**KLS’s Gogte Institute of Technology**

**Department of Information Science and Engineering**

**Lab Writeup**

**WEB TECHNOLOGIES LAB** (21IS34)

**Term Work -1.**

**Term Work 1:**

**Build a simple webpage of images with Figma login for designers.**

**Theory Related to the Experiment:**

Figma is a collaborative interface design tool that’s taking the design world by storm. Unlike Sketch, which runs as a standalone MacOS app, Figma is entirely browser-based, and therefore works not only on Macs, but also on PCs running Windows or Linux, and even on Chromebooks. It also offers a web API, and it’s free!

Another big advantage of Figma is that it allows real-time collaboration on the same file. When using conventional “offline” apps like Sketch and Photoshop, if designers want to share their work, they typically have to export it to an image file, then send it via email or instant message.

In Figma, instead of exporting static images, we can simply share a link to the Figma file for clients and colleagues to open in their browser. This in itself saves significant time and inconvenience in a designer’s workflow. But more importantly, it means that clients and colleagues can interact more richly with the work, and review the latest version of the file.

**Steps for execution of Figma app online:**

**Step 1:** Setting the page template

1. **Drag the Page Template** from the Page Templates components library (left side of your screen) into your Figma project.
2. **Select the page** by clicking on it. You should see a purple outline around the page.
3. **Detach the page component** from his instance by right-clicking directly on the page or the page's layers), and choosing the Detach Instance option
4. **Rename the page** by double-clicking the layer's name directly from the layers panel (right side of your screen).

**Step 2:** Creating a form

1. **Add the image** from the File components library (left side of your screen) into your Figma project.
2. **Select the image** by clicking on it. You should see a purple outline around the component.
3. **Customize the look and content** of the image with the options you need from the variants control panel (right side of your screen).
4. **Adjust the width of the image** by changing the value from "fixed width" to "fill container" in the Resizing panel (right side of your screen)
5. **Edit the image.** To do so, you could click on the label until the text becomes highlighted or select the label layer from the layers panel (left side of the screen). Once selected, use your keyboard to enter new text.
6. **Make the Assistive label invisible** if you are not using it.

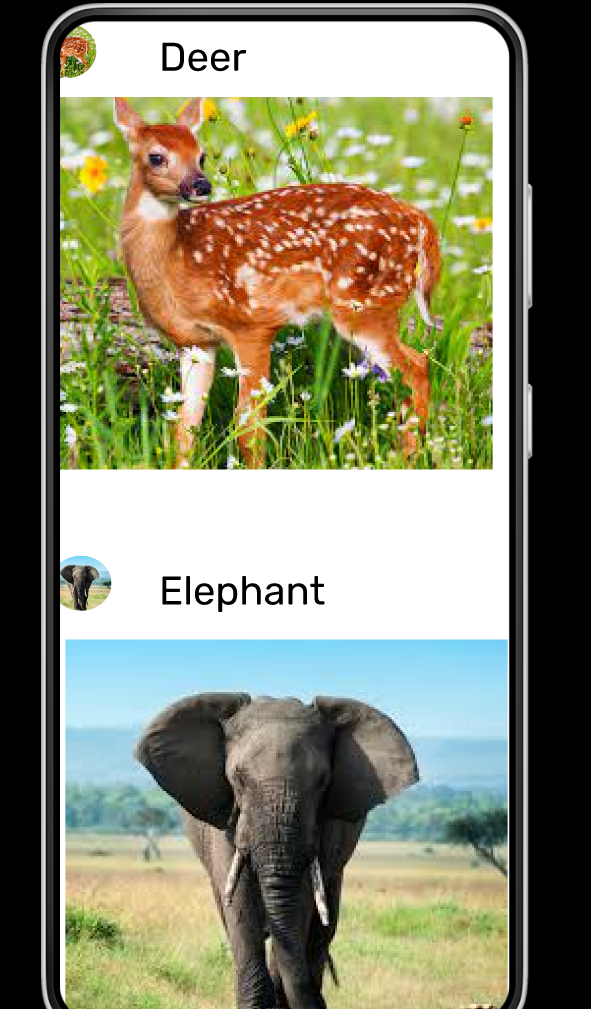
**Step 3:** Adjusting the layout

1. **Select the layer group** from the layer panel on the left side of the screen.
2. **Go to the Auto layout panel** on the right side of the screen. Change the value of the spacing between elements.
3. **Application of the experiment performed:**

**Application of the experiment performed:**

**Output and Observation:**

**(Sample Output add the required output )**



**Marks:**

|  |  |  |
| --- | --- | --- |
|  | **Maximum Marks** | **Marks Scored** |
| Conducting the experiment | 5 |  |
| Calculations, results, graph, conclusion, and outcome | 5 |  |
| Viva voce | 5 |  |
|  | **15** |  |
| Checked on |  |  |
| **Signature of the faculty member** |  |  |

