INDEX

XHTML

Introduction:

HTML is defined using the Standard Generalized Markup Language(SGML),which is an ISO standard notation for describing text-formatting languages.
XHTML stands for eXtensible HyperText Markup Language. Developed in the year 2000, it's a hybrid of the programming languages of HTML and XML; and it is used to develop the structure of your web site.
It carries the balance of browser-friendliness and popularity found in HTML with the extensibility and strictness of XML.
HTML was designed to specify document structure at a higher and more abstract level HTML specified documents had to be displayed on a variety of computer systems, often using different browsers.
The latest version of HTML,4.01 was approved by w3c in late 1999.
XHTML stands for Extensible Hypertext Markup Language:
The XHTML 1.0 standard was approved in early 2000. XHTML is a redefinition of HTML 4.01 using XML. Now the latest version is XHTML1.1.
It is not a programming language, it is used to describe the general form and layout of documents by a browser.

Difference between XHTML and HTML5

XHTML vs HTML5

As the name denotes, HTML 5 is the fifth revision of HTML. HTML is a coding language that is used in the development of online scripts. HTML refers to Hyper Text Mark-up Language and is used in the development of web scripts and is one of the earliest languages developed. XHTML on the other hand is a language that is also used in the development of web pages. It stands for Extensible Hyper Text Markup Language and is a hybrid that bridges the gap between HTML 5 and XML. The main function of XHTML is to allow for flexible displays on the net for different devices. XHTML therefore refers to HTML 5 being defined in the scope of an XML application. What differences do these two programming languages offer?

The main function of HTML 5 is to allow for the web browsers to read HTML 5 elements that have been written within tags and convert the content in tags into visual content that the end user can view. The tags are developed on the back end of the site to help in the display of the content of interest. The tags that HTML 5 uses allow for the display of text, images and video to help the display of an amazing web page. XHTML on the other hand is a markup language that extends the scope of HTML 5. This means that HTML language is defined in XHTML as an XML application. The namespaces that XHTML uses correspond to the HTML language.

The first difference that is viewed between XHTML and HTML is that XHTML can be said to be a hybrid language that bridges HTML to XML. HTML5 on the other hand is only the fifth version of the initial HTML 5. Markup representation between XHTML and HTML 5 differ and this is a very special issue that differentiates the tow.

Another difference between the two is that XHTML bears a lot of similarity between it and HTML 4.0 while HTML 5.0 is indeed different from its predecessors and thus not similar in any way to XHTML. When it comes to being strict in following of laid down rules, XHTML is very strict, requiring you to close all tags that have been opened for the tags to work. HTML 5 is less strict and allows for some leeway of error.

HTML is also less strict on restrictions such as nesting tags in already open tags. XHTML strictly specifies which tags can be nested together within tags. Parsing requirements in HTML are borrowed from XML while HTML 5 makes use of its own parsing requirements. XHTML requires one to manually declare a namespace whereas in HTML 5, this requirement is not a necessity.

Type attributes are also needed in XHTML when writing script and type elements. These style attributes are however not needed in HTML 5 in the different script and style elements. XHTML will make use of a long doc type while HTML 5 Makes use of <!DOCTYPE html> . It is also important to note that contrary to optional use of dtd url in XML doctype, HTML does not come with such an option and it is mandatory to use it.

STANDARD XHTML DOCUMENT STRUCTURE:

The first line of every XHTML document is:

<?xml version= “1.0” encoding=”utf-8” ?>

Then SGML document command

<! DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.1//EN”

http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd>

The HTML identifies root element <html>, includes an attribute, xmlns ,that specifies XHTML namespace, as shown

<html xmlns= “http://www.w3.org/1999/xhtml”>

An XHTML document consists two parts: ‘head’ and ‘body’ ,the <head> element contains information about the document and the <body> contains content of the document and also having a ‘title’ for title inclusion.

XHTML SYNTAX:-

The following general syntax rules will help you use XHTML:

1. All tags begin with< and end with >. The tag name is given immediately following the leading <.Any attribute is given following the tag-name in the form:
   <tag attribute1=”value” attribute2=”value”…>
2. Tag and attribute names are given in lower case. They are of the form:
   attribute_name=”value” where the value is case sensitive.
3. Unrecognized tags and attributes are ignored by browsers.
4. Most elements include opening and closing tags.
5. Elements must be well-informed.
6. Attributes can be required or optional and can be given in any order.
7. Extra white spaces and line breaks are allowed between the tag name and attributes and around the +sign inside an attribute.
8. The block element may contain only block-level HTML element.

CORE ATTRIBUTES:-

The following are the core attributes of XHTML:

1. Id: Uniquely identifies the elements in a page.
2. Style: gives presentation styles for the individual elements
3. Class: specifies the style class for the elements

XHTML Tags:

<p> : paragraphs in the body of document

<br> : line breaks

<pre> : to preserve a white space in text

<h i> : is used for headings size,i=1,2,3,4,5,6

<blockquote> : block of text to be set off from the normal form

<i> : italic format

<b> : for bold

<em> : specifies that textual content is special

<sub> : subscript characters can be specified

<sup> : superscript characters can be specified

<strong> : is like the emphasis and content is bold

<code> : to specify a monospace font

<hr> : horizontal line

<meta> :to provide addition information about document

To include an image:

<img src=”path” alt=”picture name”/>

To point a different document specify the address in link tag:

<a href=”filename.html”>

<ul> : creates an unordered list

<li> : for list item

<ol> : ordered list

<dl> :definition list

<dt> : each term to be defined

<dd> : dentitions themselves are specified

<table> : to define a table

<tr> : table row

<th> : table heading

<td> : table data

<form> : components of form appear

<input> : to specify the controls

<select> : a menu is specified

<option> : a value of menu item

<textarea> : to specify a text field

<frameset> : no.of frames & their layouts can be specified

Browser Support

The latest versions of Apple Safari, Google Chrome, Mozilla Firefox, and Opera all support many HTML5 features and Internet Explorer 9.0 will also have support for some HTML5 functionality.

The mobile web browsers that come pre-installed on iPhones, iPads, and Android phones all have excellent support for HTML5.

New Features in HTML5

HTML5 introduces a number of new elements and attributes that can help you in building modern websites. Here is a set of some of the most prominent features introduced in HTML5.

- **New Semantic Elements** − These are like <header>, <footer>, and <section>.
- **Forms 2.0** − Improvements to HTML web forms where new attributes have been introduced for <input> tag.
- **Persistent Local Storage** − To achieve without resorting to third-party plugins.
- **WebSocket** − A next-generation bidirectional communication technology for web applications.
- **Server-Sent Events** − HTML5 introduces events which flow from web server to the web browsers and they are called Server-Sent Events (SSE).
- **Canvas** − This supports a two-dimensional drawing surface that you can program with JavaScript.
- **Audio & Video** − You can embed audio or video on your webpages without resorting to third-party plugins.
- **Geolocation** − Now visitors can choose to share their physical location with your web application.
- **Microdata** − This lets you create your own vocabularies beyond HTML5 and extend your web pages with custom semantics.
- **Drag and drop** − Drag and drop the items from one location to another location on the same webpage.

Exercise: 1.Develop college website using HTML5 and CSS

File name as collegewebsite_HTML5_CSS.html

```
<!DOCTYPE html> <!-- The new doctype -->

<html>

<head>

  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

  <title>Coding using HTML5 and CSS3 </title>

  <link rel="stylesheet" type="text/css" href="styles.css" />

    <style type="text/css">

      .clear {
            zoom: 1;
            display: block;
         }
      </style>
</head>
<body>
   <section id="page"> <!-- Defining the #page section with the section tag -->
   <header> <!-- Defining the header section of the page with the appropriate tag -->
      <h2><center>DECCAN COLLEGE OF ENGINEERING AND
TECHNOLOGY</center></h2>
      <h4><center>(Established by Dar-Us-Salam Educational Trust)
Approved by the A.I.C.T.E. New Delhi and Affiliated to Osmania University, Hyderabad
Dar-Us-Salam, Hyderabad - 500 001, Telangana, India</center></h4><br/><br/>
      <nav class="clear"> <!-- The nav link semantically marks your main site navigation -->
         <ul>
            <li><a href="#article1">ABOUT US</a></li>
            <li><a href="#article2">Management</a></li>
            <li><a href="#article3">Departments</a></li>
         </ul>
      </nav>
   </header>
 <!-- Article 1 start -->
   <div class="line"></div>  <!-- Dividing line -->
   <article id="article1"> <!-- The new article tag. The id is supplied so it can be scrolled into
view. -->

      <h2>MISSION & VISION</h2>

      <div class="line"></div>

      <div class="articleBody clear">

         <figure> <!-- The figure tag marks data (usually an image) that is part of the article --
>

               <img src="E:\Software From Musthafa\WP\LAB RECORD 2019\dcet_pic.jpg"
width="620" height="340" /></a>

         </figure>

<h5> </h5>

<p>

To create a &quot;Centre for excellence&quot; in the developing rural areas with a view to
expose the students to the vast technical and research opportunities of high standards
available in the emerging fields of Engineering & Technology, which will contribute to the
advancement of society and humankind.

Our Mission is to identify, scientifically evaluate and implement proven, prevention oriented,
forward-looking solutions to critical scientific and technological problems and bring technical
education of international standard within the reach of all and uplift the weaker Muslim
minority community in particular

<p> It is our effort to develop every student into a well qualifies Professional of the future
India. Come and join with us to face the challenging world with new vision.</p>

      </div>

   </article>

<footer> <!-- Marking the footer section -->

         <div class="line"></div>
```

```
        <p>&copy;Copyright 2019 - deccancollege.ac.in</p> <!-- Change the copyright
notice -->

        <a href="#" class="up">Go UP</a>

      </footer>

</body>

</html>
```

