JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

Department of CSE&IT

Lab Manual

Course Code: 21B16CS323

Course Name: Web Technology Lab

B.Tech, ODD SEM 2022

2022-23

S. No.	Contents	Page
1	Department Vision	3
2	Department Mission	3
3	Programme Educational Objectives (PEO)	4
4	Programme Outcomes (PO)	4
5	Programme Specific Outcomes (PSO)	5
6	Course Outcomes (CO)	6
7	Introduction to Laboratory	7
List of Experiments		
1.	**Introduction to HTML**– Introduction to HTML, designing basic web pages using HTML, containing tables, hyperlinks, buttons, and colours.	8
2.	**Introduction to CSS**– Introduction to CSS, changing fonts, colour, other visual aspects using CSS.	21
3.	**Basics of JavaScript** – Designing HTML forms to take user input and perform operations on them.	29
4.	**JavaScript Functions** – JavaScript arithmetic functions, alert boxes changing webpage background using JavaScript.	36
5.	**Introduction to ReactJS** – Basic ReactJS programs taking input and other components	40
6.	**ReactJS** – Implementing Fibonacci Sequence, fizz buzz problem, palindromes, grade calculator	44
7.	**NodeJS** – Arithmetic operations in NodeJS, loops, string processing, server side CRUD operations.	50
8.	**Overview of PHP** – Arrays traversal, loops, sorting, arithmetic operations, and calculator.	54
9.	**Overview of MySQL**– Creating tables & databases, insertion, deletion, updation, querying basic SQL operations.	61
10.	**Database Connectivity with PHP**– Connecting PHP with MySQL, database operations from PHP, forms to take data and store in database, verify the data using PHPMyAdmin.	68
11.	**Web Development Framework Bootstrap** – Use of Bootstrap in web development, designing web pages using bootstrap.	73
12.	**Web Development Framework Django** – Use of Django in web development, designing web pages using Django and Python.	77

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING AND INFORMATIONTECHNOLOGY

VISION

To be a centre of excellence for providing quality education and carrying out cutting edge research to develop future leaders in all aspects of computing, IT and entrepreneurship.

MISSION

MISSION 1: To offer academic programme with state of art curriculum having flexibility for accommodating the latest developments in the areas of computer science and IT

MISSION 2: To conduct research and development activities in contemporary and emerging areas of computer science & engineering and IT.

MISSION 3: To inculcate IT & entrepreneurial skills to produce professionals capable of providing socially relevant and sustainable solutions.

.

Programme Name: B.TECH. IN COMPUTER SCIENCE & ENGINEERING

PROGRAMME EDUCATIONAL OBJECTIVES:

PEO 1: To provide core theoretical and practical knowledge in the domain of Computer Science & Engineering for leading successful career in industries, pursuing higher studies or entrepreneurial endeavors.

PEO 2: To develop the ability to critically think, analyze and make decisions for offering techno-commercially feasible and socially acceptable solutions to real life problems in the areas of computing.

PEO 3: To imbibe lifelong learning, professional and ethical attitude for embracing global challenges and make positive impact on environment and society.

PROGRAMME OUTCOMES:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAMME SPECIFIC OUTCOMES:

PSO 1: Able to identify suitable data structures and algorithms to design, develop and evaluate effective solutions for real-life and research problems.

PSO 2: Able to excel in various programming/project competitions and technological challenges laid by professional societies.

COURSE OUTCOMES

COURSE OUTCOMES		COGNITIVE LEVELS
C376.1	Apply the fundamental elements of Web development in design of web pages	Apply (Level 3)
C376.2	Understand the web development concepts built on Advanced Java Scripting	Understand (Level 2)
C376.3	Apply functional aspects of database handling to create database using PHP	Apply (Level 3)
C376.4	Understand event-driven programming using React JS, Node JS	Understand (Level 2)
C376.5	Use the popular web development frameworks to build web applications	Understand (Level 2)

WEB TECHNOLOGY LAB

This course focuses on web technologies such as HTML, CSS, JS, NodeJS, ReactJS, Bootstrap, Django, PHP and SQL. Students are introduced to foundation concepts of Hyper Text Markup Language (HTML), CSS which is used to design and layout webpages, JavaScript which is used for client-side scripting, PHP and SQL which are used to design dynamic web pages for accessing and manipulating databases.

Lab 1
Introduction to HTML

HTML stands for Hypertext Markup Language, and it is the most widely used language to write Web Pages. HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers. Now, HTML is being widely used to format web pages with the help of different tags available in HTML language.

- *Hypertext* is text containing hyperlinks (links) to other texts. These links allow the user to jump to some other place in the same web page, or other web pages.
- *Markup Language* is a language that allows users to mark-up a text document with special symbols called tags. These tags tell the browser how to display the text.
- Each html document or web page is basically a normal text file having *.html* file extension.
- Two tools required for working with HTML documents are:
    - HTML Editor (WYSIWYG or Text) – for creating the document
    - Web Browser – for viewing the document

HTML Tags

Tags act as building blocks for a web page. They define how the web browser must format and display the information on the web page. Tags are enclosed in angle brackets, and are placed before and after the text being marked up.

Most tags have two parts, an opening tag (aka ON tag) and a closing tag (aka OFF tag).

```
<h1>The Sun – Source of Life on Earth</h1>
<p>The Sun is a big ball of gas. It lies at the center of our solar system.</p>
```

When using nested tags, the tags must be closed in the order in which they were opened. For example:

```
<p><b>Tags opened first must be closed last.</b></p>
```

HTML Elements

HTML elements are of two types:

- Container Element

- Defined by a ON tag, some content, and an OFF tag:

  <ON tag>Content goes here</OFF tag>

  Example – Paragraph element

  The Sun is a big ball of gas. It lies at the center of our solar system.

- Void or Empty Element

  - Void elements contain only ON tags. They do not enclose any content, and have no OFF tags.

    Example - <img> (image),
 (break), <hr> (horizontal rule), and <meta> tags.

HTML Attributes

Attributes enhance the functionality of an HTML element by providing additional information. They are always specified in the opening tag. Attributes consists of two parts − name and value. The name is the property you want to set. The value is the option chosen for the property. For example,

 { Here, align is the attribute name and left is the value assigned to it }

<img src="moon.jpg" alt="A photo of the moon">

Jaypee Institute of Information Technology, Noida

<h1 style="color:blue; font-size:30px;">This is a heading</h1>

A void element can also have attributes.

Points to remember when using HTML

- Always declare the document type as the first line in your document. {<!DOCTYPE html>}
- Tag and attribute names are case-insensitive.
- Tag name is always surrounded by angle brackets.
- Tag names don’t contain spaces.
- Closing of container elements must be ensured.
- Attribute values are usually enclosed in double quotes, although single quotes can also be used.
- For most tags attributes are optional. However, for some tags specifying attributes is mandatory such as for <img> and <a>.
- When using nested tags, you have to close the inner tag before closing the outer tag.

HTML Document Structure

HTML document (web page) has two sections: Head and Body. Head provides general information about the web page, which does not get displayed on the web page, and hence is not visible to the user. It may contain the elements: <title>, <style>, <meta>, <link>, <script>, and <base>. Body encloses the content to be displayed on the web page. It includes text, graphics, tags and attributes. <HTML> tag is also called the Root Element.

