Unit: 5
DHTML

Outline

- Introduction
- Features of DHTML
- DHTML Objects
- CSS in Javascript in DHTML
- DHTML Buttons and Events

Introduction

- DHTML stands for “dynamic hypertext transfer markup language”
- DHTML is not a language.
- DHTML is a term describing the art of making dynamic & interactive Webpages.
- DHTML is a combination of web development technologies used to create dynamically changing websites.
- DHTML composes of various web technologies that can be used together to generate more interactive web pages. DHTML = CSS + JavaScript + DOM + HTML

Feature of DHTML

1. **Dynamic style**: Using style sheets, we can specify the color, style or size of the page content. We can change the style of the page without having to return to web server → reduce the time-consuming.
2. **Dynamic content**: We can change the contents of web page after they are displayed in response to user input or events like a mouse click.
3. **Real-time positioning**: We can specify the exact position (either absolute or relative), in terms of x and y co-ordinates. Once page is rendered, we can move the elements on the page, making it dynamic.

Feature of DHTML cont..

4. **Font download**: are a feature that allows us to insert our choice fonts in the web page. This is an important feature, because if the specified fonts are not available on the user's machine, the browser defaults to available our fonts.
5. **Dynamic binding**: We can bind the table in the web page to a database. When the web page is loaded, the data from the database on the server is displayed in the table.
6. **Scripting**: We can write scripts to change the style and content of the web page. This script is inserted in the web page. The browser interprets the script whenever the page displayed.

DHTML Objects

- Every element contained within a web page is referred as an object. The DHTML objects allow you to access and manipulate the element by using the different properties and methods.

- **a**: specifies the beginning and end of a hypertext link.
- **body:** specifies the start and end of the body of the document.
- **div**: divides a web page into multiple sections, where each section can render other HTML elements.
- **document**: represents the entire HTML document.
- **form**: Specifies a container for other controls.

DHTML Objects cont..

- **frame**: specifies a frame within a frameset.
- **frameset**: specifies a frameset that can hold multiple frames.
- **html**: specifies the HTML elements.
- **img**: specifies an image or video clip to be embedded in the document.
- **input**: creates different form input controls.
- **li:** specifies a list item.
- **link**: specifies a link between the existing document and external document.
- **span**: Specifies an inline element to apply styles to a part of the text.

DHTML Objects cont..

- **table**: specifies that the content should be displayed in a table.
- **td**: specifies the data to be displayed in a table cell.
- **tr**: specifies a table row.

DHTML JavaScript

- JavaScript can be included in HTML pages, which creates the content of the page as dynamic.

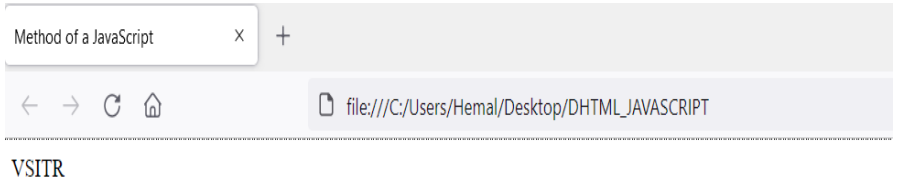
❑ **Document.write() Method**

✓ The document.write() method of JavaScript, writes the output to a web page.

➢ **Example :** The following example simply uses the document.write() method of JavaScript in the DHTML.

DHTML JavaScript cont..

