University of Global Village (UGV), Barishal

Content of the Sessional Course
University Student (UGV) Format

Program: Bachelor of Science in Computer Science Engineering (CSE)

Course Code	--
Name of Course Title	Web application project development (Level-4)
Course Type	Skill Course
Level	4th Semester
Academic Session	Winter 2025
Name(s) of AcademicCourse teacher(s)	Sohag Mollik, Lecturer, CSE. Mobile: 01304142395 E-mail: sohag.cse.just@gmail.com
Consultation Hours:	

Web Page design & development Lab Student (UGV) Format	
Course Code: --	Credits: --
Exam Hours: --	CIE Marks: 30
Course for 4th Semester, Bachelor of Science in Computer Science Engineering (CSE)	SEE Marks: 20

1. Course Learning Outcome (CLO) at the end of the course, the students will be able to-

CLO 1: Describe the core concepts and principles of web development.

CLO 2: Understand and apply JavaScript fundamentals.

CLO 3: Analyze and design web applications.

CLO 4: Create and develop dynamic and interactive web applications.

CLO 5: Handle and manipulate data.

CLO 6: Optimize web applications & evaluate and adapt to emerging trends and technologies.

2.Topics to be covered

Week	Topics	Teaching Learning Strategy(s)	Class Hour	Practice Hour	Assessment Strategy(s)	Mapping with CLO
01	Describe the core concepts and principles of web development, including client-server architecture, HTTP/HTTPS protocols, and web standards.	Lecture, Live demonstration & Hands-on exercises.	5h	4h	Participation, Lab Performance	CLO 1
02	JavaScript Fundamentals: Syntax, variables, data types, and control structures.	Lecture, Code-along & Hands-on exercises.	5h	4h	Short quiz, Participation	CLO 2
03	JavaScript Functions: Function declarations, expressions, and arrow functions.	Lecture, Practice coding & Debugging tasks.	5h	4h	Lab exercises, Code review	CLO 2
04	JavaScript Objects and Arrays: Creation, properties, methods, and manipulation.	Lecture, Interactive coding & Problem-solving	5h	4h	Lab exercises, Quiz	CLO 2
05	Introduction to HTML: Structure of web pages, tags, and attributes.	Lecture, Code-along & Hands-on tasks	5h	5h	Lab exercises, Participation	CLO 1 & CLO 3
06	Styling with CSS: Selectors, properties, responsive design concepts.	Lecture, Live demo, Hands-on exercises.	5h	4h	Lab exercises, Quiz	CLO 1 & CLO 3

07	DOM Manipulation: Selecting and updating elements, event handling.	Live demonstration, Hands-on DOM projects.	5h	5h	Lab assignments, Class discussion	CLO 3
08	Real-world Application: Integrating front-end (HTML, CSS, JS) and back-end concepts	Group coding project, Mentoring & Code review.	5h	4h	Lab performance, Group evaluation	CLO 3
09	Advanced JavaScript Concepts: Destructuring, scope, hoisting, and `this` keyword.	Lecture, Debugging tasks, Coding practice.	5h	4h	Debugging tasks, Participation	CLO 4
10	JSON and APIs: Parsing, fetching, and handling data from external sources.	Lecture, Code-along, Hands-on tasks.	5h	4h	Lab performance, Short quiz	CLO 4 & CLO 5
11	Strings and Numbers: Methods, templates, and manipulations in JavaScript.	Lecture, Code-along, Hands-on exercises.	5h	5h	Lab tasks, Quiz	CLO 5
12	JavaScript Dates and Math Operations: Date formats, get/set methods, and math functions.	Lecture, Code demonstrations & Exercises.	5h	3h	Lab performance, Participation	CLO 5
13	Optimizing Web Applications: Best practices in performance, usability, and accessibility.	Lecture, Group discussions & Refactoring tasks.	5h	4h	Participation, Code review	CLO 6
14	Error Handling: Try-catch, debugging tools, and resolving common errors & Database Integration with MySQL: Understanding Database Connection	Practical debugging sessions & Hands-on tasks.	5h	5h	Debugging exercises, Quiz & Participation.	CLO 6
15	Project Planning: Design, structure, and functional requirements of a web application	Discussion, Group mentoring, Hands-on practice.	5h	5h	Project proposal, Peer feedback	CLO 1 – CLO 6
16	Final Project Development: Building a full dynamic and interactive web application.	Project work, Instructor feedback & Guidance.	5h	5h	Progress evaluation, Participation.	CLO 1 - CLO 6
17	Final Project Deployment and Presentation: Testing, debugging, and showcasing web applications.	Live demonstrations, Project presentation.	5h	2h	Project demonstration, Evaluation	CLO 1 - CLO 6

3.Teaching-Learning Strategy:

- **Lecture:** Explain concepts with real-world examples and visual aids.
- **Live Demonstration:** Show step-by-step coding and debugging in real-time.
- **Interactive Coding Examples:** Engage students with challenges during class.
- **Hands-on Exercises:** Provide structured practice aligned with topics.
- **Code-along Sessions:** Guide students through practical coding implementations.
- **Problem-solving Tasks:** Assign real-world challenges to apply concepts.
- **Group Discussions:** Facilitate discussions on best practices and peer reviews.
- **Mini-projects:** Assign small, focused projects incorporating multiple concepts.
- **Debugging Sessions:** Teach error identification and resolution with tools.
- **Final Project Work:** Mentor students in developing a complete web application.

4.Assessment Strategy:

- **Lab Performance:** 30% (Lab participation, hands-on exercises, and weekly assessments)
- **Quizzes and Short Tests:** 20% (Regular quizzes on theoretical concepts)
- **Assignments and Reports:** 20% (Assignments related to data management, cloud integration, and security)
- **Project Evaluation:** 30% (Progress, final project implementation, and presentation)

5.Instructional Materials and References: Textbooks:

1. "Eloquent JavaScript" by Marijn Haverbeke
2. "JavaScript: The Definitive Guide" by David Flanagan
3. "HTML and CSS: Design and Build Websites" by Jon Duckett
4. "Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics" by Jennifer Niederst Robbins

Additional References:

Follow w3school [https://www.w3schools.com/] & others website.

An Overview of HTML, CSS & JavaScript

WEEK 01-04

Page
9-30

HYPER TEXT MARKUP LANGUAGE

WHAT IS HTML?

