Introduction to

DHTML

Drawbacks of HTML & CSS

- Is good for publishing only static documents.
- The content cannot be changed once it has been delivered to the browser .
- Though CSS-2 and CSS-3 have bought in some interactivity, the scope is limited.

Interactive Technologies

- Client-side techniques
  - JavaScript is
    - a language for extending HTML to embed small programs.
    - the most popular scripting language on the internet, and it works in all major browser
- Dynamic HTML Technologies
  - – Combination of HTML, Cascading Style Sheet and some scripting language.
  - – Provides more control over the appearance, layout and behavior of the web page.
- The Document Object Model (DOM)
  - Defines a standard set of objects for HTML, and a standard way to access and manipulate HTML objects.

What is JavaScript?

- JavaScript is designed by Brendan Eich, in 1995
- Many JavaScript engines are based on ECMA script specification
- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A JavaScript consists of lines of executable computer code
- A JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- JavaScript is a multiple paradigm
- Everyone can use JavaScript without purchasing a license
- Many HTML editors supply a library of common code that can be adapted and used in pages.

What can JavaScript Do?

- **JavaScript gives HTML designers a programming tool -** but JavaScript is a scripting language with a very simple syntax!
- **JavaScript can put dynamic text into an HTML page -** A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page
- **JavaScript can react to events -** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can read and write HTML elements -** A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data -** A JavaScript can be used to validate form data before it is submitted to a server.
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

Advantages of using JavaScript

- It is widely supported in Web Browsers.
- It gives easy access to the document objects and can manipulate most of them.
- Can be used for animation without download time
- Web surfers don't need special plugins to use the scripts.

Issues with JavaScript

- Access to objects differ from browser to browser.
- If script does not work, page is useless
- Web surfers may disable JavaScript support in the browser
- Can run slowly and complex scripts take long time to start up.
- JavaScript and the DOM provide the potential for scripts
- Browser authors contain this risk using two restrictions.
  - Scripts can only perform web-related actions, not general-purpose programming tasks like creating files.
  - Second, scripts are constrained by the same origin policy: scripts from one web site do not have access to information such as usernames, passwords, or cookies sent to another site.

JavaScript Statements

- Single line comments start with //.
- Multi line comments start with /* and end with */.
- Scripts require neither a main function nor an exit condition.
- JavaScript code is case sensitive.
- Each statement is executed by the browser in the sequence they are written.
- Each line of code terminated by semicolon.
- Functions
  - have parameters which are passed inside parenthesis
  - Statements inside a function can also be grouped together in blocks using {}
  - Functions will not be executed before the event occurs.

Where to Put the JavaScript?

- **Scripts in the head section:** Scripts to be executed when they are called, or when an event is triggered, go in the head section.
- <HTML>

  <HEAD>

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

  </HEAD>

  Scripts in the body section: Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates the content of the page.
- <HTML>

  <HEAD> ...

  </HEAD>

  <BODY>

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

  </BODY>

Scripts in both the body and the head section

Where to Put the JavaScript?

- **Using an External JavaScript**
- To run the same JavaScript on several pages, without having to write the same script on every page.
- Write a JavaScript in an external file. Save the external JavaScript file with a .js file extension.
- The external script cannot contain the <script> tag!
- To use the external script, point to the .js file in the "src" attribute of the <script> tag:
- <HTML>

  <HEAD> <script src="xxx.js"> …. </script>

  </HEAD>
- Best Practise :
  - Execute a JavaScript when an **event** occurs, such as when a user clicks a button.
  - When this is the case we can put the script inside a **function**.
- Events are normally used in combination with functions (like calling a function when an event occurs).

JavaScript Functions

- Functions can be defined both in the <head> and in the <body> section of a document.
- Syntax :

```
function functionname(var1,var2,...,varX){
    some code
    }
```

- Function name(parameters) In JavaScript parameters are passed as arrays.
- A function with no parameters must include the parentheses () after the function name.
- The word *function* must be written in **lowercase letters**, otherwise a JavaScript error occurs.
- **The return statement** is used to specify the value that is returned from the function.

