ADVANCED WEB TECHNOLOGY

Unit 1

What is HTML?

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

My First Heading

My first paragraph.

- The <!DOCTYPE html> declaration defines that this document is an HTML document
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the HTML page
- The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The <h1> element defines a large heading
- The element defines a paragraph

What is an HTML Element?

An HTML element is defined by a start tag, some content, and an end tag:

<tagname> Content goes here... </tagname>

The HTML *element* is everything from the start tag to the end tag:

<h1>My First Heading</h1>

My first paragraph.

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
	My first paragraph.	
 	None	*none*

HTML Attributes

HTML attributes provide additional information about HTML elements.

HTML Attributes

- All HTML elements can have **attributes**
- Attributes provide **additional information** about elements
- Attributes are always specified in **the start tag**
- Attributes usually come in name/value pairs like: **name="value"**

The href Attribute

The <a> tag defines a hyperlink. The href attribute specifies the URL of the page the link

Example

Home

The src Attribute

The <img> tag is used to embed an image in an HTML page.
The src attribute specifies the path to the image to be displayed:

Example

<img src="***image.jpg***">

The width and height Attributes

The <img> tag should also contain the width and height attributes, which specify the width and height of the image (in pixels):

Example

<img src="***img.jpg***" width="***500***" height="***600***">

The alt Attribute

The required alt attribute for the <img> tag specifies an alternate text for an image, if the image for some reason cannot be displayed. This can be due to a slow connection, or an error in the src attribute, or if the user uses a screen reader.

Example

<img src="***img.jpg***" alt=" ">

The style Attribute

The style attribute is used to add styles to an element, such as color, font, size, and more.

Example

This is a red paragraph.

The title Attribute

The title attribute defines some extra information about an element.

The value of the title attribute will be displayed as a tooltip when you mouse over the element:

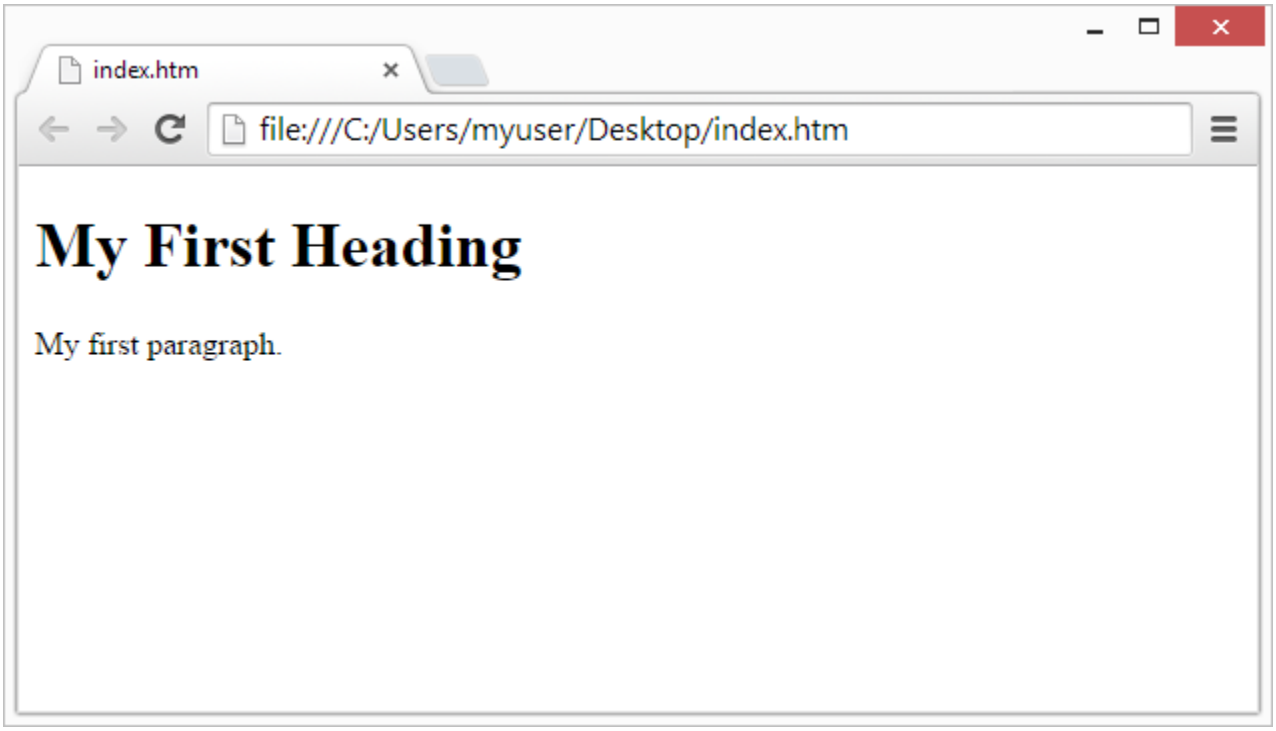
Example

```
<p title="I'm a tooltip">This is a paragraph.</p>
```

Web Browsers

The purpose of a web browser (Chrome, Edge, Firefox, Safari) is to read HTML documents and display them correctly.

A browser does not display the HTML tags, but uses them to determine how to display the document:



HTML History

Year	Version
1989	Tim Berners-Lee invented www
1991	Tim Berners-Lee invented HTML
1993	Dave Raggett drafted HTML+
1995	HTML Working Group defined HTML 2.0
1997	W3C Recommendation: HTML 3.2
1999	W3C Recommendation: HTML 4.01
2000	W3C Recommendation: XHTML 1.0

2008	WHATWG HTML5 First Public Draft
2012	WHATWG HTML5 Living Standard
2014	W3C Recommendation: HTML5
2016	W3C Candidate Recommendation: HTML 5.1
2017	W3C Recommendation: HTML5.1 2nd Edition
2017	W3C Recommendation: HTML5.2

Text formatting tags

- <b> - Bold text
- - Important text
- <i> - Italic text
- - Emphasized text
- <mark> - Marked text
- <small> - Smaller text
- <del> - Deleted text
- <ins> - Inserted text
- <sup> - Superscript text

List tags

HTML Lists are used to specify lists of information. All lists may contain one or more list elements.

1. Ordered List or Numbered List (ol)
2. Unordered List or Bulleted List (ul)
3. Description List or Definition List (dl)

Ordered List or Numbered List

In the ordered HTML lists, all the list items are marked with numbers by default.

```
<!DOCTYPE>
<html>
<body>
<ol>
 <li>M.Sc(CS)</li>
 <li>MCA</li>
 <li>M.Sc(IT)</li>
 <li>M.Sc(AI)</li>
</ol>
</body>
</html>
```

HTML Unordered List or Bulleted List

In HTML Unordered list, all the list items are marked with bullets.

```
<!DOCTYPE>
<html>
<body>
<ul>
<li>M.Sc(CS)</li>
 <li>MCA</li>
 <li>M.Sc(IT)</li>
 <li>M.Sc(AI)</li>
</ul>
</body>
</html>
```

HTML Description List or Definition List

HTML Description list is also a list style which is supported by HTML and XHTML.

1. **<dl> tag** defines the start of the list.
2. **<dt> tag** defines a term.
3. **<dd> tag** defines the term definition (description).

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

```
<body>

<dl>

  <dt>HTML</dt>

  <dd>-Hypertext markup language.</dd>

  <dt>XML</dt>

  <dd>-Extensible markup language</dd>

 <dt>Script</dt>

 <dd>-Scripting languages are usually interpreted at runtime rather than compiled.</dd>

</dl>

</body>

</html>
```

Image tag

HTML img tag is used to display image on the web page. HTML img tag is an empty tag that contains attributes only, closing tags are not used in HTML image element.

Attributes of HTML img tag

The src and alt are important attributes of HTML img tag. All attributes of HTML image tag are given below.

1) src

It is a necessary attribute that describes the source or path of the image. It instructs the browser where to look for the image on the server.

The location of image may be on the same directory or another server.

2) alt

The alt attribute defines an alternate text for the image, if it can't be displayed. The value of the alt attribute describe the image in words. The alt attribute is considered good for SEO prospective.

3) width

It is an optional attribute which is used to specify the width to display the image. It is not recommended now. You should apply CSS in place of width attribute.

4) height

It h3 the height of the image. The HTML height attribute also supports iframe, image and object elements. It is not recommended now. You should apply CSS in place of height attribute.

Example:

<**img** src="animal.jpg" height="180" width="300" alt="animal image">

<**img** src="E:/images/animal.png" height="180" width="300" alt="animal image">

Table tag

HTML table tag is used to display data in tabular form (row * column). There can be many columns in a row.

We can create a table to display data in tabular form, using <table> element, with the help of <tr> , <td>, and <th> elements.

In Each table, table row is defined by <tr> tag, table header is defined by <th>, and table data is defined by <td> tags. HTML tables are used to manage the layout of the page e.g. header section, navigation bar, body content, footer section etc.

HTML Table Tags

Tag	Description
<table>	It defines a table.
<tr>	It defines a row in a table.
<th>	It defines a header cell in a table.
<td>	It defines a cell in a table.
<caption>	It defines the table caption.
<colgroup>	It specifies a group of one or more columns in a table for formatting.
<col>	It is used with <colgroup> element to specify column properties for each column.
<tbody>	It is used to group the body content in a table.
<thead>	It is used to group the header content in a table.
<tfooter>	It is used to group the footer content in a table.

```
<!DOCTYPE>

<html>

<body>

<table>
```

<tr><th>Name</th><th>Subject</th><th>Marks</th></tr>

<tr><td>Karthik</td><td>Python</td><td>60</td></tr>

<tr><td>Mani</td><td>Python</td><td>80</td></tr>

<tr><td>Suresh</td><td>R Programming</td><td>82</td></tr>

<tr><td>Nithya</td><td>Java</td><td>72</td></tr>

</table>

</body>

</html>

Note

<table border="1">

What is HTML Form?

HTML Form is a document that stores information of a user on a web server using interactive controls. An HTML form contains different kinds of information such as username, password, contact number, email id, etc.
The elements used in an HTML form are the check box, input box, radio buttons, submit buttons, etc.

<html>

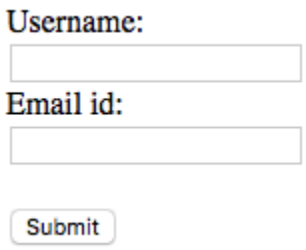
<body>

<form>

Username:

```
        <input type="text" name="username">
        <br>
        Email id:<br>
        <input type="text" name="email_id">
        <br><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Output:



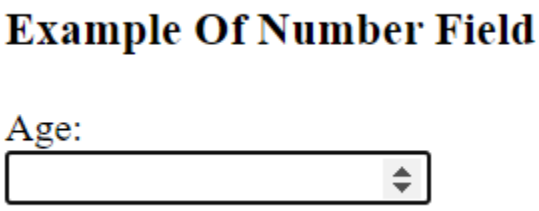
Input Element in HTML Forms: Input Elements are the most common elements which are used in HTML Forms. Various user input fields can be created such as text field, check box, password field, radio button, submit button, etc.

Text Field in HTML Forms: The text field is a one-line input field allowing the user to input text. Text Field input controls are created using the “input” element with a type attribute having the value “text”.