In its simplest form, following is an example of an HTML document:

```
<!DOCTYPE html>
<html>
<head>
        <title>This is document title</title>
</head>
<body>
        <h1>This is a heading</h1>
        <p>Document content goes here.....</p>
</body>
</html>
```

Questions

Q1. Create a webpage with HTML describing your hobbies. You can use lists, heading & tables.

Q2. Apply various colors to suitably distinguish key words. Also apply font styling like italics, underline and two other fonts to words you find appropriate. Also use header tags.

Q3. Apply hyperlink on specific words. Such as clicking football will open FIFA website.

Q4. Insert an image and create a link such that clicking on image takes user to other page.

Q5. Change the background color of the page. At the bottom create a link to take user to the top of the page

Lab 2

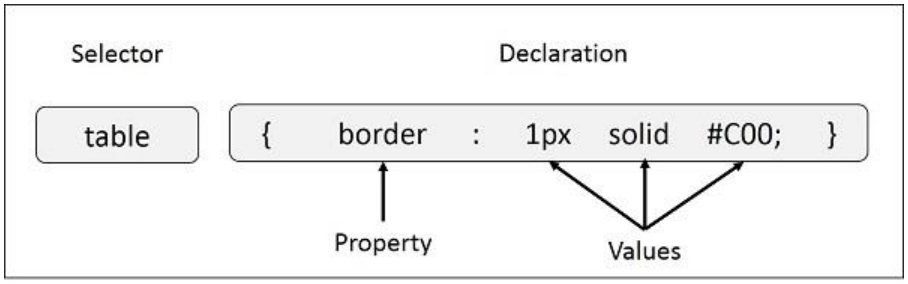
Introduction to CSS

Cascading Style Sheets (CSS) is the language we use to style an HTML document. It provides following benefits:

- **Superior styles to HTML** − CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** − Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** − Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

Syntax

selector { property: value }



Example

You can define a table border as follows:

table{ border :1px solid #C00; }

Here, table is a selector and border is a property and given value *1px solid #C00* is the value of that property.

CSS provides multiple selectors:.

- The Type Selectors- This is the same selector we have seen above. Again, one more example to give a color to all level 1 headings:

```
h1 {
  color: #36CFFF;
}
```

- The Universal Selectors - Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type −

```
* {
  color: #000000;
}
```

This rule renders the content of every element in our document in black.

- The Descendant Selectors - Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to element only when it lies inside <ul> tag.

```
ul em {
  color: #000000;
}
```

- The Class Selectors - You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black {
  color: #000000;
}
```

This rule renders the content in black for every element with class attribute set to *black* in our document. You can make it a bit more particular. For example,

```
h1.black {
  color: #000000;
}
```

This rule renders the content in black for only <h1> elements with class attribute set to *black*.

You can apply more than one class selectors to given element. Consider the following example −

```
<p class = "center bold">
    This para will be styled by the classes center and bold.
</p>
```

- The ID Selectors - You can define style rules based on the *id* attribute of the elements. All the elements having that *id* will be formatted according to the defined rule.

```
#black {
  color: #000000;
}
```

This rule renders the content in black for every element with *id* attribute set to *black* in our document. You can make it a bit more particular. For example,

```
h1#black {
  color: #000000;
}
```

This rule renders the content in black for only <h1> elements with *id* attribute set to *black*.

The true power of *id* selectors is when they are used as the foundation for descendant selectors, For example,

```
#black h2 {
  color: #000000;
}
```

In this example all level 2 headings will be displayed in black color when those headings will lie with in tags having *id* attribute set to *black*.

- The Child Selectors - You have seen the descendant selectors. There is one more type of selector, which is very similar to descendants but have different functionality. Consider the following example,

```
body > p {
  color: #000000;
}
```

This rule will render all the paragraphs in black if they are direct child of <body> element. Other paragraphs put inside other elements like or <td> would not have any effect of this rule.

- The Attribute Selectors - You can also apply styles to HTML elements with particular attributes. The style rule below will match all the input elements having a type attribute with a value of *text* −

```
input[type = "text"] {
  color: #000000;
}
```

The advantage to this method is that the <input type = "submit" /> element is unaffected, and the color applied only to the desired text fields.

Multiple Style Rules

You may need to define multiple style rules for a single element. You can define these rules to combine multiple properties and corresponding values into a single block as defined in the following example,

```
h1 {
```

```
  color: #36C;
  font-weight: normal;
  letter-spacing: .4em;
  margin-bottom: 1em;
  text-transform: lowercase;
}
```

Here all the property and value pairs are separated by a **semicolon (;)**. You can keep them in a single line or multiple lines. For better readability, we keep them in separate lines.

Grouping Selectors

You can apply a style to many selectors if you like. Just separate the selectors with a comma, as given in the following example,

```
h1, h2, h3 {
  color: #36C;
  font-weight: normal;
  letter-spacing: .4em;
  margin-bottom: 1em;
  text-transform: lowercase;
}
```

This define style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.

You can combine the various *id* selectors together as shown below −

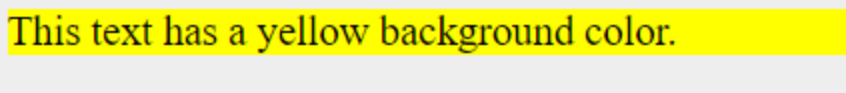
```
#content, #footer, #supplement {
  position: absolute;
  left: 510px;
  width: 200px;
}
```

Example

Backgrounds - to set the background color for an element.

```
<html>
  <head>
  </head>

  <body>
    <p style = "background-color:yellow;">
      This text has a yellow background color.
    </p>
  </body>
</html>
```

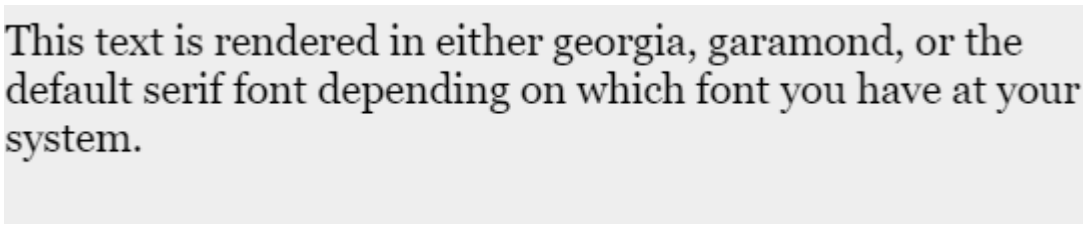
This will produce following result:



Fonts - to set the font family of an element. Possible value could be any font family name.