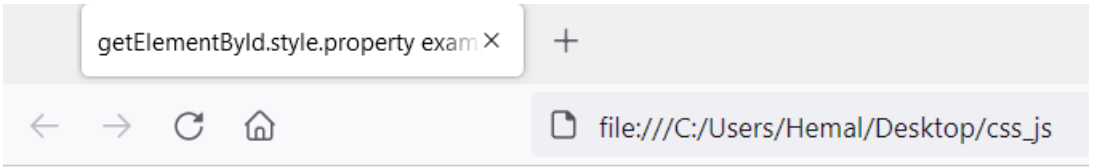
```
<html>
<head>
<title>
Method of a JavaScript
</title>
</head>
<body>
<script type="text/javascript">
document.write(“VSITR");
</script>
</body>
</html>
```

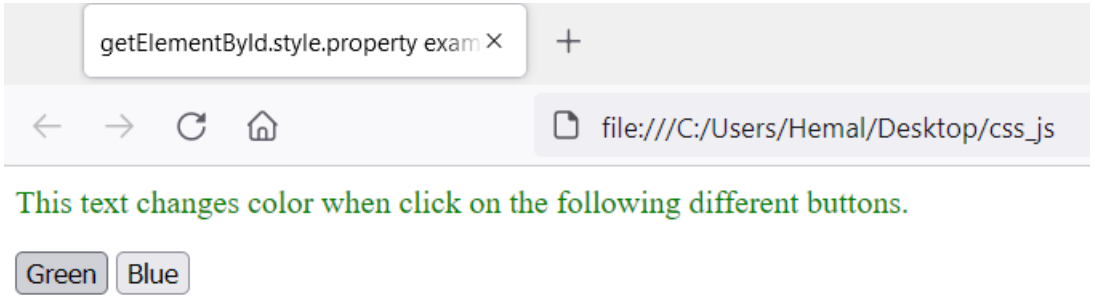


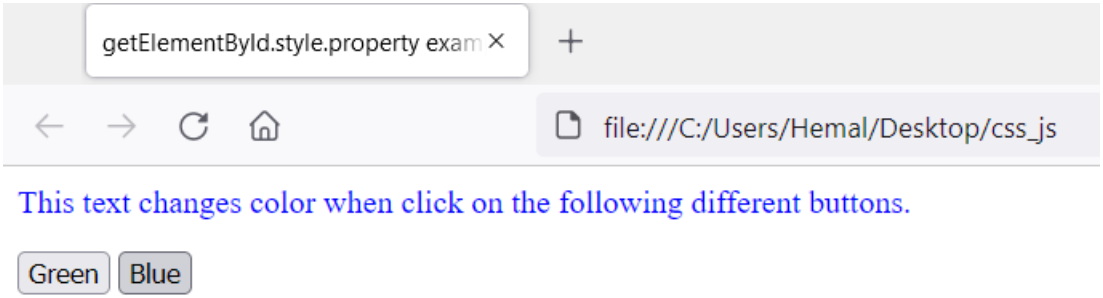
CSS with JavaScript in DHTML

```
<html>
<head>
  <title>
getElementById.style.property example
</title>
</head>
<body>
<p id="demo"> This text changes color when click on the following different buttons. </p>
  <button onclick="change_Color('green');"> Green </button>
  <button onclick="change_Color('blue');"> Blue </button>
<script type="text/javascript">
  function change_Color(newColor) {
  var element = document.getElementById('demo').style.color = newColor;
}
</script>
</body>
</html>
```

CSS with JavaScript in DHTML cont..



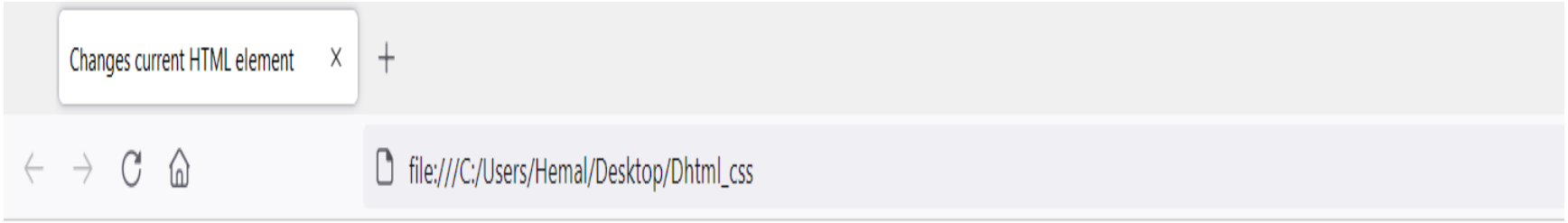




DHTML CSS

```
<html>
<head>
<title>
Changes current HTML element
</title>
</head>
<body>
<center>
<h1 onclick="this.style.color='blue'"> Vidush Somany Instituet of Technology and Research </h1>
<center>
</body>
</html>
```

DHTML CSS cont..