```
<!DOCTYPE html>
```

```
<html>
<body>
    <h3>Example Of Number Field</h3>
    <form>
        <label for="Age">Age:</label><br>
        <input type="number" name="Age" id="Age">
    </form>
</body>
</html>
```

Output:



Password Field in HTML Forms: Password fields are a type of text field in which the text entered is masked using asterisks or dots for the prevention of user identity from another person who is looking at the screen. Password Field input controls are created using the “input” element with a type attribute having the value “password”.

```
<!DOCTYPE html>
```

```
<html>
<body>
    <h3>Example of Password Field</h3>
    <form>
        <label for="user-password">
            Password:
        </label><br>
        <input type="password" name="user-pwd"
            id="user-password">
    </form>
</body>
</html>
```

Radio Buttons in HTML Form: Radio Buttons are used to let the user select exactly one option from a list of predefined options. Radio Button input controls are created using the “input” element with a type attribute having the value as “radio”.

```
<html>
<body>
    <h3>Example of Radio Buttons</h3>
```

```
    <form>
        SELECT GENDER
        <br>
        <input type="radio" name="gender" id="male">
        <label for="male">Male</label><br>
        <input type="radio" name="gender" id="female">
        <label for="female">Female</label>
    </form>
</body>
</html>
```

Checkboxes in HTML Form: Checkboxes are used to let the user select one or more options from a pre-defined set of options. Checkbox input controls are created using the “input” element with a type attribute having the value as “checkbox”.

```
<!DOCTYPE html>
<html>
<body>
    <h3>Example of HTML Checkboxes</h3>
    <form>
```

```
        <b>SELECT SUBJECTS</b>
        <br>
        <input type="checkbox" name="subject" id="maths">
        <label for="maths">Maths</label>
        <input type="checkbox" name="subject" id="science">
        <label for="science">Science</label>
        <input type="checkbox" name="subject" id="english">
        <label for="english">English</label>
    </form>
</body>
</html>
```

Select Boxes in HTML Forms: Select boxes are used to allow users to select one or more than one option from a pull-down list of options. Select boxes are created using two elements which are “select” and “option”. List items are defined within the select element.

```
<!DOCTYPE html>
<html>
<body>
    <h3>Example of a Select Box</h3>
```

```
    <form>
        <label for="country">Country:</label>
        <select name="country" id="country">
            <option value="India">India</option>
            <option value="Sri Lanka">Sri Lanka</option>
            <option value="Australia">Australia</option>
        </select>
    </form>
</body>
</html>
```

What is CSS?

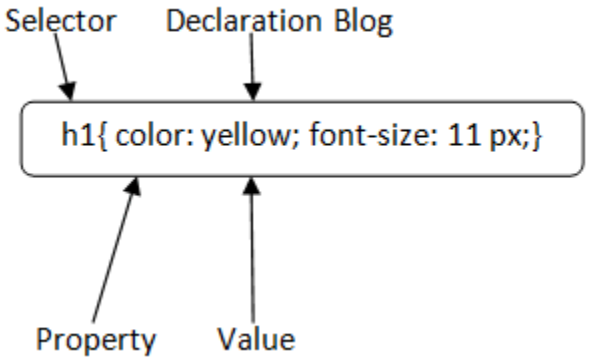
CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces.

CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

- CSS stands for Cascading Style Sheet.
- CSS is used to design HTML tags.
- CSS is a widely used language on the web.

- HTML, CSS and JavaScript are used for web designing. It helps the web designers to apply style on HTML tags.

CSS Syntax



Selector: Selector indicates the HTML element you want to style. It could be any tag like <h1>, <title> etc.

Declaration Block: The declaration block can contain one or more declarations separated by a semicolon. For the above example, there are two declarations:

1. color: yellow;
2. font-size: 11 px;

Each declaration contains a property name and value, separated by a colon.

Property: A Property is a type of attribute of HTML element. It could be color, border etc.

Value: Values are assigned to CSS properties. In the above example, value "yellow" is assigned to color property.

Inline style

The inline CSS is also a method to insert style sheets in HTML document.

Three Ways to Insert CSS

- External CSS
- Internal CSS
- Inline CSS

Inline CSS

The inline CSS is also a method to insert style sheets in HTML document.

Disadvantages of Inline CSS

- These styles cannot be reused anywhere else.
- These styles are tough to be edited because they are not stored at a single place.
- It is not possible to style pseudo-codes and pseudo-classes with inline CSS.
- Inline CSS does not provide browser cache advantages.

Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>
```

```
<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

External CSS

The external style sheet is generally used when you want to make changes on multiple pages. It is ideal for this condition because it facilitates you to change the look of the entire web site by changing just one file.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

mystyle.css

```
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

CSS Attribute Selectors

The `[attribute]` selector is used to select elements with a specified attribute.

The `[attribute="value"]` selector is used to select elements with a specified attribute and value.

Example

```
a[target="_blank"]

 {
  background-color: yellow;
}
```

Ex

```
<!DOCTYPE html>

<html>

<head>

<style>

[class^="top"] {

  background: yellow;

}

</style>

</head>

<body>

<h2>CSS [attribute^="value"] Selector</h2>

<h1 class="top-header">Welcome</h1>

<p class="top-text">Hello world!</p>

<p class="topcontent">Are you learning CSS?</p>

</body>
```

</html>

CSS content property

Values	Description
normal	It is used to set the default value
none	This value does not set the content.
counter	It sets the content as the counter. It is generally a number. It is displayed by using the **counter()** or **counters()** function.
string	It is used to set any string value. It should always be enclosed within quotation marks. It generates any string after or before the HTML element.
attr	It inserts the value of the specified attribute after or before the element. If the selector does not have a particular attribute, then an empty string will be inserted.
open-quote	It is used to insert the opening quotation mark, or it sets the content to be an opening quote.
close-quote	It is used to insert the closing quotation mark, or it sets the content to be a closing quote.
no-close-quote	If the closing quotes are specified, then it is used to remove the closing quote from the content.
no-open-quote	If the opening quotes are specified, then it is used to remove the opening quote from the content.
url	It is used to set the content to some media, which could be an image, video, audio, and many more.
Initial	It is used to set the property to its default value.
Inherit	It inherits the property from its parent element.

Example

```
<!DOCTYPE html>
<html>
<head>
   <title>
      CSS content Property
   </title>
   <style>
   body{
   text-align: center;
   }
   p{
   font-size: 25px;
   }
   p::before {
         content: "Welcome ";
      }

      #para::before {
         content: normal;
      }
      #para1::before {
         content: none;
      }
      </style>
</head>

<body>
   <h1> CSS content property </h1>
   <h2> Use of content: normal; and content: none; </h2>
   <p> To Computer Science Department </p>
   <p id = "para"> This is a paragraph using <b>normal</b> value. </p>
   <p id = "para1"> This is another paragraph using <b>none</b> value. </p>
```

</body>
</html>

Example

```
<head>
<style>
input[type="text"] {
  width: 150px;
  display: block;
  margin-bottom: 10px;
  background-color: yellow;
}
input[type="button"]
{
  width: 120px;
  margin-left: 35px;
  display: block;
}
</style>
</head>
<body>
<h2>Styling Forms</h2>
<form name="input" action="" method="get">
```

```
  Firstname:<input type="text" name="Name" value="Raja" size="20">
  Lastname:<input type="text" name="Name" value="mani" size="20">
  <input type="button" value="Example Button">
</form>
</body>
</html>
```

JavaScript

JavaScript is *an object-based scripting language* which is lightweight and cross-platform.

JavaScript is not a compiled language, but it is a translated language. The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.

Features of JavaScript

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. It is a light-weighted and interpreted language.
4. It is a case-sensitive language.
5. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
6. It provides good control to the users over the web browsers.

Application of JavaScript

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

Example

```
<!doctype html>
<html>
  <head>
    <title>Adding two numbers using JavaScript in HTML</title>
```

```
    <script type="text/javascript">
      var num1 = 10;
      var num2 = 40;
      var sum = num1+num2;
      document.write("Sum of two numbers is " + sum);
    </script>
  </head>
  <body></body>
</html>
```

Output

Sum of two numbers is 50

Example

```
<!doctype html>
<html>
  <head>
    <script type="text/javascript">
      function add()
      {
        var num1, num2, sum;
        num1 = parseInt(document.getElementById("firstnumber").value);
        num2 = parseInt(document.getElementById("secondnumber").value);
        sum = num1 + num2;
```

```
        document.getElementById("answer").value = sum;
      }
    </script>
  </head>
  <body>
    <p>First Number: <input id="firstnumber"></p>
    <p>Second Number: <input id="secondnumber"></p>
    <button onclick="add()">Add</button>
    <p>Sum = <input id="answer"></p>
  </body>
</html>
```

Output

First Number: 20

Second Number: 10

Add

Sum = 30

Data Types

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

Primitive data types

There are five types of primitive data types in JavaScript.

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

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

Examples

```
// Numbers:
let length = 16;
let weight = 7.5;
```

```
// Strings:
let color = "Yellow";
let lastName = "Johnson";

// Booleans
let x = true;
let y = false;

// Object:
const person = {firstName:"John", lastName:"Doe"};

// Array object:
const cars = ["Saab", "Volvo", "BMW"];

// Date object:
const date = new Date("2022-03-25");
```

Operators

JavaScript operators are symbols that are used to perform operations on operands.

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

Arithmetic Operators

Operator	Description	Example
+	Addition	10+20 = 30

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

Comparison Operators

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

Example

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

```
<h1>JavaScript Comparison</h1>

<h2>The == Operator</h2>

<p>Assign 5 to x, and display the value of the comparison (x == 8):</p>

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

<script>

let x = 5;

document.getElementById("demo").innerHTML = (x == 8);

</script>

</body>

</html>
```

Output

JavaScript Comparison

The == Operator

Assign 5 to x, and display the value of the comparison (x == 8):

false

Example

```
<!DOCTYPE html>

<html>

<body>
```

```
<h1>JavaScript Comparison</h1>

<h2>The != Operator</h2>

<p>Assign 5 to x, and display the value of the comparison (x != 8).</p>

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

<script>

let x = 5;

document.getElementById("demo").innerHTML = (x != 8);

</script>

</body>

</html>
```

Output

JavaScript Comparison

The != Operator

Assign 5 to x, and display the value of the comparison (x != 8).

true

Example

```
<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Comparison</h1>

<h2>The && Operator (Logical AND)</h2>
```

```
<p>The && operator returns true if both expressions are true, otherwise it returns false.</p>

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

<script>

let x = 6;

let y = 3;

document.getElementById("demo").innerHTML = (x < 10 && y > 1) + "<br>" +

(x < 10 && y < 1);

</script>

</body>

</html>
```

Output

JavaScript Comparison

The && Operator (Logical AND)

The && operator returns true if both expressions are true, otherwise it returns false.

```
true
false
```

Example

```
<!DOCTYPE html>

<html>

<body>
```

```
<h1>JavaScript Comparison</h1>

<h2>The (Logical OR)</h2>

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

<script>

let x = 6;

let y = 3;

document.getElementById("demo").innerHTML =

(x == 5 || y == 5) + "<br>" +

(x == 6 || y == 0) + "<br>" +

(x == 0 || y == 3) + "<br>" +

(x == 6 || y == 3);

</script>

</body>

</html>
```

Output

JavaScript Comparison

The (Logical OR)

false
true

true
true

Bitwise Operators

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

Logical Operators

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false

!	Logical Not	!(10==20) = true

Assignment Operators

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

Variable

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

```
<html>

<body>

<script>

var x = 10;

var y = 20;

var z=x+y;

document.write(z);
```

```
</script>

</body>

</html>
```

Local variable

A JavaScript local variable is declared inside block or function.

```
<script>
function abc(){
var x=10;//local variable
}
</script>
```

Global variable

A **JavaScript global variable** is accessible from any function.

```
<script>
var data=200;//gloabal variable
function a(){
document.writeln(data);
}
function b(){
document.writeln(data);
}
a();//calling JavaScript function
b();
</script>
```

The if Statement

The if statement to specify a block of JavaScript code to be executed if a condition is true.