```
<html>
  <head>
  </head>

  <body>
    <p style = "font-family:georgia,garamond,serif;">
      This text is rendered in either georgia, garamond, or the
      default serif font depending on which font  you have at your system.
    </p>
  </body>
</html>
```

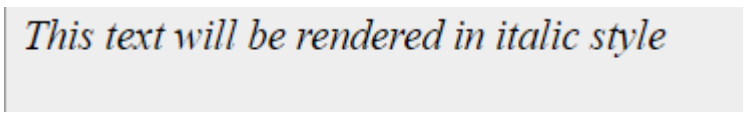
This will produce following result:



Set the Font Style - to set the font style of an element. Possible values are *normal, italic and oblique*.

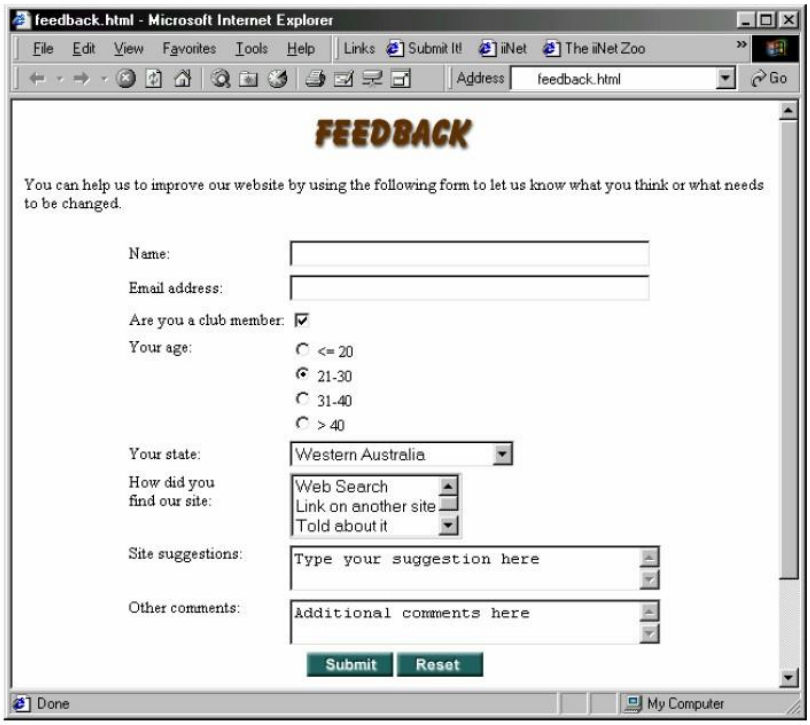
```
<html>
  <head>
  </head>

  <body>
    <p style = "font-style:italic;">
      This text will be rendered in italic style
    </p>
  </body>
</html>
```

This will produce following result:

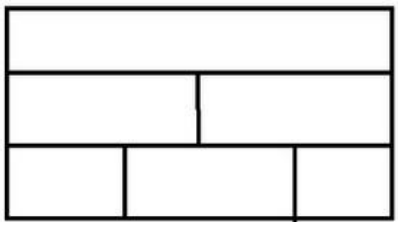


Questions

Q1. Design the following webpage feedback.html, to store user feedback. Use html form controls and attributes to allow user make appropriate choices.



Q2. Design the following table layouts:

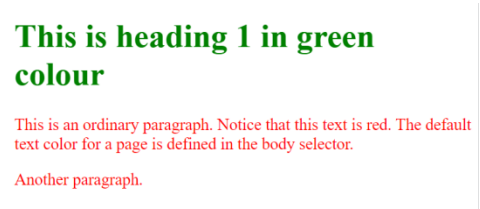


Q3. Develop a webpage for "Student Personal Details" that should allow user entry of following details: first name, last name, gender, address, guardian's name, birth date, mail-id, nationality, etc. Make use of following form controls to create the page design:

a. Text box
b. Text area
c. Radio buttons
d. Submit buttons
e. Reset buttons

Q4. Write HTML and CSS for the following output.

Q5. Write HTML and CSS for the following output.. The default text colour is red



Lab 3

Basics of JavaScript

JavaScript (JS) is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

Client-Side JavaScript

- Client-side JS is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.
- It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.
- The JS client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JS to check if the user has entered a valid e-mail address in a form field, before the page request is submitted to the server.
- The JS code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.
- JS can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

Advantages of JavaScript

The merits of using JavaScript are −

- **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces** − You can use JS to include such items as drag-and-drop components and sliders to give a rich interface to your site visitors.

Limitations of JavaScript

We cannot treat JS as a full-fledged programming language. It lacks the following important features −

- Client-side JS does not allow the reading or writing of files. This has been kept for security reason.
- JS cannot be used for networking applications because there is no such support available.
- JS doesn't have any multi-threading or multiprocessor capabilities.

JS is a lightweight, interpreted programming language that allows you to build interactivity into otherwise static HTML pages.

Syntax

JS can be implemented using JS statements that are placed within the **<script>... </script>** HTML tags in a web page. You can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags. The <script> tag alerts the browser program to start interpreting all the text between these tags as a script.

A simple syntax of your JS will appear as follows:

```
<script type= “text/javascript” language= “javascript”...>
  JavaScript code
</script>
```

The script tag takes two important attributes −

- **Language** − This attribute specifies what scripting language you are using. Typically, its value will be *javascript*. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **Type** − This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "*text/javascript*".

So your JS segment will look like:

```
<script language = "javascript" type = "text/javascript">
  JavaScript code
</script>
```

Your First JavaScript Code

Take a look at the following code.

```
<html>
  <body>
    <script language = "javascript" type = "text/javascript">
      <!--
```

```
            document.write("Hello World!")
         //-->
        </script>
      </body>
    </html>
```

This code will produce the following result −

```
Hello World!
```

Let us take a sample example to print out "Hello World". We added an optional HTML comment that surrounds our JS code. This is to save our code from a browser that does not support JS. The comment ends with a "//-->". Here "//" signifies a comment in JS, so we add that to prevent a browser from reading the end of the HTML comment as a piece of JS code. Next, we call a function *document.write* which writes a string into our HTML document. This function can be used to write text, HTML, or both.

Whitespace and Line Breaks

JavaScript ignores spaces, tabs, and newlines that appear in JS programs. You can use spaces, tabs, and newlines freely in your program and you are free to format and indent your programs in a neat and consistent way that makes the code easy to read and understand.

Semicolons are Optional

Simple statements in JS are generally followed by a semicolon character, just as they are in C, C++, and Java. JS, however, allows you to omit this semicolon if each of your statements are placed on a separate line. For example, the following code could be written without semicolons.

```
<script language = "javascript" type = "text/javascript">
  <!--
    var1 = 10
    var2 = 20
  //-->
</script>
```

But when formatted in a single line as follows, you must use semicolons −

```
<script language = "javascript" type = "text/javascript">
  <!--
    var1 = 10; var2 = 20;
  //-->
</script>
```

Note − It is a good programming practice to use semicolons.

Case Sensitivity

JS is a case-sensitive language. This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters. So the identifiers **Time** and **TIME** will convey different meanings in JS.

Comments in JavaScript

JS supports both C-style and C++-style comments, Thus −

- Any text between a // and the end of a line is treated as a comment and is ignored by JS.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JS also recognizes the HTML comment opening sequence <!--. JS treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

Placement in HTML File

There is a flexibility given to include JS code anywhere in an HTML document. However the most preferred ways to include JS in an HTML file are as follows −

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in an external file and then include in <head>...</head> section.

JavaScript in External File

As you begin to work more extensively with JS, you will be likely to find that there are cases where you are reusing identical JS code on multiple pages of a site.

You are not restricted to be maintaining identical code in multiple HTML files. The **script** tag provides a mechanism to allow you to store JS in an external file and then include it into your HTML files.

Here is an example to show how you can include an external JS file in your HTML code using **script** tag and its **src** attribute.

```
<html>
  <head>
    <script type = "text/javascript" src = "filename.js" ></script>
  </head>

  <body>
```

```
    .....
  </body>
</html>
```

To use JS from an external file source, you need to write all your JS source code in a simple text file with the extension ".js" and then include that file as shown above.

For example, you can keep the following content in **filename.js** file and then you can use **sayHello** function in your HTML file after including the filename.js file.