File name as styles.css

```
*{

        /* Universal reset: */

        margin:0;

        padding:0;

}

header,footer,

article,section,

hgroup,nav,

figure{

        /* Giving a display value to the HTML5 rendered elements: */

        display:block;

}

body{

        /* Setting the default text color, size, page background and a font stack: */

        font-size:0.825em;

        color:#fcfcfc;
```

```
	background-color:#abc123;

	font-family:Arial, Helvetica, sans-serif;

}

/* Hyperlink Styles: */

a, a:visited {

	color:#0196e3;

	text-decoration:none;

	outline:none;

}

a:hover{

	text-decoration:underline;

}

a img{

	border:none;

}

/* Headings: */

h1,h2,h3{

	font-family:"Myriad Pro","Helvetica Neue",Helvetica,Arial,Sans-Serif;

	text-shadow:0 1px 1px black;

}



h1{

	/* The logo text */
```

```
        font-size:3.5em;

        padding:0.5em 0 0;

        text-transform:uppercase;

}

h3{

        /* The slogan text */

        font-family:forte,"Myriad Pro","Helvetica Neue",Helvetica,Arial,Sans-Serif;

        font-size:2em;

        font-weight:normal;

        margin:0 0 1em;

}

h2{

        font-size:2.2em;

        font-weight:normal;

        letter-spacing:0.01em;

        text-transform:uppercase;

}

p{

        line-height:1.5em;

        padding-bottom:1em;

}

.line{

        /* The dividing line: */
        height:1px;

        background-color:#24404c;

        border-bottom:1px solid #416371;

        margin:1em 0;

        overflow:hidden;

}

article .line{

        /* The dividing line inside of the article is darker: */

        background-color:#15242a;

        border-bottom-color:#204656;

        margin:1.3em 0;

}

footer .line{

        margin:2em 0;

}

nav{

        background:url() repeat-x 50% 50% #f8f8f8;

        padding:0 5px;

        position:absolute;

        right:0;

        top:5em;

        border:1px solid #FCFCFC;

        -moz-box-shadow:0 1px 1px #333333;
```

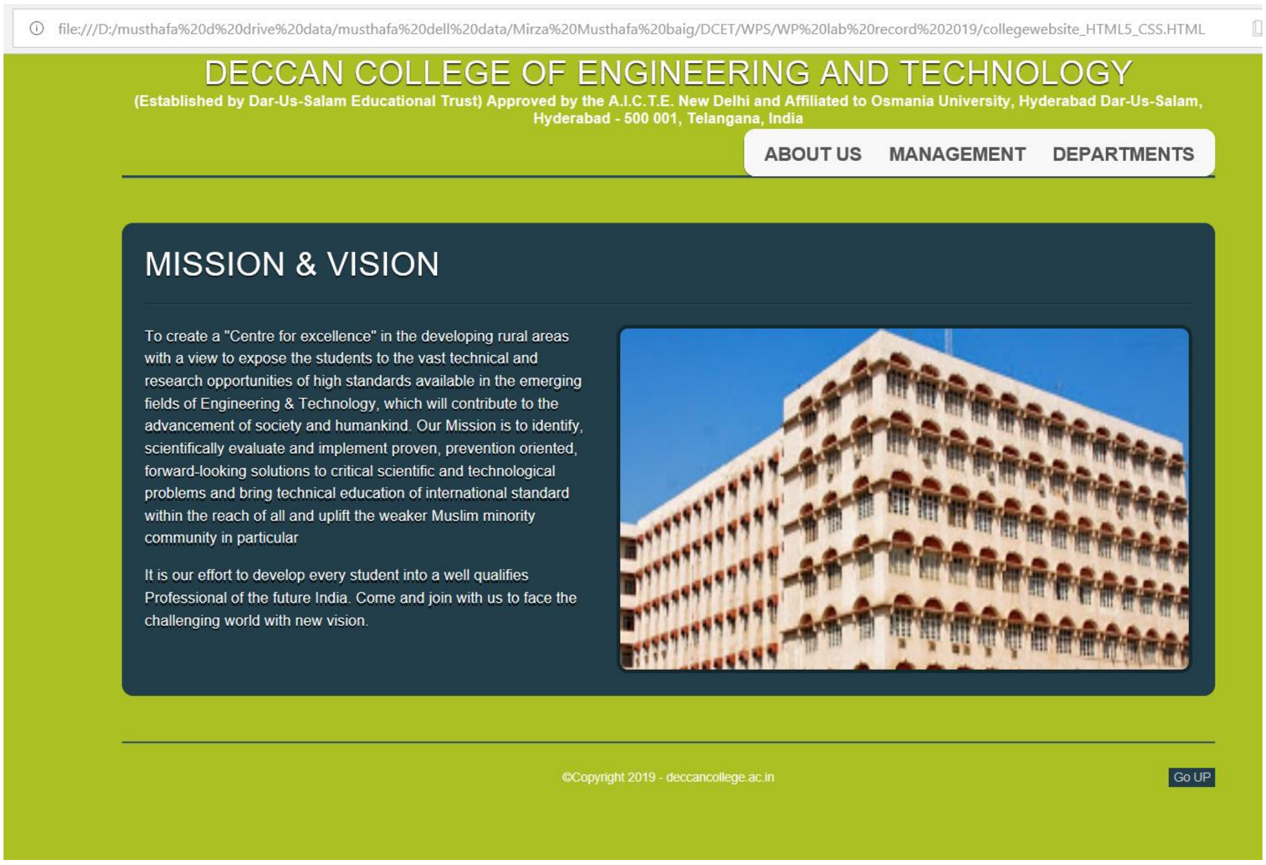
```
	-webkit-box-shadow:0 1px 1px #333333;

	box-shadow:0 1px 1px #333333;

}

/* The clearfix hack to clear the floats: */


.clear:after{

	content: ".";

	display: block;

	height: 0;

	clear: both;

	visibility: hidden;

}

/* The navigation styling: */

nav ul li{

	display:inline;

}

nav ul li a,

nav ul li a:visited{

	color:#565656;

	display:block;

	float:left;

	font-size:1.25em;

	font-weight:bold;
```

```
            margin:5px 2px;

            padding:7px 10px 4px;

            text-shadow:0 1px 1px white;

            text-transform:uppercase;

}

nav ul li a:hover{

            text-decoration:none;

            background-color:#f0f0f0;

}

nav, article, nav ul li a,figure{

            /* Applying CSS3 rounded corners: */

            -moz-border-radius:10px;

            -webkit-border-radius:10px;

            border-radius:10px;

}

/* Article styles: */

#page{

            width:960px;

            margin:0 auto;

            position:relative;

}

article{

            background-color:#213E4A;
```

```
        margin:3em 0;

        padding:20px;

        text-shadow:0 2px 0 black;

}

figure{

        border:3px solid #142830;

        float:right;

        height:300px;

        margin-left:15px;

        overflow:hidden;

        width:500px;

}

figure:hover{

        -moz-box-shadow:0 0 2px #4D7788;

        -webkit-box-shadow:0 0 2px #4D7788;

        box-shadow:0 0 2px #4D7788;

}

figure img{

        margin-left:-60px;

}

/* Footer styling: */

footer{

        margin-bottom:30px;
        text-align:center;

        font-size:0.825em;

}

footer p{

        margin-bottom:-2.5em;

        position:relative;

}

footer a,footer a:visited{

        color:#cccccc;

        background-color:#213e4a;

        display:block;

        padding:2px 4px;

        z-index:100;

        position:relative;

}

footer a:hover{

        text-decoration:none;

        background-color:#142830;

}

footer a.by{

        float:left;


}
```

```
footer a.up{

        float:right;

}
```

OUTPUT



file:///D:/musthafa%20d%20drive%20data/musthafa%20dell%20data/Mirza%20Musthafa%20baig/DCET/WPS/WP%20lab%20record%202019/collegewebsite_HTML5_CSS.HTML

DECCAN COLLEGE OF ENGINEERING AND TECHNOLOGY

(Established by Dar-Us-Salam Educational Trust) Approved by the A.I.C.T.E. New Delhi and Affiliated to Osmania University, Hyderabad Dar-Us-Salam, Hyderabad - 500 001, Telangana, India

MISSION & VISION

To create a "Centre for excellence" in the developing rural areas with a view to expose the students to the vast technical and research opportunities of high standards available in the emerging fields of Engineering & Technology, which will contribute to the advancement of society and humankind. Our Mission is to identify, scientifically evaluate and implement proven, prevention oriented, forward-looking solutions to critical scientific and technological problems and bring technical education of international standard within the reach of all and uplift the weaker Muslim minority community in particular

It is our effort to develop every student into a well qualifies Professional of the future India. Come and join with us to face the challenging world with new vision.