Syntax

```
if (condition)
 {
  //  block of code to be executed if the condition is true
}
```

Example

```
if (hour < 18)

{
  greeting = "Good day";
}
```

Output

Good day

If...else Statement

It evaluates the content whether condition is true of false.

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

Example

```
<html>
<body>
<script>
var a=20;
```

```
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
</body>
</html>
```

Loops

The *JavaScript loops* are used to iterate the piece of code using for, while, do while or for-in loops.

1. for loop
2. while loop

For loop

The *JavaScript for loop* iterates the elements for the fixed number of times

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

Example

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

```
<script>
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
</script>
</body>
</html>
```

Output

1
2
3
4
5

While loop

The **JavaScript while loop** *iterates the elements for the infinite number of times*. It should be used if number of iteration is not known.

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

Example

```
<!DOCTYPE html>
<html>
<body>
<script>
var i=11;
while (i<=15)
{
document.write(i + "<br/>");
```

```
i++;
}
</script>
</body>
</html>
```

Output:

```
11
12
13
14
15
```

Functions

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

Advantage

1. **Code reusability**: We can call a function several times so it save coding.
2. **Less coding**: It makes our program compact. We don't need to write many lines of code each time to perform a common task.

Syntax

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

Example

```
<html>
<body>
<script>
```

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

HTML DOM Events

DOM Events allow JavaScript to add event listener or event handlers to HTML elements.

- When a user clicks the mouse
- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted
- When a user strokes a key

Examples

In HTML onclick is the event listener, myFunction is the event handler:

<button onclick="myFunction()">Click me</button>

In JavaScript click is the event, myFunction is the event handler:

button.addEventListener("click", myFunction);

Example

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

<h1 onclick="this.innerHTML = 'Ooops!'">Click on this text!</h1>

</body>
</html>
```

Event	Occurs When	Belongs To
abort	The loading of a media is aborted	UiEvent, Event
afterprint	A page has started printing	Event
animationend	A CSS animation has completed	AnimationEvent

animationiteration	A CSS animation is repeated	AnimationEvent
animationstart	A CSS animation has started	AnimationEvent
beforeprint	A page is about to be printed	Event
beforeunload	Before a document is about to be unloaded	UiEvent, Event
blur	An element loses focus	FocusEvent
canplay	The browser can start playing a media (has buffered enough to begin)	Event
canplaythrough	The browser can play through a media without stopping for buffering	Event
change	The content of a form element has changed	Event
click	An element is clicked on	MouseEvent

contextmenu	An element is right-clicked to open a context menu	MouseEvent
copy	The content of an element is copied	ClipboardEvent
cut	The content of an element is cut	ClipboardEvent
dblclick	An element is double-clicked	MouseEvent
drag	An element is being dragged	DragEvent
dragend	Dragging of an element has ended	DragEvent
dragenter	A dragged element enters the drop target	DragEvent
dragleave	A dragged element leaves the drop target	DragEvent
dragover	A dragged element is over the drop target	DragEvent

dragstart	Dragging of an element has started	DragEvent
drop	A dragged element is dropped on the target	DragEvent
durationchange	The duration of a media is changed	Event
ended	A media has reach the end ("thanks for listening")	Event
error	An error has occurred while loading a file	ProgressEvent, UiEvent, Event
focus	An element gets focus	FocusEvent
focusin	An element is about to get focus	FocusEvent
focusout	An element is about to lose focus	FocusEvent
fullscreenchange	An element is displayed in fullscreen mode	Event

fullscreenerror	An element can not be displayed in fullscreen mode	Event
hashchange	There has been changes to the anchor part of a URL	HashChangeEvent
input	An element gets user input	InputEvent, Event
invalid	An element is invalid	Event
keydown	A key is down	KeyboardEvent
keypress	A key is pressed	KeyboardEvent
keyup	A key is released	KeyboardEvent
load	An object has loaded	UiEvent, Event
loadeddata	Media data is loaded	Event
loadedmetadata	Meta data (like dimensions and duration) are loaded	Event

loadstart	The browser starts looking for the specified media	ProgressEvent
message	A message is received through the event source	Event
mousedown	The mouse button is pressed over an element	MouseEvent
mouseenter	The pointer is moved onto an element	MouseEvent
mouseleave	The pointer is moved out of an element	MouseEvent
mousemove	The pointer is moved over an element	MouseEvent
mouseover	The pointer is moved onto an element	MouseEvent
mouseout	The pointer is moved out of an element	MouseEvent

mouseup	A user releases a mouse button over an element	MouseEvent
mousewheel	Deprecated. Use the wheel event instead	WheelEvent
offline	The browser starts working offline	Event
online	The browser starts working online	Event
open	A connection with the event source is opened	Event
pagehide	User navigates away from a webpage	PageTransitionEvent
pageshow	User navigates to a webpage	PageTransitionEvent
paste	Some content is pasted in an element	ClipboardEvent
pause	A media is paused	Event
play	The media has started or is no longer	Event

	paused	
playing	The media is playing after being paused or buffered	Event
popstate	The window's history changes	PopStateEvent
progress	The browser is downloading media data	Event
ratechange	The playing speed of a media is changed	Event
resize	The document view is resized	UiEvent, Event
reset	A form is reset	Event
scroll	An scrollbar is being scrolled	UiEvent, Event
search	Something is written in a search field	Event
seeked	Skipping to a media position is finished	Event

seeking	Skipping to a media position is started	Event
select	User selects some text	UiEvent, Event
show	A <menu> element is shown as a context menu	Event
stalled	The browser is trying to get unavailable media data	Event
storage	A Web Storage area is updated	StorageEvent
submit	A form is submitted	Event
suspend	The browser is intentionally not getting media data	Event
timeupdate	The playing position has changed (the user moves to a different point in the media)	Event
toggle	The user opens or closes the	Event

	<details> element	
touchcancel	The touch is interrupted	TouchEvent
touchend	A finger is removed from a touch screen	TouchEvent
touchmove	A finger is dragged across the screen	TouchEvent
touchstart	A finger is placed on a touch screen	TouchEvent
transitionend	A CSS transition has completed	TransitionEvent
unload	A page has unloaded	UiEvent, Event
volumechange	The volume of a media is changed (includes muting)	Event
waiting	A media is paused but is expected to resume (e.g. buffering)	Event

Introduction to React Native

What is React Native?

React Native is a JavaScript framework used for developing a real, native mobile application for iOS and Android. It uses only JavaScript to build a mobile application. It is like React, which uses native component rather than using web components as building blocks.

What are React Native apps?

React Native apps are not web application. They are running on a mobile device, and it does not load over the browser. It is also not a Hybrid app that builds over Ionic, Phone Gap, etc. that runs over WebView component. React Native apps are the real native app, the JavaScript code stays as JavaScript, and they run in some extra thread by the compiled app. The user interface and everything are compiled to native code.

History of React Native

Facebook develops the React Native in 2013 for their internal project **Hackathon**. Later on, it was released publically in January 2015 as React.js, and in March 2015, Facebook announced that React Native is open and available on GitHub.

React Native was initially developed for the iOS application. However, recently it also supports the Android operating system.

Advantages of React Native

1. **Cross-Platform Usage:** Provide facility of "Learn once write everywhere", it works for both platform Android as well iOS devices.
2. **Class Performance:** The code written in React Native are compiled into native code, which enables it for both operating systems as well as it functions in the same way on both the platforms.
3. **JavaScript:** The JavaScript knowledge is used to build native mobile apps.
4. **Community:** The large community of React and React Native around helps us to find any answer we require.
5. **Hot Reloading:** Making a few changes in the code of your app will be immediately visible during development. If business logic is changed, its reflection is live reloaded on screen.

1. **Improving with Time:** Some features of iOS and Android are still not supported, and the community is always inventing the best practices.
2. **Native Components:** We will need to write some platform specific code if we want to create native functionality which is not designed yet.
3. **Existence is Uncertain:** As the Facebook develop this framework, its presence is uncertain since it keeps all the rights to kill off the project anytime. As the popularity of React Native rises, it is unlikely to happen

Unit 2

What is PHP

PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development.

- PHP stands for Hypertext Preprocessor.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is faster than other scripting languages, for example, ASP and JSP.
- PHP is a server-side scripting language, which is used to manage the dynamic content of the website.
- PHP can be embedded into HTML.
- PHP is an object-oriented language.
- PHP is an open-source scripting language.
- PHP is simple and easy to learn language.

PHP what is OOP?

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the PHP code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

Creating a class

A class is defined by using the class keyword, followed by the name of the class and a pair of curly braces ({}).

1. Class is a programmer-defined data type, which includes local methods and local variables.
2. Class is a collection of objects. Object has properties and behavior.

Syntax

```
<?php
class Fruit

{
  // code goes here...
}
?>
```

Example

```
<?php
class Fruit

{
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name)

{
    $this->name = $name;
  }
  function get_name()

{
    return $this->name;
  }
}
?>
```

Define Objects

Classes are nothing without objects! We can create multiple objects from a class. Each object has all the properties and methods defined in the class, but they will have different property values.

Objects of a class are created using the `new` keyword.