```
function sayHello() {
  alert("Hello World")
}
```

Questions

Q1. Write a function in java script to print the first 10 multiples of 3.

Q2. Write a function in JS to check which out of three given number is smallest.

Q3. Write a webpage that uses JS to pop up an alert box when a button is pressed.

Q4. Design a complete calculator with all 4 basic arithmetic operations in JS.

Q5. Design a web page that changes its background colour as per user's choice. The user is given three buttons red, blue green.

Lab 4

JavaScript Functions

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions.

Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions. You must have seen functions like **alert()** and **write()** in the earlier chapters. We were using these functions again and again, but they had been written in core JavaScript only once. JavaScript allows us to write our own functions as well.

Function Definition

Before we use a function, we need to define it. The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax

The basic syntax is shown here.

```
<script type = "text/javascript">
    function functionname(parameter-list) {
      statements
    }
</script>
```

Example

Try the following example. It defines a function called *sayHello* that takes no parameters

```
<script type = "text/javascript">
    function sayHello() {
      alert("Hello there");
    }
</script>
```

Calling a Function

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
  <head>
    <script type = "text/javascript">
      function sayHello() {
```

```
        document.write ("Hello there!");
      }
    </script>
  </head>
  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello()" value = "Say Hello">
    </form>
    <p>Use different text in write method and then try...</p>
  </body>
</html>
```

Function Parameters

Till now, we have seen functions without parameters. But there is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma.

Try the following example. We have modified our sayHello function to take two parameters.

```
<html>
  <head>
    <script type = "text/javascript">
      function sayHello(name, age) {
        document.write (name + " is " + age + " years old.");
      }
    </script>
  </head>
  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "sayHello('Zara', 7)" value = "Say Hello">
    </form>
    <p>Use different parameters inside the function and then try...</p>
  </body>
</html>
```

The return Statement

A JavaScript function can have an optional return statement. This is required if you want to return a value from a function. This statement should be the last statement in a function.For

example, you can pass two numbers in a function and then you can expect the function to return their multiplication in your calling program.

Try the following example. It defines a function that takes two parameters and concatenates them before returning the resultant in the calling program.

```
<html>
  <head>
    <script type = "text/javascript">
      function concatenate(first, last) {
        var full;
        full = first + last;
        return full;
      }
      function secondFunction() {
        var result;
        result = concatenate('Zara', 'Ali');
        document.write (result );
      }
    </script>
  </head>
  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "secondFunction()" value = "Call Function">
    </form>
    <p>Use different parameters inside the function and then try...</p>
  </body>
</html>
```

Questions

Q1. Write a function in JS to check if a string is palindrome.

Q2. Write a function in JS to calculate the number of vowel in a given string.

Q3. Write a function in JS to calculate age from Date of birth.

Lab 5
Introduction to ReactJS

ReactJS is a simple, feature rich, component based JavaScript UI library. It can be used to develop small applications as well as big, complex applications. ReactJS provides minimal and solid feature set to kick-start a web application. React community compliments React library by providing large set of ready-made components to develop web application in a record time. React community also provides advanced concept like state management, routing, etc., on top of the React library.

Benefits

Few benefits of using *React library* are as follows −

- Easy to learn
- Easy to adept in modern as well as legacy application
- Faster way to code a functionality
- Availability of large number of ready-made component
- Large and active community

Applications

Few popular websites powered by *React library* are listed below −

- *Facebook*, popular social media application
- *Instagram*, popular photo sharing application
- *Netflix*, popular media streaming application
- *Code Academy*, popular online training application
- *Reddit*, popular content sharing application

As you see, most popular application in every field is being developed by *React Library*.

ReactJS - Installation

Before moving to the installation, let us verify the prerequisite first.

React provides CLI tools for the developer to fast forward the creation, development and deployment of the React based web application. React CLI tools depends on the Node.js and must be installed in your system. Hopefully, you have installed Node.js on your machine. We can check it using the below command −

```
node --version
```

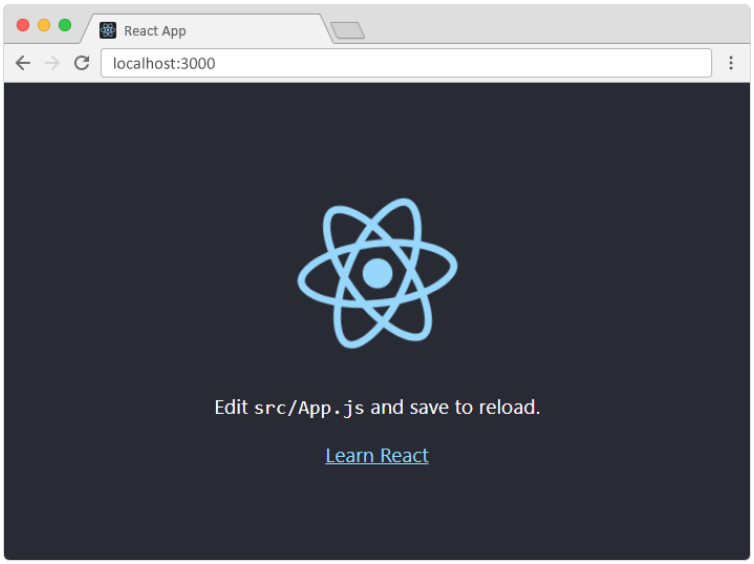
You could see the version of Nodejs you might have installed. It is shown as below for me,

```
v14.2.0
```

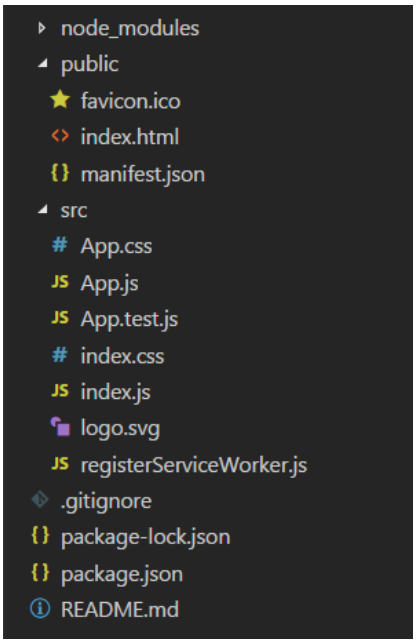
If *Nodejs* is not installed, you can download and install by visiting https://nodejs.org/en/download/.

Setting up a React environment – using Create React App

- Install Node.js
- Set up create-react-app - run command *npm install -g create-react-app* in your terminal
- Create a react app - run command *npx create-react-app your-app-name* in your terminal
- Move to the newly created directory
- Start the project - run command *npm start* in your terminal
- A new window will open in the web browser at localhost:3000 with your new React app, as shown below.



