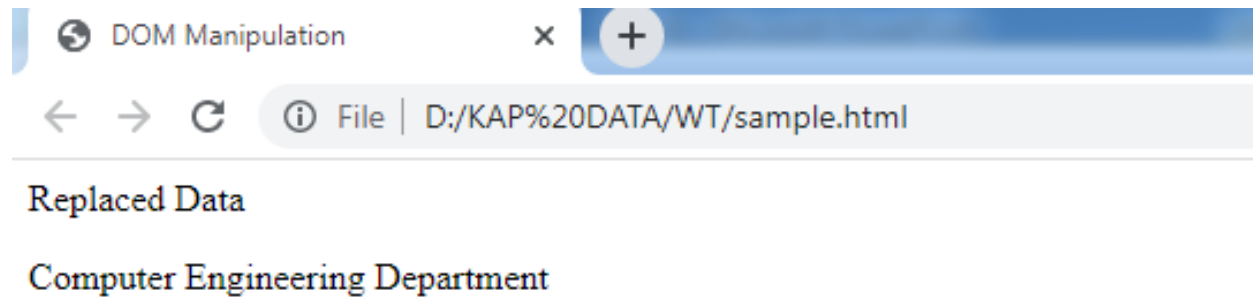
4. Replacing HTML Elements:

- Replacing an existing element by new one
- The `replaceChild()` method is used for this purpose
- Program: Write a program to replace an existing element by new element using `replaceChild()` method

```
<html>
  <head>
    <title> DOM Manipulation </title>
    <div id="div1">
      <p id="p1"> MET's Institute of Engineering, nashik</p>
      <p id="p2"> Computer Engineering Department </p>
    </div>
  </head>
  <body>
    <script>
      var prg=document.createElement("p");
      var nd=document.createTextNode("Replaced Data");
      prg.appendChild(nd);
      var parent=document.getElementById("div1");
      var child=document.getElementById("p1");
      parent.replaceChild(prg,child);
    </script>
  </body>
</html>
```

Cont...

► Output:



Intrinsic Event Handling

▶ JavaScript Events

- ▶ HTML events are "**things**" that happen to HTML elements.
- ▶ When JavaScript is used in HTML pages, JavaScript can "**react**" on these events.

▶ HTML Events

- ▶ An HTML event can be something the browser does, or something a user does.
- ▶ Here are some examples of HTML events:
 - ▶ An HTML web page has finished loading
 - ▶ An HTML input field was changed
 - ▶ An HTML button was clicked
- ▶ Often, when events happen, you may want to do something.
- ▶ JavaScript lets you execute code when events are detected.
- ▶ HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

Cont...

► Event Handler:

- Event handler is a script that gets executed in response to these events
- This event handler enables the web document to respond the user activities through the browser window

Event Handler	Description
onAbort	It executes when the user aborts loading an image
onBlur	It executes when the input focus leaves the field of a text,, textarea or a slelect option
onChange	It executes when the input focus exists the field after the user modifies its text
onClick	In this, a function is called when an object in a button is clicked, a link is pushed, a checkbox is checked or an image map is selected. It can return false to cancel the action

Cont...

Event Handler	Description
onError	It executes when an error occurs while loading a document or an image
onFocus	It executes when input focus enters the field by tabbing in or by clicking but not selecting input from the field
onLoad	It executes when a window or image finishes loading
onMouseOver	The JavaScript code is called when the mouse is placed over a specific link or an object
onMouseOut	The JavaScript code is called when the mouse leaves a specific link or an object
onReset	It executes when the user resets a form by clicking on the reset button
onSelect	It executes when the user selects some of the text within a text or textarea field
onSubmit	It calls when the form is submitted

Cont...

- ▶ Program: illustrate the use of onLoad event handler

```
<!DOCTYPE html>
<html>
<body onload="myFunction()">

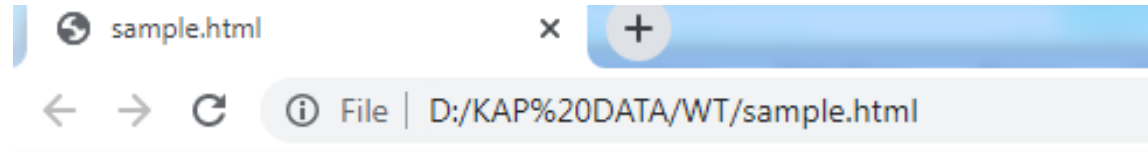
<h1>Hello World!</h1>

<script>
function myFunction()
{
    alert("Page is loaded");
}
</script>

</body>
</html>
```


Cont...

► Output:



Hello World!

Event	Description	Belongs To
<u>abort</u>	The event occurs when the loading of a media is aborted	<u>UiEvent</u> , <u>Event</u>
<u>afterprint</u>	The event occurs when a page has started printing, or if the print dialogue box has been closed	<u>Event</u>
<u>animationend</u>	The event occurs when a CSS animation has completed	<u>AnimationEvent</u>
<u>animationiteration</u>	The event occurs when a CSS animation is repeated	<u>AnimationEvent</u>
<u>animationstart</u>	The event occurs when a CSS animation has started	<u>AnimationEvent</u>
<u>beforeprint</u>	The event occurs when a page is about to be printed	<u>Event</u>
<u>beforeunload</u>	The event occurs before the document is about to be unloaded	<u>UiEvent</u> , <u>Event</u>
<u>blur</u>	The event occurs when an element loses focus	<u>FocusEvent</u>

<u>canplay</u>	The event occurs when the browser can start playing the media (when it has buffered enough to begin)	<u>Event</u>
<u>canplaythrough</u>	The event occurs when the browser can play through the media without stopping for buffering	<u>Event</u>
<u>change</u>	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)	<u>Event</u>
<u>click</u>	The event occurs when the user clicks on an element	<u>MouseEvent</u>
<u>contextmenu</u>	The event occurs when the user right-clicks on an element to open a context menu	<u>MouseEvent</u>
<u>copy</u>	The event occurs when the user copies the content of an element	<u>ClipboardEvent</u>
<u>cut</u>	The event occurs when the user cuts the content of an element	<u>ClipboardEvent</u>
<u>dblclick</u>	The event occurs when the user double-clicks on an element	<u>MouseEvent</u>
<u>drag</u>	The event occurs when an element is being dragged	<u>DragEvent</u>
<u>dragend</u>	The event occurs when the user has finished dragging an element	<u>DragEvent</u>
<u>dragenter</u>	The event occurs when the dragged element enters the drop target	<u>DragEvent</u>
<u>dragleave</u>	The event occurs when the dragged element leaves the drop target	<u>DragEvent</u>
<u>dragover</u>	The event occurs when the dragged element is over the drop target	<u>DragEvent</u>

dragstart	The event occurs when the user starts to drag an element	DragEvent
drop	The event occurs when the dragged element is dropped on the drop target	DragEvent
durationchange	The event occurs when the duration of the media is changed	Event
ended	The event occurs when the media has reach the end (useful for messages like "thanks for listening")	Event
error	The event occurs when an error occurs while loading an external file	ProgressEvent , UiEvent , Event
focus	The event occurs when an element gets focus	FocusEvent
focusin	The event occurs when an element is about to get focus	FocusEvent
focusout	The event occurs when an element is about to lose focus	FocusEvent
fullscreenchange	The event occurs when an element is displayed in fullscreen mode	Event
fullscreenerror	The event occurs when an element can not be displayed in fullscreen mode	Event
hashchange	The event occurs when there has been changes to the anchor part of a URL	HashChangeEvent
input	The event occurs when an element gets user input	InputEvent , Event