Vidush Somany Institute of Research and Technology

DHTML Events and Buttons

Sr.No	Events	When It Occurs
1.	onabort	It occurs when the user aborts the page or media file loading.
2.	onblur	It occurs when the user leaves an HTML object.
3.	onchange	It occurs when the user changes or updates the value of an object.
4.	onclick	It occurs or triggers when any user clicks on an HTML element.
5.	ondblclick	It occurs when the user clicks on an HTML element two times together.
6.	onfocus	It occurs when the user focuses on an HTML element. This event handler works opposite to onblur.

DHTML Events and Buttons cont..

Sr.No	Events	When It Occurs
7.	onkeydown	It triggers when a user is pressing a key on a keyboard device. This event handler works for all the keys.
8.	onkeypress	It triggers when the users press a key on a keyboard. This event handler is not triggered for all the keys.
9.	onkeyup	It occurs when a user released a key from a keyboard after pressing on an object or element.
10.	onload	It occurs when an object is completely loaded.
11.	onmousedown	It occurs when a user presses the button of a mouse over an HTML element.
12.	onmousemove	It occurs when a user moves the cursor on an HTML object.

DHTML Events and Buttons cont..

Sr.No	Events	When It Occurs
13.	onmouseover	It occurs when a user moves the cursor over an HTML object.
14.	onmouseout	It occurs or triggers when the mouse pointer is moved out of an HTML element.
15.	onmouseup	It occurs or triggers when the mouse button is released over an HTML element.
16.	onreset	It is used by the user to reset the form.
17.	onselect	It occurs after selecting the content or text on a web page.
18.	onsubmit	It is triggered when the user clicks a button after the submission of a form.
19.	onunload	It is triggered when the user closes a web page.

Unit: 5
XML

Outline

1. Introduction
   - Introduction to XML
   - Features of XML
   - XML Key Component
2. Document Type Definition (DTD)
3. XML Schemas
4. XSL
5. XSLT

Introduction to XML

- XML stands for e**X**tensible **M**arkup **L**anguage
- XML is a language to describe other languages.
- Its main purpose is to allow the sharing of data across different type of systems and it is particularly useful in this sense for applications that do this over the internet.
- Example :

```
<?xml version="1.0">
<person>
        <first>Narendra</first>
        <last>Modi</last>
        <birthdate>01/01/45</birthdate>
        <employed started="01/02/03">
                Prof. @ VSITR college
        </employed>
</person>
```

This line is called the XML prolog

Here person is root element

Here started is an attribute of element employed

Features of XML

- It is in a format that both **human** and **machines** can read.
- It supports **Unicode**.
- It supports **data structures**.
- It is **self-documenting**.
- It has a **strict format** that makes it easy for **parsing** to take place.
- It can be understood and exchanged between **dissimilar systems**.
- It can be useful for swapping data between **different applications**.

Entity References

- If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.
- <message>salary < 1000</message>
- To avoid this error, replace the "<" character with an **entity reference**: <message>salary < 1000</message>

<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

XML Key Component

- One of the key aspects of XML is how strict the syntax is.
- There are mainly 3 components of the XML
  1. Elements
  2. Attribute
  3. Namespace

Elements

- The strict syntax of XML contains a few rules about elements that must be adhered to:
  - Elements must have a **closing tag**.
  - Tags are **case sensitive**
  - Elements must be **nested correctly**
  - XML documents must have a **root element**.
- Example :

<**birthdate**>26th October 1788</**birthdate**> ✓

<Birthdate>26th October 1788</birthdate> ✘ (elements are case sensitive)

Hello ✘ (elements not nested properly)

Hello ✓

Attributes

- Attributes can be added to elements in XML but must always be quoted.
- For Example, here employed is a element and started is the attribute of the element employed.