- HTML STANDS FOR **H**YPER **T**EXT **M**ARKUP **L**ANGUAGE
- HTML IS A **MARKUP** LANGUAGE
- A MARKUP LANGUAGE IS A SET OF MARKUP **TAGS**
- THE TAGS **DESCRIBE** DOCUMENT CONTENT
- HTML DOCUMENTS CONTAIN HTML **TAGS** AND PLAIN **TEXT**
- HTML DOCUMENTS ARE ALSO CALLED **WEB PAGES**

HTML TAGS

- Html tags are keywords (tag names) surrounded by **ANGLE BRACKETS** like <html>
- Html tags normally **come in pairs** like and
- The first tag in a pair is the **START TAG,** the second tag is the **END TAG**
- The end tag is written like the start tag, with a **SLASH** before the tag name
- START and END tags are also called **OPENING TAGS** and **CLOSING TAGS**

EXAMPLE:

<tagname>content</tagname>

HTML EXAMPLE

```
<html>
<body>
<h1>my first heading</h1>
<p>my first paragraph.</p>
</body>
</html>
```

EXAMPLE EXPLAINED

- The text between <html> and </html> DESCRIBES THE WEB PAGE
- The text between <body> and </body> is the VISIBLE PAGE CONTENT
- The text between <h1> and </h1> is DISPLAYED AS A HEADING
- The text between and is DISPLAYED AS A PARAGRAPH

HTML EDITORS

Write html using notepad or textedit

- Html can be edited by using a professional html editor like:
  - Adobe dreamweaver
  - Microsoft expression web
  - Coffeecup html editor
- However, for learning html we recommend a text editor like notepad (pc) or textedit (mac).
- When saving an html file, use either the .htm or the .html file extension. there is no difference, it is entirely up to you.

HTML TAG

Html headings are defined with the **<h1>** to **<h6>** tags

```
<h1>this is a heading</h1>
```

Html paragraphs are defined with the **<p>** tag.

```
<p>this is a paragraph.</p>
```

Html links are defined with the **<a>** tag.

```
<a href="http://espesolutions.com">this is a link</a>
```

Html images are defined with the **<img>** tag.

```
<img src="espelogo.jpg" alt="espesolutions.com" width="105" height="105">
```

Html uses tags like **<b>** and ***<i>*** for formatting output, like **bold** or ***italic*** text.

```
<b>this text is bold</b>,<i>this text is italic</i>
```

HTML ATTRIBUTES

- Html elements can have **ATTRIBUTES**
- Attributes provide **ADDITIONAL INFORMATION** about an element
- Attributes are always specified in **THE START TAG**
- Attributes come in name/value pairs like: **name="value“**

<IMG SRC=“ESPELOGO.JPG” WIDTH=“105” HEIGHT=“105”>

NAME **VALUE** **NAME** **VALUE** **NAME** **VALUE**

HTML TABLES

- Tables are defined with the **<table>** tag.
- Tables are divided into **table rows** with the **<tr>** tag.
- Table rows are divided into **table data** with the **<td>** tag.
- A table row can also be divided into **table headings** with the **<th>** tag.

Example

```
<table>
<tr>
<th>name</th><th>qualification</th>
</tr>
<tr>
<td>sandeep</td><td>cse</td>
</tr>
</table>
```

HTML LIST

HTML CAN HAVE UNORDERED LISTS &ORDERED LISTS

- **UNORDERED HTML LIST**
  - The first item
  - The second item
  - The third item
  - The fourth item
- **ORDERED HTML LIST**
  1. The first item
  2. The second item
  3. The third item
  4. The fourth item

EXAMPLE

UNORDERD LIST:

```
<ul>
    <li>java</li>
    <li>c</li>
    <li>c++</li>
</ul>
```

ORDERD LIST:

```
<ol>
    <li>java</li>
    <li>c</li>
    <li>c++</li>
</ol>
```

HTML FORMS

- Html forms are used to select different kinds of user input.
- Html forms are used to pass data to a server.
- An html form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. a form can also contain select lists, textarea, fieldset, legend, and label elements.
- **SYNTAX:**

```
<form>
    input elements
</form>
```

INPUT ELEMENT

- The most important form element is the <input> element.
- The <input> element is used to select user information.
- An <input> element can vary in many ways, depending on the type attribute.
- An <input> element can be of type text field, checkbox, password, radio button, submit button, and more.

TEXT FIELDS

DEFINES ONE LINE INPUT FIELD WHERE USER CAN ENTER TEXT.

EXAMPLE:

```
<form>
FIRST NAME: <input type="text“ name="firstname"><br>
LAST NAME: <input type="text" name="lastname">
</form>
```

OUTPUT:

FIRST NAME:

LAST NAME:

PASSWORD FIELD

- PASSWORD defines a password field.

  <input type="password">

- the text entered in the textfield will view as *******.

Syntax:

password:<input type ="password" name=" password">

OUTPUT:

PASSWORD: ********

RADIO BUTTONS

- Radio buttons let a user select only one of a limited number of choices.

<input type="radio“>

SYNTAX:

```
<form>
  <input type="radio" name=“gender"
  value="male">male<br>
  <input type="radio" name=“gender"
  value="female">female
  </form>
```

OUTPUT:

- ○ Male
- ○ Female

CHECKBOXES

- Checkboxes let a user select zero or more options of a limited number of choices.

<input type="checkbox“>

SYNTAX:

```
<form>
<input type="checkbox" name="vehicle" value="bike">i have a bike<br>
<input type="checkbox" name="vehicle" value="car">i have a car
</form>
```

OUTPUT:

- I HAVE A BIKE
- I HAVE A CAR

SUBMIT

A submit button is used to send form data to a server.

- The data is sent to the page specified in the form's action attribute. t
  - The file defined in the action attribute usually does something with the received input.

```
<input type="submit“>
```

- **TYPE:** SUBMIT.
- **NAME:** Value used by the cgi (common gateway interface)script for processing.
- **VALUE:** Determines the text label on the button, usually submit query.
- **CGI**: External program use standard input and output for data exchange.

SUBMIT

SYNTAX:

```
<form name="input" action="demo" method="get">
  username: <input type="text" name="user">
  password:<input type="password" name="pass">
  <input type="submit" value="submit" >
  </form>
```

OUTPUT:

RESET

- It allows the surfer to clear all the input in the form.
- For reset give <input type="reset">
- The browser display reset button.

DROP-DOWN LIST

Let a user select one or more choices from limited number of options.

SYNTAX:

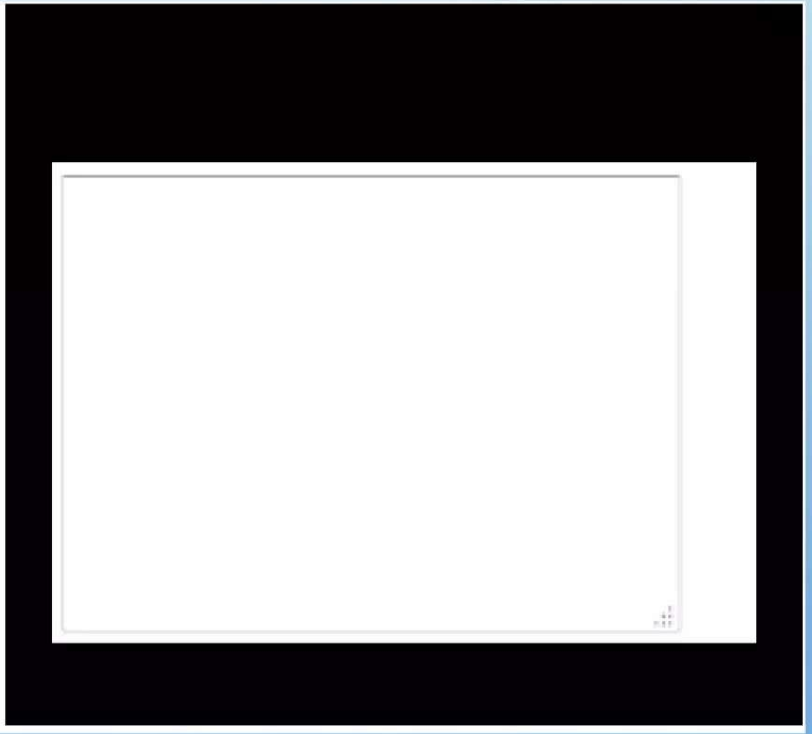
```
<html>
<body>
<select>
<option value="fiat">fiat</option>
<option value="audi">audi</option>
</select>
</body>
</html>
```