<u>invalid</u>	The event occurs when an element is invalid	<u>Event</u>
<u>keydown</u>	The event occurs when the user is pressing a key	<u>KeyboardEvent</u>
<u>keypress</u>	The event occurs when the user presses a key	<u>KeyboardEvent</u>
<u>keyup</u>	The event occurs when the user releases a key	<u>KeyboardEvent</u>
<u>load</u>	The event occurs when an object has loaded	<u>UiEvent</u> , <u>Event</u>
<u>loadeddata</u>	The event occurs when media data is loaded	<u>Event</u>
<u>loadedmetadata</u>	The event occurs when meta data (like dimensions and duration) are loaded	<u>Event</u>
<u>loadstart</u>	The event occurs when the browser starts looking for the specified media	<u>ProgressEvent</u>
<u>message</u>	The event occurs when a message is received through the event source	<u>Event</u>
<u>mousedown</u>	The event occurs when the user presses a mouse button over an element	<u>MouseEvent</u>
<u>mouseenter</u>	The event occurs when the pointer is moved onto an element	<u>MouseEvent</u>
<u>mouseleave</u>	The event occurs when the pointer is moved out of an element	<u>MouseEvent</u>
<u>mousemove</u>	The event occurs when the pointer is moving while it is over an element	<u>MouseEvent</u>

<u>mouseover</u>	The event occurs when the pointer is moved onto an element, or onto one of its children	<u>MouseEvent</u>
<u>mouseout</u>	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children	<u>MouseEvent</u>
<u>mouseup</u>	The event occurs when a user releases a mouse button over an element	<u>MouseEvent</u>
<u>mousewheel</u>	Deprecated. Use the <u>wheel</u> event instead	<u>WheelEvent</u>
<u>offline</u>	The event occurs when the browser starts to work offline	<u>Event</u>
<u>online</u>	The event occurs when the browser starts to work online	<u>Event</u>
<u>open</u>	The event occurs when a connection with the event source is opened	<u>Event</u>
<u>pagehide</u>	The event occurs when the user navigates away from a webpage	<u>PageTransitionEvent</u>
<u>pageshow</u>	The event occurs when the user navigates to a webpage	<u>PageTransitionEvent</u>
<u>paste</u>	The event occurs when the user pastes some content in an element	<u>ClipboardEvent</u>
<u>pause</u>	The event occurs when the media is paused either by the user or programmatically	<u>Event</u>

<u>play</u>	The event occurs when the media has been started or is no longer paused	<u>Event</u>
<u>playing</u>	The event occurs when the media is playing after having been paused or stopped for buffering	<u>Event</u>
popstate	The event occurs when the window's history changes	<u>PopStateEvent</u>
<u>progress</u>	The event occurs when the browser is in the process of getting the media data (downloading the media)	<u>Event</u>
<u>ratechange</u>	The event occurs when the playing speed of the media is changed	<u>Event</u>
<u>resize</u>	The event occurs when the document view is resized	<u>UiEvent</u> , <u>Event</u>
<u>reset</u>	The event occurs when a form is reset	<u>Event</u>
<u>scroll</u>	The event occurs when an element's scrollbar is being scrolled	<u>UiEvent</u> , <u>Event</u>
<u>search</u>	The event occurs when the user writes something in a search field (for <input="search">)	<u>Event</u>
<u>seeked</u>	The event occurs when the user is finished moving/skipping to a new position in the media	<u>Event</u>
<u>seeking</u>	The event occurs when the user starts moving/skipping to a new position in the media	<u>Event</u>
<u>select</u>	The event occurs after the user selects some text (for <input> and <textarea>)	<u>UiEvent</u> , <u>Event</u>
<u>show</u>	The event occurs when a <menu> element is shown as a context menu	<u>Event</u>
<u>stalled</u>	The event occurs when the browser is trying to get media data, but data is not available	<u>Event</u>

storage	The event occurs when a Web Storage area is updated	StorageEvent
submit	The event occurs when a form is submitted	Event
suspend	The event occurs when the browser is intentionally not getting media data	Event
timeupdate	The event occurs when the playing position has changed (like when the user fast forwards to a different point in the media)	Event
toggle	The event occurs when the user opens or closes the <details> element	Event
touchcancel	The event occurs when the touch is interrupted	TouchEvent
touchend	The event occurs when a finger is removed from a touch screen	TouchEvent
touchmove	The event occurs when a finger is dragged across the screen	TouchEvent
touchstart	The event occurs when a finger is placed on a touch screen	TouchEvent
transitionend	The event occurs when a CSS transition has completed	TransitionEvent
unload	The event occurs once a page has unloaded (for <body>)	UiEvent , Event
volumechange	The event occurs when the volume of the media has changed (includes setting the volume to "mute")	Event
waiting	The event occurs when the media has paused but is expected to resume (like when the media pauses to buffer more data)	Event
wheel	The event occurs when the mouse wheel rolls up or down over an element	WheelEvent

HTML DOM Event Properties and Methods

Property/Method	Description	Belongs To
altKey	Returns whether the "ALT" key was pressed when the mouse event was triggered	MouseEvent
altKey	Returns whether the "ALT" key was pressed when the key event was triggered	KeyboardEvent , TouchEvent
animationName	Returns the name of the animation	AnimationEvent
bubbles	Returns whether or not a specific event is a bubbling event	Event
button	Returns which mouse button was pressed when the mouse event was triggered	MouseEvent
buttons	Returns which mouse buttons were pressed when the mouse event was triggered	MouseEvent
cancelable	Returns whether or not an event can have its default action prevented	Event
charCode	Returns the Unicode character code of the key that triggered the onkeypress event	KeyboardEvent

<code>changeTouches</code>	Returns a list of all the touch objects whose state changed between the previous touch and this touch	TouchEvent
clientX	Returns the horizontal coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered	MouseEvent , TouchEvent
clientY	Returns the vertical coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered	MouseEvent , TouchEvent
<code>clipboardData</code>	Returns an object containing the data affected by the clipboard operation	ClipboardData
code	Returns the code of the key that triggered the event	KeyboardEvent
<code>composed</code>	Returns whether the event is composed or not	Event
ctrlKey	Returns whether the "CTRL" key was pressed when the mouse event was triggered	MouseEvent
ctrlKey	Returns whether the "CTRL" key was pressed when the key event was triggered	KeyboardEvent , TouchEvent
currentTarget	Returns the element whose event listeners triggered the event	Event
data	Returns the inserted characters	InputEvent
<code>dataTransfer</code>	Returns an object containing the data being dragged/dropped, or inserted/deleted	DragEvent , InputEvent
defaultPrevented	Returns whether or not the <code>preventDefault()</code> method was called for the event	Event

deltaX	Returns the horizontal scroll amount of a mouse wheel (x-axis)	WheelEvent
deltaY	Returns the vertical scroll amount of a mouse wheel (y-axis)	WheelEvent
deltaZ	Returns the scroll amount of a mouse wheel for the z-axis	WheelEvent
deltaMode	Returns a number that represents the unit of measurements for delta values (pixels, lines or pages)	WheelEvent
detail	Returns a number that indicates how many times the mouse was clicked	UiEvent
elapsedTime	Returns the number of seconds an animation has been running	AnimationEvent
elapsedTime	Returns the number of seconds a transition has been running	
eventPhase	Returns which phase of the event flow is currently being evaluated	Event
getTargetRanges()	Returns an array containing target ranges that will be affected by the insertion/deletion	InputEvent
getModifierState()	Returns an array containing target ranges that will be affected by the insertion/deletion	MouseEvent
inputType	Returns the type of the change (i.e "inserting" or "deleting")	InputEvent
isComposing	Returns whether the state of the event is composing or not	InputEvent , KeyboardEvent
isTrusted	Returns whether or not an event is trusted	Event