<employed started="10
</employed>

Value must be quoted

<employed started=10/11/12> VSITR, Kadi
</employed> ✘

Namespace

- Sometimes in XML there is a danger of conflicting names between documents.
- Example :
  - You create one document with name element for the fruit, it may also possible someone else create a document with name element for the vegetables name, so to avoid the conflict we can use namespaces.
- Namespace usually take the form of a URL, beginning with a domain name, an optional namespace label in the form of a directory name and finally a version number, which is also optional.

  xmlns = “http://www.mydomain.com/ns/vegetables/1.1”

Namespace (Example)

- This XML carries HTML table information:

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

- This XML carries information about a table (a piece of furniture):

```
<table>
  <name>Saag table</name>
  <width>3</width>
  <length>6</length>
  <weight>5kg</weight>
</table>
```

Namespace (Example) cont..

- To solve the conflict problem we can use namespace of html table.
- Example :

```
<h:table
xmlns:h="http://www.w3.org
/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

- To solve the conflict problem we can use namespace of furniture table.
- Example :

```
<f:table
xmlns:f="http://vsitr.ac.in/furnti
ture">
  <f:name>Saag table</f:name>
  <f:width>3</f:width>
  <f:length>6</f:length>
  <f:weight>5kg</f:weight>
</f:table>
```

XML Key Component cont..

- **White space** is **preserved** in XML where as in HTML it is truncated down to just a single space.

XML: Hello Tove
HTML: Hello Tove

- One thing does remain in common with HTML though is comments,
  - Comments can be added using the triangle brackets like this:

<!-- here are some remarks -->

Document Type Definition (DTD)

- XML is particularly concerned with being well formed or correct in syntax.
- There are two ways of checking whether the document follows expected order and structure
  - **Document Type Definitions (DTDs)**
  - **Schemas**
- A Document Type Definition (**DTD**) defines the legal building blocks of an XML document.
- A DTD can be declared **inline** inside an XML document, or as an **external** reference

Why Use a DTD?

- With a **DTD**, each of your XML files can carry a **description** of its **own format.**
- With a DTD, independent groups of people can agree to use a **standard DTD for interchanging data.**
- Your application can use a standard DTD to **verify** that the data you **receive** from the outside world is **valid.**

DTD (Example)