When we create a React project, our application's structure looks as below:

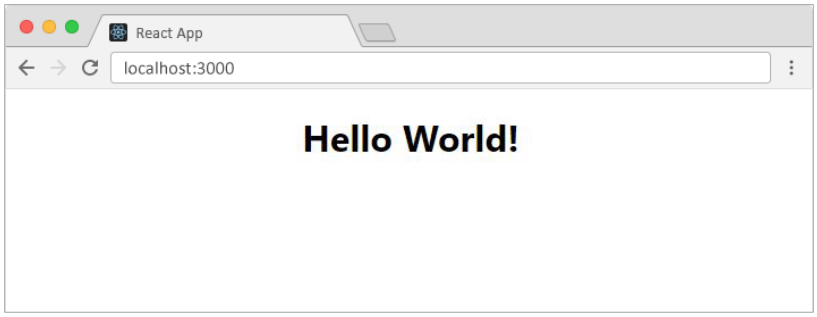


- node_modules - holds all the dependencies and sub-dependencies of our project
- Public folder - this folder contains 3 files
    - favicon.ico - contains an icon which displays on the browser.
    - index.html - due to this file, the public folder, known as “root folder”, gets served by the web server.
    - manifest.json – contains metadata about our application.
- src folder - this folder contains actual source code for developers. Pre-existing files are:
    - App.css - gives some CSS classes/ styling which is used by App.js file.
    - App.js - a sample React component called “App” which we get for free when creating a new app.
    - App.test.js - a test file
    - index.css - stores the base styling for the application.
    - index.js - stores the main Render call from ReactDOM. It imports App.js component and tells React where to render it.
    - logo.svg - contains the logo of Reactjs, visible on the browser.
    - registerServiceWorker.js - used for registering a service
- Package.json - contains the information like the name of project, versions, dependencies etc. Whenever we install a third party library, it automatically gets registered into this file

Getting Started

Type the following in index.js file. Refresh the browser to get the output as shown.

```
import React from 'react'
import ReactDOM from 'react-dom'
import './index.css'

class App extends React.Component {
  render() {
    return (
      <div className="App">
        <h1>Hello World</h1>
      </div>
    )
  }
}
ReactDOM.render(<App />, document.getElementById('root'))
```



Question

Q1. Install ReactJS on your computer and run the hello world code.

Lab 6
ReactJS

React.JS/ React library is built on a solid foundation. It is simple, flexible and extensible. It is used to create user interface (UI) for a web application. React's primary purpose is to enable the developer to create UI using pure JavaScript. Normally, every user interface library introduces a new template language (which we need to learn) to design the user interface and provides an option to write logic, either inside the template or separately. Instead of introducing new template language, React introduces three simple concepts as given below:

- **React elements**: JavaScript representation of HTML DOM. React provides an API, *React.createElement* to create React Element.
- **JSX**: A JavaScript extension to design user interface. JSX is an XML based, extensible language supporting HTML syntax with little modification. JSX can be compiled to React Elements and used to create user interface.
- **React component**: React component is the primary building block of the React application. It uses React elements and JSX to design its UI. React component is basically a JavaScript class (extends the *React.component* class) or a pure JavaScript function. React component has properties, state management, life cycle and event handler. It can be used for simple as well as advanced logic.

Getting Started

Let us understand the workflow of a React application by creating and analyzing a simple React application.

- Create a react app - run command *npx create-react-app first-react-app* in your terminal
- Move to the newly created directory
- Type the following code in the file "*App.js*" present in the *src* folder located inside your *first-react-app* folder.

```
import React, {Component} from 'react'
import ReactDOM from 'react-dom'
import './index.css'

class App extends Component {
  render() {
    return (
      <div className="App">
        <h1>Learning React!</h1>
      </div>
    )
  }
}

ReactDOM.render(<App />, document.getElementById('root'))
```

- Start the project - run command *npm start* in your terminal
- A new window will open in the web browser at localhost:3000 with the output.

The *ReactDOM.render()* function takes two arguments, HTML code and an HTML element. The purpose of the function is to display the specified HTML code inside the specified HTML element. So, where does the output of the React app get rendered?

There is a folder in your React project, named "public". Present in this folder is a file *index.html*. If you'll check, you'll find that there is a single in the body. It is inside this that our React application will be rendered.

Questions

Q1. Build a React component to compute the Fibonacci sequence up to first *n* terms. The value of *n* should be passed as an argument to the component.

Q2. Build a React component that counts incrementally, replacing any number divisible by three with the word "fizz", and any number divisible by five with the word "buzz".

Q3. Build a React component to check for palindromes. Take input from user.

Q4. Build a React component that functions as a student result calculator where user will enter student marks out of 100. The grades are given as follows:

Marks range	Grade
0<marks<30	F
30<marks<40	D
40<marks<50	C
50<marks<60	B
Marks>60	A

Lab 7
NodeJS

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux. Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Node.js = Runtime Environment + JavaScript Library

Features

Following are some of the important features that make Node.js the first choice of software architects.

- **Asynchronous and Event Driven** − All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.

- **Very Fast** − Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.

- **Single Threaded but Highly Scalable** − Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.

- **No Buffering** − Node.js applications never buffer any data. These applications simply output the data in chunks.

- **License** − Node.js is released under the MIT license.

Where to Use Node.js?

Following are the areas where Node.js is proving itself as a perfect technology partner.

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

Node.js - First Application

Before creating an actual "Hello, World!" application using Node.js, let us see the components of a Node.js application. A Node.js application consists of the following three important components −

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

Creating Node.js Application

Step 1 - Import Required Module
We use the require directive to load the http module and store the returned HTTP instance into an http variable as follows:

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

Step 2 - Create Server
We use the created http instance and call http.createServer() method to create a server instance and then we bind it at port 8081 using the listen method associated with the server instance. Pass it a function with parameters request and response. Write the sample implementation to always return "Hello World".

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

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

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

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

Step 3 - Testing Request & Response
Let's put step 1 and 2 together in a file called *main.js* and start our HTTP server as shown below:

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

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

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

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

```
$ node main.js
```

Verify the Output. Server has started. [Server running at http://127.0.0.1:8081/]

Make a Request to the Node.js Server [Open http://127.0.0.1:8081/ in any browser and observe].

Questions

Q1. Display "Hello World" in a web browser Using Node JS.
Q2. Make a x^y calculator in Node JS. The user will enter the value of X,Y
Q3. Check if a string is palindrome or not in Node JS
Q4. Create a server-side application demonstrating the implementation of all the CRUD operations.

Lab 8

Overview of PHP

PHP is a server-side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites. It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Microsoft SQL Server, etc. PHP can perform system functions like from files on a system it can create, open, read, write, and close them. It can be used to handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user. You can add, delete, modify elements within your database through PHP. Using PHP, you can restrict users to access some pages of your website.

- PHP script can be placed anywhere in the document.
- Default file extension is ".php".
- PHP statements end with a semicolon.
- Keywords, classes, functions, and user-defined functions are not case-sensitive.
- Variable names are case-sensitive.
- All PHP code must be included inside one of the three special markup tags recognized by the PHP Parser:

```
<?php PHP code goes here ?>
<?    PHP code goes here ?>
<script language = "php"> PHP code goes here </script>
```

"Hello World" Script in PHP

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script. As mentioned earlier, PHP is embedded in HTML. That means that amongst your normal HTML, you'll have PHP statements like this −

```
<html>
    <head>
    <title>Hello World</title>
    </head>
    <body>
    <?php echo "Hello, World!";?>
    </body>
</html>
```

When you've finished typing it all, save the page as *first.php*. Remember: when you're saving your work, save it to the *htdocs* folder (XAMPP) or *www* folder (WAMP).

To run the page, type this in the browser address bar: http://localhost/first.php. If you've created a folder inside the htdocs or www folder, then the address to type would be something like: http://localhost/folder-name/first.php

Output in browser:

Hello, World!