key	Returns the key value of the key represented by the event	KeyboardEvent
key	Returns the key of the changed storage item	StorageEvent
keyCode	Returns the Unicode character code of the key that triggered the onkeypress event, or the Unicode key code of the key that triggered the onkeydown or onkeyup event	KeyboardEvent
location	Returns the location of a key on the keyboard or device	KeyboardEvent
lengthComputable	Returns whether the length of the progress can be computable or not	ProgressEvent
loaded	Returns how much work has been loaded	ProgressEvent
metaKey	Returns whether the "META" key was pressed when an event was triggered	MouseEvent
metaKey	Returns whether the "meta" key was pressed when the key event was triggered	KeyboardEvent , TouchEvent
MovementX	Returns the horizontal coordinate of the mouse pointer relative to the position of the last mousemove event	MouseEvent
MovementY	Returns the vertical coordinate of the mouse pointer relative to the position of the last mousemove event	MouseEvent
newValue	Returns the new value of the changed storage item	StorageEvent

newURL	Returns the URL of the document, after the hash has been changed	HasChangeEvent
offsetX	Returns the horizontal coordinate of the mouse pointer relative to the position of the edge of the target element	MouseEvent
offsetY	Returns the vertical coordinate of the mouse pointer relative to the position of the edge of the target element	MouseEvent
oldValue	Returns the old value of the changed storage item	StorageEvent
oldURL	Returns the URL of the document, before the hash was changed	HasChangeEvent
onemptied	The event occurs when something bad happens and the media file is suddenly unavailable (like unexpectedly disconnects)	
pageX	Returns the horizontal coordinate of the mouse pointer, relative to the document, when the mouse event was triggered	MouseEvent
pageY	Returns the vertical coordinate of the mouse pointer, relative to the document, when the mouse event was triggered	MouseEvent
persisted	Returns whether the webpage was cached by the browser	PageTransitionEvent
preventDefault()	Cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur	Event

propertyName	Returns the name of the CSS property associated with the animation or transition	AnimationEvent , TransitionEvent
pseudoElement	Returns the name of the pseudo-element of the animation or transition	AnimationEvent , TransitionEvent
region		MouseEvent
relatedTarget	Returns the element related to the element that triggered the mouse event	MouseEvent
relatedTarget	Returns the element related to the element that triggered the event	FocusEvent
repeat	Returns whether a key is being hold down repeatedly, or not	KeyboardEvent
screenX	Returns the horizontal coordinate of the mouse pointer, relative to the screen, when an event was triggered	MouseEvent
screenY	Returns the vertical coordinate of the mouse pointer, relative to the screen, when an event was triggered	MouseEvent
shiftKey	Returns whether the "SHIFT" key was pressed when an event was triggered	MouseEvent
shiftKey	Returns whether the "SHIFT" key was pressed when the key event was triggered	KeyboardEvent , TouchEvent
state	Returns an object containing a copy of the history entries	PopStateEvent

stopImmediatePropagation()	Prevents other listeners of the same event from being called	Event
stopPropagation()	Prevents further propagation of an event during event flow	Event
storageArea	Returns an object representing the affected storage object	StorageEvent
target	Returns the element that triggered the event	Event
targetTouches	Returns a list of all the touch objects that are in contact with the surface and where the touchstart event occurred on the same target element as the current target element	TouchEvent
timeStamp	Returns the time (in milliseconds relative to the epoch) at which the event was created	Event
total	Returns the total amount of work that will be loaded	ProgressEvent
touches	Returns a list of all the touch objects that are currently in contact with the surface	TouchEvent
transitionend	The event occurs when a CSS transition has completed	TransitionEvent
type	Returns the name of the event	Event
url	Returns the URL of the changed item's document	StorageEvent
which	Returns which mouse button was pressed when the mouse event was triggered	MouseEvent
which	Returns the Unicode character code of the key that triggered the onkeypress event, or the Unicode key code of the key that triggered the onkeydown or onkeyup event	KeyboardEvent
view	Returns a reference to the Window object where the event occurred	UiEvent

► Program: Display Time - onLoad()

```
<html>
  <head>
    <script type="text/javascript">
      function time()
      {
        var d = new Date();
        var ty = d.getHours() + ":"+d.getMinutes()+":"+d.getSeconds();
        document.frmty.timetxt.value=ty;
        setInterval("time()",1000)
      }
    </script>
  </head>
  <body onload="time()">
    <center><h2>Displaying Time</h2>
      <form name="frmty">
        <input type="text" name="timetxt" size="8">
      </form>
    </center>
  </body>
</html>
```


Cont...

► Output:



► Program: Validation Form - onFocus() & onSubmit()

```
<html>
  <body>
    <script>
      function validateform()
      {
        var uname=document.myform.name.value;
        var upassword=document.myform.password.value;
        if (uname==null || uname=="")
        {
          alert("Name cannot be left blank");
          return false;
        }
        else if(upassword.length<6)
        {
          alert("Password must be at least 6 characters long.");
          return false;
        }
      }
    }
  }
}
```

```
function emailvalidation()
{
    var a=document.myform.email.value
    if (a.indexOf("@")==-1)
    {
        alert("Please enter valid email address")
        document.myform.email.focus()
    }
}
</script>
<body>
    <form name="myform" method="post" action="validate.html" onsubmit="return
validateform()">
        Email: <input type="text" size="20" name="email" onblur="emailvalidation()"><br>
        User Name: <input type="text" name="name"><br>
        Password: <input type="password" name="password"><br>
        <input type="submit" value="Submit" >
    </form>
</body>
</html>
```

► Validate.html

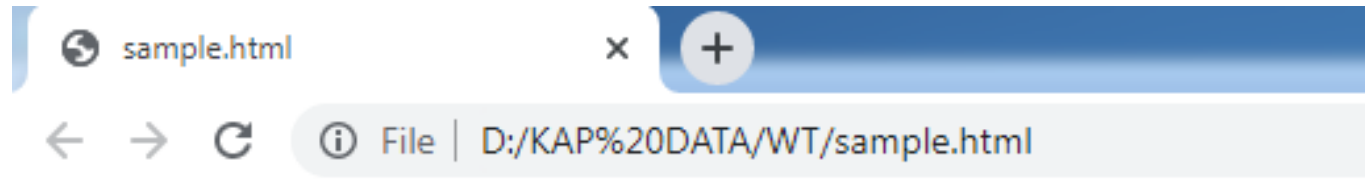
```
<html>
  <body>
    <script type="text/javascript">
      alert("You are a Valid User !!!");
    </script>
  </body>
</html>
```

Program: onmousedown() and onmouseup()

```
<html>
  <body>
    <h2 id="myid1" onmousedown="msg1()" onmouseup="msg2()"> Click on the Text!!!
  </h2>
  <script>
    function msg1()
    {
      document.getElementById("myid1").style.color="blue";
    }
    function msg2()
    {
      document.getElementById("myid1").style.color="black";
    }
  </script>
</body>
</html>
```

Cont...

► Output:



Click on the Text!!!