TEXTAREA

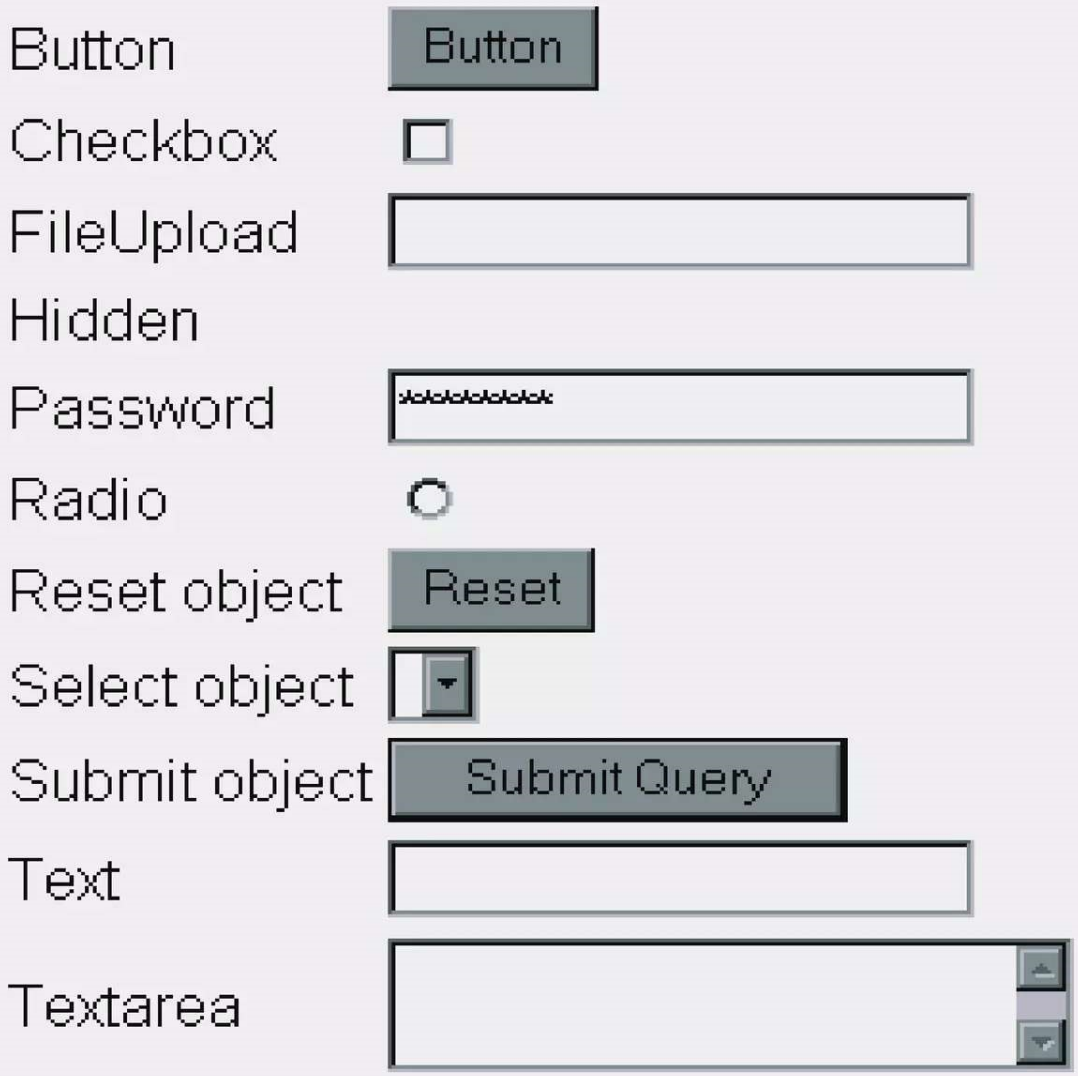
- The <textarea> tag defines a multi-line text input control.
- The size of a text area can be specified by the cols and rows attributes, or even better; through css' height and width properties.

Syntax:

```
<html>
< body>
<textarea rows="10" cols="30">
</textarea>
</body>
```



There are eleven different types of form elements:



Week 05-08

Page
32-45

CASCADING STYLE SHEETS (CSS)

WHAT IS CSS?

- **Css** stands for **c**ascading **s**tyle **s**heets
- Styles define **how to display** html elements
- Styles were added to html 4.0 **to solve a problem**
- **External style sheets** can save a lot of work
- External style sheets are stored in **css files**

CSS SYNTAX

A CSS rule set consists of a selector and a declaration block:

CSS EXAMPLE

➢A css declaration always ends with a semicolon, and declaration groups are surrounded by curly braces:

```
p {
    color: red;
    text-align: center;
  }
```

CSS SELECTORS

➢Css selectors are used to "find" (or select) html elements based on their id, classes, types, attributes, values of attributes and much more.

- ➢element selector
- ➢id selector
- ➢class selector

THE ELEMENT SELECTOR

- The element selector selects elements based on the element name.

```
p {
    text-align: center;
    color: red;
}
```

THE ID SELECTOR

- The id selector uses the id attribute of an html tag to find the specific element.
- An id should be unique within a page, so you should use the id selector when you want to find a single, unique element.

hi

#para1

{

text-align: center;

color: red;

}

THE CLASS SELECTOR

- The class selector finds elements with the specific class.
- The class selector uses the html class attribute.
- Html elements with class="center"

.center{

text-align : center;

color: red;

}

THREE WAYS TO INSERT CSS

- There are three ways of inserting a style sheet:
  - External style sheet
  - Internal style sheet
  - Inline style

EXTERNAL STYLE SHEET

- An external style sheet is ideal when the style is applied to many pages. with an external style sheet, you can change the look of an entire web site by changing just one file.

```
<head>
<link rel="stylesheet" type="text/css“ href="mystyle.css">
</head>
```

INTERNAL STYLE SHEET

- An internal style sheet should be used when a single document has a unique style. you define internal styles in the head section of an html page, inside the <style> tag, like this:

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

INLINE STYLES

- An inline style loses many of the advantages of a style sheet (by mixing content with presentation). use this method sparingly!
- To use inline styles, add the style attribute to the relevant tag. the style attribute can contain any css property.