**Term Work -2.**

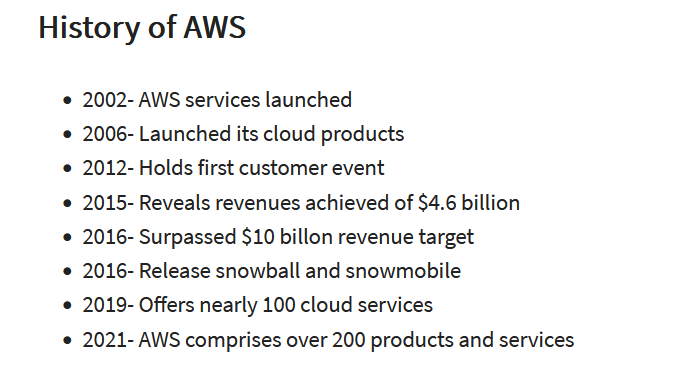
**Term Work 2:**

**Build a simple responsive website that has the basic Web controls.**

**.**

**Theory Related to the Experiment:**

* **The full form of AWS is Amazon Web Services. It is a platform that offers flexible, reliable, scalable, easy-to-use and, cost-effective cloud computing solutions.**
* **AWS is a comprehensive, easy to use computing platform offered Amazon. The platform is developed with a combination of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings.**

****

**Important AWS Services:**

* **Amazon Web Services offers a wide range of different business purpose global cloud-based products. The products include storage, databases, analytics, networking, mobile, development tools, enterprise applications, with a pay-as-you-go pricing model.**

****

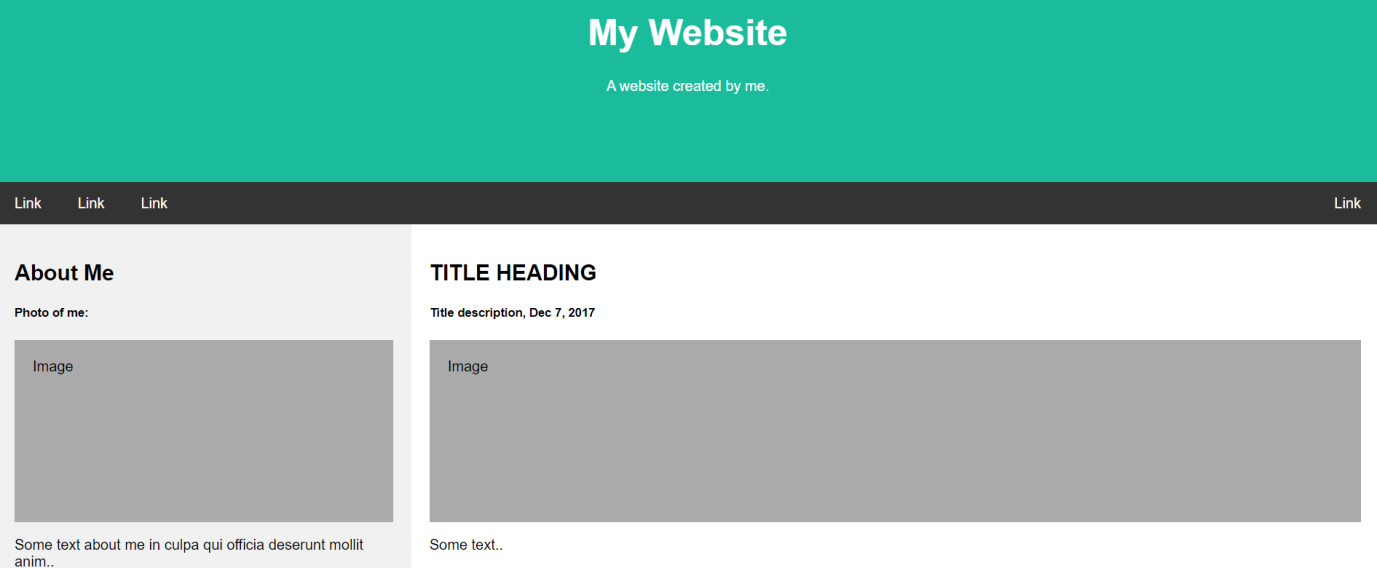
**Amazon Web services Applications Category:**

* **Application Services**
* **Deployment and Management**
* **Developer Tools**
* **Mobile Services**
* **Business Productivity**
* **Desktop & App Streaming**
* **Artificial Intelligence**
* **AR & VR (Augmented Reality & Virtual Reality)**
* **Customer Engagement**

**Application of the experiment performed:**

**Output and Observation:**

**(Sample Output add the required output ):**

****

**Marks:**

|  |  |  |
| --- | --- | --- |
|  | **Maximum Marks** | **Marks Scored** |
| Conducting the experiment | 5 |  |
| Calculations, results, graph, conclusion, and outcome | 5 |  |
| Viva voce | 5 |  |
|  | **15** |  |
| Checked on |  |  |
| **Signature of the faculty member** |  |  |

**Term Work 3.**

**Create multi column article using HTML tags. Integrate social**

**sharing feature.**

**Theory Related to the Experiment:**

Multiple Columns

The CSS3 multi-column layout allows easy definition of multiple columns of text - just like in newspapers:

The numbers in the table specify the first browser version that fully supports the property.