Program: Write JavaScript that handles following mouse event

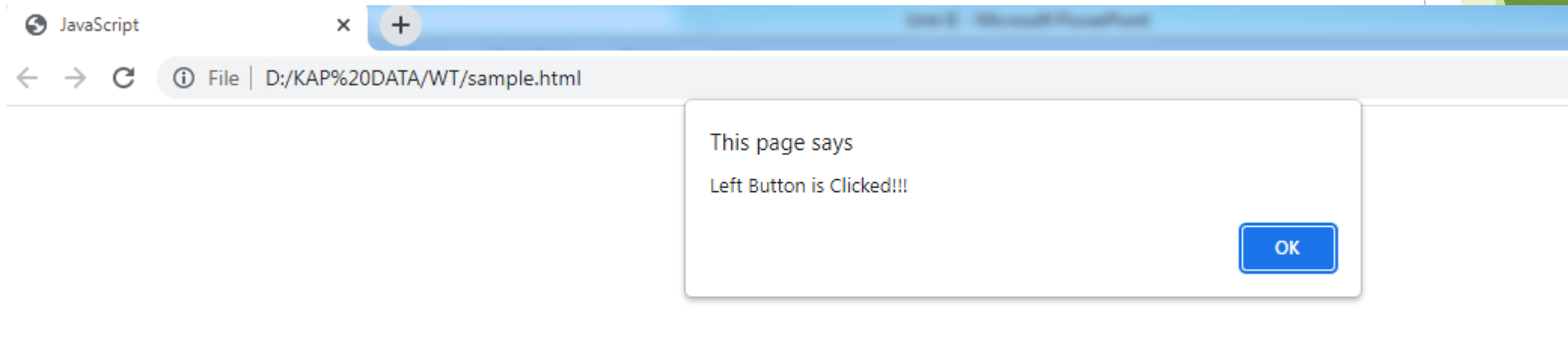
- a. If mouse left button pressed on browser it displayed message “Left Clicked”
- b. If mouse right button pressed on browser it displayed message “Right Clicked”

```
<html>
  <head>
    <title> JavaScript </title>
  </head>
  <body>
    <script language="JavaScript">
      var TestEventType='mousedown';
      function handleevent(e)
      {
        var evt=(e==null ? event:e);
        var clickType='LEFT';
        if(evt.type!=TestEventType) return true;
        if(evt.which)
        {
          if(evt.which==3) clickType='RIGHT';
        }
      }
    </script>
  </body>
</html>
```

```
if(clickType=='LEFT')
    alert("Left Button is Clicked!!!");
else if(clickType=='RIGHT')
    alert("Right Button is Clicked!!!");
return true;
}
document.onmousedown=handleevent;
document.onmouseup=handleevent;
document.onclick=handleevent;
</script>
</body>
</html>
```


Cont...

► Output:

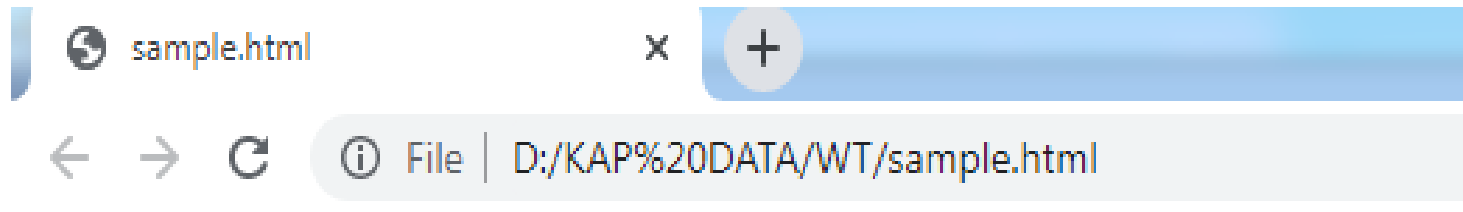


Program: Design HTML form which include two fields username and password. Write JavaScript code to show and hide password

```
<html>
<body>
  <script>
    (function()
    {
      var pass=function(element,field){
        this.element=element;
        this.field=field;
        this.toggle();
      };
      pass.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 pass(checkbox,pwd);
});
})();
</script>
<form action="" method="post" id="login">
    <font size=5><div>
        User Name: <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>
</div>
</form>
</body>
</html>
```



User Name:

Password:

☒ Show password

Login

Modifying Element Style

- We can access or change the contents of document by using various methods

Method	Description
<code>getElementById()</code>	Returns the element with the specified ID
<code>getElementsByTagName()</code>	Returns a list of nodes containing all elements with the specified tag name
<code>getElementsByClassName()</code>	Returns a list of all the elements with the specified node contains the name of the class
<code>appendChild()</code>	To add a new child node to the specified node
<code>removeChild()</code>	Removes the child node
<code>replaceChild()</code>	Replace child nodes
<code>insertBefore()</code>	Insert new child node in front of the specified child node

Cont...

Method	Description
<code>createAttribute()</code>	Create attribute node
<code>createElement()</code>	Create an element node
<code>createTextNode()</code>	Create a text node
<code>getAttribute()</code>	Returns the specified attribute values
<code>setAttribute()</code>	To set or modify the specified value of the specified property
<code>innerHTML</code>	It is useful for getting the text value of a node
<code>parentNode</code>	This DOM property is useful for obtaining the parent node of the specific node
<code>childNodes</code>	This DOM property is useful for obtaining the child node of the specific node
<code>attributes</code>	This property is used to get the attribute node of the specific node

Introduction to JQuery

- ▶ JQuery was initially released by John Resig at BarCamp NYC in January 2006
- ▶ Now a day is maintained by Timmy Willison and his team
- ▶ JQuery is a lightweight, "write less, do more", JavaScript library.
- ▶ The purpose of JQuery is to make it much easier to use JavaScript on your website.
- ▶ JQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
- ▶ JQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.
- ▶ The JQuery library contains the following features:
 - ▶ DOM manipulation
 - ▶ Cross Browser Support
 - ▶ Event Handling
 - ▶ Effects and animations
 - ▶ AJAX support
 - ▶ Lightweight
 - ▶ Latest technology

Features of JQuery

- ▶ **DOM Manipulation:**
 - ▶ It 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
- ▶ **Event Handling:**
 - ▶ Handling a wide variety of events like link click, mouse over etc. are effectively handled in the jquery using event handlers
- ▶ **AJAX Support:**
 - ▶ AJAX technology can be used with jquery to develop responsive websites having a rich set of advanced features
- ▶ **Animations:**
 - ▶ Number of built-in animation effects are provided by jquery

Cont...

- ▶ **Lightweight:**
 - ▶ JQuery is a lightweight library which takes minimum time to load
- ▶ **Cross Browser Support:**
 - ▶ JQuery is supported by various latest browsers like chrome,mozilla,IE etc.
- ▶ **Latest technology:**
 - ▶ Advanced technologies like CSS3 selectors and basic XPath syntax are supported by jquery

Local Installation

- ▶ Download the jQuery library from [jQuery.com](https://jquery.com)
- ▶ There are two versions of jQuery available for downloading:
- ▶ Production version - this is for your live website because it has been minified and compressed
- ▶ Development version - this is for testing and development (uncompressed and readable code)
- ▶ Both versions can be downloaded from [jQuery.com](https://jquery.com).
- ▶ The jQuery library is a single JavaScript file, and you reference it with the HTML `<script>` tag (notice that the `<script>` tag should be inside the `<head>` section):

```
<head>  
  <script src="jquery-3.6.0.min.js"></script>  
</head>
```

CDN Based Version

- ▶ If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).
- ▶ Google is an example of someone who host jQuery:
- ▶ `<head>`
`<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>`
`</head>`
- ▶ Big Advantage:
Many users already have downloaded jQuery from Google when visiting another site. As a result, it will be loaded from cache when they visit your site, which leads to faster loading time. Also, most CDN's will make sure that once a user requests a file from it, it will be served from the server closest to them, which also leads to faster loading time.