EXAMPLE:

```
<h1 style="color:blue;margin-left:30px;">this is aheading.</h1>
```

STYLING LINKS

- Links can be styled with any css property (e.g. color, font-family, background, etc.).
- The four links states are:
  - A:LINK - A normal, unvisited link
  - A:VISITED - A link the user has visited
  - A:HOVER - A link when the user mouses over it
  - A:ACTIVE - A link the moment it is clicked

EXAMPLE:

```
/* UNVISITED LINK */
a:link {
    color: #ff0000;
}

/* VISITED LINK */
a:visited {
    color: #00ff00;
}

/* MOUSE OVER LINK */
a:hover {
    color: #ff00ff;
}

/* SELECTED LINK */
a:active {
    color: #0000ff;
}
```

LIST

In html, there are two types of lists:

- Unordered lists - the list items are marked with bullets
- Ordered lists - the list items are marked with numbers or letters

```
ul {
    list-style-image: url('sqpurple.gif');
}
```

```
ul {
    list-style-type: circle;
}
```

```
ol{
    list-style-type: upper-roman;
}
```

TABLE BORDERS

- To specify table borders in css, use the border property.

```
table,th,td
{
border : 1px solid black;
}
```

COLLAPSE BORDERS

The border-collapse property sets whether the table borders are collapsed into a single border or separated:

```
table{border-collapse: collapse;}
table,th,td
{
border : 1px solid black;
}
```

TABLE WIDTH, HEIGHT, TEXT ALIGNMENT AND PADDING

Width and height of a table is defined by the width and height properties.

```
table{
width: 100%;
}
th{
height: 50px;
}
td{
text-align: right;
padding: 15px;
}
```

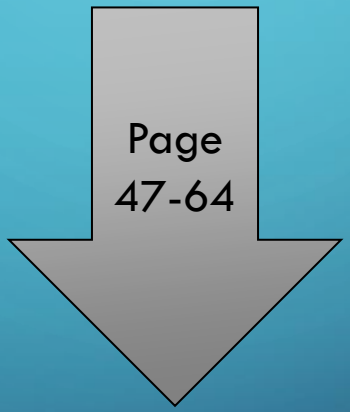
THE CSS BOX MODEL

- All html elements can be considered as boxes. in css, the term "box model" is used when talking about design and layout.
- The image below illustrates the box model:

Explanation of the different parts:

Content - The content of the box, where text and images appear
Padding - Clears an area around the content. The padding is transparent
Border - A border that goes around the padding and content
Margin - Clears an area outside the border. The margin is transparent

WEEK 09-12



JAVA SCRIPT

Client-side programming with JavaScript

- Scripts vs. programs
  - JavaScript vs. JScript vs. VBScript
  - Common tasks for client-side scripts

- JavaScript
  - Data types & expressions
  - Control statements
  - Functions & libraries
  - Strings & arrays
  - Date, document, navigator, user-defined classes

CLIENT-SIDE PROGRAMMING

- Client-side programming
  - Programs are written in a separate programming (or scripting) language
    - e.g., JavaScript, JScript, VBScript
  - Programs are embedded in the HTML of a Web page, with (HTML) tags to identify the program component
    - e.g., <script type="text/javascript"> … </script>
  - The browser executes the program as it loads the page, integrating the dynamic output of the program with the static content of HTML
  - Could also allow the user (client) to input information and process it, might be used to validate input before it's submitted to a remote server

JAVASCRIPT

Javascript code can be embedded in a web page using <script> tags

```
<html>
<!-- COMP519  js01.html  16.08.06 -->

<head>
  <title>JavaScript Page</title>
</head>

<body>
  <script type="text/javascript">
    // silly code to demonstrate output

    document.write("<p>Hello
world!</p>");

    document.write(" <p>How are <br/> "
+
                   " <i>you</i>?</p> ");
  </script>

  <p>Here is some static text as
well.</p>

</body>
</html>
```

document.write displays text in the page text to be displayed can include HTML tags the tags are interpreted by the browser when the text is displayed as in C++/Java, statements end with ; but a line break might also be interpreted as the end of a statement (depends upon browser).JavaScript comments similar to C++/Java

// starts a single line comment

/*…*/ enclose multi-line comments

JAVASCRIPT DATA TYPES & VARIABLES

Javascript has only three primitive data types

STRING : "FOO" 'HOW DO YOU DO?' "I SAID 'HI'." ""

NUMBER: 12 3.14159 1.5E6

BOOLEAN : TRUE FALSE *FIND INFO ON NULL, UNDEFINED

```
<html>
<!-- COMP519  js02.html  16.08.06 -->

<head>
  <title>Data Types and Variables</title>
</head>

<body>
  <script type="text/javascript">
    var x, y;
    x= 1024;
    y=x;  x = "foobar";
    document.write("<p>x = " + y + "</p>");
    document.write("<p>x = " + x + "</p>");
  </script>
</body>
</html>
```

JAVASCRIPT OPERATORS & CONTROL STATEMENTS

```
<html>
<!-- COMP519 js03.html  08.10.10 -->

<head>
  <title>Folding Puzzle</title>
</head>

<body>
 <script type="text/javascript">
    var distanceToSun = 93.3e6*5280*12;
    var thickness = .002;

    var foldCount = 0;
    while (thickness < distanceToSun) {
       thickness *= 2;
       foldCount++;
    }
    document.write("Number of folds = " +
                foldCount);
  </script>
</body>
</html>
```

standard C++/Java operators & control statements are provided in JavaScript

- +, -, *, /, %, ++, --, …
- ==, !=, <, >, <=, >=
- &&, ||, !,===,!==
- if , if-else, switch
- while, for, do-while, …

PUZZLE: Suppose you took a piece of paper and folded it in half, then in half again, and so on.

How many folds before the thickness of the paper reaches from the earth to the sun?

*Lots of information is available online

JAVASCRIPT MATH ROUTINES

```
<html>
<!-- COMP519  js04.html  08.10.10 -->

<head>
  <title>Random Dice Rolls</title>
</head>

<body>
  <div style="text-align:center">
    <script type="text/javascript">
      var roll1 = Math.floor(Math.random()*6) + 1;
      var roll2 = Math.floor(Math.random()*6) + 1;

      document.write("<img src='http://www.csc.liv.ac.uk/"+
          "~martin/teaching/comp519/Images/die" +
          roll1 + ".gif' alt='dice showing ' + roll1 />");
      document.write("  ");
      document.write("<img src='http://www.csc.liv.ac.uk/"+
          "~martin/teaching/comp519/Images/die" +
          roll2 + ".gif' alt='dice showing ' + roll2 />");
    </script>
  </div>
</body>
</html>
```

The built-in Math object contains functions and constants

Math.sqrt
Math.pow
Math.abs
Math.max
Math.min
Math.floor
Math.ceil
Math.round

Math.PI
Math.E

Math.random function returns a real number in [0..1)