```
<!DOCTYPE html>
<html>
<body>
<?php
class Fruit
 {
  // Properties
  public $name;
  public $color;
  // Methods
  function set_name($name)
{
    $this->name = $name;
  }
  function get_name()
{
    return $this->name;
  }
```

```
}

$apple = new Fruit();

$banana = new Fruit();

$apple->set_name('Apple');

$banana->set_name('Banana');


echo $apple->get_name();

echo "<br>";

echo $banana->get_name();

?>

</body>

</html>
```

PHP Constructors and Destructors

Constructors are special member functions for initial settings of newly created object instances from a class, which is the key part of the object-oriented concept in **PHP5**.

Constructors are the very basic building blocks that define the future object and its nature. You can say that the Constructors are the blueprints for object creation providing values for member functions and member variables.

Once the object is initialized, the constructor is automatically called. Destructors are for destroying objects and automatically called at the end of execution.

Syntax:

- **__construct():**

```
function __construct()
    {
    // initialize the object and its properties by assigning
    //values
    }
```

- **__destruct():**

```
function __destruct()
    {
    // destroying the object or clean up resources here
    }
```

- **Default Constructor:**It has no parameters, but the values to the default constructor can be passed dynamically.
- **Parameterized Constructor:** It takes the parameters, and also you can pass different values to the data members.
- **Copy Constructor:** It accepts the address of the other objects as a parameter.

Program

```
<?PHP
class Tree
{
    function Tree()
    {
        echo "Its a User-defined Constructor of the class Tree";
    }
    function __construct()
```

```
    {
        echo "Its a Pre-defined Constructor of the class Tree";
    }
}

$obj= new Tree();
?>
```

Output:

Its a Pre-defined Constructor of the class Tree

Parameterized Constructor: The constructor of the class accepts arguments or parameters.
The -> operator is used to set value for the variables. In the constructor method, you can assign values to the variables during object creation.

Program

```
<?php
class Employee
{
    Public $name;
    Public $position;
    function __construct($name,$position)
    {
        // This is initializing the class properties
        $this->name=$name;
```

```
        $this->position=$position;
    }
    function show_details()
    {
        echo $this->name." : ";
        echo "Your position is ".$this->profile."\n";
    }
}
$employee_obj= new Employee("Rakesh","developer");
$employee_obj->show_details();
$employee2= new Employee("Vikas","Manager");
$employee2->show_details();
?>
```

Output:

```
Rakesh : Your position is developer
Vikas : Your position is Manager
```

Advantages of using Constructors:

- Constructors provides the ability to pass parameters which are helpful in automatic initialization of the member variables during creation time .
- The Constructors can have as many parameters as required and they can be defined with the default arguments.
- They encourage re-usability avoiding re-initializing whenever instance of the class is created .

- You can start session in constructor method so that you don’t have to start in all the functions everytime.
- They can call class member methods and functions.
- They can call other Constructors even from Parent class.

Destructor: Destructor is also a special member function which is exactly the reverse of constructor method and is called when an instance of the class is deleted from the memory. Destructors (__destruct (void): void) are methods which are called when there is no reference to any object of the class or goes out of scope or about to release explicitly.
They don’t have any types or return value. It is just called before de-allocating memory for an object or during the finish of execution of PHP scripts or as soon as the execution control leaves the block.

Program

```
<?php

class SomeClass

        {

                function __construct()

                {

                        echo "In constructor, ";

                        $this->name = "Class object! ";

                }

                function __destruct()

                {

                        echo "destroying " . $this->name . "\n";

                }
```

```
    }
$obj = new Someclass();
?>
```

Advantages of destructors:

- Destructors give chance to objects to free up memory allocation , so that enough space is available for new objects or free up resources for other tasks.
- It effectively makes programs run more efficiently and are very useful as they carry out clean up tasks.

Constructors	Destructors
Accepts one or more arguments.	No arguments are passed. Its void.
function name is _construct().	function name is _destruct()
It has same name as the class.	It has same name as the class with prefix ~tilda.
Constructor is involved automatically when the object is created.	Destructor is involved automatically when the object is destroyed.
Used to initialize the instance of a class.	Used to de-initialize objects already existing to free up memory for new accommodation.
Used to initialize data members of class.	Used to make the object perform some task before it is destroyed.
Constructors can be overloaded.	Destructors cannot be overloaded.
It is called each time a class is instantiated or object is created.	It is called automatically at the time of object deletion .

Constructors	Destructors
Allocates memory.	It deallocates memory.
Multiple constructors can exist in a class.	Only one Destructor can exist in a class.
If there is a derived class inheriting from base class and the object of the derived class is created, the constructor of base class is created and then the constructor of the derived class.	The destructor of the derived class is called and then the destructor of base class just the reverse order of constructor.
The concept of copy constructor is allowed where an object is initialized from the address of another object.	No such concept is allowed.

PHP - What are Interfaces?

Interfaces allow you to specify what methods a class should implement.

Interfaces make it easy to use a variety of different classes in the same way. When one or more classes use the same interface, it is referred to as "polymorphism".

Interfaces are declared with the `interface` keyword:

Syntax

```
<?php
interface InterfaceName {
  public function someMethod1();
  public function someMethod2($name, $color);
  public function someMethod3() : string;
}
?>
```

Program

```
<!DOCTYPE html>
<html>
<body>
<?php
interface Animal {
  public function makeSound();
}
class Cat implements Animal
{
  public function makeSound()
 {
    echo "Meow";
  }
}
$animal = new Cat();
$animal->makeSound();
?>
</body>
</html>
```

Encapsulation in PHP

The OOPs concept of Encapsulation in PHP means, enclosing the internal details of the object to protect from external sources. It describes, combining the class, data variables and member functions that work on data together within a single unit to form an object.

- Encapsulation is a concept where we encapsulate all the **data and member functions** together to form an object.
- **Wrapping up data member and method together** into a single unit is called Encapsulation.
- Encapsulation also allows a **class to change its internal implementation** without hurting the overall functioning of the system.
- Binding the data with the code that manipulates it.
- It keeps the data and the code **safe from external interference**.

Example 1

```
<?php
class person
{
public $name;
public $age;
function __construct($n, $a)
{
$this->name=$n;
$this->age=$a;
}
public function setAge($ag)
 {

$this->ag=$ag;

}
```

```
public function display()

{

echo  "welcome ".$this->name."<br/>";

return $this->age-$this->ag;

}

}

$person=new person("sonoo",28);

$person->setAge(1);

echo "You are ".$person->display()." years old";

?>
```

Advantages of Encapsulation:

- **Data Hiding and Abstraction:** Unnecessary details, internal representation and implementation are hidden from the end-users for protection of data structure and the Class. Data access is prohibited from members of other classes by creating private methods. It protects the internal state of any object by keeping member variables private and preventing any inconsistent state. It is the enclosing of data and related operations into that object.
  Note: Encapsulation is used to hide internal views from the client.

- **Data security:** Encapsulation helps in making data very robust and secured, as the data and member functions are wrapped together to form an object. All the tasks are done inside without any external worry and it also makes life very easy.
- **Reduces complexity:** Encapsulation helps in reducing development complexity of the software by hiding the details of implementation and exposing the methods or operations.
- **Reusability:** There are instances, you don't have to re-write same functionality that you inherited from the parent class.
- **Reliability:** You can make the class read-only or write-only by writing SET or GET methods.
- **Easier testing of code:** Encapsulated PHP code is easy to test as the functions used for testing child class ensures the testing of parent class functions also.
- **Increased flexibility:** Class variables can be accessed by GET or SET methods increasing flexibility. It is easy to maintain as the internal implementation can be changed without changing the code.

Web Techniques

The web runs on HTTP, the HyperText Transfer Protocol. This protocol governs how web browsers request files from web servers and how the servers send the files back.

When a web browser requests a web page, it sends an HTTP request message to a web server. The request message always includes some header information, and it sometimes also includes a body. The web server responds with a reply message, which always includes header information and usually contains a body.

Introduction Variables

In PHP, a variable is declared using a **$ sign** followed by the variable name.

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.
- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.

Syntax

$variablename=value;

- A variable must start with a dollar ($) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, _).
- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.
- PHP variables are case-sensitive, so $name and $NAME both are treated as different variable.

Example

```
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
echo "float is: $y <br/>";
```

?>

Example

```
<?php
$x=5;
$y=6;
$z=$x+$y;
echo $z;
?>
```

Server Information

- SERVER_SOFTWARE –---Apache
- GATEWAY_INTERFACE--- CGI
- SERVER_PROTOCOL –HTTP
- SERVER_PORT—80
- REQUEST_METHOD---GET

Form Processing

HTML forms are used to send the user information to the server and returns the result back to the browser. For example, if you want to get the details of visitors to your website, and send them good thoughts, you can collect the user information by means of form processing. Then, the information can be validated either at the client-side or on the server-side. The final result is sent to the client through the respective web browser. To create a HTML form, **form** tag should be used.

Attributes of Form Tag:

Attribute	Description
name or	It specifies the name of the form and is used to identify individual

id	forms.
action	It specifies the location to which the form data has to be sent when the form is submitted.
method	It specifies the HTTP method that is to be used when the form is submitted. The possible values are **get** and **post**. If **get** method is used, the form data are visible to the users in the url. Default HTTP method is **get**.
encType	It specifies the encryption type for the form data when the form is submitted.
novalidate	It implies the server not to verify the form data when the form is submitted.

Controls used in forms

Form processing contains a set of controls through which the client and server can communicate and share information. The controls used in forms are:

- **Textbox:** Textbox allows the user to provide single-line input, which can be used for getting values such as names, search menu and etc.
- **Textarea:** Textarea allows the user to provide multi-line input, which can be used for getting values such as an address, message etc.
- **DropDown:** Dropdown or combobox allows the user to provide select a value from a list of values.
- **Radio Buttons:** Radio buttons allow the user to select only one option from the given set of options.
- **CheckBox:** Checkbox allows the user to select multiple options from the set of given options.

- **Buttons:** Buttons are the clickable controls that can be used to submit the form.

Example

```
<?php
if (isset($_POST['submit']))
{
        if ((!isset($_POST['firstname'])) || (!isset($_POST['lastname'])) ||
                (!isset($_POST['address'])) || (!isset($_POST['emailaddress'])) ||
                (!isset($_POST['password'])) || (!isset($_POST['gender'])))
        {
                $error = "*" . "Please fill all the required fields";
        }
        else
        {
                $firstname = $_POST['firstname'];
                $lastname = $_POST['lastname'];
                $address = $_POST['address'];
                $emailaddress = $_POST['emailaddress'];
                $password = $_POST['password'];
                $gender = $_POST['gender'];
        }
}
?>
<html>
```

```
<head>
	<title>Simple Form Processing</title>
</head>

<body>
	<h1>Form Processing using PHP</h1>
	<fieldset>
		<form id="form1" method="post" action="form.php">
			<?php
				if (isset($_POST['submit']))
				{
					if (isset($error))
					{
						echo "<p style='color:red;'>"
						. $error . "</p>";
					}
				}
				?>

				FirstName:
				<input type="text" name="firstname"/>
				<span style="color:red;">*</span>
				<br>
				<br>
				Last Name:
				<input type="text" name="lastname"/>
```

```
<span style="color:red;">*</span>
<br>
<br>
Address:
<input type="text" name="address"/>
<span style="color:red;">*</span>
<br>
<br>
Email:
<input type="email" name="emailaddress"/>
<span style="color:red;">*</span>
<br>
<br>
Password:
<input type="password" name="password"/>
<span style="color:red;">*</span>
<br>
<br>
Gender:
<input type="radio"
        value="Male"
        name="gender"> Male
<input type="radio"
        value="Female"
        name="gender">Female
<br>
```

```
                    <br>
                    <input type="submit" value="Submit" name="submit" />
        </form>
</fieldset>
<?php
if(isset($_POST['submit']))
{
        if(!isset($error))
        {
                    echo"<h1>INPUT RECEIVED</h1><br>";
                    echo "<table border='1'>";
                    echo "<thead>";
                    echo "<th>Parameter</th>";
                    echo "<th>Value</th>";
                    echo "</thead>";
                    echo "<tr>";
                    echo "<td>First Name</td>";
                    echo "<td>".$firstname."</td>";
                    echo "</tr>";
                    echo "<tr>";
                    echo "<td>Last Name</td>";
                    echo "<td>".$lastname."</td>";
                    echo "</tr>";
                    echo "<tr>";
                    echo "<td>Address</td>";
                    echo "<td>".$address."</td>";
```