Numbers followed by -webkit- or -moz- specify the first version that worked with a prefix.

CSS3 Multi-column Properties:

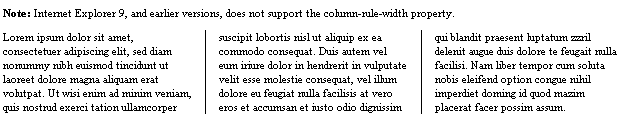
* column-count
* column-gap
* column-rule-style
* column-rule-width
* column-rule-color
* column-rule
* column-span
* column-width

The column-count property specifies the number of columns an element should be divided into.

div {  
    -webkit-column-count: 3; /\* Chrome, Safari, Opera \*/  
    -moz-column-count: 3; /\* Firefox \*/  
    column-count: 3;  
}

The column-rule-width property specifies the width of the rule between columns:

div {  
    -webkit-column-rule-width: 1px; /\* Chrome, Safari, Opera \*/  
    -moz-column-rule-width: 1px; /\* Firefox \*/  
    column-rule-width: 1px;  
}



5 Ways To Integrate Social Media On Your Website:

**Social share and follow buttons**

Make your content shareable with [social share buttons](https://shareaholic.com/). Not only do they help you increase awareness of your content, but when you provide your visitors the opportunity to share your content easily, you’re also improving [user experience](https://blog.hootsuite.com/online-user-experience).

**Social login**  
Did you know that [73%](http://www.loginradius.com/social-login) of users prefer to log in to a site with social login, as opposed to providing an email address and creating a new account. Improve your website visitors’ experience with social login, and increase your website registration conversions and retention.

**Social Video**   
[Social videos](https://blog.hootsuite.com/how-to-use-social-video-for-marketing/) are videos that are created and shared on social networks. The added bonus of social videos for marketers is that they provide an easy social media integration for your website. With so many social media platforms including video in the mix, it would be a missed opportunity to not include your social videos on your website too.

**Instagram photos**This social media integration feature is perfect for those of you who want to showcase your Instagram photos on your website. If you’re also trying to increase your [Instagram following](https://blog.hootsuite.com/8-ways-get-likes-followers-instagram/), embedding your Instagram photos in blog posts allows you to capture new followers from your website visitors.

**Social Proof**Simply put, when you integrate [social proof](https://blog.hootsuite.com/what-is-social-proof/) to your website you’re providing your visitors the opportunity to trust you more. With [79% of consumers trusting social proof as much as personal recommendations](http://searchengineland.com/2013-study-79-of-consumers-trust-online-reviews-as-much-as-personal-recommendations-164565), it’s important you integrate the proper social widgets on your website to increase sales and website conversions.

**Pseudocode:**

<!DOCTYPE html>

<html>

<head>

<style>

// add all the required CSS classes

.newspaper {

-webkit-column-count: 3; /\* Chrome, Safari, Opera \*/

-moz-column-count: 3; /\* Firefox \*/

column-count: 3;

}

</style>

<body>

<div class="newspaper">

Gogte Institute of Technology, the flagship Institute of Karnatak Law Society, Belagavi was incepted in 1979 </div>