INTERACTIVE PAGES USING PROMPT

```
<html>
<!-- COMP519  js05.html  08.10.10 -->

<head>
  <title>Interactive page</title>
</head>

<body>
<script type="text/javascript">
 var userName = prompt("What is your name?", "");

 var userAge = prompt("Your age?", "");
 var userAge = parseFloat(userAge);

    document.write("Hello " + userName + ".")
    if (userAge < 18) {
      document.write("  Do your parents know " +
                     "you are online?");
    }
    else {
      document.write("  Welcome friend!");
    }
</script>

  <p>The rest of the page...</p>
</body>
</html>
```

crude user interaction can take place using prompt

- 1st argument: the prompt message that appears in the dialog box
- 2nd argument: a default value that will appear in the box (in case the user enters nothing)the function returns the value entered by the user in the dialog box (a string)
- if value is a number, must use parseFloat (or parseInt) to convert
- forms will provide a better interface for interaction *(later)*

USER-DEFINED FUNCTIONS

- Function definitions are similar to c++/java, except:
  - No return type for the function (since variables are loosely typed)
  - No variable typing for parameters (since variables are loosely typed)
  - By-value parameter passing only (parameter gets copy of argument)

```
function isPrime(n)
// Assumes: n > 0
// Returns: true if n is prime, else false
{
  if (n < 2) {
    return false;
  }
  else if (n == 2) {
    return true;
  }
  else {
      for (var i = 2; i <= Math.sqrt(n); i++) {
        if (n % i == 0) {
          return false;
        }
      }
      return true;
  }
}
```

Can limit variable scope to the function.

if the first use of a variable is preceded with var, then that variable is local to the function

for modularity, should make all variables in a function local

STRING EXAMPLE: PALINDROMES

```
function strip(str)
// Assumes: str is a string
// Returns: str with all but letters removed
{
  var copy = "";
  for (var i = 0; i < str.length; i++) {
    if ((str.charAt(i) >= "A" && str.charAt(i) <= "Z")
||
        (str.charAt(i) >= "a" && str.charAt(i) <=
"z")) {
      copy += str.charAt(i);
    }
  }
  return copy;
}

function isPalindrome(str)
// Assumes: str is a string
// Returns: true if str is a palindrome, else false
{
  str = strip(str.toUpperCase());

  for(var i = 0; i < Math.floor(str.length/2); i++) {
    if (str.charAt(i) != str.charAt(str.length-i-1)) {
      return false;
    }
  }
  return true;
}
```

suppose we want to test whether a word or phrase is a palindrome

```
<html>
<!-- COMP519  js09.html  11.10.2011 -->

<head>
 <title>Palindrome Checker</title>

  <script type="text/javascript">
        function strip(str)
        {
     // CODE AS SHOWN ON PREVIOUS SLIDE
        }

        function isPalindrome(str)
        {
          // CODE AS SHOWN ON PREVIOUS SLIDE
        }
  </script>
</head>

<body>
  <script type="text/javascript">
    text = prompt("Enter a word or phrase", "Madam, I'm Adam");

    if (isPalindrome(text)) {
      document.write("" + text + "' <b>is</b> a palindrome.");
    }
    else {
      document.write("" + text + "' <b>is not</b> a palindrome.");
    }
  </script>
</body>
</html>
```

JAVASCRIPT ARRAYS

- Arrays store a sequence of items, accessible via an index
  since javascript is loosely typed, elements do not have to be the same type
  - To create an array, allocate space using new (or can assign directly)
    - ITEMS = NEW ARRAY(10); // ALLOCATES SPACE FOR 10 ITEMS
    - ITEMS = NEW ARRAY(); // IF NO SIZE GIVEN, WILL ADJUST DYNAMICALLY
    - ITEMS = [0,0,0,0,0,0,0,0,0,0]; // CAN ASSIGN SIZE & VALUES []
  - To access an array element, use [] (as in c++/java)
    - FOR (I = 0; I < 10; I++) {
    - ITEMS[I] = 0; // STORES 0 AT EACH INDEX
    - }
  - The length property stores the number of items in the array
    - FOR (I = 0; I < ITEMS.LENGTH; I++) {
    - DOCUMENT.WRITE(ITEMS[I] + "<BR>"); // DISPLAYS ELEMENTS
    - }

ARRAY EXAMPLE

```
<html>
<!-- COMP519  js10.html  11.10.2011 -->
<head>
 <title>Die Statistics</title>
 <script type="text/javascript"
src="http://www.csc.liv.ac.uk/~martin/teaching/comp519/JS/random.js">
 </script>
</head>
<body>
 <script type="text/javascript">
    numRolls = 60000;
    dieSides = 6;

    rolls = new Array(dieSides+1);
    for (i = 1; i < rolls.length; i++) {
        rolls[i] = 0;
    }

    for(i = 1; i <= numRolls; i++) {
        rolls[randomInt(1, dieSides)]++;
    }

    for (i = 1; i < rolls.length; i++) {
        document.write("Number of " + i + "'s = " +
                       rolls[i] + "<br />");
    }
  </script>
</body>
</html>
```

suppose we want to simulate die rolls and verify even distribution

keep an array of counters:

- initialize each count to 0
- each time you roll `X`, increment `rolls[X]`
- display each counter

DATE OBJECT

- String & array are the most commonly used objects in javascript
  - Other, special purpose objects also exist
- The date object can be used to access the date and time
  - To create a date object, use new & supply year/month/day/… as desired
  - Today = new date(); // sets to current date & time
  - Newyear = new date(2002,0,1); //sets to jan 1, 2002 12:00am
  - METHODS INCLUDE:
    - newyear.getyear()
    - newyear.getmonth()
    - newyear.getday()
    - newyear.gethours()
    - newyear.getminutes()
    - newyear.getseconds()
    - newyear.getmilliseconds()

DATE EXAMPLE

```
<html>
<!-- COMP519  js11.html  16.08.2006 -->

<head>
  <title>Time page</title>
</head>

<body>
  Time when page was loaded:
  <script type="text/javascript">
    now = new Date();

    document.write("<p>" + now + "</p>");

    time = "AM";
    hours = now.getHours();
    if (hours > 12) {
        hours -= 12;
        time = "PM"
    }
    else if (hours == 0) {
        hours = 12;
    }
    document.write("<p>" + hours + ":" +
                   now.getMinutes() + ":" +
                   now.getSeconds() + " " +
                   time + "</p>");
  </script>
</body>
</html>
```

by default, a date will be displayed in full, e.g.,

Sun Feb 03 22:55:20 GMT-0600 (Central Standard Time) 2002

can pull out portions of the date using the methods and display as desired

here, determine if "AM" or "PM" and adjust so hour between 1-12

10:55:20 PM

JavaScript and HTML validators

- In order to use an HTML validator, and not get error messages from the JavaScript portions, you must "mark" the JavaScipt sections in a particular manner. Otherwise the validator will try to interpret the script as HTML code.
- To do this, you can use a markup like the following in your inline code (this isn't necessary for scripts stored in external files).