```
                    echo "</tr>";
                    echo "<tr>";
                    echo "<td>Email Address</td>";
                    echo "<td>" .$emailaddress."</td>";
                    echo "</tr>";
                    echo "<tr>";
                    echo "<td>Password</td>";
                    echo "<td>".$password."</td>";
                    echo "</tr>";
                    echo "<tr>";
                    echo "<td>Gender</td>";
                    echo "<td>".$gender."</td>";
                    echo "</tr>";
                    echo "</table>";
            }
        }
        ?>
</body>
</html>
```

PHP - What is Inheritance?

Inheritance in OOP = When a class derives from another class.

The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.

An inherited class is defined by using the extends keyword.

- **It supports the concept of hierarchical classification.**

- **Inheritance has three types, single, multiple and multilevel Inheritance.**
- **PHP supports only single inheritance, where only one class can be derived from single parent class.**
- **We can simulate multiple inheritances by using interfaces.**

Example 1

```
<?php
class a
   {
      function fun1()
      {
         echo "javatpoint";
      }
   }
   class b extends a
   {
      function fun2()
      {
         echo "SSSIT";
      }
   }
   $obj= new b();
   $obj->fun1();
?>
```

What is Database?

A database is an application that stores the organized collection of records. It can be accessed and manage by the user very easily. It allows us to organize data

into tables, rows, columns, and indexes to find the relevant information very quickly.

What is MySQL?

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications.

How MySQL Works?

MySQL follows the working of Client-Server Architecture. This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services. Here, the clients make requests through a graphical user interface (GUI), and the server will give the desired output as soon as the instructions are matched.

- MySQL is an open-source database, so you don't have to pay a single penny to use it.
- MySQL is a very powerful program that can handle a large set of functionality of the most expensive and powerful database packages.
- MySQL is customizable because it is an open-source database, and the open-source GPL license facilitates programmers to modify the SQL software according to their own specific environment.

- MySQL is quicker than other databases, so it can work well even with the large data set.
- MySQL supports many operating systems with many languages like PHP, PERL, C, C++, JAVA, etc.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL is very friendly with PHP, the most popular language for web development.

PHP MySQL Database

What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

PHP MySQL Connect Example

Example

```
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$conn = mysqli_connect($host, $user, $pass);
if(! $conn )
{
  die('Could not connect: ' . mysqli_error());
}
echo 'Connected successfully';
mysqli_close($conn);
?>
```

Output:

Connected successfully

There are three ways of working with MySQl and PHP

1. MySQLi (object-oriented)
2. MySQLi (procedural)
3. PDO

Using MySQLi object-oriented procedure

Syntax:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
// Creating connection
$conn = new mysqli($servername, $username, $password);
// Checking connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Output:

Connected successfully";

PHP MySQL Create Table

```
CREATE TABLE MyGuests
(
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)
```

- NOT NULL - Each row must contain a value for that column, null values are not allowed
- DEFAULT value - Set a default value that is added when no other value is passed
- UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
- AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
- PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

Example (MySQLi Object-oriented)Get your own PHP Server

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// sql to create table
```

```
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

MySQL Functions

Creating a function

A function always returns a value using the return statement. The function can be used in SQL queries.

Syntax

CREATE FUNCTION function_name [(parameter datatype [, parameter datatyp e])]

RETURNS return_datatype

BEGIN

Declaration_section

Executable_section

END;

Parameter:

Function_name: name of the function

Parameter: number of parameter. It can be one or more than one.

return_datatype: return value datatype of the function

declaration_section: all variables are declared.

executable_section: code for the function is written here.

Relational Database Management System

- RDBMS stands for Relational Database Management System.
- is a program used to maintain a relational database.
- RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access.
- RDBMS uses SQL queries to access the data in the database.

Unit 3

Introduction to XML

XML is a markup language that looks a lot like HTML. An XML document is plain text and contains tags delimited by < and >.

What is XML?

- XML stands for eXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Difference Between XML and HTML

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags are

XML is a markup language that looks a lot like HTML. An XML document is plain text and contains tags delimited by < and >

XML – Documents

An XML *document* is a basic unit of XML information composed of elements and other markup in an orderly package. An XML *document* can contains wide variety of data. For example, database of numbers, numbers representing molecular structure or a mathematical equation.

XML Document Example

```
<?xml version = "1.0"?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```

Document Prolog Section

Document Prolog comes at the top of the document, before the root element. This section contains −

- XML declaration
- Document type declaration

Document Elements Section

Document Elements are the building blocks of XML. These divide the document into a hierarchy of sections, each serving a specific purpose. You can separate a document into multiple sections so that they can be rendered differently, or used by a search engine. The elements can be containers, with a combination of text and other elements.

XML – Declaration

Syntax

```
<?xml

  version = "version_number"

  encoding = "encoding_declaration"

  standalone = "standalone_status"
?>
```

Rules

- If the XML declaration is present in the XML, it must be placed as the first line in the XML document.
- If the XML declaration is included, it must contain version number attribute.

- The Parameter names and values are case-sensitive.
- The names are always in lower case.
- The order of placing the parameters is important. The correct order is: *version, encoding and standalone.*
- Either single or double quotes may be used.
- The XML declaration has no closing tag i.e. **</?xml>**

XML Parser

SimpleXML Parser

SimpleXML is a tree-based parser.

SimpleXML provides an easy way of getting an element's name, attributes and textual content if you know the XML document's structure or layout.

SimpleXML turns an XML document into a data structure you can iterate through like a collection of arrays and objects.

Compared to DOM or the Expat parser, SimpleXML takes a fewer lines of code to read text data from an element.

PHP SimpleXML - Read From String

The PHP simplexml_load_string() function is used to read XML data from a string.

```
$myXMLData =
"<?xml version='1.0' encoding='UTF-8'?>
<note>
```

<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>";

Example

```
<?php
$myXMLData =
"<?xml version='1.0' encoding='UTF-8'?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>";

$xml=simplexml_load_string($myXMLData) or die("Error: Cannot create
object");
print_r($xml);
?>
```

output

SimpleXMLElement Object ([to] => Tove [from] => Jani [heading] => Reminder [body] => Don't forget me this weekend!)

SimpleXML - Read From File

The PHP `simplexml_load_file()` function is used to read XML data from a file.

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

XML document structure

An **XML (EXtensible Markup Language) Document** contains declarations, elements, text, and attributes. It is made up of entities (storing units) and It tells us the structure of the data it refers to. It is used to provide a standard format of data transmission. As it helps in message delivery, it is not always stored physically, i.e. in a disk but generated dynamically but its structure always remains the same.

XML Standard structure and its rules:

Rule 1: Its standard format consists of an **XML prolog** which contains both XML Declaration and XML DTD (Document Type Definition) and the body. If the XML prolog is present, it should always be the beginning of the document. The XML Version, by default, is **1.0**, and including only this forms the shortest XML Declaration. **UTF-8** is the default character encoding and is one of seven character-encoding schemes. If it is not present, it can result in some encoding errors.

Syntax of XML Declaration:

<?xml version="1.0" encoding="UTF-8"?>

Syntax of DTD:

<!DOCTYPE root-element [<!element-declarations>]>

<!DOCTYPE website [

<!ELEMENT website (name,company,phone)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT company (#PCDATA)>

<!ELEMENT phone (#PCDATA)>

]>

<website>

<name>GeeksforGeeks</name>

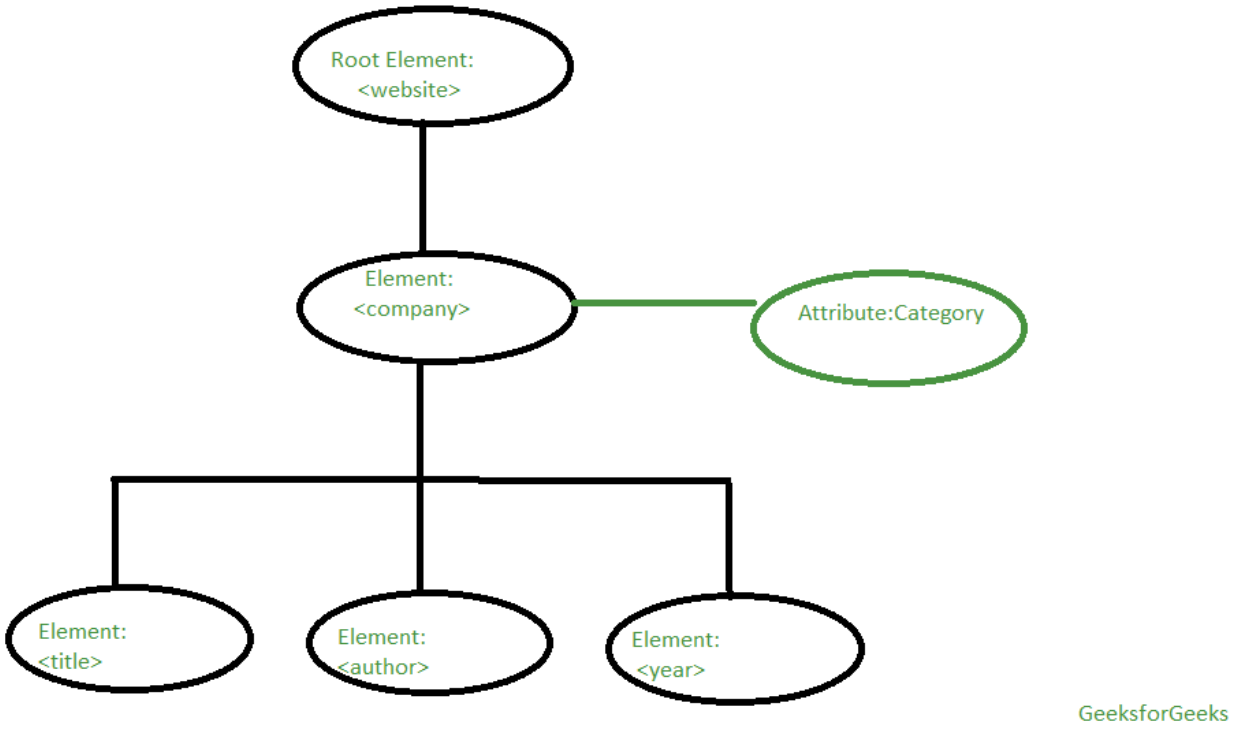
<company>GeeksforGeeks</company>

<phone>011-24567981</phone>

</website>

Rule 2: XML Documents must have a root element (the supreme parent element) and its child elements (sub-elements). To have a better view of the hierarchy of

the data elements, XML follows the XML tree structure which comprises of one single root (parent) and multiple leaves (children).