```
<?xml version="1.0"?>
<!DOCTYPE note [
   <!ELEMENT note (to,from,title,message)>
   <!ELEMENT to (#PCDATA)>
   <!ELEMENT from (#PCDATA)>
   <!ELEMENT title (#PCDATA)>
   <!ELEMENT message (#PCDATA)>
]>
```

DTD cont..

- DTD can be internal or external
- If it is **internal** than simply put previous code to the **top of the XML** file
- If it is external than save it as **.dtd** file extension and refer it from XML,

<!DOCTYPE note SYSTEM “note.dtd”>

DTD Elements

- We can specify the number of occurrences of the elements using +, *, ? and | operators (works ~ similar to Regular Expression)
- Example :
  - <!ELEMENT note(to+,from,title?,message*) />
- Above example suggest that **root** element of the xml must be **note** and should have one or more (+) **recipients**, **sender** should be only one, **title** must be one or zero(?) and **messages** can be zero or more(*).
- We can also specify to have either one of the elements using | operator
  - <!ELEMENT note(to,from,title,message|information) >
- In above declaration we have specified that either message should be there or the information element should be there in the note

DTD Attribute

We can specify the attributes also using DTD using **ATTLIST** declaration.

- Syntax:

  <!ATTLIST element-name attribute-name attribute-type default value >

- Example :

  <!ATTLIST employed started CDATA “01/01/01”>

This is the default value for started

We can also specify required or fixed for the attribute

- Example :

  <!ATTLIST employed started CDATA #REQUIRED>

  <!ATTLIST employed started CDATA #FIXED “01/01/01”>

This suggest that started is mandatory field

XML Schema

- XML **Schema** is an XML-based **alternative** to **DTD**
- An XML schema describes the structure of an XML document.
- The XML Schema language is also referred to as **XML Schema Definition** (**XSD**)
- An XML Schema
  - defines **elements** that can appear in a document
  - defines **attributes** that can appear in a document
  - defines which elements are **child elements**
  - defines the **order** of child elements
  - defines **data types** for elements and attributes
  - defines **default** and **fixed** values for elements and attributes

XML Schema cont..

- XML Schemas are the Successors of DTDs
  - XML Schemas are **extensible** to future additions
  - XML Schemas are **richer** and **more powerful** than DTDs
  - XML Schemas are **written** in **XML**
  - XML Schemas support **data types**
  - XML Schemas support **namespaces**

XML Schema (Example)

note.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
    >
     <xs:element name="note“>
        <xs:complexType>
          <xs:sequence>
               <xs:element name="to" type="xs:string"/>
               <xs:element name="from" type="xs:string"/>
               <xs:element name="heading" type="xs:string"/>
                <xs:element name="body" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
     </xs:element>
</xs:schema>
```

XML Schema (Example) cont..

```
<?xml version="1.0"?>
<note
xmlns:xsi=http://www.w3.org/2001/XMLSchema-
    instance
xsi:schemaLocation="note.xsd">
    <to>VSITR</to>
    <from>Student</from>
    <heading>Reminder</heading>
    <body>Don't forget to attend lecture this
    weekend!</body>
</note>
```

Data Types in XSD

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time
- Etc…..

Complex Types in XSD

- Complex elements can be built that contain other elements and attributes.
- For example,

```
<xs:complexType name="productinfo">
   <xs:sequence>
     <xs:element name="item" type="xs:string" />
     <xs:element name="itemcode" type="xs:string" />
   </xs:sequence>
</xs:complexType>
<xs:element name="food" type="productinfo"/>
<xs:element name="magazine" type="productinfo"/>
<xs:element name="clothes" type="productinfo"/>
```

Default ,Fixed and Required Values

- To define attribute a similar style is adopted, for example for the XML element :

  <firstname lang=“English”>Narendra</firstname>
- XSD would be

  <xs:attribute firstname="lang" type="xs:string"/>
- Default value attribute in XSD

  <xs:attribute firstname="lang" type="xs:string" **default="EN"**/>
- Fixed value attributes in XSD

  <xs:attribute firstname ="lang" type="xs:string" **fixed="EN"**/>
- Required value attributes in XSD

  <xs:attribute firstname ="lang" type="xs:string" **use="required"**/>

What is XSL

- XSL stands for e**X**tensible **S**tylesheet **L**anguage.
  - CSS = Style Sheets for HTML
  - XSL = Style Sheets for XML
- XSL describes how the XML document should be displayed!
- XSL - More Than a Style Sheet Language
- XSL consists of three parts:
  - XSLT - a language for transforming XML documents
  - XPath - a language for navigating in XML documents
  - XSL-FO - a language for formatting XML documents

What is XSLT?

- XSLT stands for XSL Transformations
- XSLT is the most important part of XSL
- XSLT transforms an XML document into another XML document (ex. (X)HTML)
- XSLT uses XPath to navigate in XML documents

XSL Transformation

The style sheet provides the template that transforms the document from one structure to another

- <**xsl:template**> starts the definition of the actual template, as the root of the source XML document
- The **match** = “/” attribute makes sure this begins applying the template to the root of the source XML document
- The style sheet is linked into the XML by adding the connecting statement to the XML document:

  <?xml-stylesheet type=”text/xsl” href=”abc.xsl” ?>

XSLT Example

book.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type = "text/xsl" href = "book.xsl"?>
<library>
        <book>
                <title>Developing Web Application</title>
                <author>Ralph Moseley</author>
                <price>109</price>
        </book>
        <book>
                <title>Software engineering</title>
                <author>Roger Pressmen</author>
                <price>120</price>
        </book>
        <book>
                <title>Java head first</title>
                <author>Bob Dylan</author>
                <price>400</price>
        </book>
</library>
```

XSLT Example cont..

book.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
	<body>
		<h2>Library Book Collection</h2>
		<table border="1">
		<xsl:for-each select="library/book">
			<tr>
				<td><xsl:value-of select="title"/></td>
				<td><xsl:value-of select="author"/></td>
			</tr>
		</xsl:for-each>
		</table>
	</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

XSL Elements

- stylesheet
- template
- value-of
- for-each
- sort
- If
- choose
- when
- otherwise

XSL Elements cont..

- **<xsl:stylesheet>**
  - <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  - **<xsl:stylesheet>**, defines that this document is an XSLT style sheet document (along with the version number and XSLT namespace attributes).
- **<xsl:template match="/">**
  - The <xsl:template> element is used to build templates.
  - Attribute match is used to set the starting point for the XPath
  - Match="/" specifies that XPath is calculated from the root of the document
- **<xsl:value-of>**
  - The <xsl:value-of> element can be used to extract the value of an XML element
  - <xsl:value-of **select**="**XPath**"/>
  - It will provide output to the stream of transformation
  - XPath index will start from **1 (not 0)**

XSL Elements cont..

- **<xsl:for-each>**
  - The XSL <xsl:for-each> element can be used to select every XML element of a specified node-set

    <xsl:for-each select="XPath">

    </xsl:for-each>
  - We can also filter the output from the XML file by

    <xsl:for-each select="kadicolleges/chat[to='ABC']">
  - Legal filter operators are:
    - = (equal)
    - ! = (not equal)
    - < less than
    - > greater than

XSL Elements cont..

- **<xsl:sort>**
  - The <xsl:sort> element is used to sort the output.
  - <xsl:sort> is always within <xsl:for-each>

```
<xsl:for-each select="kadicolleges/chat">
        <xsl:sort select="message"
    order="ascending|descending"/>
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="artist"/></td>
        </tr>
</xsl:for-each>
```

XSL Elements cont..

- **<xsl:if>**
  - To put a conditional if test against the content of the XML file, add an <xsl:if> element to the XSL document.

```
<xsl:if test="expression">
    ...some output if the expression is true...
</xsl:if>
```

  - <xsl:if test="price > 10">

```
<tr>
 <td><xsl:value-of select="title"/></td>
 <td><xsl:value-of select="price"/></td>
</tr>
</xsl:if>
```

XSL Elements cont..

- **<xsl:choose>**
  - The <xsl:choose> element is used in conjunction with <xsl:when> and <xsl:otherwise> to express multiple conditional tests

```
<xsl:choose>
    <xsl:when test="expression">
      ... some output ...
    </xsl:when>
    <xsl:otherwise>
      ... some output ....
    </xsl:otherwise>
</xsl:choose>
```

Unit: 5
AJAX

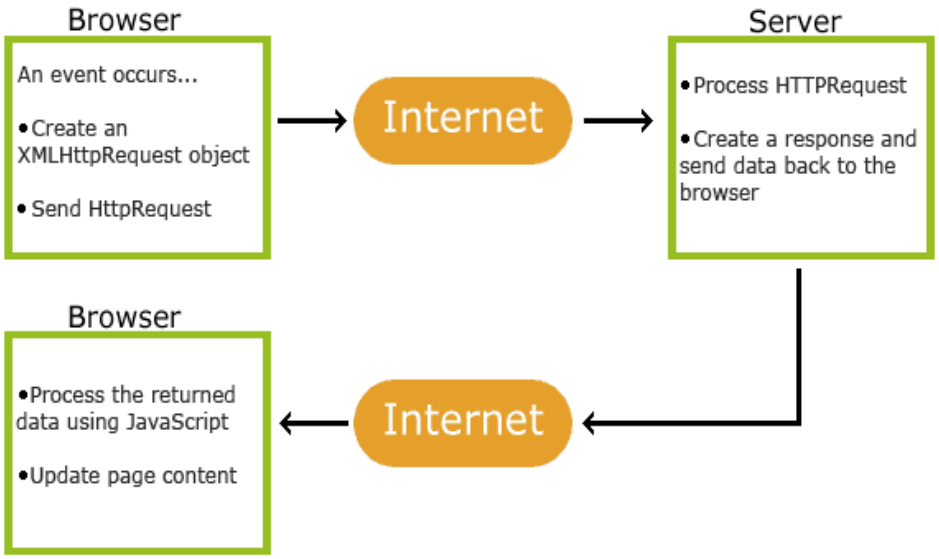
Outline

- Introduction
- How AJAX Works?
- XMLHttpRequest Object Methods
- Advantages of Ajax
- Disadvantages of Ajax
- Asynchronous call in Ajax

Introduction

- AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.
- AJAX is not a programming language.
- AJAX just uses a combination of:
  - A browser built-in XMLHttpRequest object (to request data from a web server)
  - JavaScript and HTML DOM (to display or use the data)
- AJAX is about updating parts of a web page, without reloading the whole page.

How Ajax Works?



How Ajax Works cont..

1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

XMLHttpRequest Object Methods

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

Advantages of Ajax

- **Reduce server traffic and increase speed**

➢ Advantage of Ajax is its ability to improve the performance and usability of web applications.

- **Enable asynchronous calls**

➢ Ajax allows its users to make asynchronous calls to the web server without reloading the whole web page. As a web visitor, you don’t have to wait for the entire page to load entirely in order to access the entire page content.

- **XMLHttpRequest**

➢ XMLHttpRequest transfers and manipulates the XML data to and from a web service using HTTP. Its purpose is to establish an independent connection between the webpage’s client-side and server.

Advantages of Ajex cont..

- **Reduce bandwidth usage**

➢ Ajax makes the best use of the server's bandwidth by fetching particle contents instead of transmitting the entire page's content. This means that you can bring data from the database and store it into the database to perform background without reloading the page.

- **Form Validation**

➢ In contrast to traditional form submission, where client-side validations occur after submission, the AJAX method enables precise and immediate form validation. AJAX provides speed, which is also one of its significant benefits.

Disadvantages of Ajax

- Hard for the users to bookmark specific sate of the web page.
- Open-source
- Any user whose browser does not support JavaScript or XMLHttpRequest, or has this functionality disabled, will not be able to properly use pages that depend on Ajax.
- **Indexing problem*:*** Search Engine is unable to list AJAX application as it is not integrated with browser.

Asynchronous call in Ajax

- Asynchronous means that the script will send a request to the server, and continue it's execution without waiting for the reply. As soon as reply is received a browser event is fired, which in turn allows the script to execute associated actions.
- Synchronous AJAX call is made when async setting of jQuery AJAX function is set to false while Asynchronous AJAX call is made when async setting of jQuery AJAX function is set to true.

Asynchronous call in Ajax cont..

```
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
<script type="text/javascript">
    function ShowCurrentTime() {
        $.ajax({
            async: false,
            type: "POST",
            url: "Default.aspx/GetCurrentTime",
            data: '{name: "Mudassar" }',
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            success: function (response) {
                alert(response.d);}
        }); }
</script>
```

Asynchronous call in Ajax cont..

```
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
<script type="text/javascript">
    function ShowCurrentTime() {
        $.ajax({
            async: true,
            type: "POST",
            url: "Default.aspx/GetCurrentTime",
            data: '{name: "Mudassar" }',
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            success: function (response) {
                alert(response.d);}
        });}
</script>
```