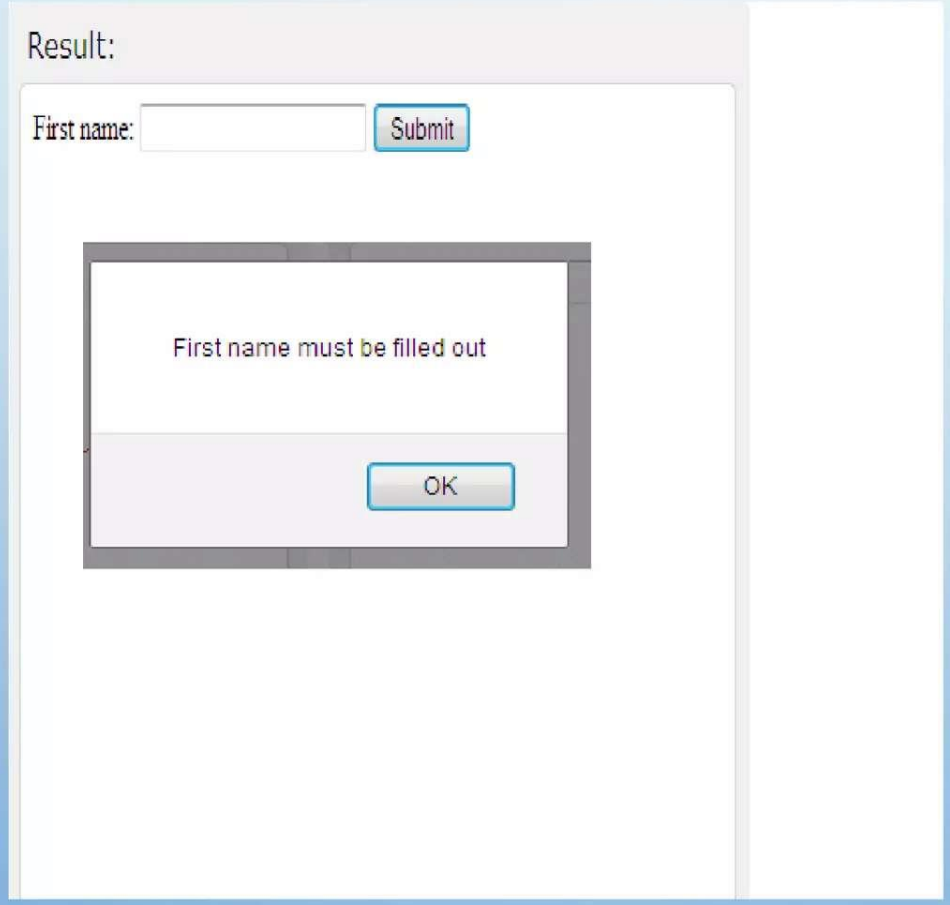
```
<script type="text/javascript"> //  <![CDATA[
document.write("<p>The quick brown fox jumped over the lazy dogs.</p>"); //  **more code here, etc.
</script>
```

```
<!DOCTYPE html>
<html>
<head>
<script>
function validateForm() {
    var x = document.forms["myForm"]["fname"].value;
    if (x==null || x=="") {
        alert("First name must be filled out");
        return false;
    }
}
</script>
</head>

<body>
<form name="myForm" action="demo_form.asp" onsubmit="return
validateForm()" method="post">
First name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
</body>

</html>
```

Output



SQL & MySQL

Week 13-15

Page
67-108

MySql Overview

Objective

- In this presentation you will learn about-
- Introduction & features of MySQL
- Types of SQL Commands- DDL, DML, TCL & DCL
- Data types in MySQL
- Creating Database & Tables
- Inserting, Deleting and modifying records
- Making Simple Queries
- Altering Table Structure

Introduction to MySQL

- MySQL is an Open Source, Fast and Reliable Relational Database Management System (RDBMS) software like Oracle, Sybase, MS SQL Server etc. It was developed by Michael Widenius and AKA Monty.
- The main features of MySQL are-

Open Source & Free of Cost:

- It is Open Source and available at free of cost.

Portability:

- It can be installed and run on any types of Hardware and OS like Linux, MS Windows or Mac etc.

Security :

- It creates secured database protected with password.

Connectivity

- It may connect various types of Network client using different protocols and Programming Languages .

Query Language

- It uses SQL (Structured Query Language) for handling database.

MySQL & SQL

- In order to access data from the MySQL database, all program and user must use SQL (Structured Query Language). SQL is a set of commands that are recognized by all the RDBMSs and has become a standard language for database handling.
- SQL is a language that enables you to create and manage a relational database, in which all the information are kept in tables.
- There are numerous version of SQL. The original version was developed at IBM's San Jose Research Laboratory with a name of Sequel, as a part of System R project in 1970s. It was standardized by ANSI in 1986 by the name of SQL.

Types of SQL Commands

- MySQL follows SQL specifications for its commands . These SQL commands can be categorized as -

Data Definition Language (DDL)

- These SQL commands are used to create, alter and delete database objects like table, views, index etc.
- Example : CREATE , ALTER , DROP etc.

Data Manipulation Language (DML)

- These commands are used to insert, delete, update and retrieve the stored records from the table.
- Ex. SELECT…., INSERT…, DELETE…, UPDATE…. etc.

Transaction Control Language (TCL)

- These commands are used to control the transaction. Ex. COMMIT, ROLLBACK, SAVEPOINT etc.

Data Control Language (DCL)

- These commands are used to manipulate permissions or access rights to the tables etc.
- Ex. GRANT , REVOKE etc.

Data type in MySQL

- **Numeric Data Types:**
  - **INTEGER or INT** – up to 11 digit number without decimal.
  - **SMALLINT** – up to 5 digit number without decimal.
  - **FLOAT (M,D) or DECIMAL(M,D) or NUMERIC(M,D)**
    - Stores Real numbers upto **M** digit length (including .) with **D**
    - decimal places.
    - e.g. Float (10,2) can store 1234567.89
- **Date & Time Data Types:**
  - **DATE** - Stores date in YYYY-MM-DD format.
  - **TIME** - Stores time in HH:MM:SS format.
- **String or Text Data Type:**
  - **CHAR(Size)**
    - A fixed length string up to 255 characters. (default is 1)
  - **VARCHAR(Size)**
    - A variable length string up to 255 characters.