<?xml version="1.0" encoding="UTF-8"?>

<website>

<company category="geeksforgeeks">

<title>Machine learning</title>

<author>aarti majumdar</author>

<year>2022</year>

</company>

```
        <company category="geeksforgeeks">
            <title>Web Development</title>
            <author>aarti majumdar</author>
            <year>2022</year>
        </company>
        <company category="geeksforgeekse">
            <title>XML</title>
            <author>aarti majumdar</author>
            <year>2022</year>
        </company>
</website>
```

Rule 3: All XML Elements are required to have Closing and opening Tags(similar to HTML).

```
<message>Welcome to GeeksforGeeks</message>
```

Rule 4: The opening and closing tags are case-sensitive. For Example, <Message> is different from <message> from above example.

Rule 5: Values of XML attributes are required to have quotations:

```
<website category="open source">
        <company>geeksforgeeks</company>
</website>
```

Rule 6: **White-Spaces are retained and maintained in XML.**

<message>welcome to computer science</message>

Rule 7: Comments can be defined in XML enclosed between *<!–* and –> tags.

<!-- XML Comments are defined like this -->

Rule 8: XML elements must be nested properly.

<message>

<company>GeeksforGeeks</company>

</message>

AJAX Introduction

- Read data from a web server - after the page has loaded
- Update a web page without reloading the page
- Send data to a web server - in the background

What is AJAX?

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

AJAX is not a programming language.

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX Web Application Model

- AJAX Wen application model uses JavaScript and XMLHttpRequest object for asynchronous data exchange. The JavaScript uses the XMLHttpRequest object to exchange data asynchronously over the client and server.
- AJAX Web application model resolve the major problem of synchronous request-response model of communication associated with classical Web application model, which keeps the user in waiting state and does not provide the best user experience.
- AJAX, a new approach to Web applications, which is based on several technologies that help in developing applications with better user experience. It uses JavaScript and XML as the main technology for developing interactive Web applications
- The AJAX application eliminates the start-stop-start-stop nature or the click, wait, and refresh criteria of the client-server interaction by introducing intermediary layer between the user and the Web server.
- Instead of loading the Web page at the beginning of the session, the browser loads the AJAX engine written in JavaScript.
- Every user action that normally would generate an HTTP request takes the form of a JavaScript call to the AJAX Engine.
- The server response comprises data and not the presentation, which implies that the data required by the client is provided by the server as the response, and presentation is implemented on that data with the help of markup language from Ajax engine.
- The JavaScript does not redraw all activities instead only updates the Web page dynamically.

- In JavaScript, it is possible to fill Web forms and click buttons even when the JavaScript has made a request to the Web server and the server is still working on the request in the background. When server completes its processing, code updates just the part of the page that has changed. This way client never has to wait around. That is the power of asynchronous requests.
- AJAX Engine between the client and the application, irrespective of the server, does asynchronous communication. This prevents the user from waiting for the server to complete its processing.
- The AJAX Engine takes care of displaying the UI and the interaction with the server on the user's behalf.

AJAX - PHP

- AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and JavaScript.
- Conventional web applications transmit information to and from the server using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX, when submit is pressed, JavaScript will make a request to the server, interpret the results and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.

AJAX - PHP framework

PHP Ajax framework is able to deal with database, search data, and build pages or parts of

- Page and publish the page or return data to the XMLHttpRequest object.

- Quicknet is an Ajax framework that provides secure data transmission, uses PHP on the server side.It is designed to develop web applications or websites that use passwords to identify correct users.
- Using this framework, no cleartext password would be sent over the network or stored in the server.
- Quicknet supports multi-language, AJAX call, session and password management,modular structure, XML content and Javascript animation where PHP is used on the server side. AjaxAC :
- It is an open source framework written in PHP, used to develop/create/generate AJAX applications.
- The fundamental idea behind AJAX (Asynchronous Javascript and XML) is to use the XMLHttpRequest object to change a web page state using background HTTP subrequests without reloading the entire page.

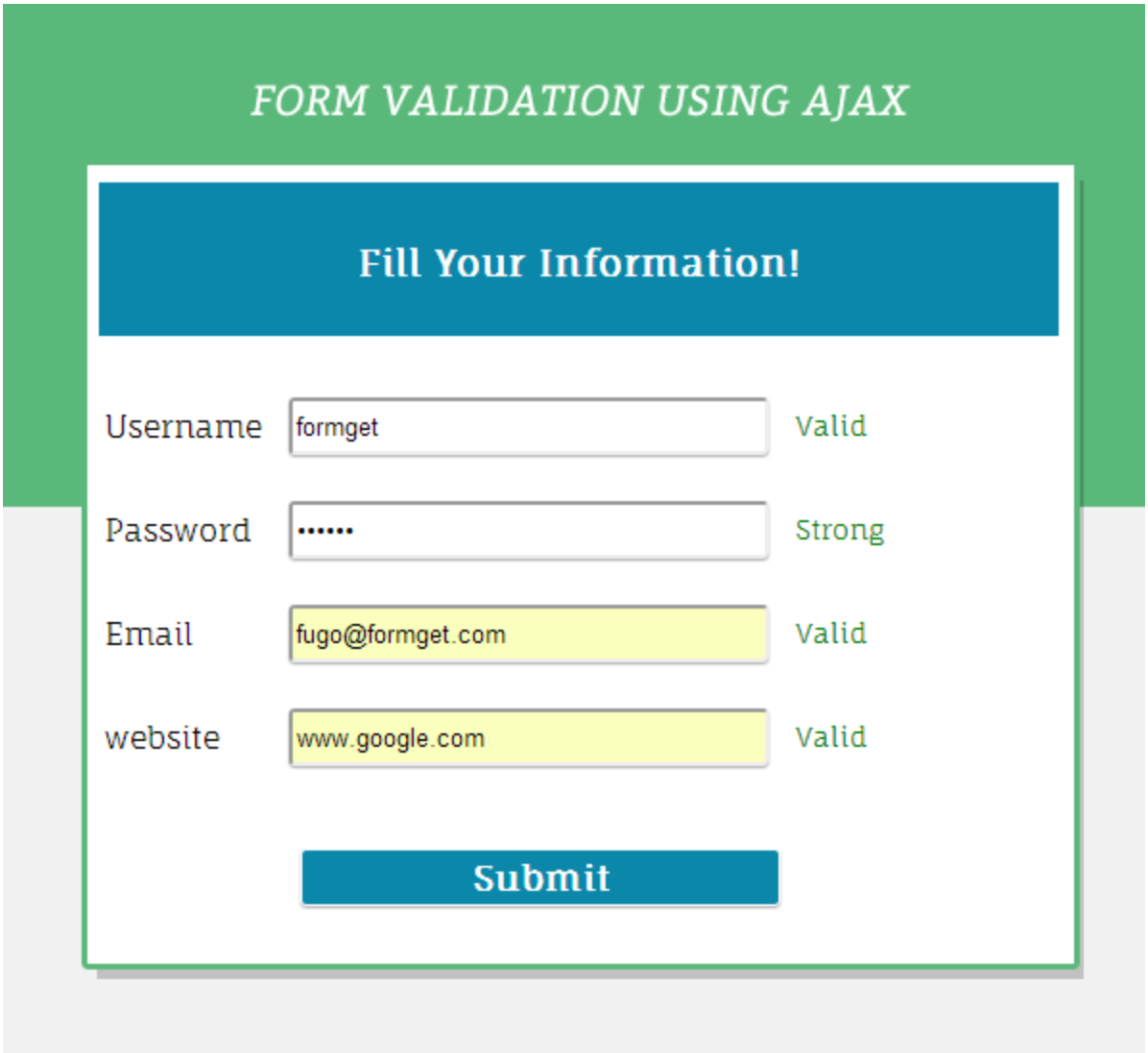
Features:

1. All application code is self contained in a single class (also additional Javascript libraries)
2. Calling PHP file/HTML page is veryclean. All that is required is creating the application classes, then referencing the application Javascript and attaching any required elements to the application.
3. Built in functionality for easy handling Javascript events.
4. Easy to hook in to existing PHP Classes or MySQL database for returning data from subrequests.

Form Validation Using Ajax

AJAX (Asynchronous JavaScript and XML) is the art of exchanging data with a server, and updating parts of a web page – without reloading the whole page. Therefore, Ajax makes web page quick responsive.

We have created an HTML form with four input fields and validation of each field is done by using combine logic of Ajax, PHP and Javascript.



HTML File: index.html

<!DOCTYPE html>

<html>

<head>

```
<link href="style.css" rel="stylesheet" type="text/css">
<script src="script.js"></script>
</head>
<body>
<div id="mainform">
<div class="innerdiv">
<!-- Required Div Starts Here -->
<h2>Form Validation Using AJAX</h2>
<form action='#' id="myForm" method='post' name="myForm">
<h3>Fill Your Information!</h3>
<table>
<tr>
<td>Username</td>
<td><input id='username1' name='username' onblur="validate('username',
this.value)" type='text'></td>
<td>
<div id='username'></div>
</td>
</tr>
```

<tr>

<td>Password</td>

<td><input id='password1' name='password' onblur="validate('password', this.value)" type='password'></td>

<td>

</td>

</tr>

<tr>

<td>Email</td>

<td><input id='email1' name='email' onblur="validate('email', this.value)" type='text'></td>

<td>

</td>

</tr>

<tr>

<td>website</td>

```
<td><input id='website1' name='website' onblur="validate('website', this.value)" type='text'></td>

<td>

<div id='website'></div>

</td>

</tr>

</table>

<input onclick="checkForm()" type='button' value='Submit'>

</form>

</div>

</body>

</html>
```

PHP File: validation.php

```
<?php

$value = $_GET['query'];

$formfield = $_GET['field'];

// Check Valid or Invalid user name when user enters user name in username input field.

if ($formfield == "username") {
```

```
if (strlen($value) < 4) {
echo "Must be 3+ letters";
} else {
echo "<span>Valid</span>";
}
}
// Check Valid or Invalid password when user enters password in password input field.
if ($formfield == "password") {
if (strlen($value) < 6) {
echo "Password too short";
} else {
echo "<span>Strong</span>";
}
}
// Check Valid or Invalid email when user enters email in email input field.
if ($formfield == "email") {
if (!preg_match("/([w-]+@[w-]+.[w-]+)/", $value)) {
echo "Invalid email";
```

```
} else {

echo "<span>Valid</span>";

}

}

// Check Valid or Invalid website address when user enters website address in website input field.

if ($formfield == "website") {

if (!preg_match("/b(?:(?:https?|ftp)://|www.)[-a-z0-9+&@#/%?=~_|!:,.;]*[-a-z0-9+&@#/%=~_|]/i", $value)) {

echo "Invalid website";

} else {

echo "<span>Valid</span>";

}

}

?>
```