Selecting Elements

- ▶ jQuery selectors are one of the most important parts of the jQuery library.
- ▶ jQuery selectors allow you to select and manipulate HTML element(s).
- ▶ jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing [CSS Selectors](#), and in addition, it has some own custom selectors.
- ▶ All selectors in jQuery start with the dollar sign and parentheses: \$().
- ▶ Types of Selector:
 - ▶ Element Selector
 - ▶ #id Selector
 - ▶ .class Selector

Element Selector

- ▶ The jQuery element selector selects elements based on the element name.
- ▶ You can select all <p> elements on a page like this:

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

- ▶ **Program:** When a user clicks on a button, all <p> elements will be hidden:

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script>
      $(document).ready(function(){
        $("button").click(function(){
          $("p").hide();
        });
      });
    </script>
  </head>
  <body>

    <h2>This is a heading</h2>

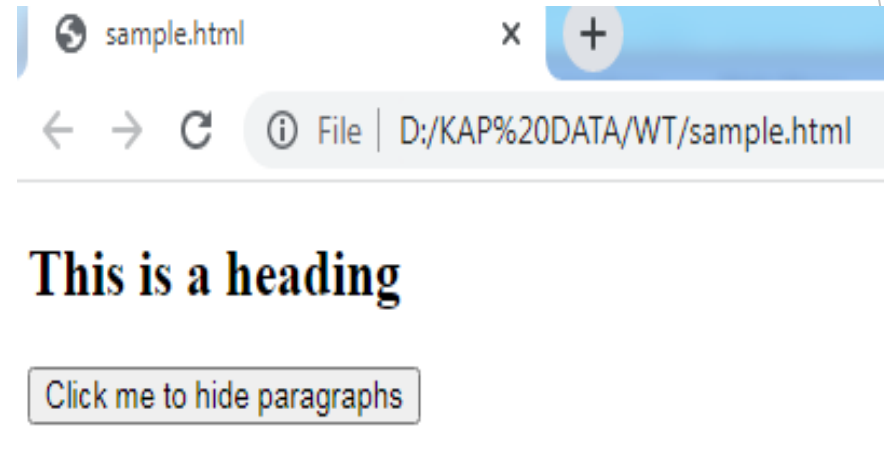
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>

    <button>Click me to hide paragraphs</button>

  </body>
</html>
```

Cont...

► Output:



#id Selector

- ▶ The jQuery *#id* selector uses the id attribute of an HTML tag to find the specific element.
- ▶ An id should be unique within a page, so you should use the *#id* selector when you want to find a single, unique element.
- ▶ To find an element with a specific id, write a hash character, followed by the id of the HTML element:

`$("#test")`

- ▶ **Program:** When a user clicks on a button, the element with id="test" will be hidden:


```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script>
      $(document).ready(function(){
        $("button").click(function(){
          $("#test").hide();
        });
      });
    </script>
  </head>
  <body>

    <h2>This is a heading</h2>

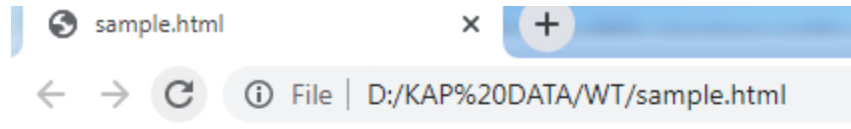
    <p>This is a paragraph.</p>
    <p id="test">This is another paragraph.</p>

    <button>Click me</button>

  </body>
</html>
```

Cont...

► Output:

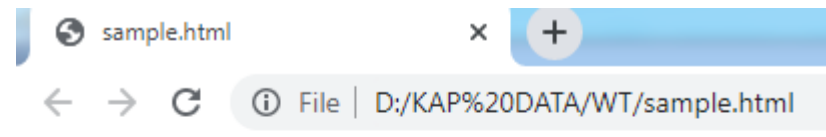


This is a heading

This is a paragraph.

This is another paragraph.

Click me



This is a heading

This is a paragraph.

Click me

.class Selector

- ▶ The jQuery *.class* selector finds elements with a specific class.
- ▶ To find elements with a specific class, write a period character, followed by the name of the class:

`$(".test")`

- ▶ **Program:** When a user clicks on a button, the elements with class="test" will be hidden:

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script>
      $(document).ready(function(){
        $("button").click(function(){
          $(".test").hide();
        });
      });
    </script>
  </head>
  <body>

    <h2 class="test">This is a heading</h2>

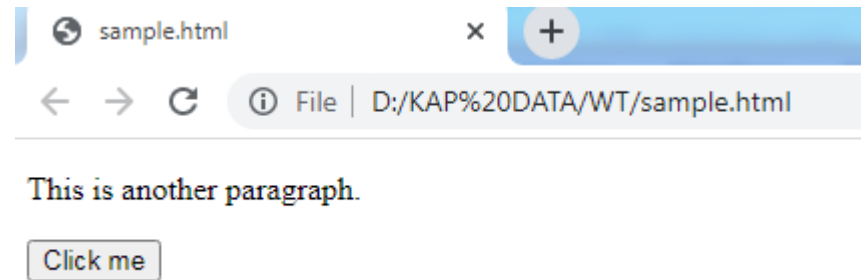
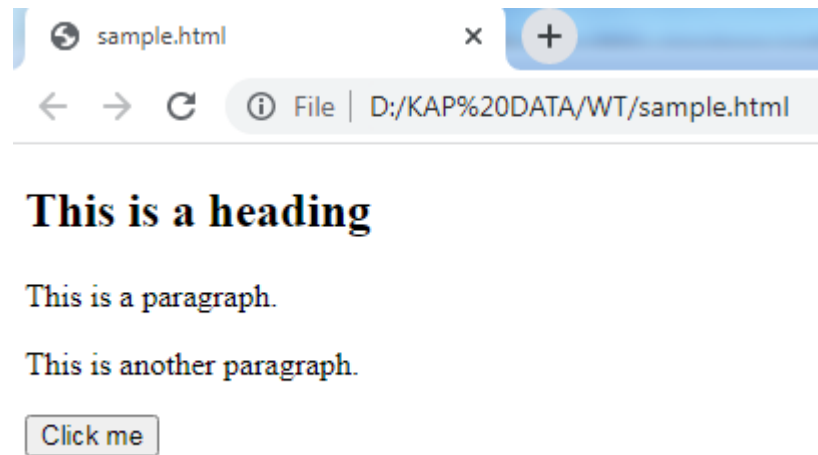
    <p class="test">This is a paragraph.</p>
    <p>This is another paragraph.</p>

    <button>Click me</button>

  </body>
</html>
```

Cont...

► Output:



Changing Styles

▶ jQuery `css()` Method

- ▶ The `css()` method sets or returns one or more style properties for the selected elements.