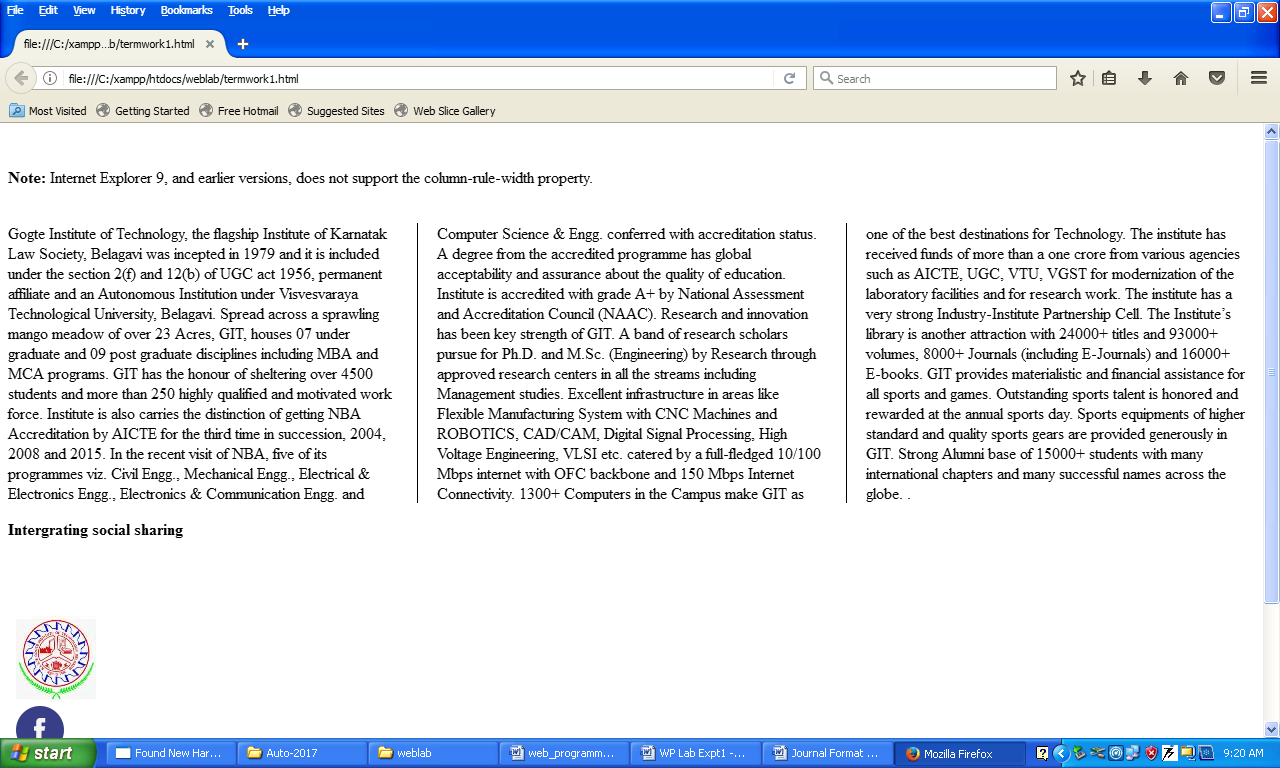
</body>

</html>

**Application of the experiment performed:**

### Output and Observation:

(sample)

****

**Marks:**

|  |  |  |
| --- | --- | --- |
|  | **Maximum Marks** | **Marks Scored** |
| Conducting the experiment | 5 |  |
| Calculations, results, graph, conclusion, and outcome | 5 |  |
| Viva voce | 5 |  |
|  | **15** |  |
| Checked on |  |  |
| **Signature of the faculty member** |  |  |

**Term Work 4.**

**Creating simple transformations using CSS.**

**Example: square to circle.**

**Theory Related to the Experiment:**

**CSS(cascading Style Sheets):**

It’s as easy as 1-2-3!

1. Start with a document that has been marked up in HTML.

2. Write style rules for how you’d like certain elements to look.

3. Attach the style rules to the document. When the browser displays the document, it follows your rules for rendering elements.

**Writing the rules :**

<html><head><title></title>

<style type="text/css">

h1 {

color: green;

}

p {

font-size: small;

font-family: sans-serif;

}

</style>

</head>

<body>

<p> Hello</P>

</body>

</html>

**Transitions, Transforms and Animation:**

* A state change such as :hover, :focus, or :active makes a good trigger.
* The transition properties are applied to the a element in its normal state.

The markup

<a href="" class="smooth">awesomes</a>

**transition-property:** Transition-property identifies the CSS property that we want to transition smoothly. In our example, it’s the background-color. You can also change the foreground color, borders, dimensions, font- and text-related attributes, and many more.

**transition-duration:**

Transition-duration sets the amount of time it will take for the animation to complete in seconds (s) or milliseconds (ms). Chosen .3 seconds, which is just enough to notice something happened but not so long that the transition feels sluggish or slows the user down. There is no correct duration, of course, but in my travels I’ve found that .2s seems to be a popular transition time for UI elements.

**CSS Transforms:**

***Values: transform function(s) | none***

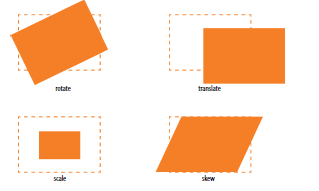
***Default: none***

***Applies to: transformable elements***

***Inherits: no***

The CSS3 Transforms module gives authors a way to rotate, relocate, resize, and skew HTML elements in both two- and three-dimensional space.You can apply a transform to the normal state of an element and it will appear in its transformed state when the page loads.

Figure 1 shows a representation of four types of two dimentional transforms: rotate, translate, scale and skew

****

**Transforming the position (translate)**

transform: translateX(50px);

transform: translateY(25px);

transform: translate(50px, 25px);

Transforming the size (scale)

a img {

transform: scaleX(1.5)}

**Keyframe Animation:**

The animation process has two parts: first, establish the keyframes with a @keyframes rule, and then add animation properties to the elements that will be animated.