HTML DOM Event Object

Attribute	Description	W3C
onblur	The event occurs when an element loses focus	Yes
onchange	The event occurs when the content of an element, the selection, or the checked state have changed	Yes
onclick	The event occurs when the user clicks on an element	Yes
ondblclick	The event occurs when the user double-clicks on an element	Yes
onerror	The event occurs when an error occurs while loading an external file	Yes
onfocus	The event occurs when an element gets focus	Yes
onkeydown	The event occurs when the user is pressing a key or holding down a key	Yes
onkeypress	The event occurs when the user is pressing a key or holding down a key	Yes
onkeyup	The event occurs when a keyboard key is released	Yes
onload	The event occurs when an object has been loaded	Yes
onmousedown	The event occurs when a user presses a mouse button over an element	Yes
onmousemove	The event occurs when a user moves the mouse pointer over an element	Yes
onmouseout	The event occurs when a user moves the mouse pointer out of an element	Yes
onmouseover	The event occurs when a user mouse over an element	Yes
onmouseup	The event occurs when a user releases a mouse button over an element	Yes
onresize	The event occurs when the size of an element has changed	Yes
onselect	The event occurs after some text has been selected in an element	Yes
onunload	The event occurs before the browser closes the document	Yes

Alert Box

- An alert box is often used if you want to make sure information comes through to the user.
- When an alert box pops up, the user will have to click "OK" to proceed.
- Syntax : alert("sometext");

Alert box example

```
<html>
<head>
  <script type="text/javascript">
  function displaymessage()
  { alert("Hello World!");
   }
  </script>
 </head>
<body>
 <form> <input type="button" value="Click me!"
  onclick="displaymessage()" >
</form>
</body>
</html>
```

Confirm box

- is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.
- **Syntax:** confirm("sometext");

Confirm Box example

```
<html>
<head>
<script type="text/javascript">
function disp_confirm()
{
var r=confirm("Press a button");
if (r==true)
  {
  document.write("You pressed OK!");
  }
else
  {
  document.write("You pressed Cancel!");
  }
}
</script>
</head>
<body>
<input type="button"
    onclick="disp_confirm()"
    value="Display a confirm box" />
</body>
</html>
```

Prompt Box

- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.
- **Syntax:** prompt("sometext","defaultvalue");

Prompt Box example

```
<html>
<head>
<script type="text/javascript">
function disp_prompt()
{
var name=prompt("Please enter your
   name","");
if (name!=null&&name!="" )
  {
  document.write("Hello " + name +
   "! How are you today?");
  }
}
</script>
</head>
<body>
 <input type="button"
  onclick="disp_prompt()"
  value="Display a prompt
  box" />
</body>
</html>
```

Data Types

- Primitive
  - String
  - Number
  - Boolean
- Composite
  - Array
  - Object
- Special
  - Null
  - Undefined

Date object

- new Date() // current date and time
- new Date(milliseconds) //milliseconds since 1970/01/01
- new Date(dateString)
  - var d = new Date("July 21, 1983 01:15:00");
- new Date(year, month, day, hours, minutes, seconds, milliseconds)
  - var d = new Date(1986,07,09,08,17,06,88);

Example

```
<html>
<head><script>
function myFunction() {
   var d = new Date(1986,07,09,08,17,06,88);
   document.getElementById("demo").innerHTML = d.toString();
}</script></head>
<body>
<p>Click the button to display the date.</p>
<input type="button" onclick="myFunction()">Click</button>
<p id="demo"></p>
</body>
</html>
```

- Sat Aug 09 1986 08:17:06 GMT+0530 (India Standard Time)