▶ Return a CSS Property

- ▶ To return the value of a specified CSS property, use the following syntax:

`css("propertyname");`

- ▶ Program: Return the background-color value of the FIRST matched element

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script>
      $(document).ready(function(){
        $("button").click(function(){
          alert("Background color = " + $("p").css("background-color"));
        });
      });
    </script>
  </head>
  <body>

    <h2>This is a heading</h2>

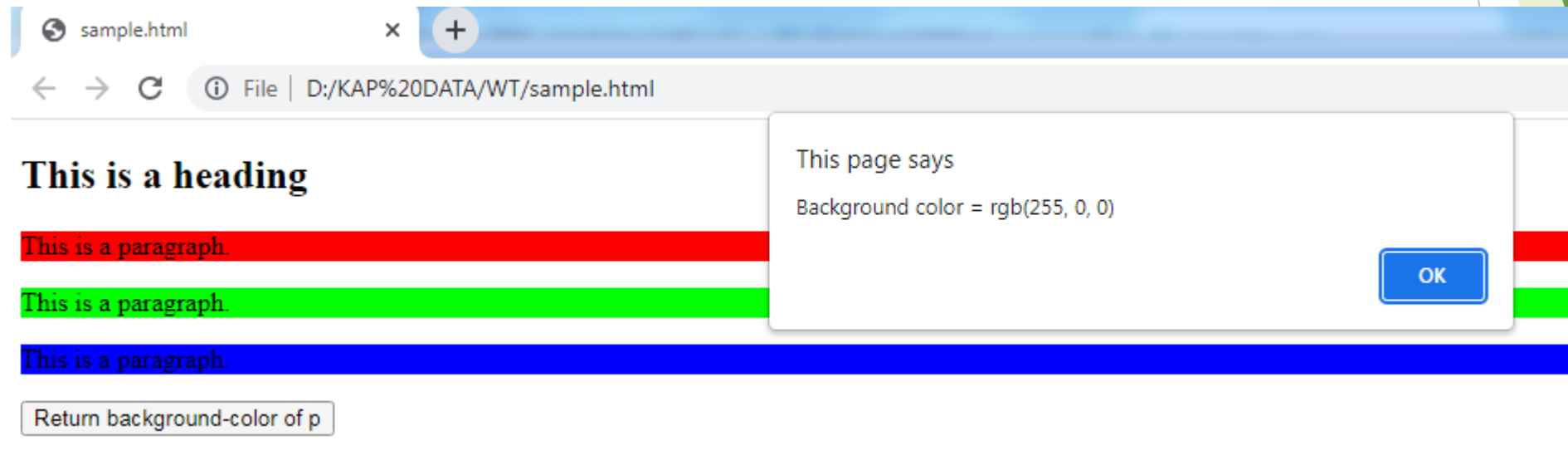
    <p style="background-color:#ff0000">This is a paragraph.</p>
    <p style="background-color:#00ff00">This is a paragraph.</p>
    <p style="background-color:#0000ff">This is a paragraph.</p>

    <button>Return background-color of p</button>

  </body>
</html>
```

Cont...

► Output:



Cont...

- ▶ **Set Multiple CSS Properties**

- ▶ To set multiple CSS properties, use the following syntax:
- ▶ `css({"propertyname":"value","propertyname":"value",...});`

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script>
      $(document).ready(function(){
        $("button").click(function(){
          $("p").css({"background-color": "yellow", "font-size": "200%"});
        });
      });
    </script>
  </head>
  <body>

    <h2>This is a heading</h2>

    <p style="background-color:#ff0000">This is a paragraph.</p>
    <p style="background-color:#00ff00">This is a paragraph.</p>
    <p style="background-color:#0000ff">This is a paragraph.</p>

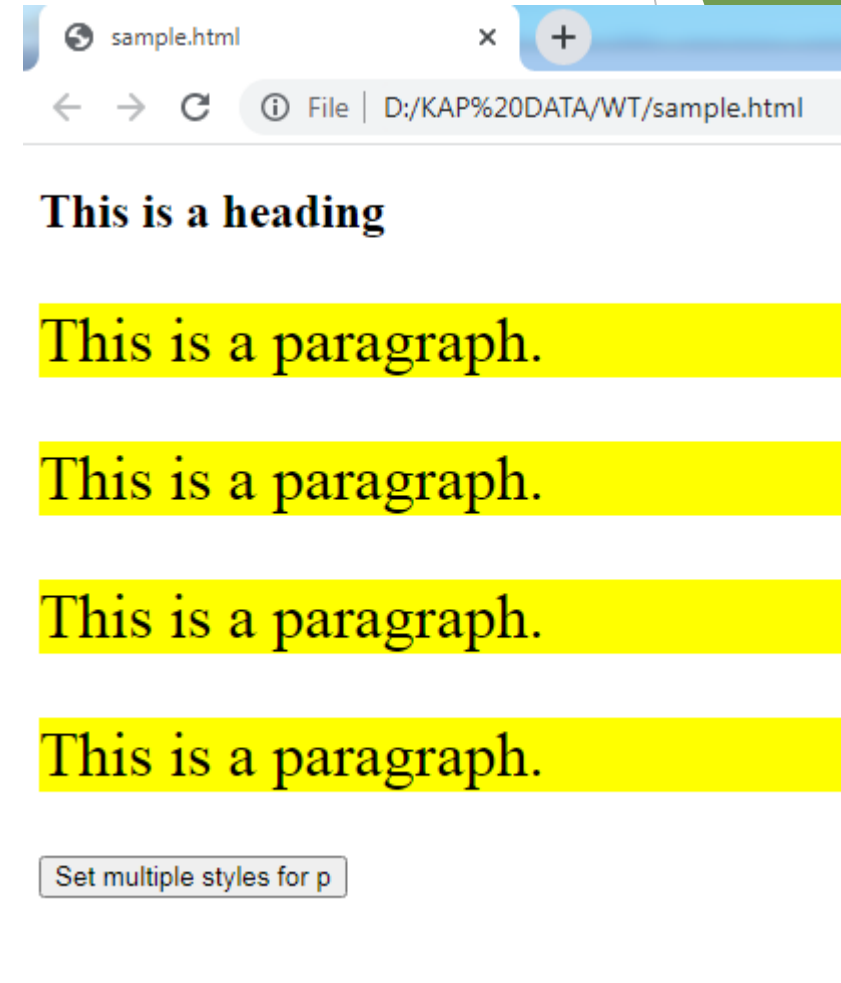
    <p>This is a paragraph.</p>

    <button>Set multiple styles for p</button>

  </body>
</html>
```

Cont...