@keyframes colors {

0% { background-color: red; }

20% { background-color: orange; }}

**Procedure for Execution :**

Step 1. Installation of XXamp software.

Step 2. Create a new html file in xxamp/htdocs folder.

Step 3. Use of all the required tags in html page.

Step 4. Interpret the result and display it in Web browser.

**Pseudocode:**

<!DOCTYPE html>

<html>

<head><style>

@keyframes square-to-circle {

0% {

border-radius:0 0 0 0; background:coral;

transform-origin: center, right; transform:rotate(0deg);

}

// add all the styles

</style>

</head>

<body>

<div>

<pre>

Square to Circle

</pre>

</div>

</body>

</html>

**Application of the experiment performed:**

**Output and Observation:**

(Sample)



**Marks:**

|  |  |  |
| --- | --- | --- |
|  | **Maximum Marks** | **Marks Scored** |
| Conducting the experiment | 5 |  |
| Calculations, results, graph, conclusion, and outcome | 5 |  |
| Viva voce | 5 |  |
|  | **15** |  |
| Checked on |  |  |
| **Signature of the faculty member** |  |  |

**Term Work 5**

**To create a HTML registration form and validate the form using JavaScript code with appropriate event handlers.**

**Theory Related to the Experiment:**

**Scrolling menus:**

To make the menu display as a scrolling list, simply specify the number of lines you’d like to be visible using the size attribute.

The multiple attribute allows users to make more than one selection from the scrolling list. Note that pull-down menus do not allow multiple selections; when the browser detects the multiple attribute, it displays a small scrolling menu automatically by default.

Use the selected attribute in an option element to make it the default value for the menu control.

<p>What 80s bands did you listen to?

<select name="EightiesBands" size="6" multiple>

<option>The Cure</option>

<option>Cocteau Twins</option>

<option selected>Tears for Fears</option>

<option selected>Thompson Twins</option>

<option value="EBTG">Everything But the Girl</option>

<option>Depeche Mode</option>

<option>The Smiths</option>

<option>New Order</option>

</select>

</p>

**Buttons:**

<input type="button" value="Roll Dice" onclick="checkItem()"><br>

Creates a button

**Psuedocode:**

<!DOCTYPE html>

<html><head>

<title>JavaScript Form Validation using a sample registration form</title>

<link rel='stylesheet' href='cssvalidation.css' type='text/css' />

<script src="jsvalidation.js"></script>

</head>

<body onload="document.registration.userid.focus();">

<form name='registration' onSubmit="return formValidation();">

**// all the required controls of HTML**

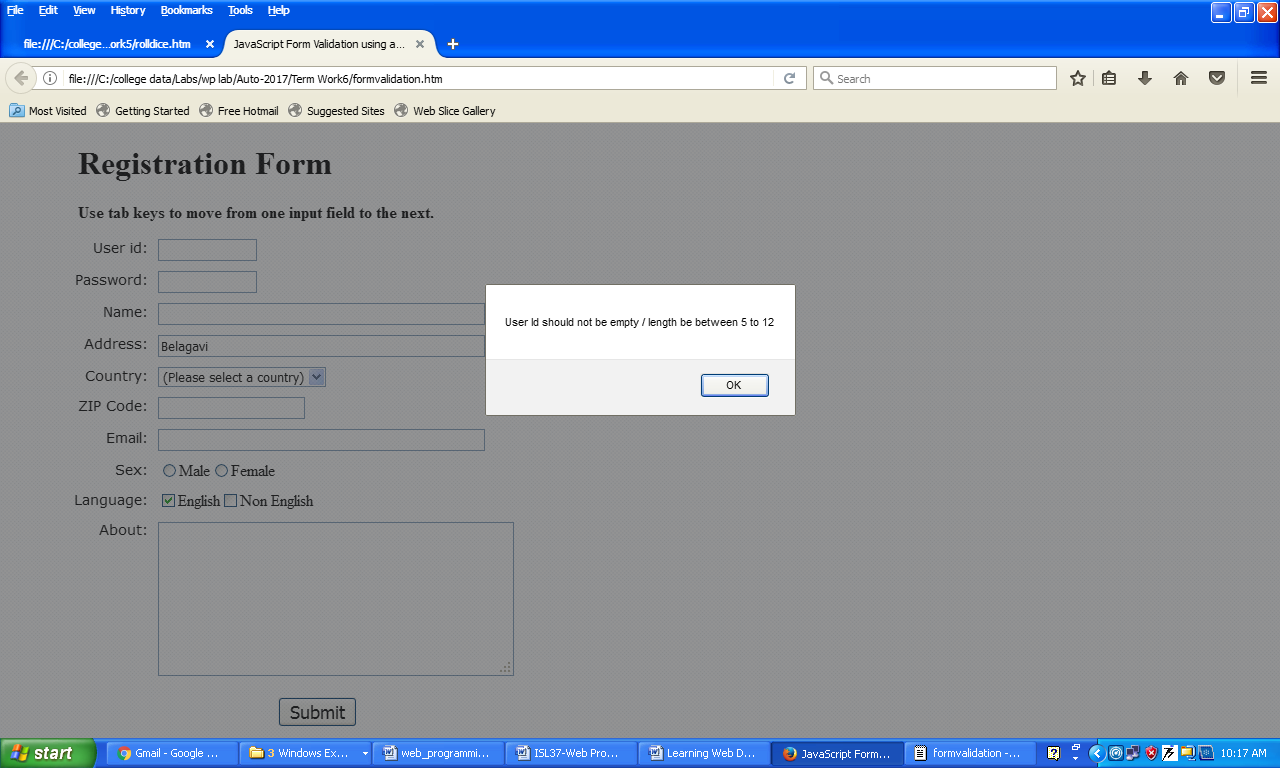
**//** jsvalidation.js contains the javascript code for validation