JAVASCRIPT File: script.js

```
function checkForm()

{

// Fetching values from all input fields and storing them in variables.
```

```
var name = document.getElementById("username1").value;

var password = document.getElementById("password1").value;

var email = document.getElementById("email1").value;

var website = document.getElementById("website1").value;

//Check input Fields Should not be blanks.

if (name == '' || password == '' || email == '' || website == '') {

alert("Fill All Fields");

} else {

//Notifying error fields

var username1 = document.getElementById("username");

var password1 = document.getElementById("password");

var email1 = document.getElementById("email");

var website1 = document.getElementById("website");

//Check All Values/Informations Filled by User are Valid Or Not.If All Fields Are invalid Then Generate alert.

if (username1.innerHTML == 'Must be 3+ letters' || password1.innerHTML == 'Password too short' || email1.innerHTML == 'Invalid email' || website1.innerHTML == 'Invalid website') {

alert("Fill Valid Information");
```

```
} else {
//Submit Form When All values are valid.
document.getElementById("myForm").submit();
}
}
}
// AJAX code to check input field values when onblur event triggerd.
function validate(field, query) {
var xmlhttp;
if (window.XMLHttpRequest) { // for IE7+, Firefox, Chrome, Opera, Safari
xmlhttp = new XMLHttpRequest();
} else { // for IE6, IE5
xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange = function() {
if (xmlhttp.readyState != 4 && xmlhttp.status == 200) {
document.getElementById(field).innerHTML = "Validating..";
} else if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
document.getElementById(field).innerHTML = xmlhttp.responseText;
```

```
} else {
document.getElementById(field).innerHTML = "Error Occurred. <a href='index.php'>Reload Or Try Again</a> the page.";
}
}
xmlhttp.open("GET", "validation.php?field=" + field + "&query=" + query, false);
xmlhttp.send();
}
```

CSS File: style.css

```
@import "http://fonts.googleapis.com/css?family=Fauna+One|Muli";
#mainform{
width:960px;
margin:20px auto;
padding-top:20px;
font-family:'Fauna One',serif
}
#mainform h2{
width:100%;
float:left;
```

```
text-align:center;
margin-top:35px
}
.innerdiv{
width:65%;
float:left
}
form{
background-color:#fff;
color:#123456;
box-shadow:0 1px 1px 1px gray;
width:500px;
margin:50px 250px 0 50px;
float:left;
height:400px;
padding:10px
}
h3{
margin-top:0;
```

```
color:#fff;
background-color:#0B87AA;
text-align:center;
width:100%;
height:50px;
padding-top:30px
}
input{
width:250px;
height:30px;
margin-top:10px;
border-radius:3px;
padding:2px;
box-shadow:0 1px 1px 0 #a9a9a9;
margin:10px
}
input[type=button]{
background-color:#0B87AA;
border:1px solid #fff;
```

```
font-family:'Fauna One',serif;

font-weight:700;

font-size:18px;

color:#fff;

width:50%;

margin-left:105px;

margin-top:30px

}

span{

color:green

}

#myForm div{

color:red;

font-size:14px

}
```

PHP and AJAX

- **Stands for**: PHP Hypertext Preprocessor (originally stood for Personal Home Page).

- **Purpose**: PHP is a server-side scripting language designed for web development. It is used to create dynamic web pages and applications.
- **How it works**: When a user requests a webpage, the PHP code on the server processes the request, interacts with databases if necessary, and generates the HTML content to be sent back to the user's browser.
- **Key features**:
    - It can be embedded within HTML code.
    - It supports a wide range of databases, making it versatile for handling data.
    - It is platform-independent and can run on various operating systems.
    - It is open-source and has a large community, which means there is a wealth of resources and libraries available.

AJAX:

- Stands for: Asynchronous JavaScript and XML.

Purpose: AJAX is a set of web development techniques that allow web pages to send and receive data from a server asynchronously (in the background) without the need to reload the entire page.

How it works: Instead of the traditional request-response model, where a user clicks a link or submits a form and the whole page is refreshed, AJAX uses JavaScript to send requests to the server and handle the response without requiring a full page reload.

Key features:

It provides a smoother and more interactive user experience, as it can update parts of a page without the need for a complete refresh.

It can be used to fetch data from a server, submit form data, and perform other operations without interrupting the user's interaction with the page.

It can work with various data formats, not just XML, despite the name. Nowadays, JSON (JavaScript Object Notation) is the most commonly used data format with AJAX.

Connecting database using PHP and AJAX

1. **Set Up the Database**:
    - Make sure you have a database server installed (e.g., MySQL, PostgreSQL).
    - Create a database and the necessary tables to store your data.
2. **Create a PHP Script for Database Connection**:
    - Create a PHP file (e.g., **db_connection.php**) to handle the database connection.
    - Use PHP's **mysqli** or PDO (PHP Data Objects) to connect to the database.

```
<?php

$servername = "localhost";

$username = "username"; // Your database username
```

```
$password = "password"; // Your database password

$dbname = "database_name"; // Your database name

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {

   die("Connection failed: " . $conn->connect_error);

}

?>
```

Create AJAX Script to Send and Receive Data:

- In your HTML file, include a JavaScript section where you'll write the AJAX code.
- Use the **XMLHttpRequest** object or a library like jQuery to send and receive data asynchronously.

```
function sendDataToServer(data) {

   var xhttp = new XMLHttpRequest();

   xhttp.onreadystatechange = function() {

      if (this.readyState == 4 && this.status == 200) {

         // Handle the response from the server

         console.log(this.responseText);

      }
```

```
    };
    xhttp.open("POST", "process_data.php", true);
    xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhttp.send("data=" + data);
}
```

Create a PHP Script to Process Data:

- Create a PHP file (e.g., **process_data.php**) that will receive data from the AJAX request and interact with the database.

```
<?php
include 'db_connection.php'; // Include the database connection script
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $data = $_POST['data']; // Get data sent from AJAX
    // Sanitize and validate data (to prevent SQL injection)
    // Perform necessary database operations
    // For example, you might insert data into a table
    $response = "Data processed successfully!";
    echo $response;
}
```

?>

Call the JavaScript Function from Your HTML:

- In your HTML file, call the JavaScript function (e.g., **sendDataToServer(data)**) when you want to send data to the server.

<button onclick="sendDataToServer('Some Data')">Send Data</button>

Unit 4

What is Node.js?

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36.

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

What Can Node.js Do?

- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database

What is a Node.js File?

- Node.js files contain tasks that will be executed on certain events
- A typical event is someone trying to access a port on the server

- Node.js files must be initiated on the server before having any effect
- Node.js files have extension ".js"

Features of Node.js

- **Asynchronous and Event Driven** − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
- **Very Fast** − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but Highly Scalable** − Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- **No Buffering** − Node.js applications never buffer any data. These applications simply output the data in chunks.
- **License** − Node.js is released under the MIT license.

Download Node.js archive

Download latest version of Node.js installable archive file from Node.js Downloads. At the time of writing this tutorial, following are the versions available on different OS.

OS	Archive name
Windows	node-v6.3.1-x64.msi
Linux	node-v6.3.1-linux-x86.tar.gz
Mac	node-v6.3.1-darwin-x86.tar.gz
SunOS	node-v6.3.1-sunos-x86.tar.gz

Installation on Windows

Use the MSI file and follow the prompts to install the Node.js. By default, the installer uses the Node.js distribution in C:\Program Files\nodejs. The installer should set the C:\Program Files\nodejs\bin directory in window's PATH environment variable. Restart any open command prompts for the change to take effect.

Verify installation: Executing a File

Create a js file named **main.js** on your machine (Windows or Linux) having the following code.

```
console.log("Hello, World!")
```

Now execute main.js file using Node.js interpreter to see the result −

```
$ node main.js
```

If everything is fine with your installation, this should produce the following result −

Hello, World!

Node.js - First Application

- **Import required modules** − We use the **require** directive to load Node.js modules.
- **Create server** − A server which will listen to client's requests similar to Apache HTTP Server.
- **Read request and return response** − The server created in an earlier step will read the HTTP request made by the client which can be a browser or a console and return the response.

Creating Node.js Application

Step 1 - Import Required Module

We use the **require** directive to load the http module and store the returned HTTP instance into an http variable as follows −

var http = require("http");

Step 2 - Create Server

We use the created http instance and call **http.createServer()** method to create a server instance and then we bind it at port 8081 using the **listen** method associated with the server instance. Pass it a function with parameters request and response. Write the sample implementation to always return "Hello World".

```
http.createServer(function (request, response) {
   // Send the HTTP header
   // HTTP Status: 200 : OK
   // Content Type: text/plain
   response.writeHead(200, {'Content-Type': 'text/plain'});

   // Send the response body as "Hello World"
   response.end('Hello World\n');
}).listen(8081);

// Console will print the message
console.log('Server running at http://127.0.0.1:8081/');
```

The above code is enough to create an HTTP server which listens, i.e., waits for a request over 8081 port on the local machine.

Step 3 - Testing Request & Response

Let's put step 1 and 2 together in a file called **main.js** and start our HTTP server as shown below −

```
var http = require("http");

http.createServer(function (request, response) {
   // Send the HTTP header
   // HTTP Status: 200 : OK
   // Content Type: text/plain
   response.writeHead(200, {'Content-Type': 'text/plain'});

   // Send the response body as "Hello World"
   response.end('Hello World\n');
}).listen(8081);

// Console will print the message
console.log('Server running at http://127.0.0.1:8081/');
```

Now execute the main.js to start the server as follows −

```
$ node main.js
```

Verify the Output. Server has started.

Server running at http://127.0.0.1:8081/

Make a Request to the Node.js Server

Open http://127.0.0.1:8081/ in any browser and observe the following result.

Hello World

Node.js Modules

What is a Module in Node.js?

Consider modules to be the same as JavaScript libraries.

A set of functions you want to include in your application.

Built-in Modules

Node.js has a set of built-in modules which you can use without any further installation.

Include Modules

To include a module, use the `require()` function with the name of the module:

```
var http = require('http');
```

Now your application has access to the HTTP module, and is able to create a server:

```
http.createServer(function (req, res)
 {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Hello World!');
}
```

listen(8080);

Create Your Own Modules

You can create your own modules, and easily include them in your applications.

The following example creates a module that returns a date and time object:

ExampleGet your own Node.js Server

Create a module that returns the current date and time:

```
exports.myDateTime = function ()
{
  return Date();
};
```

Use the `exports` keyword to make properties and methods available outside the module file.

Save the code above in a file called "myfirstmodule.js"

Include Your Own Module

Example

Use the module "myfirstmodule" in a Node.js file:

```
var http = require('http');
var dt = require('./myfirstmodule');
```

```
http.createServer(function (req, res)

{
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write("The date and time are currently: " + dt.myDateTime());
  res.end();
}).listen(8080);
```

Notice that we use ./ to locate the module, that means that the module is located in the same folder as the Node.js file.

Save the code above in a file called "demo_module.js", and initiate the file:

Initiate demo_module.js:

C:\Users*Your Name*>node demo_module.js

Node.js HTTP Module

The Built-in HTTP Module

Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol (HTTP).

To include the HTTP module, use the `require()` method:

```
var http = require('http');
```

Node.js as a Web Server

The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.

Use the `createServer()` method to create an HTTP server:

Example

```
var http = require('http');


//create a server object:
http.createServer(function (req, res)

{
  res.write('Hello World!'); //write a response to the client
  res.end(); //end the response
}).listen(8080); //the server object listens on port 8080
```

The function passed into the http.createServer() method, will be executed when someone tries to access the computer on port 8080.

Save the code above in a file called "demo_http.js", and initiate the file:

Initiate demo_http.js:

C:\Users*Your Name*>node demo_http.js

File System Module

Node.js as a File Server

The Node.js file system module allows you to work with the file system on your computer.