©Copyright 2019 - deccancollege.ac.in

Exercise 2: Develop HTML5 form with client validations using java script

The 'required' attribute

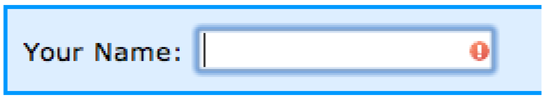
The simplest change you can make to your forms is to mark a text input field as 'required':

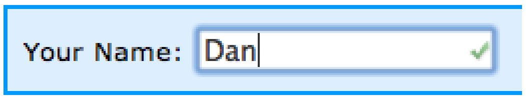
```
Your Name: <input type="text" name="name" required>
```

Your Name: [ ]

This informs the (HTML5-aware) web browser that the field is to be considered mandatory. Different browsers may mark the input box in some way (Firefox 4 Beta adds a red box-shadow by default), display a warning (Opera) or even prevent the form from being submitted if this field has no value. Hopefully these behaviours will converge in future releases.

For these examples we have created our own valid/invalid CSS formatting to override the browser default. More on that below. That's why you may see something like the following:





Before you type anything into the box a red marker is shown. As soon as a single character has been entered this changes to a green marker to indicate that the input is 'valid'.

Using CSS you can place markers inside or alongside the input box, or simply use background colours and borders as some browsers do by default.

The required attribute can also apply to checkboxes which we've covered separately.

2. New text INPUT types

This is where HTML5 really gets interesting and more useful. Along with the text input type, there are now a host of other options, including email, url, number, tel, date and many others.

On the iPhone/iPad the different input types are associated with different keyboards, making it easier for people to complete your online forms. In other web browsers they can be used in combination with the required attribute to limit or give advice on allowable input values.

INPUT type="email"

By changing the input type to email while also using the required attribute, the browser can be used to validate (in a limited fashion) email addresses:

```
Email Address: <input type="email" name="email" required placeholder="Enter a
valid email address">
```

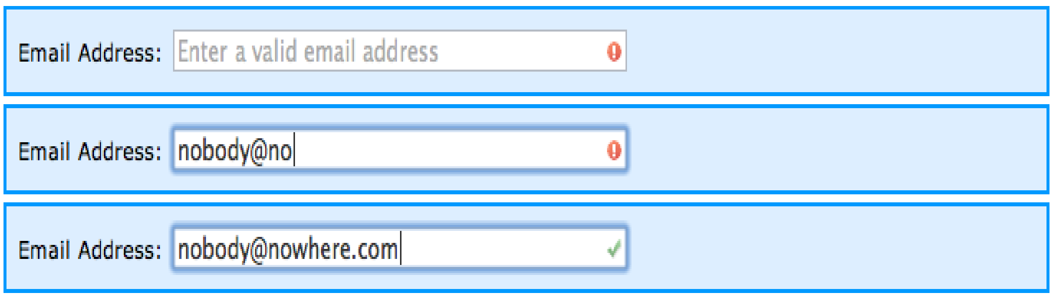
Note that for this example we've made use of another HTML5 attribute placeholder which lets us display a prompt or instructions inside the field - something that previously had to be implemented using messy onfocus and onblur JavaScript events.

The above code displays an input box as follows:

Email Address:

Again, different browsers implement this differently. In Opera it's sufficient to enter just *@* for the input to be accepted. In Safari, Chrome and Firefox you need to enter at least *@-.-. Obviously neither example is very limiting, but it will prevent people from entering completely wrong values, such as phone number, strings with multiple '@'s or spaces.

Here is how it appears in Safari (with our CSS formatting to show the (in)valid state):


Email Address: Enter a valid email address

Email Address: nobody@no

Email Address: nobody@nowhere.com

INPUT type="url"

In a similar fashion to the email input type above, this one is designed to accept only properly-formatted URLs. Of course it currently does nothing of the kind, but later you will see how to improve it's behaviour using the pattern attribute.

```
Website: <input type="url" name="website" required>
```

Again, the input box appears as normal:

Website:

This time the minimum requirement for most browsers is one or more letters followed by a colon. Again, not very helpful, but it will stop people trying to input their email address or other such nonsense.

As mentioned above, we can improve on this by making use of the pattern attribute which accepts a JavaScript regular expression. So the code above becomes:

```
Website: <input type="url" name="website" required pattern="https?://.+">
```

Now our input box will only accept text starting with http:// or https:// and at least one additional character:

Website: starting with http

If you're not yet familiar with regular expressions, you really should make it a priority to learn. For those already familiar, note that the ^ and $ are already implicit so the input has to match the entire expression. Pattern modifiers are not supported.

If anyone wants to contribute a more thorough expression to test for valid email or url format, feel free to post it using the Feedback option above..

INPUT type="number" and type="range"

The number and range input types also accept parameters for min, max and step. In most cases you can leave out step as it defaults to 1.

Here you see an example including both a number input, typically displayed as a 'roller' and a range input displayed as a 'slider':

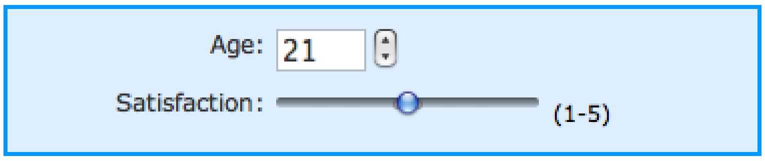
```
Age: <input type="number" size="6" name="age" min="18" max="99"
value="21"><br>

Satisfaction: <input type="range" size="2" name="satisfaction" min="1" max="5"
value="3">
```

As with other HTML5 input types, browsers that don't recognise the new options will default to simple text inputs. For that reason it's a good idea to include a size for the input box.

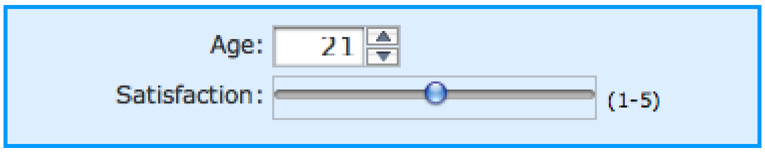
AgeSatisfaction (1-5)

The slider option is a bit bizarre in that no values are displayed, but may be useful for more 'analog' inputs. There are some bugs with the number input in that if you don't set a max value, clicking 'down' with the input blank will result in a very large number.

Here is how the two inputs are displayed in Safari:



and in Opera:



They are currently not supported in Firefox 4 Beta.

If you want to restrict the input of a text field to numbers without having the up/down arrows associated with the input box, you can always just set the input type to text and use a pattern of "\d+" (one or more numbers).

INPUT type="password"

We have a separate article with details on validating passwords using HTML5, including JavaScript code for customising the browser generated alert messages.

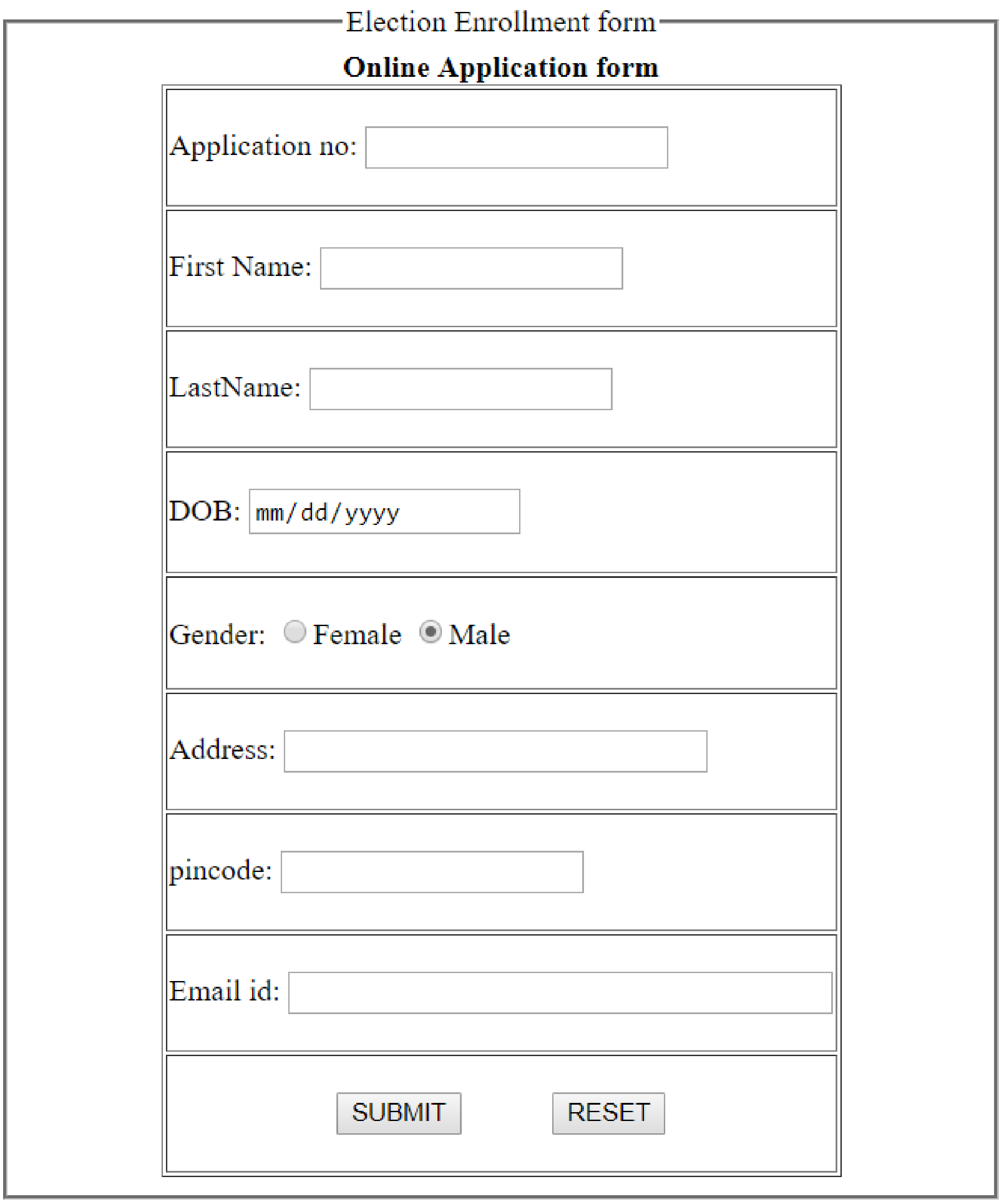
Program on validation.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"><html><head><META http-equiv="Content-Type" content="text/html;
charset=utf-8"></head><body>