</body>

</html>

**Application of the experiment performed:**

**Output and Observation (Sample):**

****

**Marks:**

|  |  |  |
| --- | --- | --- |
|  | **Maximum Marks** | **Marks Scored** |
| Conducting the experiment | 5 |  |
| Calculations, results, graph, conclusion, and outcome | 5 |  |
| Viva voce | 5 |  |
|  | **15** |  |
| Checked on |  |  |
| **Signature of the faculty member** |  |  |

**Termwork-6**

**To Create HTML grid using tailwind CSS**

**Theory related :**

Tailwind Elements is a plugin that extends the functionality of the library with many interactive components.

In some dynamic components (like dropdowns or modals) we add Font Awesome icons and custom JavaScript. However, they do not require any additional installation, all the necessary code is always included in the example and copied to any Tailwind project - it will work.

**Steps:**

NPM is a recommended way to work with Tailwind Elements and others libraries.

Make sure that you have Node.js installed. In order to verify that, open a command line and type node -v. If node is installed you should see it's version in a console.

1. Before starting the project make sure to install [Node.js (LTS)](https://nodejs.org/en/) and [TailwindCSS](https://tailwindcss.com/).

2. Run the following command to install the package via NPM:

3. Tailwind Elements is a plugin and should be included inside the tailwind.config.js file. It is also recommended to extend the content array with a js file that loads dynamic component classes:

4. Dynamic components will work after adding the js file:

Pseudocode:

**<body>**

**<h1 class="text-3xl font-bold underline text-clifford">**

**Hello world!**

**</h1>**

**<!-- Full width column -->**

**<div class="flex mb-4">**

**<div class="w-full bg-gray-300 h-12"></div>**

**</div>**

**<!-- Two columns -->**

**<div class="flex mb-4">**

**<div class="w-1/2 bg-gray-500 h-12"></div>**

**<div class="w-1/2 bg-gray-300 h-12"></div>**

**</div>**

**<!-- Three columns -->**

**<div class="flex mb-4">**

**<div class="w-1/3 bg-gray-500 h-12"></div>**

**<div class="w-1/3 bg-gray-300 h-12"></div>**

**<div class="w-1/3 bg-gray-500 h-12"></div>**

**</div>**

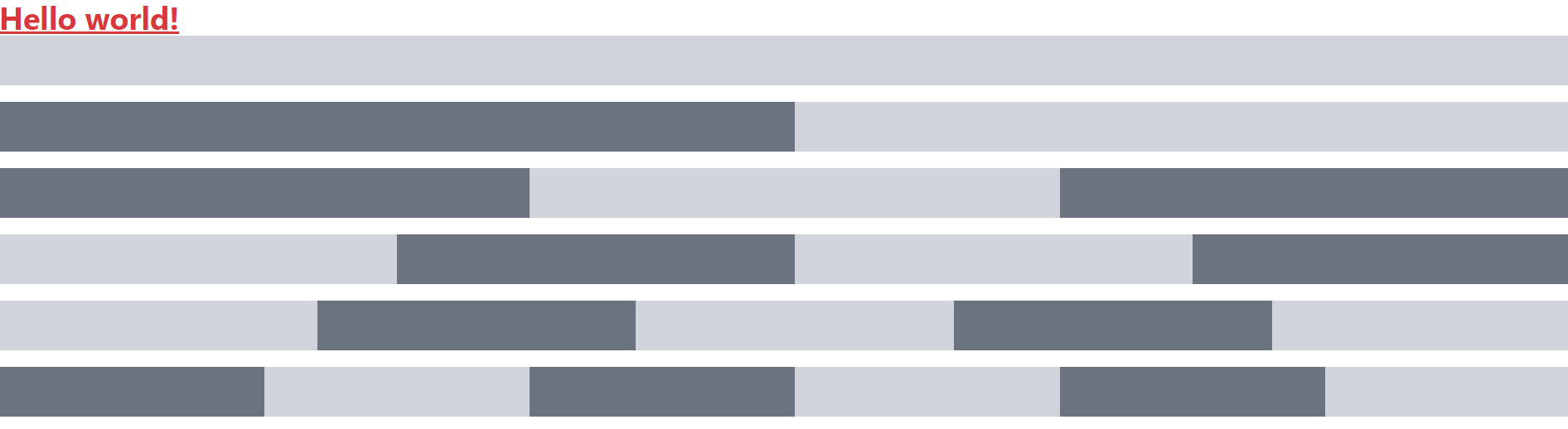
**</body>**

**</html>**

**Application of the experiment performed:**

**Output and Observation:**

**(sample)**

****

**Marks:**

|  |  |  |
| --- | --- | --- |
|  | **Maximum Marks** | **Marks Scored** |
| Conducting the experiment | 5 |  |
| Calculations, results, graph, conclusion, and outcome | 5 |  |
| Viva voce | 5 |  |
|  | **15** |  |
| Checked on |  |  |
| **Signature of the faculty member** |  |  |

**TermWork-7:**

To create a Bootstrap Navigational Bar using color schemes.

### It provides solutions for faster and easier web development.

### It's powerful grid system allows you to create fluid and responsive design very easily.

### It comes with typography and custom UI elements.

### It provides easy to use dynamic UI components like tabs and accordion.

### It is highly compatible across the browser.

### It is fully customizable.

### Psuedocode:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<div class="navbar navbar-fixed-top navbar-inverse">

<div class="navbar-inner">

<div class="container">

<a href="#" class="brand">Gadget Store</a>

<div class="nav-collapse collapse navbar-responsive-collapse">

<ul class="nav">

<li><a href="#">Home</a></li>

<li><a href="#">Services</a></li>

<li class="dropdown">

<a href="#" data-toggle="dropdown" class="dropdown-toggle">Products <b class="caret"></b></a>

<ul class="dropdown-menu">

<li><a href="#">Gadgets</a></li>

<li><a href="#">Accessories</a></li>

<li class="divider"></li>

<li><a href="#">More &raquo;</a></li>

</ul></li></ul>

<form action="#" class="navbar-search pull-left">

<input type="text" placeholder="Search &hellip;" class="search-query">

</form>

<ul class="nav pull-right">

<li><a href="#myModal" data-toggle="modal">Login</a></li>

<li class="divider-vertical"></li>

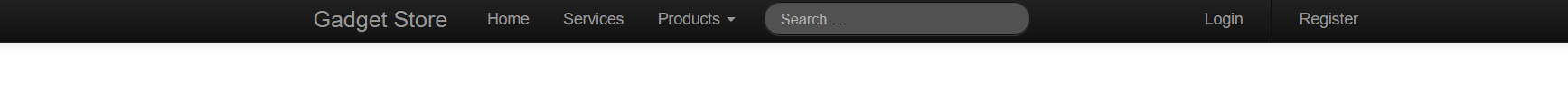
<li><a href="#">Register</a></li>

</ul>

</div></div></div></div>

**Application of the experiment performed:**

**Output and Observation: (sample)**

****

**Marks:**

|  |  |  |
| --- | --- | --- |
|  | **Maximum Marks** | **Marks Scored** |
| Conducting the experiment | 5 |  |
| Calculations, results, graph, conclusion, and outcome | 5 |  |
| Viva voce | 5 |  |
|  | **15** |  |
| Checked on |  |  |
| **Signature of the faculty member** |  |  |

Termwork-8:

web design which automatically creates buttons with different colors.

Theory related:

Bootstrap 4 Jumbotron

A jumbotron indicates a big grey box for calling extra attention to some special content or information.

**Tip:** Inside a jumbotron you can put nearly any valid HTML, including other Bootstrap elements/classes.

Jumbotron is a full width [card](https://mdbootstrap.com/docs/standard/components/cards/) that is usually displayed at the top of the page, sometimes also referred to as a "Hero component". Jumbotron with [rounded corners](https://mdbootstrap.com/docs/standard/utilities/borders/), [vertically centered](https://mdbootstrap.com/docs/standard/layout/vertical-alignment/) content, and a call to action [button](https://mdbootstrap.com/docs/standard/components/buttons/) is the most common component found on [landing pages](https://mdbootstrap.com/docs/standard/templates/landing-page/). The [jumbotron color](https://mdbootstrap.com/docs/standard/content-styles/colors/) should be only slightly off the background of the page to make your design feel "light".