To include the File System module, use the require() method:

```
var fs = require('fs');
```

Common use for the File System module:

- Read files
- Create files
- Update files
- Delete files
- Rename files

Read Files

The fs.readFile() method is used to read files on your computer.

Assume we have the following HTML file (located in the same folder as Node.js):

demofile1.html

```
<html>
<body>
<h1>My Header</h1>
<p>My paragraph.</p>
</body>
</html>
```

Example

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res)
{
  //Open a file on the server and return its content:
  fs.readFile('demofile1.html', function(err, data)
{
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(8080);
```

Output

My Header

My paragraph.

Initiate demo_readfile.js:

C:\Users*Your Name*>node demo_readfile.js

Create Files

The File System module has methods for creating new files:

- fs.appendFile()
- fs.open()
- fs.writeFile()

The fs.appendFile() method appends specified content to a file.

Exapmle

```
var fs = require('fs');


//create a file named mynewfile1.txt:
fs.appendFile('mynewfile1.txt', 'Hello content!', function (err)
{
  if (err) throw err;
  console.log('Saved!');
});
```

Output

```
Saved!
```

The `fs.open()` method takes a "flag" as the second argument, if the flag is "w" for "writing", the specified file is opened for writing. If the file does not exist, an empty file is created:

Example

Create a new, empty file using the open() method:

```
var fs = require('fs');

fs.open('mynewfile2.txt', 'w', function (err, file)

{
  if (err) throw err;
  console.log('Saved!');
});
```

The fs.writeFile() method replaces the specified file and content if it exists. If the file does not exist, a new file, containing the specified content, will be created:

Example

Create a new file using the writeFile() method:

```
var fs = require('fs');

fs.writeFile('mynewfile3.txt', 'Hello content!', function (err)

{
  if (err) throw err;
  console.log('Saved!');
});
```

Update Files

The File System module has methods for updating files:

- fs.appendFile()
- fs.writeFile()

Example

Append "This is my text." to the end of the file "mynewfile1.txt":

```
var fs = require('fs');

fs.appendFile('mynewfile1.txt', ' This is my text.', function (err)

{
  if (err) throw err;
  console.log('Updated!');
});
```

The `fs.writeFile()` method replaces the specified file and content:

Example

Replace the content of the file "mynewfile3.txt":

```
var fs = require('fs');

fs.writeFile('mynewfile3.txt', 'This is my text', function (err) {
  if (err) throw err;
  console.log('Replaced!');
});
```

Delete Files

To delete a file with the File System module, use the `fs.unlink()` method.

The `fs.unlink()` method deletes the specified file:

Example

Delete "mynewfile2.txt":

```
var fs = require('fs');

fs.unlink('mynewfile2.txt', function (err)
{
  if (err) throw err;
  console.log('File deleted!');
});
```

Rename Files

To rename a file with the File System module, use the `fs.rename()` method.

The `fs.rename()` method renames the specified file:

Example

Rename "mynewfile1.txt" to "myrenamedfile.txt":

```
var fs = require('fs');

fs.rename('mynewfile1.txt', 'myrenamedfile.txt', function (err)
{
  if (err) throw err;
  console.log('File Renamed!');
});
```

URL Module

The Built-in URL Module

The URL module splits up a web address into readable parts.

To include the URL module, use the require() method:

```
var url = require('url');
```

Parse an address with the url.parse() method, and it will return a URL object with each part of the address as properties:

Example

```
var url = require('url');
var adr = 'http://localhost:8080/default.htm?year=2017&month=february';
var q = url.parse(adr, true);

console.log(q.host);
console.log(q.pathname);
```

```
console.log(q.search);

var qdata = q.query;


console.log(qdata.month);
```

Output

localhost:8080

/default

?year=2023&month=October

October

Node.js File Server

summer.html

```
<!DOCTYPE html>
<html>
<body>
<h1>Summer</h1>
<p>I love the sun!</p>
</body>
</html>
```

winter.html

```
<!DOCTYPE html>
<html>
<body>
<h1>Winter</h1>
<p>I love the snow!</p>
</body>
</html>
```

demo_fileserver.js:

```
var http = require('http');
var url = require('url');
var fs = require('fs');

http.createServer(function (req, res)
{
  var q = url.parse(req.url, true);
  var filename = "." + q.pathname;
  fs.readFile(filename, function(err, data)
 {
    if (err)
 {
      res.writeHead(404, {'Content-Type': 'text/html'});
      return res.end("404 Not Found");
    }
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
```

```
    return res.end();
  });
}).listen(8080);
```

Initiate demo_fileserver.js:

C:\Users*Your Name*>node demo_fileserver.js

Node.js NPM

What is NPM?

NPM is a package manager for Node.js packages, or modules if you like.

What is a Package?

A package in Node.js contains all the files you need for a module.

Modules are JavaScript libraries you can include in your project.

Download a Package

Downloading a package is very easy.

Open the command line interface and tell NPM to download the package you want.

I want to download a package called "upper-case":

```
Download "upper-case":
```

```
C:\Users\Your Name>npm install upper-case
```

Now you have downloaded and installed your first package!

NPM creates a folder named "node_modules", where the package will be placed. All packages you install in the future will be placed in this folder.

My project now has a folder structure like this:

C:\Users*My Name*\node_modules\upper-case

Using a Package

Once the package is installed, it is ready to use.

Include the "upper-case" package the same way you include any other module:

```
var uc = require('upper-case');
```

Create a Node.js file that will convert the output "Hello World!" into upper-case letters:

```
var http = require('http');
var uc = require('upper-case');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  /*Use our upper-case module to upper case a string:*/
  res.write(uc.upperCase("Hello World!"));
  res.end();
}).listen(8080);
```

Node.js Events

Events in Node.js

Every action on a computer is an event. Like when a connection is made or a file is opened.

Objects in Node.js can fire events, like the readStream object fires events when opening and closing a file:

Example

```
var fs = require('fs');

var readStream = fs.createReadStream('./demofile.txt');

/*Write to the console when the file is opened:*/
readStream.on('open', function ()
 {
  console.log('The file is open');
});
```

Output

```
The file is open
```

Events Module

Node.js has a built-in module, called "Events", where you can create-, fire-, and listen for- your own events.

To include the built-in Events module use the require() method. In addition, all event properties and methods are an instance of an EventEmitter object. To be able to access these properties and methods, create an EventEmitter object:

```
var events = require('events');
var eventEmitter = new events.EventEmitter();
```

The EventEmitter Object

We can assign event handlers to your own events with the EventEmitter object.

In the example below we have created a function that will be executed when a "scream" event is fired.

To fire an event, use the `emit()` method.

Example

```
var events = require('events');
var eventEmitter = new events.EventEmitter();

//Create an event handler:
var myEventHandler = function () {
  console.log('I hear a scream!');
}
```

```
eventEmitter.on('scream', myEventHandler);

eventEmitter.emit('scream');
```

Output

```
I hear a scream!
```

Node.js Upload Files

The Formidable Module

There is a very good module for working with file uploads, called "Formidable".

The Formidable module can be downloaded and installed using NPM:

```
C:\Users\Your Name>npm install formidable
```

After you have downloaded the Formidable module, you can include the module in any application:

```
var formidable = require('formidable');
```

Upload Files

Now you are ready to make a web page in Node.js that lets the user upload files to your computer:

Step 1: Create an Upload Form

Example

```
var http = require('http');


http.createServer(function (req, res)

{
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('<form action="fileupload" method="post" enctype="multipart/form-
data">');
  res.write('<input type="file" name="filetoupload"><br>');
  res.write('<input type="submit">');
  res.write('</form>');
  return res.end();
}).listen(8080);
```

Step 2: Parse the Uploaded File

Include the Formidable module to be able to parse the uploaded file once it reaches the server.

When the file is uploaded and parsed, it gets placed on a temporary folder on your computer.

Example

The file will be uploaded, and placed on a temporary folder:

```
var http = require('http');
var formidable = require('formidable');

http.createServer(function (req, res)
 {
  if (req.url == '/fileupload') {
    var form = new formidable.IncomingForm();
    form.parse(req, function (err, fields, files)
{
      res.write('File uploaded');
      res.end();
    });
  } else {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write('<form action="fileupload" method="post" enctype="multipart/form-data">');
    res.write('<input type="file" name="filetoupload"><br>');
    res.write('<input type="submit">');
    res.write('</form>');
    return res.end();
  }
}).listen(8080);
```

Step 3: Save the File

When a file is successfully uploaded to the server, it is placed on a temporary folder.

The path to this directory can be found in the "files" object, passed as the third argument in the `parse()` method's callback function.

To move the file to the folder of your choice, use the File System module, and rename the file:

Example

Include the fs module, and move the file to the current folder:

```
var http = require('http');
var formidable = require('formidable');
var fs = require('fs');

http.createServer(function (req, res)

{
  if (req.url == '/fileupload')

{
    var form = new formidable.IncomingForm();
    form.parse(req, function (err, fields, files)

{
      var oldpath = files.filetoupload.filepath;
      var newpath = 'C:/Users/Your Name/' +
```

```
files.filetoupload.originalFilename;
    fs.rename(oldpath, newpath, function (err)
{
      if (err) throw err;
      res.write('File uploaded and moved!');
      res.end();
    });
});
 } else
{
   res.writeHead(200, {'Content-Type': 'text/html'});
   res.write('<form action="fileupload" method="post" enctype="multipart/form-data">');
   res.write('<input type="file" name="filetoupload"><br>');
   res.write('<input type="submit">');
   res.write('</form>');
   return res.end();
 }
}).listen(8080);
```

Node.js Send an Email

The Nodemailer Module

The Nodemailer module makes it easy to send emails from your computer.

The Nodemailer module can be downloaded and installed using npm:

```
C:\Users\Your Name>npm install nodemailer
```

```
var nodemailer = require('nodemailer');
```

Send an Email

Now you are ready to send emails from your server.

Use the username and password from your selected email provider to send an email. This example will show you how to use your Gmail account to send an email:

Example Get your own Node.js Server

```
var nodemailer = require('nodemailer');

var transporter = nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: 'youremail@gmail.com',
    pass: 'yourpassword'
  }
});

var mailOptions = {
  from: 'youremail@gmail.com',
  to: 'myfriend@yahoo.com',
  subject: 'Sending Email using Node.js',
  text: 'That was easy!'
```

```
};

transporter.sendMail(mailOptions, function(error, info){
  if (error) {
    console.log(error);
  } else {
    console.log('Email sent: ' + info.response);
  }
});
```

Multiple Receivers

To send an email to more than one receiver, add them to the "to" property of the mailOptions object, separated by commas:

Example

Send email to more than one address:

```
var mailOptions =

{
  from: 'youremail@gmail.com',
  to: 'myfriend@yahoo.com, myotherfriend@yahoo.com',
  subject: 'Sending Email using Node.js',
  text: 'That was easy!'
}
```

Send HTML

To send HTML formatted text in your email, use the "html" property instead of the "text" property:

Example

Send email containing HTML:

```
var mailOptions =

{
  from: 'youremail@gmail.com',
  to: 'myfriend@yahoo.com',
  subject: 'Sending Email using Node.js',
  html: '<h1>Welcome</h1><p>That was easy!</p>'
}
```