<div bgcolor="aqua">
<form action="http://reg.html" align="left" target="_blank" onsubmit="try {return
window.confirm(&quot;This form may not function properly due to certain security
constraints.\nContinue?&quot;);} catch (e) {return false;}">
<center>
<fieldset style="width:500px">
<legend align="center"> Election Enrollment form </legend>
<center>
<table border="1" align="center">
<caption><b>Online Application form</b></caption>
<tr><td><br>
<label>Application no:</label>
```

```
<input maxlength="10"><br><br>
</td></tr>
<tr><td><br>
<label>First Name:</label>
<input><br><br>
</td></tr>
<tr><td><br>
<label>LastName:</label>
<input><br><br>
</td></tr>
<tr><td><br>
<label>DOB:</label>
<input type="date" name="dob" value="DD/MM/YYYY"><br><br>
</td></tr>
<tr><td><br>
<label>Gender:</label>
<input type="radio" value="Female">Female
<input type="radio" value="Male">Male
<br><br>
</td></tr>
<tr><td><br>
<label>Address:</label>
<input size="30"><br><br>
</td></tr>
<tr><td><br>
<label>pincode:</label>
<input type="number" maxlength="6"><br><br>
</td></tr>
<tr><td><br>
<label>Email id:</label>
<input type="email" name="email" size="40"><br><br>
</td></tr>
<tr><td align="center"><br>
<input type="submit" value="SUBMIT">

<input type="reset" value="RESET"><br><br>
</td></tr>
</table>
</center></fieldset>
</center>
</form>
```

```
</div>
</body></html>
```

OUTPUT



Election Enrollment form

Online Application form

Application no:	
First Name:	
LastName:	
DOB:	mm/dd/yyyy
Gender:	○ Female [✓ ● Male]
Address:	
pincode:	
Email id:	
SUBMIT	RESET

Exercise 3: Publishing XML document using XSLT

Difference between XML and HTML:

XML was designed to carry data, not displaying data

- XML is not a replacement for HTML.
- Different goals:

XML was designed to describe data and to focus on what data is.
HTML was designed to display data and to focus on how data looks.

- HTML is about displaying information, XML is about describing information.

How Can XML be Used?

- XML is used in many aspects of web development, often to simplify data storage and sharing.
- XML is Used to Create New Internet Languages
- XML Makes Your Data More Available
- XML Simplifies Data Transport

XML Tree:

- XML documents form a tree structure that starts at "the root" and branches to "the leaves".

Why Is XML Important?

- Plain Text
  - Easy to edit
  - Useful for storing small amounts of data
  - Possible to efficiently store large amounts of XML data through an XML front end to a database
- Data Identification
  - Tell you what kind of data you have
  - Can be used in different ways by different applications
- Stylability
  - Inherently style-free
  - XSL---Extensible Stylesheet Language

  - Different XSL formats can then be used to display the same data in different ways
- Hierarchical
  - Faster to access
  - Easier to rearrange

XML Building blocks:

- Element

  Delimited by angle brackets

  Identify the nature of the content they surround

  General format: <element> … </element>

  Empty element: <empty-Element/>
- Attribute

  Name-value pairs that occur inside start-tags after element name, like: <element attribute="value">

XML Building blocks—Prolog:

- The part of an XML document that precedes the XML data
- Includes

  A declaration: [version , encoding, standalone]

  An optional DTD (Document Type Definition)
- Example

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

XML Syntax:

- All XML elements must have a closing tag
- XML tags are case sensitive
- All XML elements must be properly nested
- All XML documents must have a root tag
- Attribute values must always be quoted
- Comments in XML: <!-- This is a comment -->

XML Elements:

- XML Elements are Extensible

  XML documents can be extended to carry more information
- XML Elements have Relationships

  Elements are related as parents and children
- Elements have Content

  Elements can have different content types: element content, mixed content, simple content, or empty content and attributes
- XML elements must follow the naming rules

XML Naming Rules:

- XML elements must follow these naming rules:
- Names can contain letters, numbers, and other characters
- Names must not start with a number or punctuation character
- Names must not start with the letters xml (or XML, or Xml, etc)
- Names cannot contain spaces
- Any name can be used, no words are reserved.

XML Attributes:

- Located in the start tag of elements
- Provide additional information about elements
- Often provide information that is not a part of data
- Must be enclosed in quotes
- Should I use an element or an attribute?

  metadata (data about data) should be stored as attributes, and that data itself should be stored as elements

Entity References:

- Some characters have a special meaning in XML.
- If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.
- <message>if salary < 1000 then</message>
- To avoid this error, replace the "<" character with an entity reference:
- <message>if salary &lt; 1000 then</message>

PCDATA:

- PCDATA means parsed character data
- Text found between the start tag and the end tag of an XML element
- Text that will be parsed
- Tags inside the text will be treated as markup and entities will be expanded

CDATA:

- CDATA means character data
- Text that will NOT be parsed
- Tags inside the text will NOT be treated as markup and entities will not be expanded

XML Namespaces:

- XML Namespaces provide a method to avoid element name conflicts.
- *What is Namespace? Why is it necessary?*
- *Solving the Name Conflict Using a Prefix*
- *Namespace Declaration*
- *Namespace Declaration Scope*
- *Namespaces with Attributes*
- *Default Namespaces*
- *Overwriting a Default Namespace*

Well Formed Document:

- An xml declaration should begin the document

- Include one or more elements
- Include Both Start and End Tags for Elements that aren't empty
- Close empty tags with />
- Root Elements must contains All other elements
- Nest Elements Correctly
- Use unique attribute names

XML Validation:

- "Well Formed" XML document
  --correct XML syntax
- "Valid" XML document
  - "well formed"
  - Conforms to the rules of a DTD (Document Type Definition)
- XML DTD
  - defines the legal building blocks of an XML document
  - Can be inline in XML or as an external reference
- XML Schema
  - an XML based alternative to DTD, more powerful
  - Support namespace and data types

Displaying XML:

- XML documents do not carry information about how to display the data
- We can add display information to XML with
  - CSS (Cascading Style Sheets)
  - XSL (eXtensible Stylesheet Language) --- preferred

XML Applications:

XML can Separate Data from HTML

- Store data in separate XML files
- Using HTML for layout and display
- Using Data Islands

- Data Islands can be bound to HTML elements

Benefits:

Changes in the underlying data will not require any changes to your HTML

XML is used to Exchange Data

- Text format
- Software-independent, hardware-independent
- Exchange data between incompatible systems, given that they agree on the same tag definition.
- Can be read by many different types of applications

Benefits:

- Reduce the complexity of interpreting data
- Easier to expand and upgrade a system

XML can be used to Store Data

- Plain text file
- Store data in files or databases
- Application can be written to store and retrieve information from the store
- Other clients and applications can access your XML files as data sources

Benefits:

Accessible to more applications

XML can be used to Create new Languages

- WML (Wireless Markup Language) used to markup Internet applications for handheld devices like mobile phones (WAP)
- MusicXML used to publishing musical scores
- XML is a self-descriptive language
- XML is a powerful language to describe structure data for web application
- XML is currently applied in many fields
- Many vendors already supports or will support XML

Experiment 3: Publishing XML document using XSLT

STUDENT.XSL

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
```

```
<head>
<title>Student Information</title>
</head>
<body bgcolor="pink" text="blue">
<font size="6" color="red">
<marquee direction=" ">
<u>WELCOME TO STUDENT DATABASE</u>
</marquee>
</font>
<h1 align="center">STUDENT INFORMATION</h1>
<hr/>
<hr/>
<table width="100%" border="2" title="student">
<tr>
<th rowspan="2">Roll no</th>
<th rowspan="2">Name</th>
<th rowspan="2">Marks1</th>
<th rowspan="2">Marks2</th>
<th rowspan="2">Marks3</th>
<th colspan="5">Address</th>
<tr>
<th>house_no</th>
<th>Street</th>
<th>City</th>
<th>Pincode</th>
<th>Phone_no</th>
</tr>
</tr>
<xsl:for-each select="student/student_info">
<tr>
<td><xsl:value-of select="stud_rollno"/></td>
<td><xsl:value-of select="stud_name"/></td>
<td><xsl:value-of select="stud_marks1"/></td>
<td><xsl:value-of select="stud_marks2"/></td>
```