Pseudocode:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css">

<script src="https://cdn.jsdelivr.net/npm/jquery@3.6.1/dist/jquery.slim.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/js/bootstrap.bundle.min.js"></script>

</head>

<body>

<div class="container">

<h2>Button Styles</h2>

<button type="button" class="btn">Basic</button>

<button type="button" class="btn btn-primary">Primary</button>

<button type="button" class="btn btn-secondary">Secondary</button>

<button type="button" class="btn btn-success">Success</button>

<button type="button" class="btn btn-info">Info</button>

<button type="button" class="btn btn-warning">Warning</button>

<button type="button" class="btn btn-danger">Danger</button>

<button type="button" class="btn btn-dark">Dark</button>

<button type="button" class="btn btn-light">Light</button>

<button type="button" class="btn btn-link">Link</button>

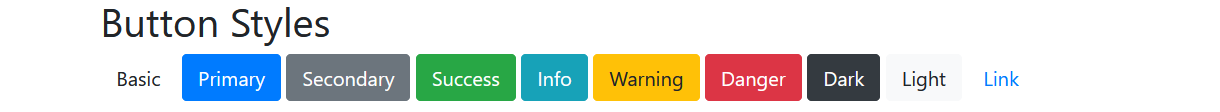
</div>

</body>

</html>

**Application of the experiment performed:**

**Output and Observation: (sample)**

****

**Marks:**

|  |  |  |
| --- | --- | --- |
|  | **Maximum Marks** | **Marks Scored** |
| Conducting the experiment | 5 |  |
| Calculations, results, graph, conclusion, and outcome | 5 |  |
| Viva voce | 5 |  |
|  | **15** |  |
| Checked on |  |  |
| **Signature of the faculty member** |  |  |

**TermWok-9**

**Locate the Users Position.**

**Theory:**

## Locate the User's Position

The HTML Geolocation API is used to get the geographical position of a user.

Since this can compromise privacy, the position is not available unless the user approves it.

## Using the Geolocation API

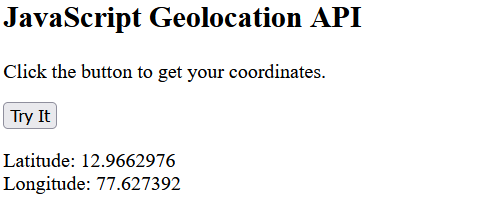
The getCurrentPosition() method is used to return the user's position.

Psuedo Code:

<script>  
const x = document.getElementById("demo");  
function getLocation() {  
  if (navigator.geolocation) {  
    navigator.geolocation.getCurrentPosition(showPosition);  
  } else {  
    x.innerHTML = "Geolocation is not supported by this browser.";  
  }  
}  
function showPosition(position) {  
  x.innerHTML = "Latitude: " + position.coords.latitude +   
  "<br>Longitude: " + position.coords.longitude;   
}  
</script>

**Application of the experiment performed:**

**Output and Observation: (sample)**



**Marks:**

|  |  |  |
| --- | --- | --- |
|  | **Maximum Marks** | **Marks Scored** |
| Conducting the experiment | 5 |  |
| Calculations, results, graph, conclusion, and outcome | 5 |  |
| Viva voce | 5 |  |
|  | **15** |  |
| Checked on |  |  |
| **Signature of the faculty member** |  |  |

**TermWork-10:**

Demonstrate how a web page can fetch information from an XML file with AJAX:

Theory:

When a user clicks on the "Get CD info" button above, the loadDoc() function is executed.

The loadDoc() function creates an XMLHttpRequest object, adds the function to be executed when the server response is ready, and sends the request off to the server.

When the server response is ready, an HTML table is built, nodes (elements) are extracted from the XML file, and it finally updates the element "demo" with the HTML table filled with XML data:

**Psuedocode:**

function loadDoc() {  
  var xhttp = new XMLHttpRequest();  
  xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
    myFunction(this);  
    }  
  };  
  xhttp.open("GET", "cd\_catalog.xml", true);  
  xhttp.send();  
}  
function myFunction(xml) {  
  var i;  
  var xmlDoc = xml.responseXML;  
  var table="<tr><th>Title</th><th>Artist</th></tr>";  
  var x = xmlDoc.getElementsByTagName("CD");  
  for (i = 0; i <x.length; i++) {   
    table += "<tr><td>" +  
    x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +  
    "</td><td>" +  
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +  
    "</td></tr>";  
  }  
  document.getElementById("demo").innerHTML = table;  
}

**Application of the experiment performed:**

**Output and Observation: (sample)**

****

****

**Marks:**

|  |  |  |
| --- | --- | --- |
|  | **Maximum Marks** | **Marks Scored** |
| Conducting the experiment | 5 |  |
| Calculations, results, graph, conclusion, and outcome | 5 |  |
| Viva voce | 5 |  |
|  | **15** |  |
| Checked on |  |  |
| **Signature of the faculty member** |  |  |