Database Handling commands in MySQL

- **Creating a Database.**
  - The following command will create School database in MySQL.
  - mysql> **CREATE DATABASE School;**
- **Opening a database**
  - To open an existing database, following command is used.
  - mysql> **USE school ;**
- **Getting listings of database and tables**
  - mysql> **SHOW DATABASES;**
  - mysql> **SHOW TABLES;**
- **Deleting a Database and Table**

  mysql> **DROP DATABASE School;**

  DROP TABLE Student;
- **Viewing Table Structure**

  mysql> **DESCRIBE Student;**

Creating Tables & Inserting records

- **Creating Simple Tables:**
  - **CREATE TABLE < Table Name>**
  - **(<Col name1><data type>[(size)],….);**
  - Data types- INTEGER, NUMERIC(P,D), CHAR(n), VARCHAR(n), DATE etc.

```
mysql> CREATE TABLE Employee
(empID Integer,
 ename char(30),
 city char(25),
 pay decimal(10,2));
```

- **Inserting Records:**
  - **INSERT INTO <Table Name> VALUES (value1, vale2, …...);** String and Date type values must be enclosed in single or double quotes.
  - **mysql> INSERT INTO Employee VALUES (1,'Amitabh','Allahabad',15000);**

Making Simple Queries Using SELECT

- The SELECT command of SQL, empower you to make a request (queries) to retrieve stored records from the database.
- The syntax of SQL is given below-

  SELECT < [Distinct | ALL] *| column name(s)> FROM <table(s)>

  WHERE <condition>

  ORDER BY <column name> [ASC | DESC] ;
- Consider the table ***Student*** having some records as –

StID	Name	Fname	DOB	City	Class
S1	Amitabh	Harivansh Rai	1948-11-10	Allahabad	12
S2	Sharukh	Firoz	1970-05-10	Delhi	11
S3	Irphan	Akbar	1970-10-05	Jaipur	11
S4	Salman	Salim Javed	1972-04-10	Mumbai	10

- **Selecting all columns**
- If you want to view all columns of the student table, then you should give the following command-
- mysql> **SELECT * FROM Student ;**
- MySQL will display the all records with all columns in the Student table.
- Is used to represent all columns.

StID	Name	Fname	DOB	City	Class
S1	Amitabh	Harivansh Rai	1948-11-10	Allahabad	12
S2	Sharukh	Firoz	1970-05-10	Delhi	11
S3	Irphan	Akbar	1970-10-05	Jaipur	11
S4	Salman	Salim Javed	1972-04-10	Mumbai	10

- **Selecting columns**
- If you want to view only **Name** and **City** columns of the student table
- mysql> **SELECT Name, City FROM Student ;**

Name	City
Amitabh	Allahabad
Sharukh	Delhi
Irphan	Jaipur
Salman	Mumbai
Abhishek	Mumbai

mysql> **SELECT City, Name FROM Student ;**

City	Name
Allahabad	Amitabh
Delhi	Sharukh
Jaipur	Irphan
Mumbai	Salman
Mumbai	Abhishek

- **Eliminating Duplicate values in a column - DISTINCT**
- mysql> **SELECT City FROM Student ;**

City
Allahabad
Delhi
Jaipur
Mumbai
Mumbai

Mumbai is repeated

MySQL assumes ALL keyword, if you are not using DISTICT keyword.

mysql> **SELECT DISTINCT City FROM Student ;**

City
Allahabad
Delhi
Jaipur
Mumbai

Only Unique Cities are displayed

Doing simple calculations

- We can also perform simple calculations with SQL Select command. SQL provide a dummy table named DUAL, which can be used for this purpose.
- mysql> **SELECT 4*3 ;**
- We can also extend this idea with a columns of the existing table.

SELECT Name, Sal *12 FROM EMP ;

Using Column Aliases

- We can give a different name to a column or expression (Alias) in the

mysql> **SELECT Name, Sal*12 AS 'Annual Salary' FROM EMP;**

mysql> **SELECT Name, DOB AS 'Date of Birth' FROM Student;**

mysql> **SELECT 22/7 AS PI FROM Dual;**

Alias for Sal*12

When Alias name is a single word then single quotes is not required.

Where Condition

- **WHERE <Condition>**
  - We can select specific records by specifying conditions with WHERE clause.
- mysql> **SELECT * FROM Student WHERE City='Mumbai';**

StID	Name	Fname	DOB	City	Class
S4	Salman	Salim Javed	1972-04-10	Mumbai	10
S5	Abhishek	Amitabh	1975-03-12	Mumbai	10

mysql> **SELECT Name, Fname, City from Student**
WHERE Class >10;

Name	Fname	City	Class
Amitabh	Harivansh Rai	Allahabad	12
Sharukh	Firoz	Delhi	11
Irphan	Akbar	Jaipur	11

Selecting Specific Records – WHERE clause

Relational Operators

We can use the following Relational operators in condition.

=, > , < , >=, <=, <>, IS , LIKE, IN, BETWEEN

Logical Operators

We can use the following Logical Operators to connect two conditions.

OR , AND , NOT (!)

```
mysql> SELECT Name, City from Student
       WHERE City <> 'Mumbai' AND Class>10;

mysql> SELECT * FROM Emp
       WHERE Sal >10000 OR Job ='Manager';

mysql> SELECT * FROM Student
       WHERE NOT Grade='A';
```

Selecting Specific Rows – WHERE clause

- **Specifying Range of Values – BETWEEN Operator**

```
mysql> SELECT * FROM Emp
       WHERE Sal BETWEEN 5000 AND 10000 ;
```

The same query can also be written as -

```
mysql> SELECT * FROM Emp
       WHERE Sal >= 5000 AND Sal<=10000 ;
```

Other Logical operators also can be applied-

```
mysql> SELECT * FROM Emp
       WHERE  NOT Sal BETWEEN 5000 AND 10000 ;
```

- **Specifying List – IN Operator**

```
mysql> SELECT * FROM Emp
       WHERE Sal IN (5000, 10000) ;
```

The same query can also be written as -

```
mysql> SELECT * FROM Emp
       WHERE Sal = 5000 OR Sal =10000 ;

mysql> SELECT * FROM Student
       WHERE  City IN ('Mumbai', 'Delhi','Kanpur') ;
```

Selecting Specific Rows – WHERE clause

☐ Searching NULL Values – IS Operator

mysql> **SELECT * FROM Student WHERE City IS NULL ;**

The **NOT** Operator can also be applied -

mysql> **SELECT * FROM Student WHERE City IS NOT NULL;**

☐ Ordering Query Result – ORDER BY Clause

A query result can be orders in ascending (A-Z) or descending (Z-A) order as per any column. Default is Ascending order.

mysql> **SELECT * FROM Student ORDER BY City;**

To get descending order use **DESC** key word.

```
mysql> SELECT * FROM Student ORDER BY City DESC;
```

You can also use

```
mysql> SELECT Name, Fname, City FROM Student
       Where Name LIKE 'R%'  ORDER BY Class;

mysql> SELECT Name, Basic+DA AS 'PAY' FROM Student ORDER BY
PAY;
```

Inserting Records in a Table

- You can insert record in the table by using by using the following DML command.
- **INSERT INTO<Table Name> [<Column list>] VALUES<list of values>**
- If value is not available for a column, NULL can be used.
- Suppose a table STUDENT has been created as per given structure-

StID	NAME	FNAME	DOB	CITY	CLASS

We can insert a record as follows-

```
mysql> INSERT INTO Student VALUES
      ('s1','Amitabh', 'Harivansh','1955-10-25', 'Mumbai', 12);
mysql> INSERT INTO Student VALUES
      ('s2','Sharukh Khan', NULL,'1972-5-25', 'Delhi', 10);
```