Methods

getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year (four digits)
getHours()	Returns the hour (from 0-23)
getMilliseconds()	Returns the milliseconds (from 0-999)
getMinutes()	Returns the minutes (from 0-59)
getMonth()	Returns the month (from 0-11)
getSeconds()	Returns the seconds (from 0-59)
getTime()	Returns the number of milliseconds since midnight Jan 1 1970, and a specified date
getTimezoneOffset()	Returns the time difference between UTC time and local time, in minutes

Methods

getUTCDate()	Returns the day of the month, according to universal time (from 1-31)
getUTCDay()	Returns the day of the week, according to universal time (from 0-6)
getUTCFullYear()	Returns the year, according to universal time (four digits)
getUTCHours()	Returns the hour, according to universal time (from 0-23)
getUTCMilliseconds()	Returns the milliseconds, according to universal time (from 0-999)
getUTCMinutes()	Returns the minutes, according to universal time (from 0-59)
getUTCMonth()	Returns the month, according to universal time (from 0-11)
getUTCSeconds()	Returns the seconds, according to universal time (from 0-59)

Methods

now()	Returns the number of milliseconds since midnight Jan 1, 1970
parse()	Parses a date string and returns the number of milliseconds since January 1, 1970
setDate()	Sets the day of the month of a date object
setFullYear()	Sets the year (four digits) of a date object
setHours()	Sets the hour of a date object
setMilliseconds()	Sets the milliseconds of a date object
setMinutes()	Set the minutes of a date object
setMonth()	Sets the month of a date object
setSeconds()	Sets the seconds of a date object
setTime()	Sets a date to a specified number of milliseconds after/before January 1, 1970

Methods

setUTCDate()	Sets the day of the month of a date object, according to universal time
setUTCFullYear()	Sets the year of a date object, according to universal time (four digits)
setUTCHours()	Sets the hour of a date object, according to universal time
setUTCMilliseconds()	Sets the milliseconds of a date object, according to universal time
setUTCMinutes()	Set the minutes of a date object, according to universal time
setUTCMonth()	Sets the month of a date object, according to universal time
setUTCSeconds()	Set the seconds of a date object, according to universal time
toDateString()	Converts the date portion of a Date object into a readable string

Methods

toISOString()	Returns the date as a string, using the ISO standard
toJSON()	Returns the date as a string, formatted as a JSON date
toLocaleDateString()	Returns the date portion of a Date object as a string, using locale conventions
toLocaleTimeString()	Returns the time portion of a Date object as a string, using locale conventions
toLocaleString()	Converts a Date object to a string, using locale conventions
toString()	Converts a Date object to a string
toTimeString()	Converts the time portion of a Date object to a string
toUTCString()	Converts a Date object to a string, according to universal time
UTC()	Returns the number of milliseconds in a date since midnight of January 1, 1970, according to UTC time

Example

```
const event = new Date(Date.UTC(2019, 11, 20, 3, 0, 0));
const options = { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric'};
console.log(event.toLocaleDateString('hi', options));
console.log(event.toLocaleDateString(undefined, options));
```

"शुक्रवार, 20 दिसंबर 2019“

"Friday, December 20, 2019"

String object

- Syntax: var txt = new String("*string*");

String methods

charAt()	Returns the character at the specified index (position)
charCodeAt()	Returns the Unicode of the character at the specified index
concat()	Joins two or more strings, and returns a new joined strings
endsWith()	Checks whether a string ends with specified string/characters
includes()	Checks whether a string contains the specified string/characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string

String methods

match()	Searches a string for a match against a regular expression, and returns the matches
repeat()	Returns a new string with a specified number of copies of an existing string
replace()	Searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced
search()	Searches a string for a specified value, or regular expression, and returns the position of the match
slice()	Extracts a part of a string and returns a new string
split()	Splits a string into an array of substrings
startsWith()	Checks whether a string begins with specified characters

String methods

substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices
toLowerCase()	Converts a string to lowercase letters
toString()	Returns the value of a String object
toUpperCase()	Converts a string to uppercase letters
trim()	Removes whitespace from both ends of a string