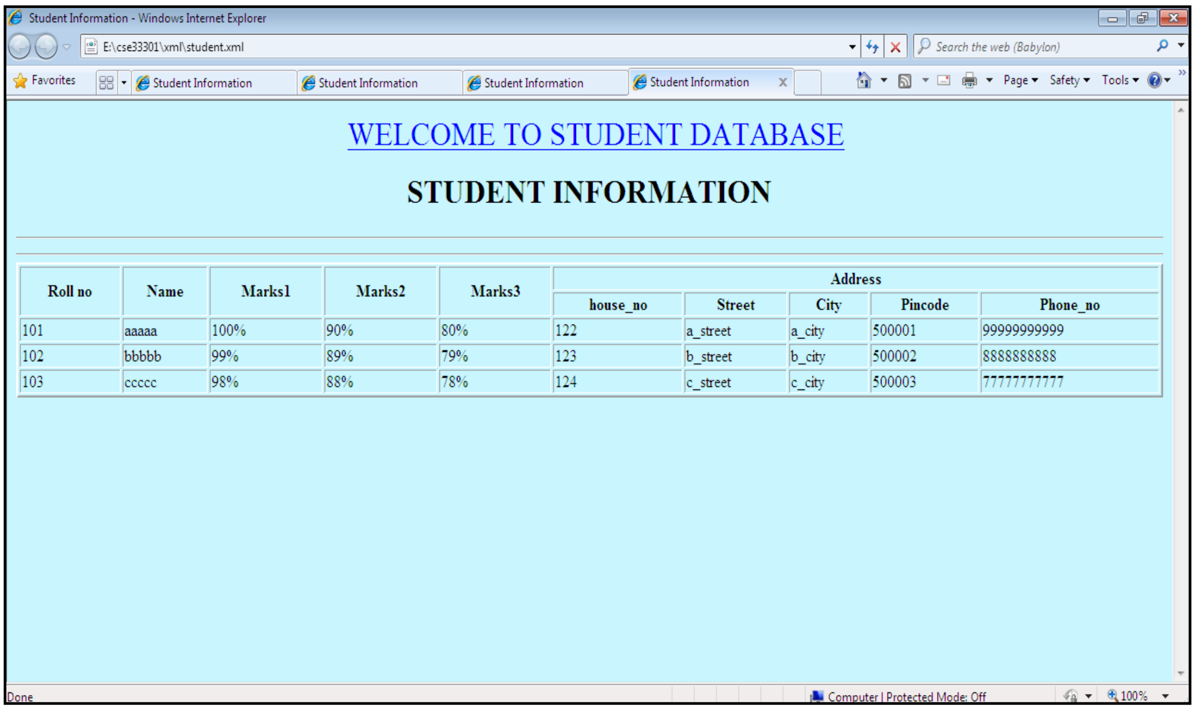
```
<td><xsl:value-of select="stud_marks3"/></td>
<td><xsl:value-of select="stud_addr/house_no"/></td>
<td><xsl:value-of select="stud_addr/street"/></td>
<td><xsl:value-of select="stud_addr/city"/></td>
<td><xsl:value-of select="stud_addr/pin"/></td>
<td><xsl:value-of select="stud_addr/phone_no"/></td>
</tr>

</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

STUDENT.XML

```
<!--Transformation of XML to HTML using XSL-->
<?xml-stylesheet type="text/xsl" href="E:\cse33301\xml\student.xsl"?>
<student>
<student_info>
<stud_rollno>101</stud_rollno>
<stud_name>aaaaa</stud_name>
<stud_marks1>100%</stud_marks1>
<stud_marks2>90%</stud_marks2>
<stud_marks3>80%</stud_marks3>
<stud_addr>
<house_no>122</house_no>
<street>a_street</street>
<city>a_city</city>
<pin>500001</pin>
<phone_no>99999999999</phone_no>
</stud_addr>
```

```
</student_info>

<student_info>
<stud_rollno>102</stud_rollno>
<stud_name>bbbbb</stud_name>
<stud_marks1>99%</stud_marks1>
<stud_marks2>89%</stud_marks2>
<stud_marks3>79%</stud_marks3>
<stud_addr>
<house_no>123</house_no>
<street>b_street</street>
<city>b_city</city>
<pin>500002</pin>
<phone_no>8888888888</phone_no>
</stud_addr>
</student_info>

<student_info>
<stud_rollno>103</stud_rollno>
<stud_name>ccccc</stud_name>
<stud_marks1>98%</stud_marks1>
<stud_marks2>88%</stud_marks2>
<stud_marks3>78%</stud_marks3>
<stud_addr>
<house_no>124</house_no>
<street>c_street</street>
<city>c_city</city>
<pin>500003</pin>
<phone_no>77777777777</phone_no>
</stud_addr>
</student_info>
</student>
```

Output:


Student Information - Windows Internet Explorer

E:\cse33301\xml\student.xml

WELCOME TO STUDENT DATABASE

STUDENT INFORMATION

Roll no	Name	Marks1	Marks2	Marks3	Address				
					house_no	Street	City	Pincode	Phone_no
101	aaaaa	100%	90%	80%	122	a_street	a_city	500001	99999999999
102	bbbbb	99%	89%	79%	123	b_street	b_city	500002	8888888888
103	ccccc	98%	88%	78%	124	c_street	c_city	500003	77777777777

Exercise 4: XML document processing using DOM

DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

Features of DOM Parser

A DOM Parser creates an internal structure in memory which is a DOM document object and the client applications get information of the original XML document by invoking methods on this document object.

DOM Parser has a tree based structure.

Advantages

1) It supports both read and write operations and the API is very simple to use.

2) It is preferred when random access to widely separated parts of a document is required.

Disadvantages

1) It is memory inefficient. (consumes more memory because the whole XML document needs to loaded into memory).

2) It is comparatively slower than other parsers.

Implementation of DOM parser

Create employees.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<employees>
<employee id="111">
<firstName>Smith</firstName>
<lastName>warn</lastName>
<location>India</location>
</employee>
<employee id="222">
<firstName>Alex</firstName>
<lastName>Gussin</lastName>
<location>Russia</location>
</employee>
<employee id="333">
<firstName>David</firstName>
<lastName>Feezor</lastName>
<location>USA</location>
</employee>
</employees>
```

Create DOM program called newDomProgram.java in NetBeans IDE

```
import javax.xml.parsers.*;
import org.w3c.dom.*;
import java.io.*;
public class NewDomProgram {
  public static void main(String[] args) {
try{
     DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
```

```
//Build Document

Document document = builder.parse(new File("C:\\Users\\MIRZA AHMED
BAIG\\Documents\\NetBeansProjects\\newDomProgram\\src\\newdomprogram\\employees.xml"))
;


//Normalize the XML Structure; It's just too important !!

document.getDocumentElement().normalize();


//Here comes the root node

Element root = document.getDocumentElement();

System.out.println(root.getNodeName());


//Get all employees

NodeList nList = document.getElementsByTagName("employee");

System.out.println("============================");


for (int temp = 0; temp < nList.getLength(); temp++)

{

 Node node = nList.item(temp);

 System.out.println("");    //Just a separator

 if (node.getNodeType() == Node.ELEMENT_NODE)

 {

    //Print each employee's detail

    Element eElement = (Element) node;

    System.out.println("Employee id : "    + eElement.getAttribute("id"));

    System.out.println("First Name : "  +
eElement.getElementsByTagName("firstName").item(0).getTextContent());

    System.out.println("Last Name : "   +
eElement.getElementsByTagName("lastName").item(0).getTextContent());
```

```
   System.out.println("Location : "    +
eElement.getElementsByTagName("location").item(0).getTextContent());

 }

}

   }

   catch (Exception e) {

      e.printStackTrace();

    }

   }

}
```

Features of SAX Parser

It does not create any internal structure.

Clients does not know what methods to call, they just overrides the methods of the API and place his own code inside method.

It is an event based parser, it works like an event handler in Java.

Advantages

1) It is simple and memory efficient.

2) It is very fast and works for huge documents.

Disadvantages

1) It is event-based so its API is less intuitive.

2) Clients never know the full information because the data is broken into pieces.

Exercise 5: XML document processing using SAX

Implementing SAX program need to create the following files

1. Create employee.xml file contain data
2. Create plain java class called UserHandler
3. Create SaxProgram in java

```
//UserHandler.java

import org.xml.sax.Attributes;

import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
public class UserHandler  extends DefaultHandler{

   boolean bFirstName = false;
  boolean bLastName = false;
  boolean bNickName = false;
  boolean bMarks = false;

  @Override
  public void startElement(String uri,
  String localName, String qName, Attributes attributes) throws SAXException {

    if (qName.equalsIgnoreCase("student")) {
      String rollNo = attributes.getValue("rollno");
      System.out.println("Roll No : " + rollNo);
    } else if (qName.equalsIgnoreCase("firstname")) {
      bFirstName = true;
    } else if (qName.equalsIgnoreCase("lastname")) {
      bLastName = true;
    } else if (qName.equalsIgnoreCase("nickname")) {
      bNickName = true;
    }
    else if (qName.equalsIgnoreCase("marks")) {
      bMarks = true;
    }
  }
  @Override
  public void endElement(String uri,
  String localName, String qName) throws SAXException {
    if (qName.equalsIgnoreCase("student")) {
      System.out.println("End Element :" + qName);
    }
  }

  @Override
  public void characters(char ch[], int start, int length) throws SAXException {

    if (bFirstName) {
      System.out.println("First Name: "
        + new String(ch, start, length));
      bFirstName = false;
    } else if (bLastName) {
      System.out.println("Last Name: " + new String(ch, start, length));
      bLastName = false;
    } else if (bNickName) {
      System.out.println("Nick Name: " + new String(ch, start, length));
      bNickName = false;
    } else if (bMarks) {
      System.out.println("Marks: " + new String(ch, start, length));
      bMarks = false;
    }
  }
}
```

```
//SaxProgram.java

import java.io.File;

import javax.xml.parsers.SAXParser;

import javax.xml.parsers.SAXParserFactory;


import org.xml.sax.Attributes;

import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

public class SaxProgram {

   public static void main(String[] args) {

      // TODO code application logic here

      try {

       File inputFile = new File("C:\\Users\\MIRZA AHMED
BAIG\\Documents\\NetBeansProjects\\saxProgram\\src\\saxprogram\\input.txt");

       SAXParserFactory factory = SAXParserFactory.newInstance();

       SAXParser saxParser = factory.newSAXParser();

       UserHandler userhandler = new UserHandler();

       saxParser.parse(inputFile, userhandler);

     } catch (Exception e) {

       e.printStackTrace();

     }

   }


}
```

EXERCISE 6: Write a program to encrypt the given number to display the encrypted data using java script.