PHP code is executed on server, and plain HTML response is displayed on client's browser. If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

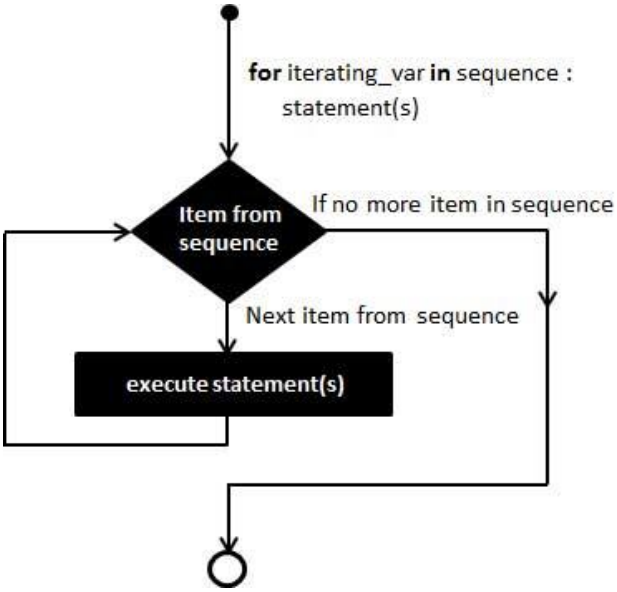
Loops

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

- **for** − loops through a block of code a specified number of times.
- **while** − loops through a block of code if and as long as a specified condition is true.
- **do...while** − loops through a block of code once, and then repeats the loop as long as a special condition is true.
- **foreach** − loops through a block of code for each element in an array.

The for loop statement

The for statement is used when you know how many times you want to execute a statement or a block of statements.



Syntax

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

The initializer is used to set the start value for the counter of the number of loop iterations. A variable may be declared here for this purpose and it is traditional to name it $i.

Example

The following example makes five iterations and changes the assigned value of two variables on each pass of the loop −

```
<html>
  <body>
    <?php
      $a = 0;
      $b = 0;
      for($i = 0; $i<5; $i++ ) {
        $a += 10;
        $b += 5;
      }
      echo ("At the end of the loop a = $a and b = $b”);
    ?>

  </body>
</html>
```

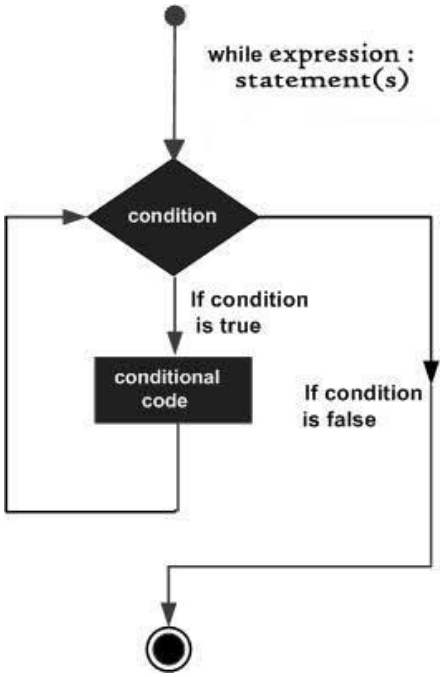
This will produce the following result −

At the end of the loop a = 50 and b = 25

The while loop statement

The while statement will execute a block of code if and as long as a test expression is true.

If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.



Syntax

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

Example

This example decrements a variable value on each iteration of the loop and the counter increments until it reaches 10 when the evaluation is false and the loop ends.

```
<html>
  <body>
    <?php
      $i = 0;
      $num = 50;
      while( $i < 10) {
        $num--;
        $i++;
      }
      echo ("Loop stopped at i = $i and num = $num" );
    ?>
  </body>
</html>
```

This will produce the following result −

Loop stopped at i = 10 and num = 40

The do...while loop statement

The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

Syntax

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

Example

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 10 −

```
<html>
  <body>
    <?php
      $i = 0;
      $num = 0;
      do {
        $i++;
      }
      while( $i < 10 );
      echo ("Loop stopped at i = $i" );
    ?>
  </body>
</html>
```

This will produce the following result −

Loop stopped at i = 10

The foreach loop statement

The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax

```
foreach (array as value) {
  code to be executed;
}
```

Example

Try out following example to list out the values of an array.

```
<html>
  <body>
    <?php
      $array = array( 1, 2, 3, 4, 5);

      foreach( $array as $value ) {
        echo "Value is $value <br />";
      }
    ?>
  </body>
</html>
```

This will produce the following result −

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
```

Questions

Write PHP script

1. To find number of elements in an array and print each element.
2. To sort elements in an array in ascending order and print the numbers as a comma separated list.
3. To find the sum and product of elements in an array.
4. To remove the duplicate values from an array.
5. To create a Simple Calculator.
6. To compute the value of 2^n.
7. To check whether a given year is a leap year or not.
8. To print the following pattern

   1
   1 2
   1 2 3
   1 2 3 4
   1 2 3 4 5

Lab 9

Overview of MySQL

SQL is a standard language for accessing, manipulating and retrieving data in databases. To build a dynamic web site you will need:

- An RDBMS database program (i.e. MS Access, SQL Server, MySQL)
- A server-side scripting language, like PHP or ASP
- SQL to access and manipulate the data
- HTML / CSS to style the page

SQL Datatypes

Category	Data Type	Description
STRING	CHAR(n)	Fixed length
	VARCHAR(n)	Variable length
NUMERIC	TINYINT, SMALLINT, MEDIUMINT, INT, INTEGER, BIGINT,	Integer (whole number)
	FLOAT, DOUBLE	Floating point number
	BOOLEAN, BOOL	Boolean value
	DECIMAL(n,d), DEC(n,d)	Exact fixed point number
DATE AND TIME	DATE	Date (YYYY-MM-DD)
	DATETIME(fsp)	Date and time combination (YYYY-MM-DD hh:mm:ss)
	TIMESTAMP(fsp)	Timestamp (YYYY-MM-DD hh:mm:ss)
	TIME(fsp)	Time (hh:mm:ss)
	YEAR	A year in four-digit format

SQL Operators

Category	Data Type	Description
ARITHMETIC	+, -, *, /, %	
COMPARISON	=, !=, <>, <, >, >=, <=, !<, !>	! depicts NOT operator
LOGICAL	ALL	TRUE if all of the subquery values meet the condition
	AND	TRUE if all the conditions separated by AND are TRUE
	ANY	TRUE if any of the subquery values meet the condition
	BETWEEN	TRUE if the operand is within the range of comparisons
	EXISTS	TRUE if the subquery returns one or more records
	IN	TRUE if the operand is equal to one of a list of expressions
	LIKE	TRUE if the operand matches a pattern
	NOT	Displays a record if the condition(s) is NOT TRUE
	OR	TRUE if any of the conditions separated by OR is TRUE
	SOME	TRUE if any of the subquery values meet the condition

SQL Commands

Command	Description	
SELECT	Retrieves data from database	Data Manipulation Language (DML)
UPDATE	Modifies data in the database	
DELETE	Deletes data from database	
INSERT INTO	Inserts data into database	
CREATE DATABASE/ CREATE TABLE	Creates a new database/ Creates a new table	Data Definition Language (DDL)
ALTER TABLE	Add, delete, or modify columns in an existing table	
DROP DATABASE/ DROP TABLE	Deletes a database/ Deletes a table	

CREATE DATABASE

- Used to create a new database
- Syntax: `CREATE DATABASE databasename;`
- To view a list of existing databases use the command:

```
SHOW DATABASES;
```

```
CREATE DATABASE collegeDB;
```

CREATE TABLE

- Used to create a new table
- Syntax:

```
CREATE TABLE table_name
(
    column1 datatype,
    column2 datatype,
    ....
);
```

```
CREATE TABLE Student_Info
(
    Enrolment_No int,
    First_Name varchar(15),
    Last_Name varchar(15),
    Address varchar(100),
    Branch int
);
```

Enrolment_No	First_Name	Last_Name	Address	Branch

DROP DATABASE

- Used to drop (delete) an existing database
- Syntax: `DROP DATABASE databasename;`
- Dropping a database, deletes all the associated tables as well.

DROP TABLE

- Used to drop (delete) an existing table
- Syntax:

```
DROP TABLE table_name;

DROP TABLE IF EXISTS table_name;
```

```
DROP TABLE Student_Info;
```

- Dropping a table, deletes all the data stored in the table.

TRUNCATE TABLE

- Used to delete the data in a table, and not the table itself.
- Syntax:

```
TRUNCATE TABLE table_name;
```

```
TRUNCATE TABLE Student_Info;
```

ALTER TABLE

- Used to add, delete, or modify columns in an existing table
- Syntax for adding a new column:

```
ALTER TABLE table_name
ADD column_name datatype;
```

```
ALTER TABLE table_name
ADD (column1_name datatype,
     column2_name datatype,
     ……);
```

```
ALTER TABLE Student_Info
ADD (Email varchar(50),
     Phone int);
```

Enrolment_No	First_Name	Last_Name	Address	Branch	Email	Phone

INSERT INTO Statement

- Used to insert records into table.
- Multiple records can be inserted using a single query.
- INSERT statement can be written in two ways:

Specifying both column names and values

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

If adding values for all the columns of the table, column names can be omitted. But the order of the values should be the same as the order of columns in the table

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

SELECT Statement

- Used to retrieve data from table(s)
- Syntax:

```
SELECT column1, column2, ... FROM table_name;
```

- If data is to be retrieved from all the columns of a table

```
SELECT * FROM table_name;
```

- A column may contain duplicate values. To retrieve only distinct (different) values

```
SELECT DISTINCT column1, column2, ... FROM table_name;
```

WHERE Clause

- Used to filter records and retrieve only those records that fulfill the specified condition
- Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

- Following operators can be used in WHERE clause:

= , > , < , >= , <= , <> or != , BETWEEN , LIKE , IN

Questions

Q1. Given the following table designs, write the MySQL queries for making these tables.

Bus (BusNo: String, Source: String, Destination: String, CoachType: String, NoOfSeats: Integer)
Seat (SeatNo: Integer, BusNo: String, Status: Character)
Ticket (TicketNo: String, PassengerID: String, Booking Date: Date, Journey Date: Date, BusNo: String, SeatNo: Integer, Source: String, Destination: String)
Passenger (PassengerID: String, Name: String, ContactNo: String, Age: Integer, Gender: Character, Address: String)
Cancellation (TicketNo: String, Cancelled On: Date)

Q2. Perform the following operations on the tables created, after populating them with data.

a) Display names of all senior citizens.
b) Display the ticket numbers and names of all male passengers travelling from Delhi to Varanasi.
c) Find the ticket numbers of all passengers whose name start with ‘r’ and ends with ‘h’.

d) Find the names of passengers whose age is between 30 and 45.
e) Display all passenger names beginning with 'A'.
f) Display all passengers having more than one tickets.
g) Display all ticket numbers whose date of journey is in November 2022.
h) Display the total number of buses for each Coach Type.
i) Display names of all passengers who cancelled tickets between May 2022 and July 2022.
j) Display all ticket numbers that have a gap of six months between booking date and date of journey.

Lab 10

Database Connectivity with PHP

PHP provides **mysqli** contruct or **mysqli_connect()** function to open a database connection. This function takes six parameters and returns a MySQL link identifier on success or FALSE on failure.

```
$mysqli = new mysqli($host, $username, $passwd, $dbName, $port, $socket)
```

$host: (Optional) The host name running the database server. If not specified, then the default value will be **localhost:3306**.

$username: (Optional) The username accessing the database. If not specified, then the default will be the name of the user that owns the server process.

$passwd: (Optional) The password of the user accessing the database. If not specified, then the default will be an empty password.

$dbName: (Optional) database name on which query is to be performed.

$port: (Optional) the port number to attempt to connect to the MySQL server.

$socket: (Optional) socket or named pipe that should be used.

You can disconnect from the MySQL database anytime using another PHP function **close()**.

```
$mysqli->close();
```

Try the following example to connect to a MySQL server: Copy and paste the following code as *mysql_example.php*

```
<html>
  <head>
    <title>Connecting MySQL Server</title>
  </head>
  <body>
    <?php
          $dbhost = 'localhost';
          $dbuser = 'root';
          $dbpass = 'root@123';   // verify database password as set in your lab
          $conn = mysqli_connect($dbhost, $dbuser, $dbpass);
          if(! $conn )
                  die('Connection Failed: ' . mysqli_connect_error ());
          else echo 'Connection Successful';
          mysqli_close($conn);
    ?>
  </body>
```

</html>

Executing query using PHP Script: Pass the SQL query statement to the mysqli_query() function to execute the command

Copy and paste the following code as *mysql_example.php* to create a database "*mydb*"

```
<?php
        $dbhost = 'localhost';
        $dbuser = 'root';
        $dbpass = 'root@123';   // verify database password as set in your lab
        $conn = mysqli_connect($dbhost, $dbuser, $dbpass);
        if(! $conn )
                die('Connection Failed: ' . mysqli_connect_error ());
        else echo 'Connection Successful';
        mysqli_close($conn);

        // Create database
        $sql = "CREATE DATABASE mydb";
        if (mysqli_query($conn, $sql)) {
          echo "Database created successfully";
        } else {
          echo "Error creating database: " . mysqli_error($conn);
        }