► Output:



Creating and Appending Elements

- ▶ it is easy to add new elements/content
- ▶ `append()` - Inserts content at the end of the selected elements
- ▶ `prepend()` - Inserts content at the beginning of the selected elements
- ▶ `after()` - Inserts content after the selected elements
- ▶ `before()` - Inserts content before the selected elements
- ▶ **jQuery `append()` Method:**
 - ▶ The jQuery `append()` method inserts content AT THE END of the selected HTML elements.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script>
      $(document).ready(function(){
        $("#btn1").click(function(){
          $("p").append(" <b>Appended text</b>.");
        });

        $("#btn2").click(function(){
          $("ol").append("<li>Appended item</li>");
        });
      });
    </script>
  </head>
  <body>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>

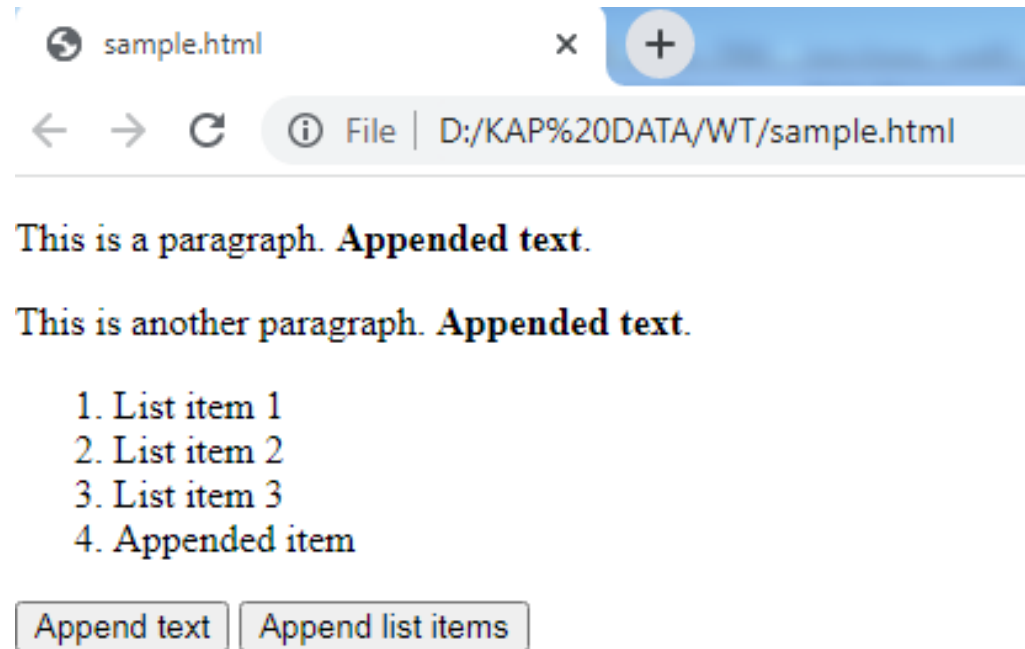
    <ol>
      <li>List item 1</li>
      <li>List item 2</li>
      <li>List item 3</li>
    </ol>

    <button id="btn1">Append text</button>
    <button id="btn2">Append list items</button>

  </body>
</html>
```

Cont...

► Output:

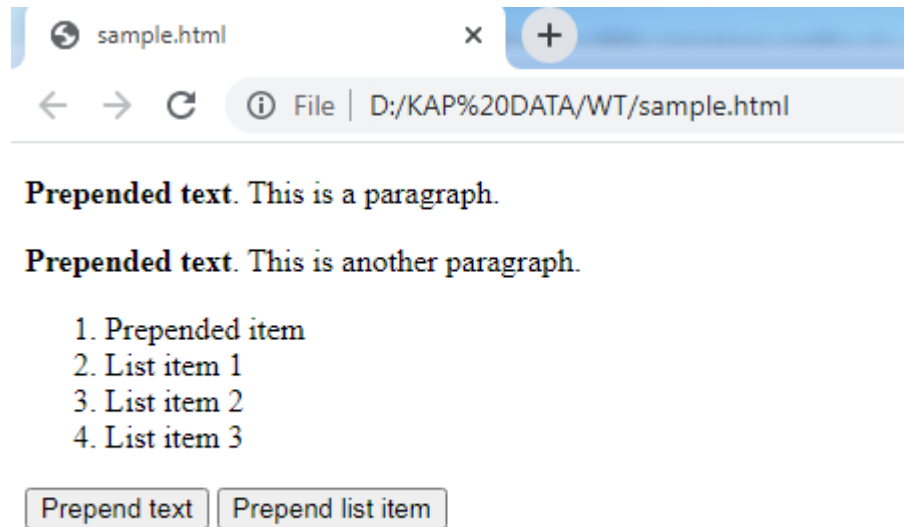


The jQuery prepend() method inserts content AT THE BEGINNING of the selected HTML elements.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script>
      $(document).ready(function(){
        $("#btn1").click(function(){
          $("p").prepend("<b>Prepended text</b> . ");
        });
        $("#btn2").click(function(){
          $("ol").prepend("<li>Prepended item</li>");
        });
      });
    </script>
  </head>
  <body>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
    <ol>
      <li>List item 1</li>
      <li>List item 2</li>
      <li>List item 3</li>
    </ol>
    <button id="btn1">Prepend text</button>
    <button id="btn2">Prepend list item</button>
  </body>
</html>
```

Cont...

► Output:



The jQuery after() method inserts content AFTER the selected HTML elements.
The jQuery before() method inserts content BEFORE the selected HTML elements.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script>
      $(document).ready(function(){
        $("#btn1").click(function(){
          $("p").before("<b>Before</b>");
        });

        $("#btn2").click(function(){
          $("p").after("<i>After</i>");
        });
      });
    </script>
  </head>
  <body>

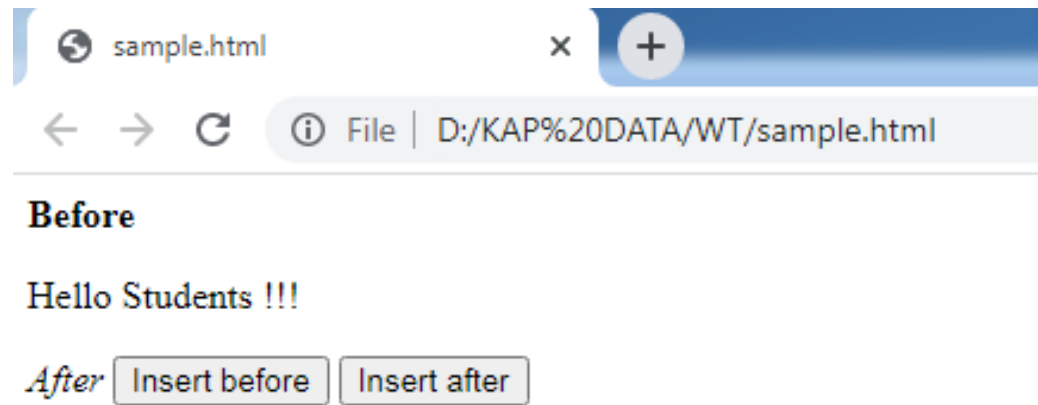
    <p> Hello Students !!! </p>

    <button id="btn1">Insert before</button>
    <button id="btn2">Insert after</button>