Python Introduction

Python is a general purpose, dynamic, high level and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is *easy to learn* yet powerful and versatile scripting language which makes it attractive for Application Development.

Python's syntax and *dynamic typing* with its interpreted nature, makes it an ideal language for scripting and rapid application development.

Python supports *multiple programming pattern*, including object oriented, imperative and functional or procedural programming styles.

Python is not intended to work on special area such as web programming. That is why it is known as *multipurpose* because it can be used with web, enterprise, 3D CAD etc.

We don't need to use data types to declare variable because it is *dynamically typed* so we can write a=10 to assign an integer value in an integer variable.

Python makes the development and debugging *fast* because there is no compilation step included in python development and edit-test-debug cycle is very fast.

Python Features

Python provides lots of features that are listed below.

1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

5) Free and Open Source

Python language is freely available at offical web address.The source-code is also available. Therefore it is open source.

6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

8) Large Standard Library

Python has a large and broad library and prvides rich set of module and functions for rapid application development.

9) GUI Programming Support

Graphical user interfaces can be developed using Python.

10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

Exercise 7: Write a python program which generates an output file based on the one-line instructions in an input file

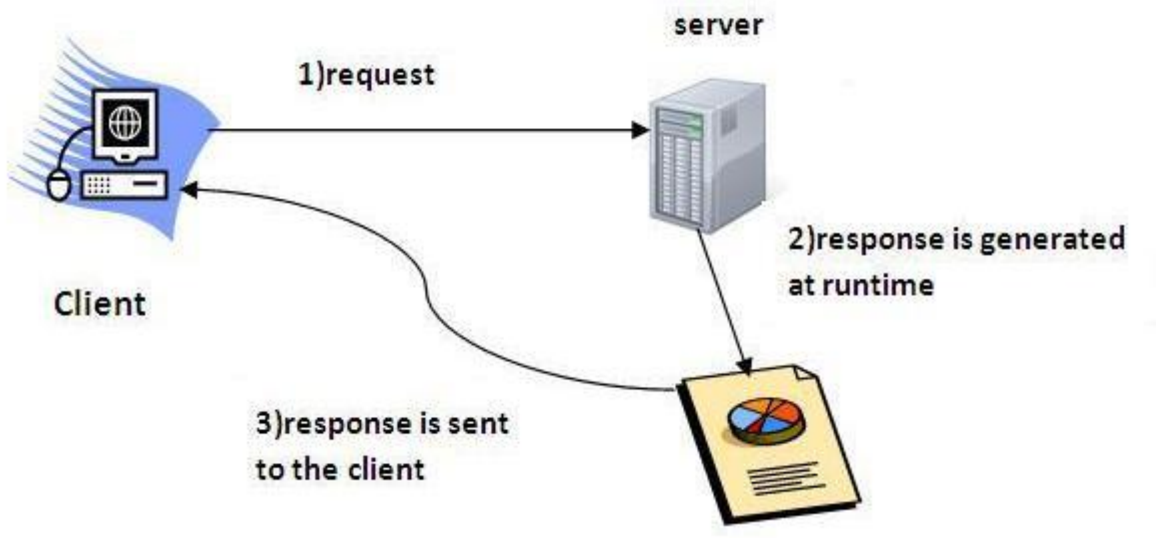
SERVLETS

Introduction

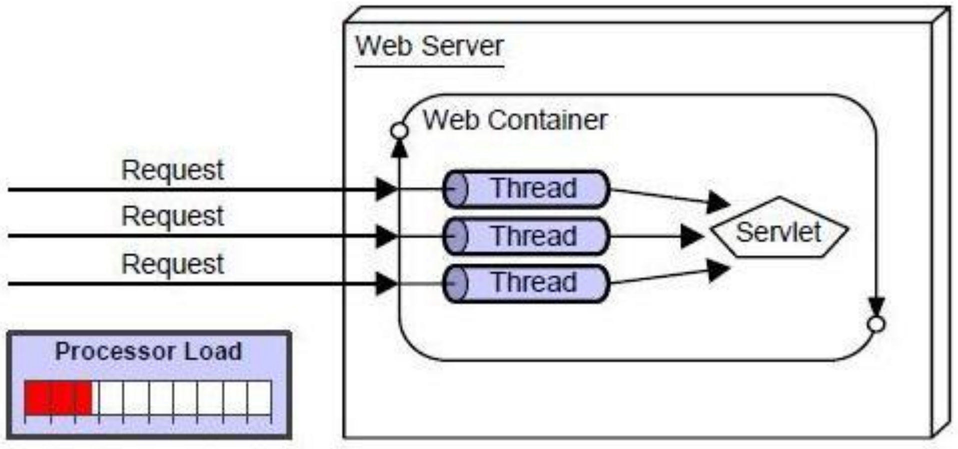
What is a Servlet?

Servlet can be described in many ways, depending on the context.

- Servlet is a technology i.e. used to create web application.
- Servlet is an API that provides many interfaces and classes including documentations.
- Servlet is an interface that must be implemented for creating any servlet.
- Servlet is a class that extend the capabilities of the servers and respond to the incoming request. It can respond to any type of requests.
- Servlet is a web component that is deployed on the server to create dynamic web page.


server
1)request
Client
2)response is generated at runtime
3)response is sent to the client

Advantage of Servlet



There are many advantages of servlet over CGI. The web container creates threads for handling the multiple requests to the servlet. Threads have a lot of benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low. The basic benefits of servlet are as follows:

1. **better performance:** because it creates a thread for each request not process.
2. **Portability:** because it uses java language.
3. **Robust:** Servlets are managed by JVM so we don't need to worry about memory leak, garbage collection etc.
4. **Secure:** because it uses java language..

Web Terminology

Servlet Terminology	**Description**
Website: static vs dynamic	It is a collection of related web pages that may contain text, images, audio and video.
HTTP	It is the data communication protocol used to establish communication between client and server.
HTTP Requests	It is the request send by the computer to a web server that contains all sorts of potentially interesting information.
Get vs Post	It give the difference between GET and POST request.
Container	It is used in java for dynamically generate the web pages on the server side.
Server: Web vs Application	It is used to manage the network resources and for running the program or software that provides services.
Content Type	It is HTTP header that provides the description about what are you sending to the browser.

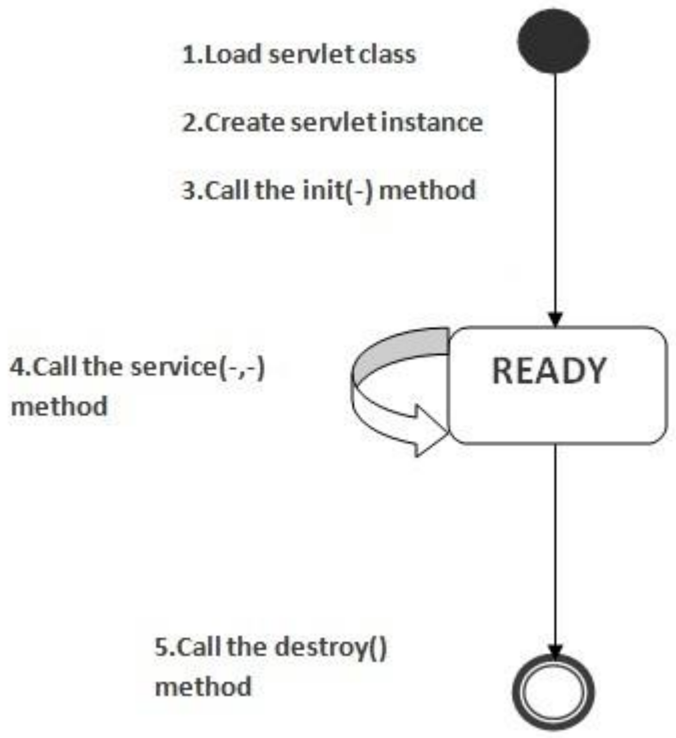
Static vs Dynamic website

Static Website	**Dynamic Website**

Prebuilt content is same every time the page is loaded.	Content is generated quickly and changes regularly.
It uses the **HTML** code for developing a website.	It uses the server side languages such as **PHP,SERVLET, JSP, and ASP.NET** etc. for developing a website.
It sends exactly the same response for every request.	It may generate different HTML for each of the request.
The content is only changes when someone publishes and updates the file (sends it to the web server).	The page contains "server-side" code it allows the server to generate the unique content when the page is loaded.
Flexibility is the main advantage of static website.	Content Management System (CMS) is the main advantage of dynamic website.

Life Cycle of a Servlet (Servlet Life Cycle)

1. Life Cycle of a Servlet
1. Servlet class is loaded
2. Servlet instance is created
3. init method is invoked
4. service method is invoked
5. destroy method is invoked


1.Load servlet class
2.Create servlet instance
3.Call the init(-) method
4.Call the service(-,-) method
READY
5.Call the destroy() method

As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

USES OF SERVLETS:

1. Servlets process and store the data submitted by an HTML form.
2. Servlets are useful for providing dynamic contents.
3. Servlets are used in cookies and in session tracking.

Cookies are small

programs that make use of information submitted on currently accessed web pages .Similarly, session tracking keeps track of all previously accessed web-pages.

DEPLOYMENT DESCRIPTOR:

It helps to manage the configuration of web applications once they are developed.

For web-containers,the deployment descriptor is an XML file called web.xml stored in the/WEB-INF directory for the web-application.

Experiment 7: Develop a simple java Servlet application

i) Creating index.jsp for (view page)

ii) Creating Servlet for Controller(Having Application Logic)

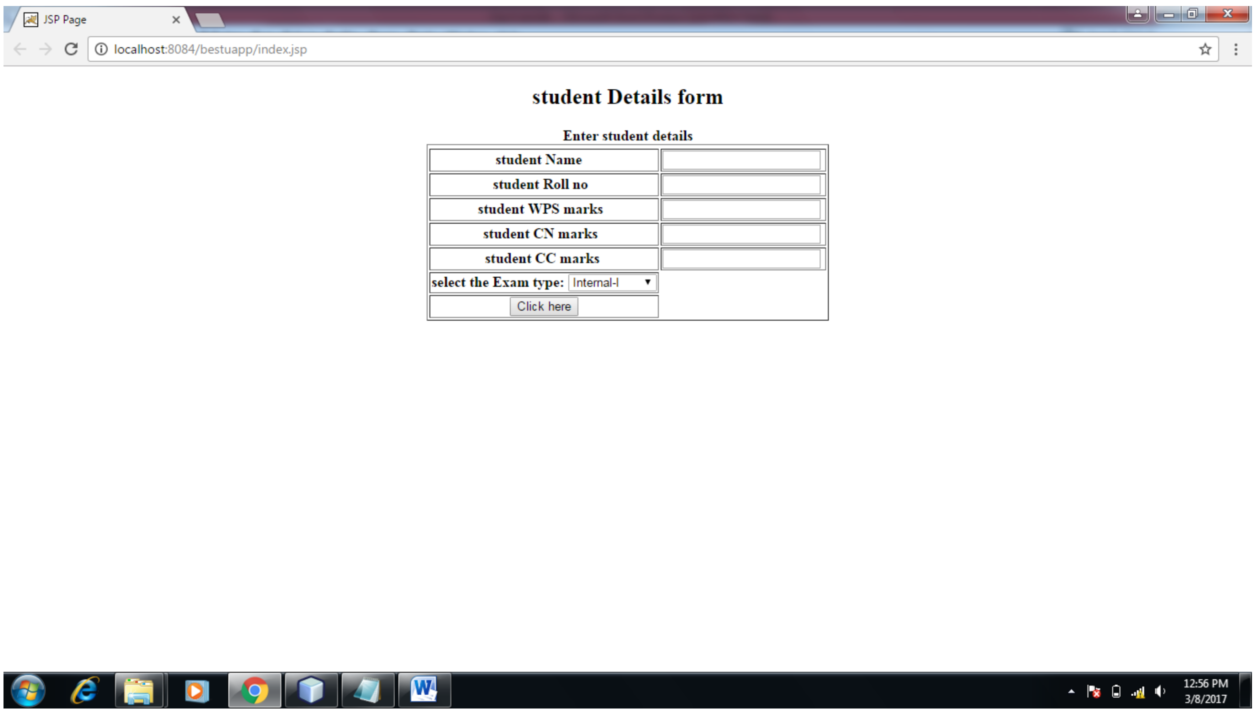
Fistly,

File name as Index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

</head>

<body >

<h2 align="center"> student Details form</h2>

<table align="center" border="1">

<caption><b>Enter student details</b></caption>

<form action="stu_marks" metho="get">


<tr>

<tr><th>student Name &nbsp;&nbsp;</th>

<td><input type="text" name="t1"</td><br/></tr>

<tr><th >student Roll no &nbsp;</th>

<td><input type="text" name="t2"</td><br/></tr>

<tr><th >student WPS marks &nbsp;</th>

<td><input type="text" name="t3"</td><br/></tr>

<tr><th >student CN marks &nbsp;</th><td><input type="text" name="t4" </td><br/></tr>

<tr><th >student CC marks&nbsp;</th><td><input type="text" name="t5"</td><br/></tr>

<tr><th>select the Exam type:

<select name="examselectopt" size="1">
```

```
<option value="BE 3by2- Sem int1">Internal-I </option>

<option value="BE 3by2- Sem int2">Internal-II </option>

<option value="BE Main Exams">Main Exams </option>

</select><br></th></tr>

<tr><th><input type="submit" value="Click here"/><br></th>

</tr>

</tr>

</table>

</form>

</body>

</html>
```

Output of index.jsp



student Details form

Enter student details

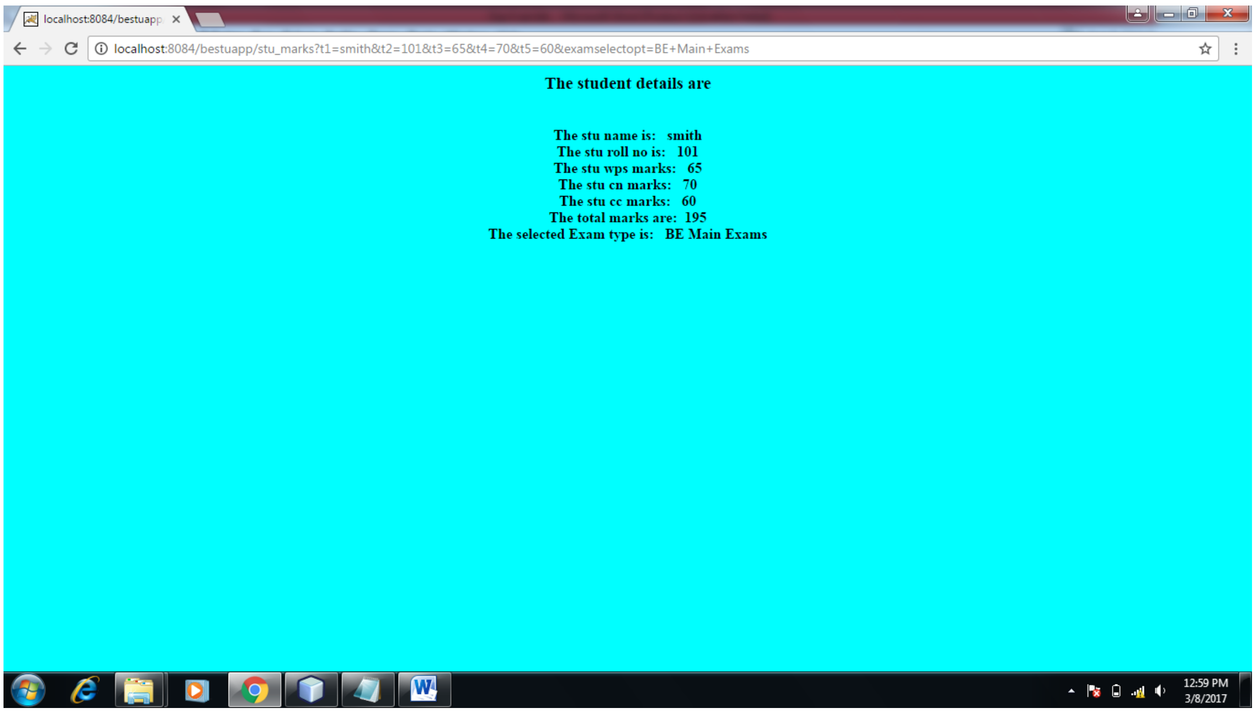
student Name	
student Roll no	
student WPS marks	
student CN marks	
student CC marks	
select the Exam type: Internal-I	
Click here	

File name as stu_marks.java

```
import java.io.IOException;

import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.math.BigInteger;

public class stu_marks extends HttpServlet {
  @Override
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    PrintWriter out=response.getWriter();
    response.setContentType("text/html");
    String name=request.getParameter("t1");
    String rno=request.getParameter("t2");
    String wpss=request.getParameter("t3");
    String cns=request.getParameter("t4");
    String ccs=request.getParameter("t5");
    String exam=request.getParameter("examselectopt");
    long r=Integer.parseInt(rno);
    int wps=Integer.parseInt(wpss);
    int cn=Integer.parseInt(cns);
    int cc=Integer.parseInt(ccs);
    int tot;
    out.println("<html>");
    out.println("<body align=center style=background:aqua>");

    out.println("<h3> The student details are </h3>" +"<br>");
```

```
        out.println("<b> The stu name is: &nbsp;&nbsp;" +name +"<br>");
        out.println("<b> The stu roll no is: &nbsp;&nbsp;" +r +"<br>");
        out.println("<b> The stu wps marks: &nbsp;&nbsp;" +wps +"<br>");
        out.println("<b> The stu cn marks: &nbsp;&nbsp;" +cn +"<br>");
        out.println("<b> The stu cc marks: &nbsp;&nbsp;" +cc +"<br>");
        tot=wps+cn+cc;
        out.println("<b>The total marks are: &nbsp;"+tot +"<br>");
        out.println("<b>The selected Exam type is: &nbsp;&nbsp;" +exam);
    }
}
```

Output of stu_marks.java



localhost:8084/bestuapp/stu_marks?t1=smith&t2=101&t3=65&t4=70&t5=60&examselectopt=BE+Main+Exams

```
The student details are

The stu name is:   smith
The stu roll no is:   101
The stu wps marks:   65
The stu cn marks:   70
The stu cc marks:   60
The total marks are:  195
The selected Exam type is:   BE Main Exams
```

Experiment 8: Providing Datastore support for website using JDBC

1. Open -> Start -> Programs -> Oracle database 10g/ 11g

2. Enter the user name as **scott**password as "tiger" host as “be” and type the following code

```
create table student(name varchar2(20),mobileno number(20));
insert into student values('smith',1234);
select * from student;
```

Create The following program

Firstly index.jsp

Secondly done.jsp

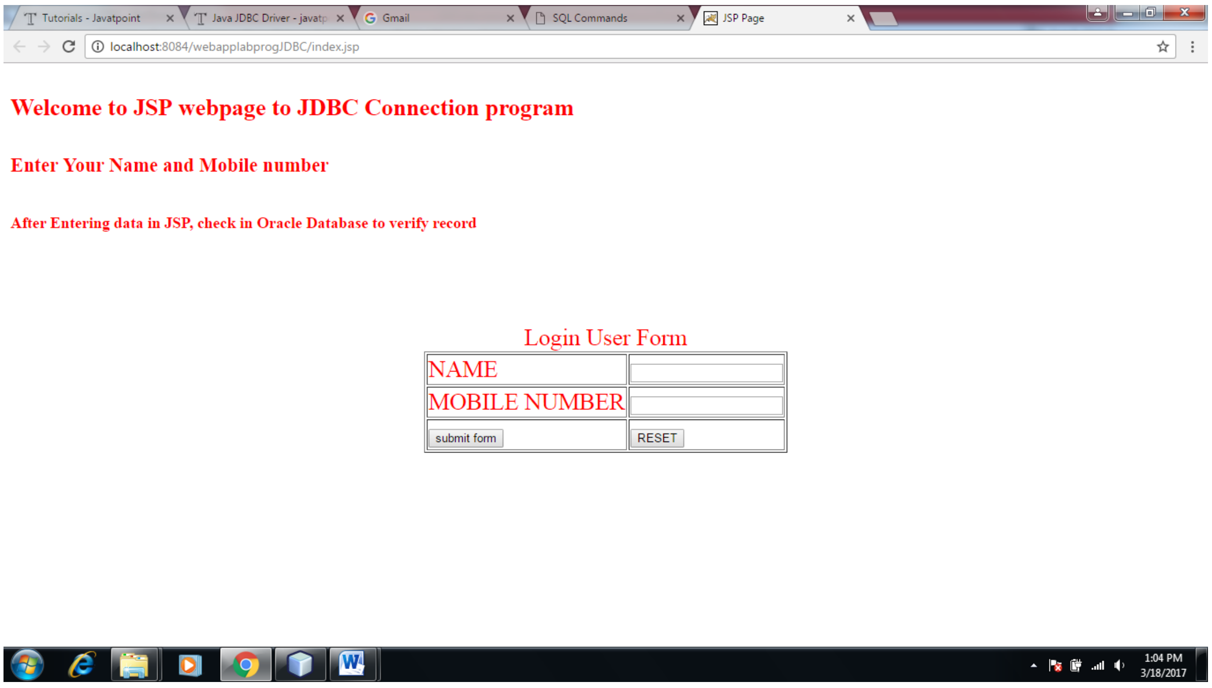
Connect jdbc drivers using JAR file(ojdbc14.jar) added to Library folder of Webapplication

File name as index.jsp

```
<%--
    Document   : index
    Created on : Mar 18, 2017, 12:16:38 PM
    Author     : musthafa
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<h4> Welcome to JSP webpage to JDBC Connection program</h4>
<h5> Enter Your Name and Mobile number </h5>
<h6> After Entering data in JSP, check in Oracle Database to verify record</h6>
<form action="done.jsp">
<br/>
<br/>
<table border="1" align="center"  >
<caption> Login User Form</caption>
<style>
            body{
                position: relative;
                color:red;
                font-size: 20pt;

            }
```

```
</style>
<tbody>
<tr>
<td>NAME</td>
<td><input type="text" name="nm" value=""/></td>
</tr>

<tr>
<td>MOBILE  NUMBER</td>
<td><input type="text" name="mb" value=""/></td>
</tr>
<tr>
<td><input type="submit" value="submit form"</td>
<td><input type="reset" value="RESET"/></td>
</tr>
</tbody>
</body>
</html>
```

Output of index.jsp


Tutorials - Javatpoint
Java JDBC Driver - javatp
Gmail
SQL Commands
JSP Page
localhost:8084/webapplabprogJDBC/index.jsp
1:04 PM
3/18/2017

Welcome to JSP webpage to JDBC Connection program

Enter Your Name and Mobile number

After Entering data in JSP, check in Oracle Database to verify record

Login User Form

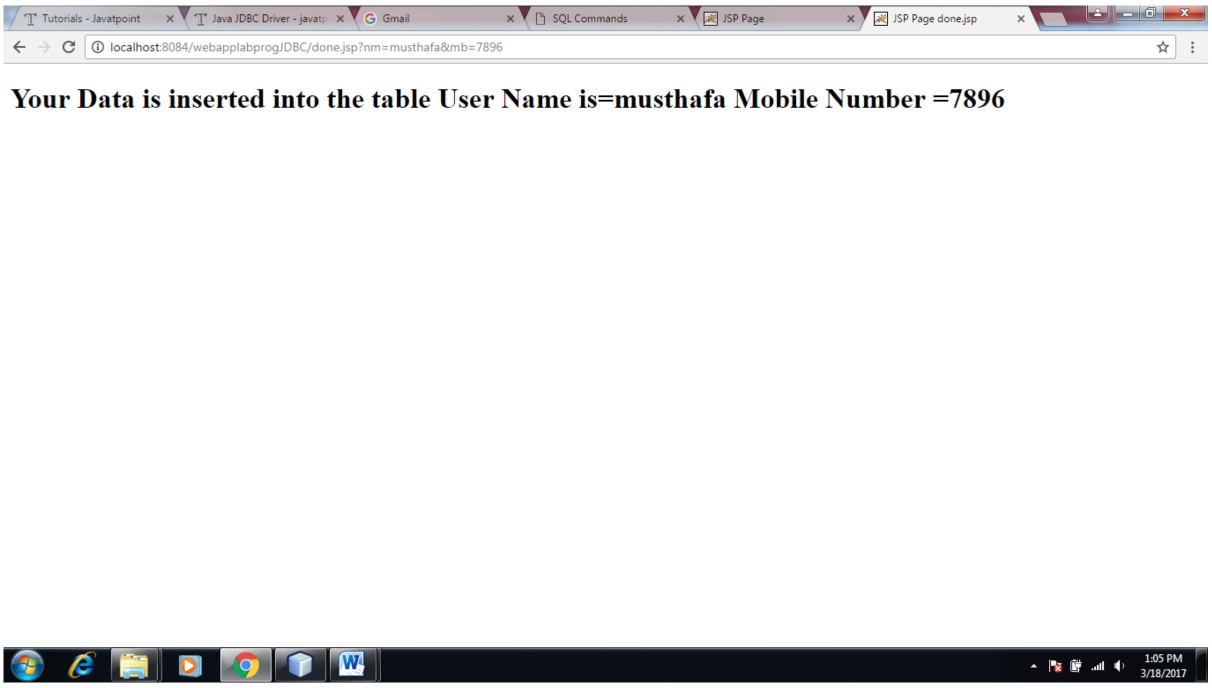
NAME	
MOBILE NUMBER	
submit form	RESET

File name as done.jsp

```
<%--
    Document   : done
    Created on : Mar 18, 2017, 12:31:12 PM
    Author     : musthafa
--%>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page done.jsp</title>
</head>
<body>

<%@page import="java.sql.*;"%>
<%
      String s1=request.getParameter("nm");
      String s2=request.getParameter("mb");



      try{
          Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","del
l");
Statement st=con.createStatement();
int i=st.executeUpdate("insert into student values('"+s1+"','"+s2+"')");
if (i>0)
      {
   System.out.println("DONE!");
   %>
<h1>Your Data is inserted into the table
 User Name is=<%=s1%>
 Mobile Number =<%=s2%></h1>
<%}
      }
      catch(Exception e)
      {
          System.out.println(e);
      }
%>
</body>
</html>
```

Output of done.jsp


Tutorials - Javatpoint
Java JDBC Driver - javatp
Gmail
SQL Commands
JSP Page
JSP Page done.jsp
localhost:8084/webapplabprogJDBC/done.jsp?nm=musthafa&mb=7896
1:05 PM
3/18/2017

Your Data is inserted into the table User Name is=musthafa Mobile Number =7896