        mysqli_close($conn);
?>
```

Questions

Create a dynamic web application in PHP and it should have following functionalities:

1. The application name is "First_PHPApp_<your roll number>" i.e. folder name inside htdocs folder

2. Create a home page index.php. It should contain:
    a. A welcome message
    b. Hyperlink for login.php (for registered user) and registration.php (for new user)

3. Create a page registration.php and it should contain:
    a. A form with basic details emp_id, password (constraints: length 6 digits to 50 digits with one lower case character, one upper case character, one number and one special character), Employee Name, Department, mobile_no (10 digit constraint and starting with 9,8,7,6,5), email_id (proper email format validation), Salary (numeric value with two decimal points), DoB (calendar should appear to select DoB), gender (radio button male, female, other) and Address.

   b. Two buttons: Submit and reset (or clear)
   c. Hyperlink for login.php and index.php pages

4. Create a page login.php and it should contain:
   a. Two fields for emp_id and password
   b. Two buttons: Submit and reset
   c. Hyperlink for registration.php, index.php and forget_password.php pages
   d. Don't create any page forget_password.php and redirect on login.php itself with # symbol

5. Create a table employee in the database and make constraints as per registration.php (Create tables inside the database manually, no need to create a separate web page for table creation)
   a. emp_id should be a primary key and gender should accept only male, female and others
   b. Make all other constraints in the table as per your intelligence

6. Write PHP code inside the php files to fulfill the requirements of database connectivity, data insertion and data retrieval after submission the login.php and registration.php.

7. On submission of correct login details, page should redirect to welcome.php. It should:
   a. Print "Welcome <employee name>!" at the center of the page
   b. Print all employee details in the tabular form. Table rows should be dynamically created as per the number of rows in the database.
   c. Contain Logout button. On clicking that button, the page should be redirected to the login page.

Lab 11

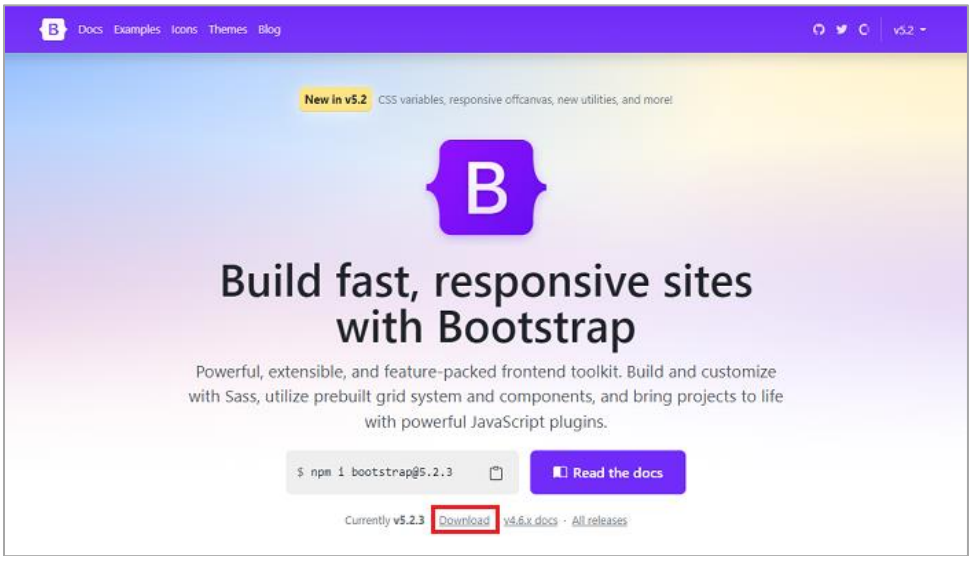
Web Development Framework Bootstrap

Bootstrap - Environment Setup

It is very easy to setup and start using Bootstrap. Here we will explain how to download and setup Bootstrap. We will also discuss the Bootstrap file structure, and demonstrate its usage with an example.

Download Bootstrap

You can download the latest version of Bootstrap from https://getbootstrap.com/. When you click on this link, you will get to see a screen as below −



On clicking *Download* link you will get following options

- ***Download Bootstrap*** − Clicking this, you can download the precompiled and minified versions of Bootstrap CSS, JavaScript, and fonts. No documentation or original source code files are included.
- ***Download Source*** − Clicking this, you can get the latest Bootstrap LESS and JavaScript source code directly from GitHub.
- ***Use Bootstrap directly from CDN*** - Skip the download to use cached version of Bootstrap's compiled CSS and JS to your project. Copy-paste the stylesheet <link> inside <head> before all other stylesheets.

If you work with Bootstrap's uncompiled source code, you need to compile the LESS files to produce usable CSS files. For compiling LESS files into CSS, Bootstrap officially supports only Recess, which is Twitter's CSS hinter based on less.js. (For more information visit: https://getbootstrap.com/docs/5.2/getting-started/download/)

HTML Template: A basic HTML template using Bootstrap would look like this −

```
<!DOCTYPE html>
<html>
<head>
        <!-- Required meta tags -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <!-- Bootstrap CSS -->
        <link href="link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css"
        rel="stylesheet" >
        <title>Hello, world!</title>
</head>

<body>
        <h1>Hello, world!</h1>
        <!-- Optional JavaScript - jQuery first, then Bootstrap JS -->
        <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" ></script>
        <script
        src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Here you can see the **jquery.js**, **bootstrap.min.js** and **bootstrap.min.css** files that are included to make a normal HTML file to the Bootstrapped Template. Many Bootstrap components require jQuery and some JavaScript plugins. Place the <script>s right before the closing </body> tag. jQuery must come first, and then the JavaScript plugins.

Bootstrap Grid System

Bootstrap is mobile first in the sense that the code for Bootstrap now starts by targeting smaller screens like mobile devices, tablets, and then "expands" components and grids for larger screens such as laptops, desktops.

Mobile First Strategy

- **Content**
  - Determine what is most important.
- **Layout**
  - Design to smaller widths first.
  - Base CSS address mobile device first; media queries address for tablet, desktops.
- **Progressive Enhancement**
  - Add elements as screen size increases.

Working of Bootstrap Grid System

Grid systems are used for creating page layouts through a series of containers, rows and columns that house your content. Here's how the Bootstrap grid system works −

- Rows must be placed within a **.container or .container-fluid** class for proper alignment and padding.
- Use rows to create horizontal groups of columns.
- Content should be placed within the columns, and only columns may be the immediate children of rows.
- Predefined grid classes like. row and. col-xs-4 are available for quickly making grid layouts. LESS mixins can also be used for more semantic layouts.
- Columns create gutters (gaps between column content) via padding. That padding is offset in rows for the first and the last column via negative margin on. row.
- Grid columns are created by specifying the number of twelve available columns you wish to span. For example, three equal columns would use three. col-xs-4 class.

Questions

Q1. Install Bootstrap

Q2. Design a simple website of your choice in Bootstrap using your imagination. Explore the usage of the various form controls and components provided by Bootstrap.

Lab 12

Web Development Framework Django

Django is a Python web framework and Django supports the MVC pattern. First let's see what is the Model-View-Controller (MVC) pattern, and then we will look at Django's specificity for the Model-View-Template (MVT) pattern.

MVC Pattern

When talking about applications that provides UI (web or desktop), we usually talk about MVC architecture. And as the name suggests, MVC pattern is based on three components: Model, View, and Controller.

DJANGO MVC - MVT Pattern

The Model-View-Template (MVT) is slightly different from MVC. In fact, the main difference between the two patterns is that Django itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template. The template is a HTML file mixed with Django Template Language (DTL).

Creating a Project

Now that we have installed Django, let's start using it. In Django, every web app you want to create is called a project; and a project is a sum of applications. An application is a set of code files relying on the MVT pattern. As example let's say we want to build a website, the website is our project and, the forum, news, contact engine are applications. This structure makes it easier to move an application between projects since every application is independent.

Create a Project: Whether you are on Windows or Linux, just get a terminal or a *cmd* prompt and navigate to the place you want your project to be created, then use this code −

```
$ django-admin startproject myproject
```

This will create a "myproject" folder with the following structure −

```
myproject/
  manage.py
  myproject/
    __init__.py
    settings.py
    urls.py
    wsgi.py
```

The Project Structure: The "*myproject*" folder is just your project container, it actually contains two elements −

- *manage.py* − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

  $ python manage.py help

- "*myproject*" subfolder − This folder is the actual python package of your project. It contains four files −

  - *__init__.py* − Just for python, treat this folder as package.
  - *settings.py* − As the name indicates, your project settings.
  - *urls.py* − All links of your project and the function to call. A kind of ToC of your project.
  - *wsgi.py* − If you need to deploy your project over WSGI.

Questions

Q1. Perform the installation of Django

Q2. Design a simple hello world app in Django