Inserting Records from Other Table

- You can insert all or selected record(s) in the table from another table by using Select … command in place of Values.
- Suppose a table named NEWSTUDENT has been created and records to be inserted from OLDSTUDENT table having the same structure of columns.

```
mysql> INSERT INTO Newstudent VALUES
       (SELECET * FROM Oldstudent);
mysql>INSERT INTO Newstudent VALUES
      (SELECT * FROM Oldstudent WHERE City='Mumbai');
mysql> INSERT INTO Newstudent (StID, Name, Class)
       VALUES (Select StID, Name,Class FROM Oldstudent
       WHERE Class>=11);
```

Deleting Records from the Table

- You can delete all or selected record(s) from the table by using the following DML command.
- **DELETE FROM<Table Name> [WHERE <Condition>]**

```
mysql> DELETE FROM Student ;

mysql> DELETE FROM Student WHERE  City='Mumbai' ;
mysql> DELETE FROM Student WHERE Class >=11 ;
mysql> DELETE FROM Student WHERE Class <9 AND City='Delhi';
```

- **You can recall (Undelete) records by giving ROLLBACK command.**

```
mysql> ROLLBACK ;
```

- **You can issue COMMIT command to record the changes permanently.**

```
mysql> COMMIT;
```

Modifying Records –UPDATE Command

- You can modify the values of columns of all or selected records in the table by using the following DML command.
- **UPDATE <Table Name>**
- **SET <Column> = <Expression> [WHERE <Condition>]**

- **mysql> UPDATE Student SET Class =10;**

```
mysql> UPDATE Student SET FName= CONACT('Mr.', FName') ;

mysql> UPDATE Emp SET Sal = Sal+(Sal*10/100);

mysql> UPDATE Emp SET Sal = Sal+(Sal*10/100)
       WHERE Sal <=10000;
```

Working with Tables

Creating Tables:

CREATE TABLE < Table Name>

(<Col name><data type>[(size)][Constraints], ...)

- **Data types -** Commonly used data types are- INTEGER, DECIMAL(P,D), NUMERIC(P,D), CHAR(n), VARCHAR(n), DATE etc.

Employee (EmpID, Ename, Sex, DOB, Pay)

```
mysql> CREATE TABLE Employee
      (EmpID integer,
       Ename char(20),
       Sex  char(1),
       Dob  Date,
       Pay decimal (8,2));
```

Creating Table with Constraints

- One of the major responsibility of a DBMS is to maintain the Integrity of the data i.e Data being stored in the Database must be correct and valid.
- An Integrity Constraints are condition or checks applicable to a column or table which ensures the integrity and validity of data.
- The following constraints are available in MySQL.

Constraints	Description
NOT NULL	Ensures that a column cannot have NULL value.
PRIMARY KEY	Used to identify a row uniquely.
DEFAULT*	Provides a default value for a column, if no value is given.
UNIQUE*	Ensures that all values in a column are different.
CHECK*	Ensures that value for a column should satisfy certain condition.
FOREIGN KEY*	Used to ensure Referential Integrity of the data.

Implementing Constraints in the Table

- **NOT NULL**

  This constraints specifies that column must not contain NULL value i.e. value for the column must be given (**mandatory**)

- **PRIMARY KEY**

  This constraints declare a column as the primary key. Since **Primary key must not have NULL value**, so it is used with NOT NULL constraints.

- **UNIQUE**

  This constraints ensures that the value for the column should be Unique i.e. **no two records have the same** (duplicate) value.

```
mysql> CREATE TABLE Student
       (StCode char(3) NOT NULL PRIMARY KEY,
        Stname char(20)  NOT NULL,
       StAdd varchar(40),
       AdmNo char(5) UNIQUE,
       StAge integer CHECK (StAge>=5) );
```

- UNIQUE allows NULL values but PRIMERY KEY does not.
- A table may have multiple UNIQUE constraints, but there must be only one PRIMERY KEY constraints in a table.

Implementing Primary Key Constraints

- **Defining Primary Key at Column Level:**

```
mysql> CREATE TABLE Student
       ( StCode   char(3)   NOT NULL PRIMARY KEY,
         Stname   char(20)  NOT NULL,
         StAge    int(2) );
```

- **Defining Primary Key at Table Level:**

```
mysql> CREATE TABLE Student
       ( StCode   char(3) NOT NULL,
         Stname   char(20) NOT NULL,
         StAge    int(2),
         PRIMARY KEY (StCode) );
```

A Composite (multi-column) Primary key can be defined as only a Table level whereas Single-column Primary key can be defined in both way i.e. Column level or Table level.

Handling Tables

- **Viewing Table Structure:**
  You can view structure of any table after using database as-
  DESC[RIBE] <table name>
  ```
  mysql> DESC Student;
  ```
- **Deleting Table:**
  You can delete an existing table as-
  DROP TABLE [IF EXIST] <table name>
  ```
  mysql> DROP TABLE Student;
  ```
- **Creating Table from Existing Table:**
  CREATE TABLE <Table name>
  AS (<Select Query>);

```
mysql> CREATE TABLE Staff
       ( Select empID, ename, sex  From Emp);
```

```
mysql> CREATE TABLE Staff
       ( Select * From Emp);
```

Modifying Table Structure

You can alter (modify) the structure of existing table by the using **ALTER TABLE....** Command of MySQL.

You can do the following with the help of ALTER TABLE.. Command.

- **Add a new Column or Constraints**
- **Modifying existing column (name, data type, size etc.)**
- **Delete an existing column or Constraints**
- **Changing Column Name**

ALTER TABLE <Table Name>
ADD|MODIFY|DROP|CHANGE <Column Definition(s)>

You can add/Delete/Modify multiple columns with single ALTER Command.

Modifying Table Structure

- **Adding new column**

  ALTER TABLE <Table Name>
  ADD <Column>[<data type> <size>][<Constraints>]

```
mysql> ALTER TABLE Student ADD (TelNo Integer);
mysql> ALTER TABLE Student ADD (Age Integer DEFAUL 10);
```

- **Modifying Existing Column**

  ALTER TABLE <Table Name>
  MODIFY <Column>[<data type> <size>] [<Constraints>]

```
mysql> ALTER TABLE Student MODIFY Name VARCHAR(40);
mysql> ALTER TABLE Employee MODIFY (Pay DECIMAL (10,2));
```

- **Removing Column & Constraints**

  ALTER TABLE <Table Name>
  DROP <Column name> |<Constraints>

```
mysql> ALTER TABLE Student DROP TelNo;
mysql> ALTER TABLE Emp DROP JOB, DROP Pay;
```

LIVE DATABASE CONNECTION WITH a PROJECT

Week 16 - 17

Page
98-99

PROJECT PLANNING, PRESENTATION, FINAL PROJECT SHOW & SOLUTION


CSS
HTML
C++
</>

THANKYOU