  </body>
</html>
```

Cont...

► Output:



Removing Elements

- ▶ To remove elements and content, there are mainly two jQuery methods:
 - ▶ `remove()` - Removes the selected element (and its child elements)
 - ▶ `empty()` - Removes the child elements from the selected element

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script>
      $(document).ready(function(){
        $("button").click(function(){
          $("#div1").remove();
        });
      });
    </script>
  </head>
  <body>

    <div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;">

      This is some text in the div.
      <p>This is a paragraph in the div.</p>
      <p>This is another paragraph in the div.</p>

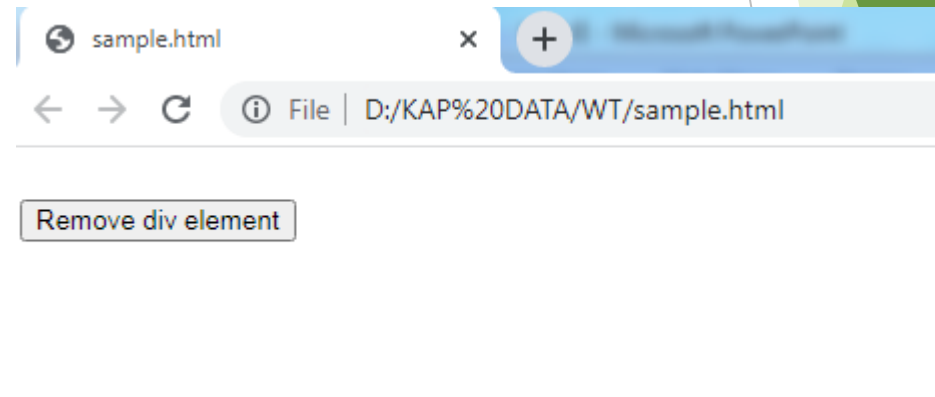
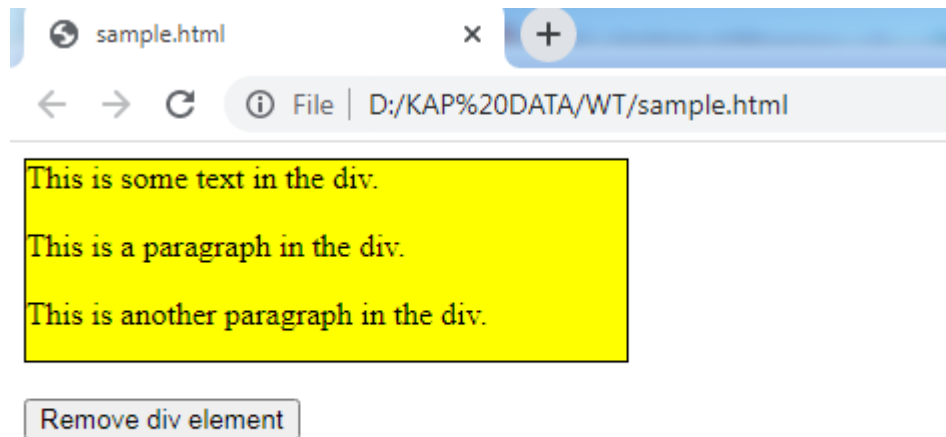
    </div>
    <br>

    <button>Remove div element</button>

  </body>
</html>
```

Cont...

► Output:

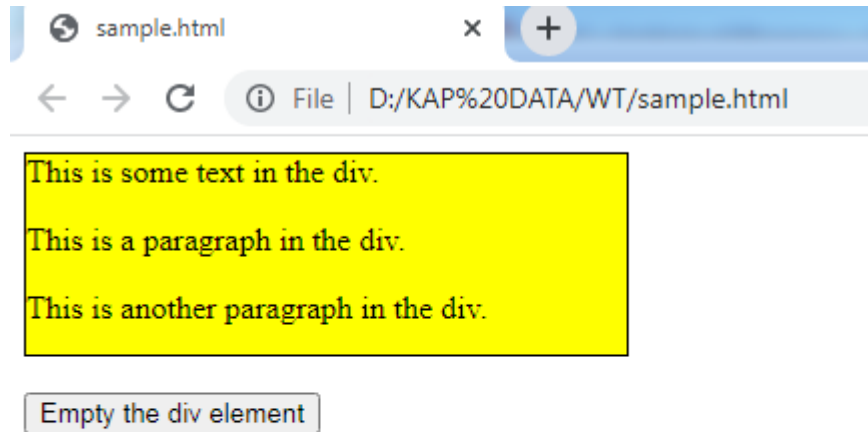


Program: empty() method

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script>
      $(document).ready(function(){
        $("button").click(function(){
          $("#div1").empty();
        });
      });
    </script>
  </head>
  <body>
    <div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;">
      This is some text in the div.
      <p>This is a paragraph in the div.</p>
      <p>This is another paragraph in the div.</p>
    </div>
    <br>
    <button>Empty the div element</button>
  </body>
</html>
```

Cont...

► Output:



Event handling in jquery

- ▶ Event methods trigger or attach a function to an event handler for the selected elements.

Method / Property	Description
<u>bind()</u>	Attaches event handlers to elements
<u>blur()</u>	Attaches/Triggers the blur event
<u>change()</u>	Attaches/Triggers the change event
<u>click()</u>	Attaches/Triggers the click event
<u>dblclick()</u>	Attaches/Triggers the double click event
<u>delegate()</u>	Attaches a handler to current, or future, specified child elements of the matching elements
<u>die()</u>	Removes all event handlers added with the live() method

error()

Attaches/Triggers the error event

event.currentTarget

The current DOM element within the event bubbling phase

event.data

Contains the optional data passed to an event method when the current executing handler is bound

event.delegateTarget

Returns the element where the currently-called jQuery event handler was attached

event.isDefaultPrevented()

Returns whether event.preventDefault() was called for the event object

event.isImmediatePropagationStopped()

Returns whether event.stopImmediatePropagation() was called for the event object

event.isPropagationStopped()

Returns whether event.stopPropagation() was called for the event object

event.namespace

Returns the namespace specified when the event was triggered

event.pageX

Returns the mouse position relative to the left edge of the document

event.pageX

Returns the mouse position relative to the top edge of the document

event.preventDefault()

Prevents the default action of the event

event.relatedTarget

Returns which element being entered or exited on mouse movement

event.result

Contains the last/previous value returned by an event handler triggered by the specified event

event.stopImmediatePropagation()

Prevents other event handlers from being called

event.stopPropagation()

Prevents the event from bubbling up the DOM tree, preventing any parent handlers from being notified of the event

event.target

Returns which DOM element triggered the event

event.timeStamp

Returns the number of milliseconds since January 1, 1970, when the event is triggered

event.type

Returns which event type was triggered

event.which

Returns which keyboard key or mouse button was pressed for the event

focus()

Attaches/Triggers the focus event

focusin()

Attaches an event handler to the focusin event

focusout()

Attaches an event handler to the focusout event

hover()

Attaches two event handlers to the hover event

keydown()

Attaches/Triggers the keydown event

keypress()

Attaches/Triggers the keypress event

keyup()

Attaches/Triggers the keyup event

live()

Adds one or more event handlers to current, or future, selected elements

load()

Attaches an event handler to the load event

mousedown()

Attaches/Triggers the mousedown event

mouseenter()

Attaches/Triggers the mouseenter event

mouseleave()

Attaches/Triggers the mouseleave event

mousemove()

Attaches/Triggers the mousemove event

mouseout()

Attaches/Triggers the mouseout event

mouseover()

Attaches/Triggers the mouseover event

mouseup()

Attaches/Triggers the mouseup event

off()

Removes event handlers attached with the on() method

on()

Attaches event handlers to elements

one()

Adds one or more event handlers to selected elements. This handler can only be triggered once per element

\$.proxy()

Takes an existing function and returns a new one with a particular context

ready()

Specifies a function to execute when the DOM is fully loaded

resize()

Attaches/Triggers the resize event

scroll()

Attaches/Triggers the scroll event

select()

Attaches/Triggers the select event

submit()

Attaches/Triggers the submit event

toggle()

Attaches two or more functions to toggle between for the click event

trigger()

Triggers all events bound to the selected elements

triggerHandler()

Triggers all functions bound to a specified event for the selected elements

unbind()

Removes an added event handler from selected elements

undelegate()

Removes an event handler to selected elements, now or in the future

unload()

Attaches an event handler to the unload event

Bind Event Handler

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").bind("click", function(){
        alert("Button is Clicked");
    });
});
</script>
</head>
<body>

<button >Click on Button</button>

</body>
</html>
```

Cont...

► Output:

