CYBAGE
Delivering Value. Scientifically.

Contents

Document name	**FET Booklet**
Version no.	*1.0*
Release date	**26-06-2018**
Classification	Departmental

FET Booklet

Document History

Ver. No	Release Date	Created By / Modified By and Date	Reviewed By and Date	Approved By and Date	Remarks and Changes Made
0.1	26-06-2018	Asfiya Khan Snehal Pawar	Yogesh Gaikwad	Yogesh Gaikwad and Kirti Mahadik	Initial draft.

Notes for assignments:

- Make assumptions wherever necessary, and mention the same against the question.
- Use Exception Handling wherever required.
- Paste the solution application on the share folder (information will be provided during training).
- Submit the assignmnets on time.

Contents

HTML5	4
CSS3	7
JavaScript	12
Jquery	33
SASS	34
RWD	40
Bootstrap	41
Photoshop	42
Angular	43
ReactJs	51
NodeJs	52

HTML5

Agenda

- What is HTML5?
- Why HTML5?
- HTML5 Semantic Elements
- HTML5 Form Elements and Attributes
- HTML5 & Multimedia (Audio, Video)
- JS API's

Assignments
Assignment #1 Create an HTML5 application to locate your browser location (in the form of longitude and latitude).
Assignment #2 Implement the Selector API to color the Para which is inside the div tag by adding a button to the web page named as COLOR. i.e.   -------
Assignment #3 Use Drag and Drop API and show the working.
Assignment #4 Explain Web Worker implementation by giving proper script.

Assignment #5

Create a canvas to draw

CSS3

Agenda

- What is CSS?
- Different types of CSS
- Types of Selectors
- Box Model
- Priority Scheme
- Various Properties
- Transformation and Transition
- Pseudo classes
- CSS 3 Animations

Assignments

Assignment #1

In this exercise you will create a CSS3 Zebra Striping a Table.

On completion, the page will look like following

Id	Name	Major
1001	Gopl Murthy	Physics
1002	Joy Sen	Economics
1003	Chandu Yadav	Chemistry
1004	Shalini Gupta	Zoology
1004	Vivek Kumar	Math
1004	Sameer Ali	Botany

Assignment #2

In this exercise you will create a navigation bar. On completion, the page will look like following

Assignment 3:

Design a web page named “Registration.html” to accept the following employee details:

Employee ID
First Name
Last Name
Department
Date Of Joining
Email
Contact No
Address
Training Programs Attended

The web page should look as shown below:

Registration Form

Employee ID:	
First Name:	First Name
Last Name:	
Department:	IT HR Finance Admin
Date of Joining:	mm/dd/yyyy
Email:	
Contact No:	
Address:	
Training Programs to be attended:	Web Basics C# Entity Framework ASP.NET MVC
	Register Cancel

Following validations should be taken care of while designing the web page:

1) All fields are mandatory (Department and Training Program to be attended should be made mandatory fields after learning JavaScript. Rest of the fields should be made mandatory during HTML5).

2) Employee ID validations:

a) Employee ID field should contain only numbers between 10001 and 15000.

b) Employee ID field should be highlighted when the form is loaded.

3) First Name should consist of only characters and should be between 3 to 15 characters.

4) Last Name should consist of only characters and should be between 5 to 25 characters.

5) User should be able to select only one department (Use radio buttons to implement this functionality).

6) Date of Joining should be valid date and should not exceed the current date.

7) Email Address should be entered in proper format.

8) ContactNo should contain only numbers and must be of 10 digits only.

9) User can attend multiple training programs (Use check boxes to implement this functionality).

10) After clicking on Register button, user should be redirected to page named “Success.html” if all inputs are valid.

11) After clicking on Cancel button, all the fields entered by the user should be cleared.

Style the “Registration.html” page appropriately using Cascading Style Sheets (CSS).

JavaScript

- Core JavaScript
  - Introduction
  - Advantages of JavaScript
  - Script Tag
  - Variables
  - Arrays
  - Operators
  - Control Structures
  - Functions
  - Events
  - JavaScript Browser Objects
- Object Oriented JavaScript
  - Strict Mode
  - Variable Hoisting
  - Objects
  - Scope
  - Functions
  - Closures
  - Inheritance

Assignments

Goals	• Learn to embed script tags in different parts of the HTML document.
Time	120 minutes

1.1: Create a page to display "Welcome to JavaScript".

Solution:

Step 1: Start the editor to be used.

Step 2: Write the JavaScript code and save it as **Assign1.html**

```
<!DOCTYPE html>
<html>
<head>
        <title> Welcome to JavaScript</title>
</head>
<body>
   <script type="text/javascript">
        document.write ("Welcome to JavaScript - The Scripting Language of the Web");
   </script>
</body>
</html>
```

Example 1: Lab 1: Assign1.html

Step 3: Select **Start → All Programs → Internet Explorer.** Alternatively select **Start → All Programs → Google Chrome.**

Step 4: In the Internet Explorer, select **File → Open → Browse**, and select the file you have just saved.

Step 5: Click **OK** in the browser pop-up window.

Step 6: Verify that you get the output as shown in the figure given below.

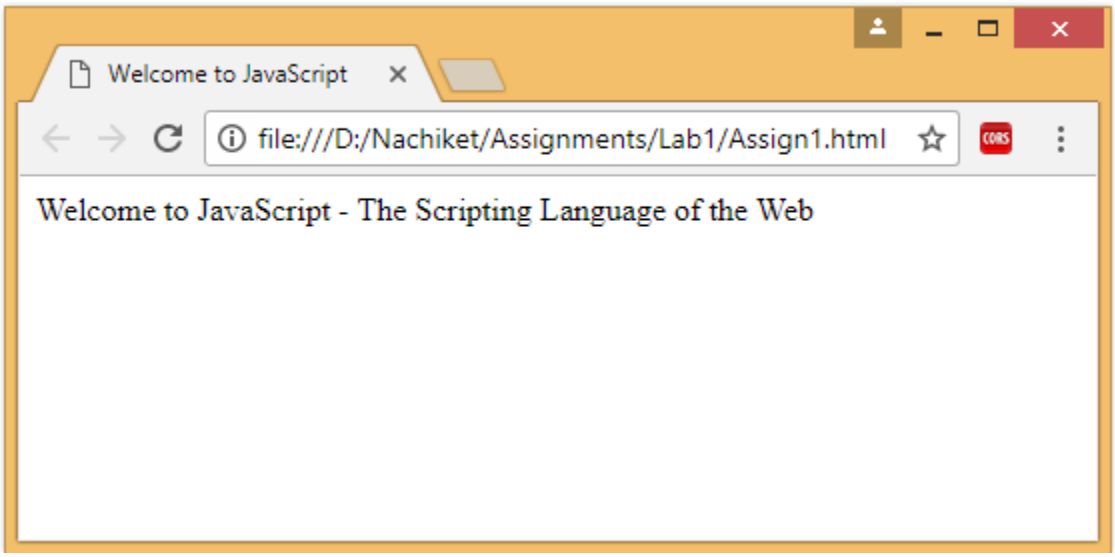


Figure 1: Welcome to JavaScript

Note: Follow the above steps (2 - 6) for every Lab problem for verifying the output (Please note that the file will be saved with .html or .js extension as the case may be. The filename will be different in every assignment). You can also use any other text editor like WordPad, Notepad++ (if installed) to create your **html** and **.js** pages.

1.2: Create Assign2.html to display Formatted Hello JavaScript by using JavaScript by embedding Hello JavaScript in <p> tag.

Solution:

Step 1: Create **Assign2.html** page to complete the following code and save in lab1 directory.

```
<html>
<head>
<title>Displaying Formatted Text using JavaScript</title>
</head>
<body>
<script type="text/javascript">
  //Display Hello JavaScript embedded in p tag with align attribute value right
</script>
</body>
</html>
```

Example 2: Lab 1: Assign2.html

Step 2: Open **Assign2.html** page in the browser, and verify that you get the same output as required.

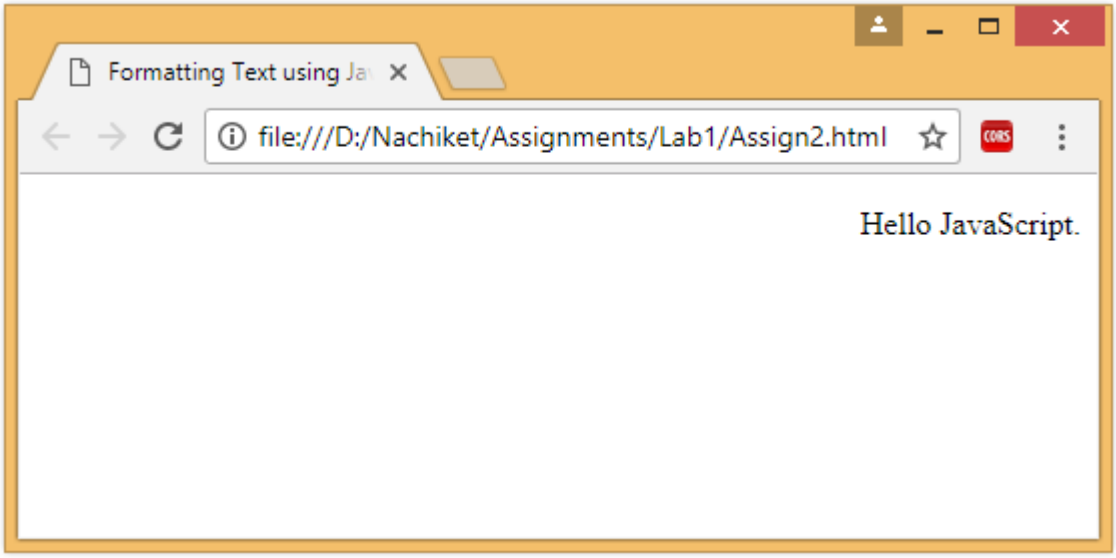


Figure 2: Formatting Text in JavaScript

1.3 : Create page to demonstrate use of external JavaScript

Solution:

Step 1: Create **Assign3.html** to complete the following code and save it in lab1 directory.

```
<!DOCTYPE html>
<html>
<head>
        <title>Using External Script file in HTML Document</title>
        <script src="HelloWorld.js"></script>
</head>
<body>
      <hr>
      <p>Script is located in external script file named "Hello.js"</p>
       <script>
                              //Insert the code here to invoke the function sayHello() in the file
HelloWorld.js
       </script>
       <hr>
</body>
</html>
```

Example 3: Lab 1: Assign3.html

Step 2: Create a file **Hello.js** which should have a function **sayHello()** that returns a string “Hello from JavaScript”.

```
function sayHello()
{
   //return the string “Hello from JavaScript“
}
```

Example 4: Lab 1: Hello.js

Step 3: Open **Assign3.html** page in the browser, and verify that you get the same output as required.

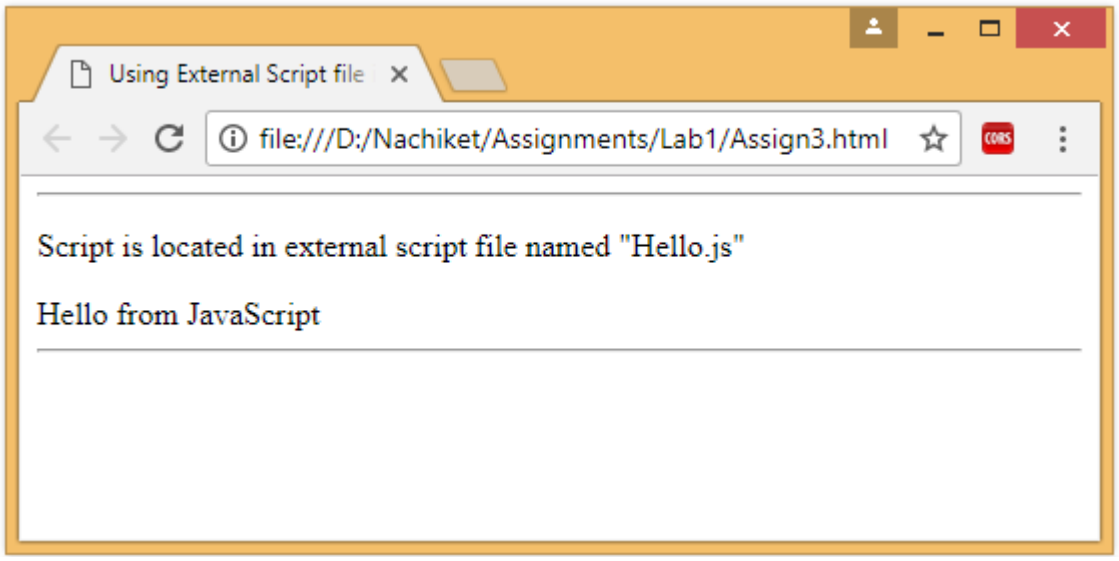


Figure 3: Using external JavaScript File

1.4: Using Variable in multiple Script tags

Step 1: Create **Assign4.html** page, and complete the following code and save it in lab1 directory.

```
<!DCOTYPE html>
<html>
<head>
	<title>Embedding Script tag in HTML Document</title>
	<script>
	  /* define variable headParam and initialize it to some integer value and
display
	     the value as shown in the figure 4. */
	</script>
	<hr>
</head>
<body>
	<script>
	   /*define variable bodyParam and initialize it to some integer value and
display
	     the value as shown in the figure 4 */
	</script>
	<hr>
	<script src="common.js">
	</script>
```

```
        <script>
          /* Invoke the method addNumbers(headParam, bodyParam)  defined in
            common.js file and pass the two variables headVar and bodyVar defined in
            the head and the body script tag and display the added result as shown in
            the Figure 4 */
        </script>
        <hr>
</body>
</html>
```

Example 4: Lab 1: Assign4.html

Step 2: Create a file **mathFunctions.js** which has a function **addNumbers()** that adds two numbers and returns the addition of two numbers.

```
var msg;
msg="<p><code>The script is located in external script file called math.js</code></p>";

function addNumbers(headParam, bodyParam)
{
    /* display the contents of the variable "msg" */

    /*  display the addition of two numbers */
}
```

Step 3: Open **Assign4.html** page in the browser, and verify that you get the same output as required.

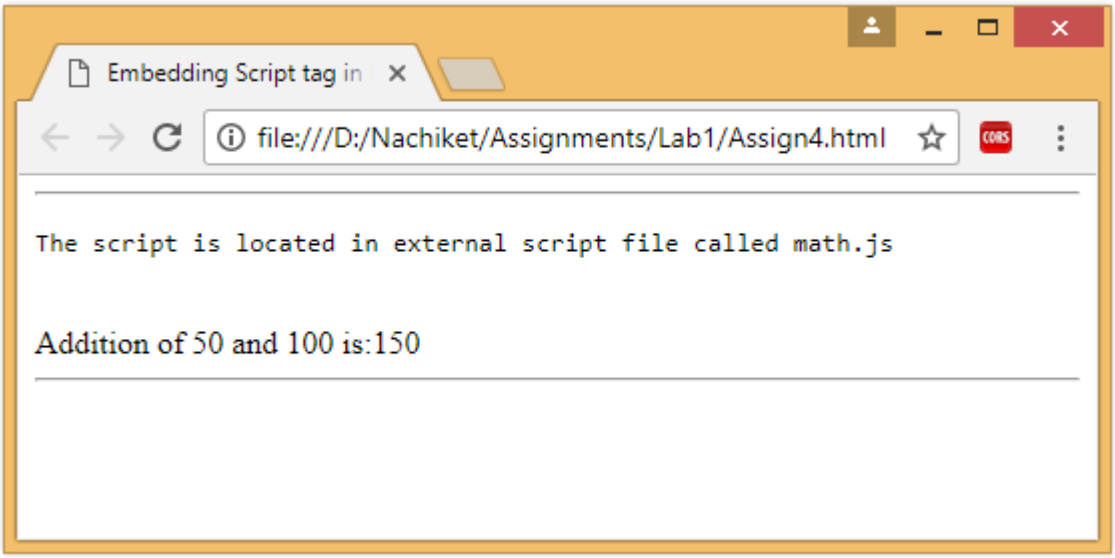


Figure 4: Using Variable in multiple Script tag

2.1: For loop in JavaScript

Create a web page as shown in the figure below.

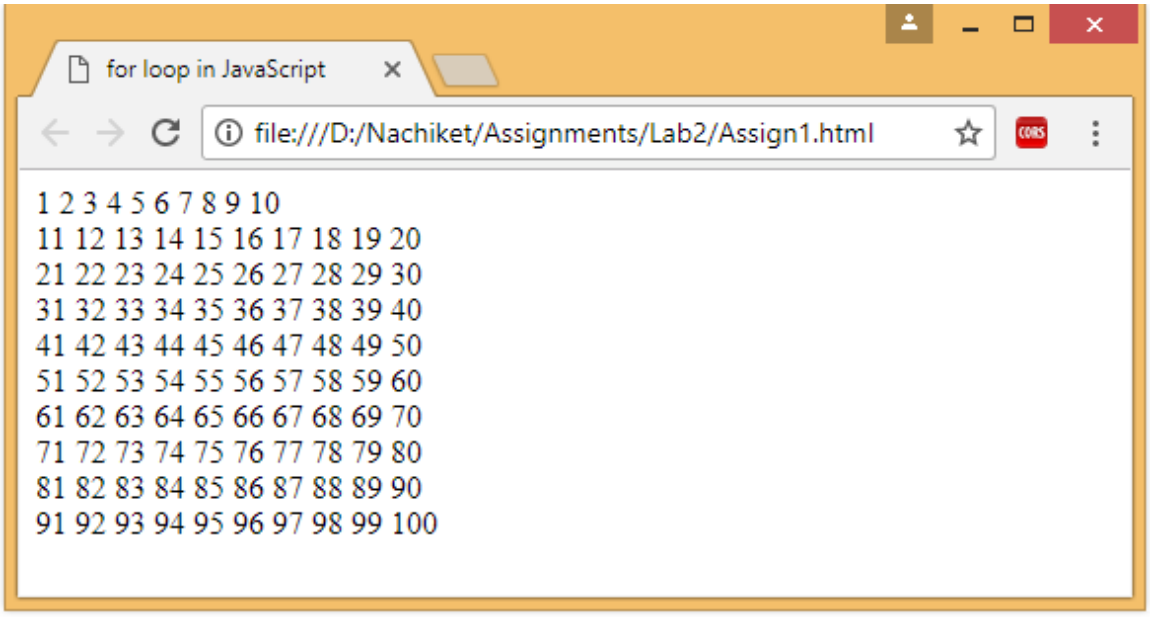


Figure 5: For loop in JavaScript

Use for loop to display the output.

After completing the loop, the variable used, that is **"index"**, should be equal to **101**.

Solution:

Step 1: Write the code and save it as **Assign1.html** in lab2 directory.

Step 2: Open **Assign.html** page in the browser, and verify that you get the same output as required.

Step 3: Create **Assign1_dowhile.html** and **Assign1_while.html** page using **do…while** and **while** control statements respectively to display similar output as shown in the figure given above.

2.2: Create a web page to calculate the Compound Interest using the formula given below:

Compund Interest $\left[p * \left(1 + \frac{r}{100} \right)^n \right] - P$

Where:

p = Principal,

r = Rate of Interest,

n = period in years

Accept the Principal (p), Rate of Interest (r) and period in years (n) values from the user and display the total interest payable by the user.

3.1: Displaying Date using Date Object

Create a web page **Assign1.html**. In this web page, create a **date** object and use the appropriate functions of the date object to display today's date in the format as shown below in the figure and also greet the user with the appropriate message depending on the time the user visits the page. The message to be displayed is given in the following table. The time column shows the current date hour value.

Time	Message to be displayed
< 12	Good Morning
>= 12 and <= 17	Good Afternoon
> 17	Good Evening

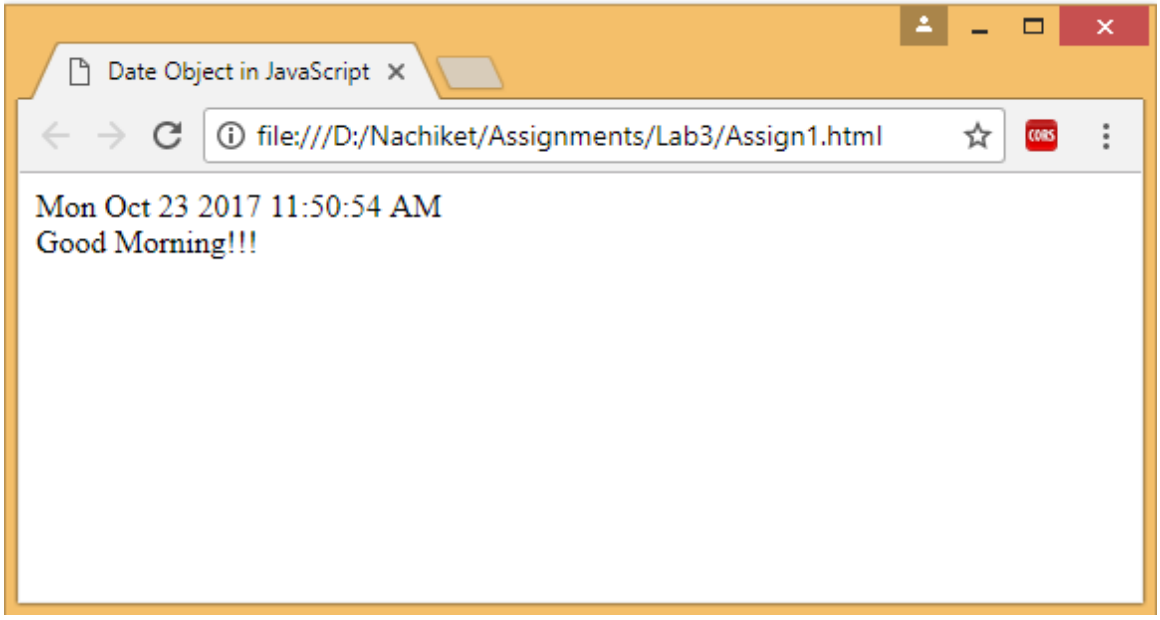


Figure 6: Displaying Date using Date Object

Solution:

Step 1: Write the Code, and save it as **Assign1.html** in lab3 directory.

Step 2: Open prob1.html page in the browser, and verify that you get the same output as required.

3.2: Using methods of String object

Create a web page **Assign2.html.** Create a string variable as follows:

var companyName=”Cybage Software Pvt Ltd”;

Accept character to be searched in this string from the user. Display the index of the first occurrence of the character if character is present in the string. Display appropriate error message if string does not contain the character entered by the user.

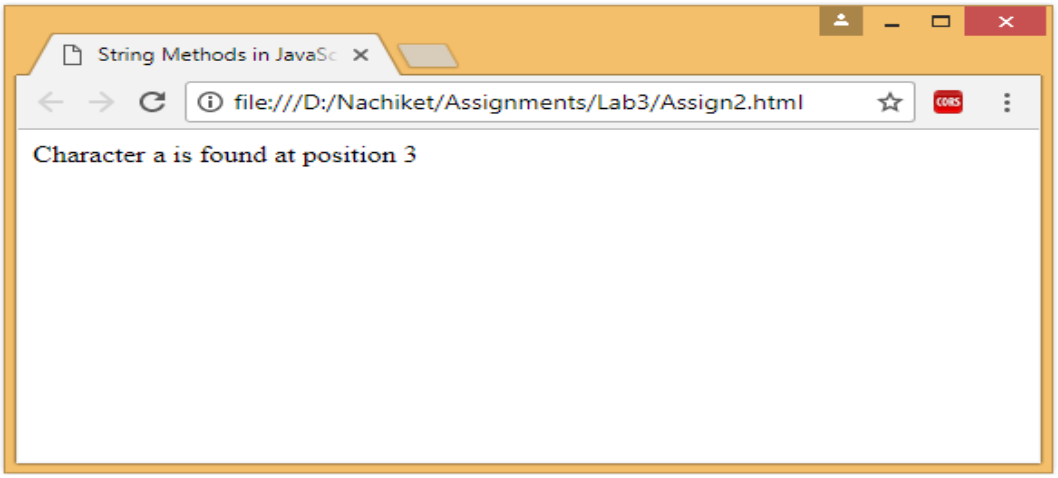


Figure 7: Using indexOf method of String object

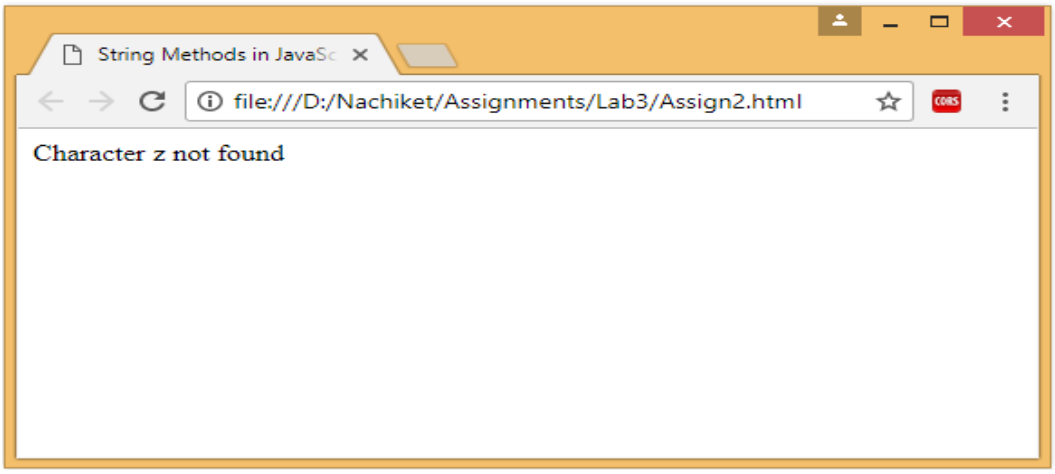


Figure 8: Using indexOf method of String object

Use appropriate methods of string object and perform the following operations on the companyName variable:

1) Extract the string “Cybage Software” and display the same in the same in the browser.

2) Convert the contents of companyName variable to lower case.

3) Convert the contents of companyName variable to upper case.

The output should look as shown in the following figure:

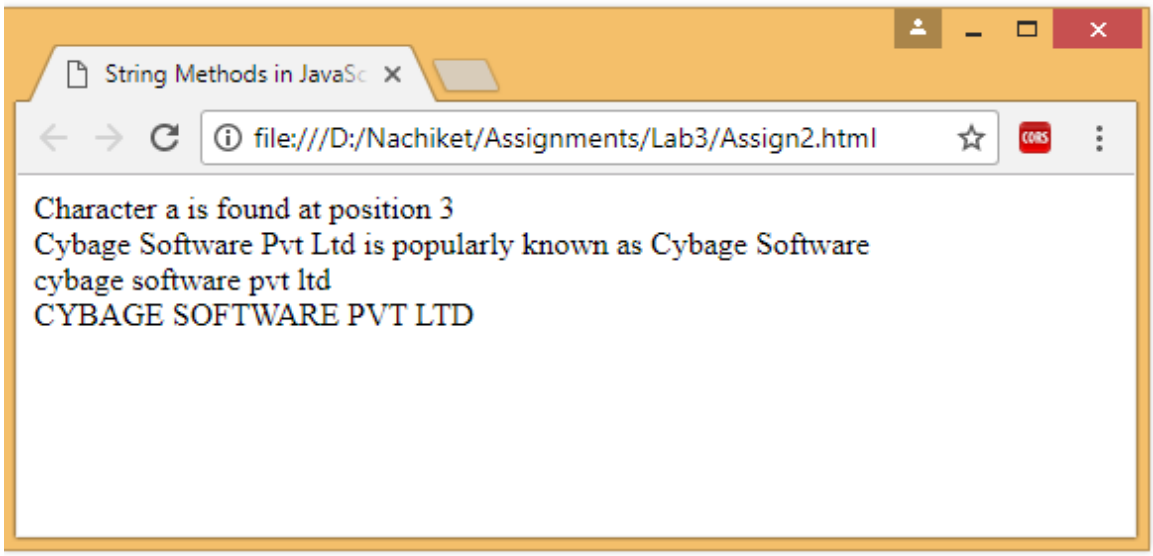


Figure 9: Using methods of String object

Solution:

Step 1: Write the Code and save it as **Assign2.html**.

Step 2: Open **Assign2.html** page in the browser, and verify that you get the same output as required.

4 .1: Using Array to display values

Write a JavaScript code to accept 4 student names from the user. Add those student names to an array and display them in the browser when user clicks Display button.

Please refer the following sample screens:

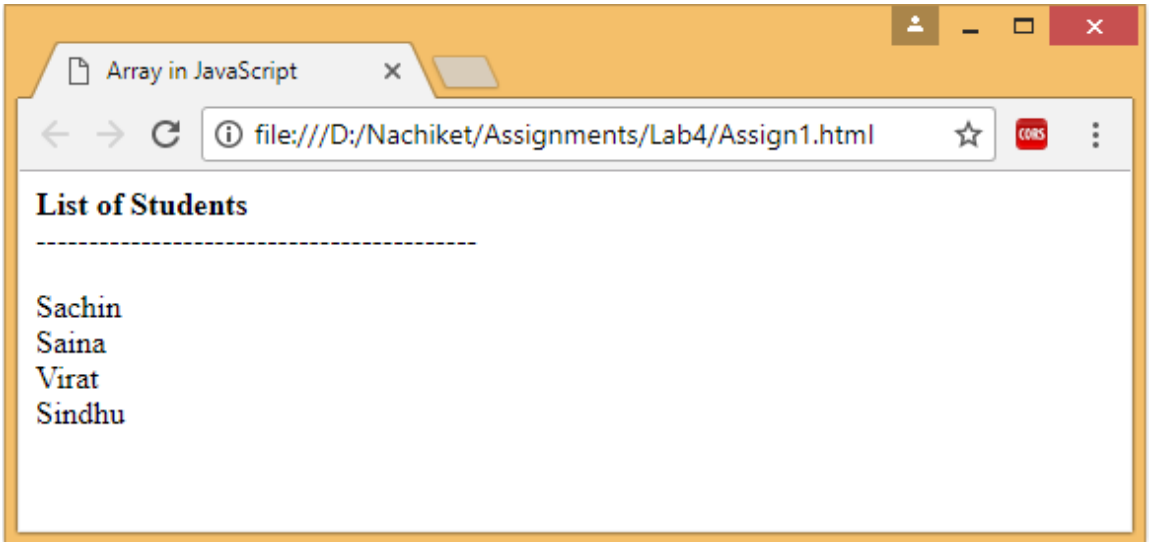


Figure 10: Displaying Array Elements

Solution:

Step 1: Write the Code, and save it as **Assign1.html**.

Step 2: Open **Assign1.html** page in the browser, and verify that you get the same output as required.

5.1: Create an Assign1.html web page as given below:

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Use the DOM to change my text!</p>
<script>
// Add code here
</script>
</body>
</html>
```

Write a code to change the text of the paragraph. The sample screen after changing the text of the paragraph is shown below:

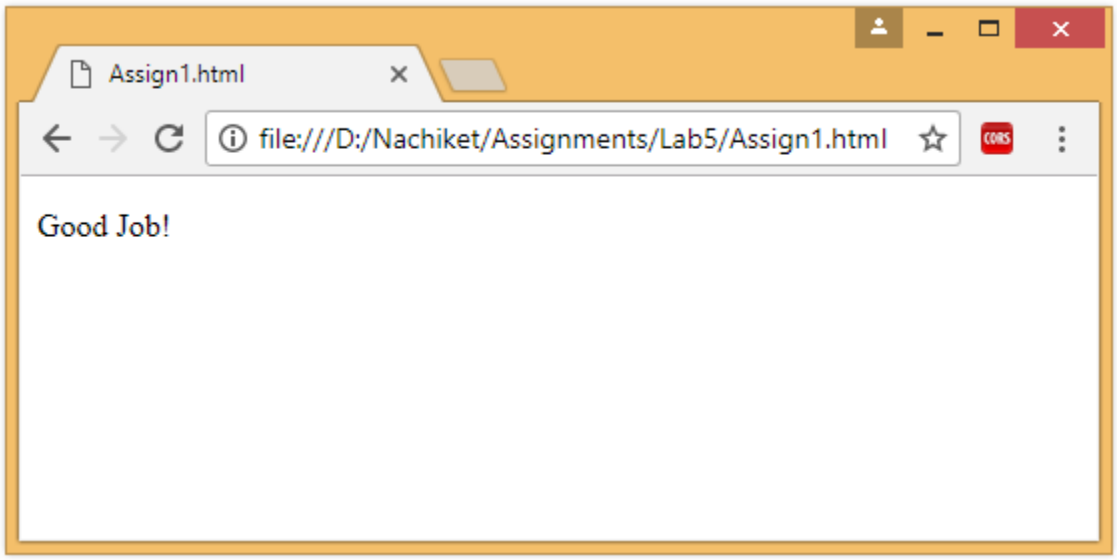


Figure 11: Changing element contents using DOM

5.2: Create an Assign2.html web page as given below:

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Use the DOM to change my text!</p>
<p id="test">This text won't be changed!</p>
<script>
// Add code here
</script>
</body>
</html>
```

Write a code to change the text of the first paragraph. The sample screen after changing the text of the paragraph is shown below:

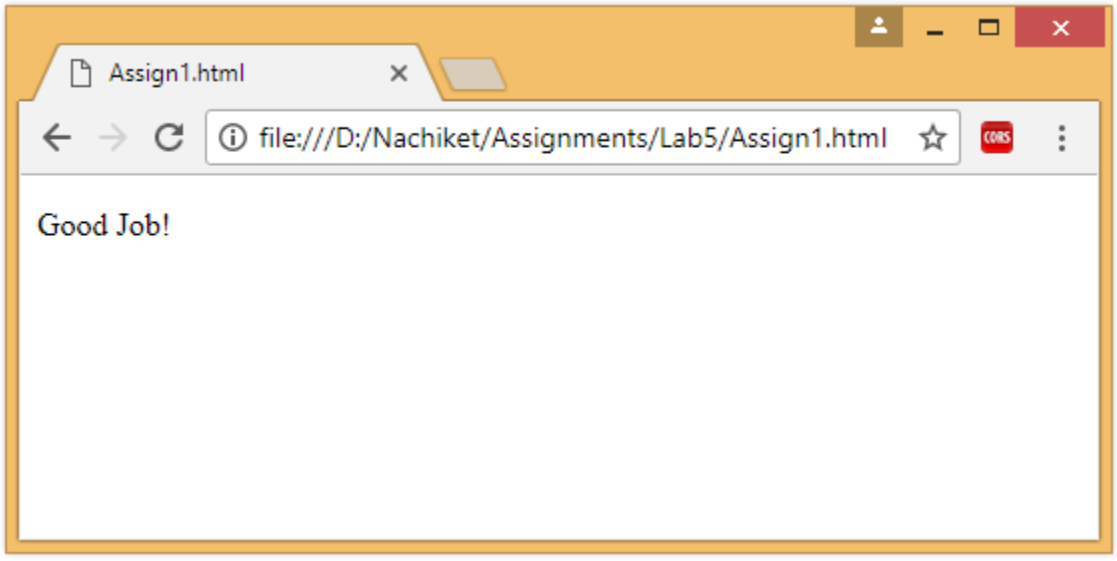


Figure 12: Changing element contents using DOM

5.3: Create an Assign3.html web page as given below:

```
<!DOCTYPE html>
<html>
<body>
   <div id="div1">
        <p id="p1">This is a paragraph.</p>
        <p id="p2">This is another paragraph.</p>
   </div>
</body>
</html>
```

Use the appropriate DOM method to add one paragraph element to this HTML page.
The sample screen after adding the paragraph element is shown below:

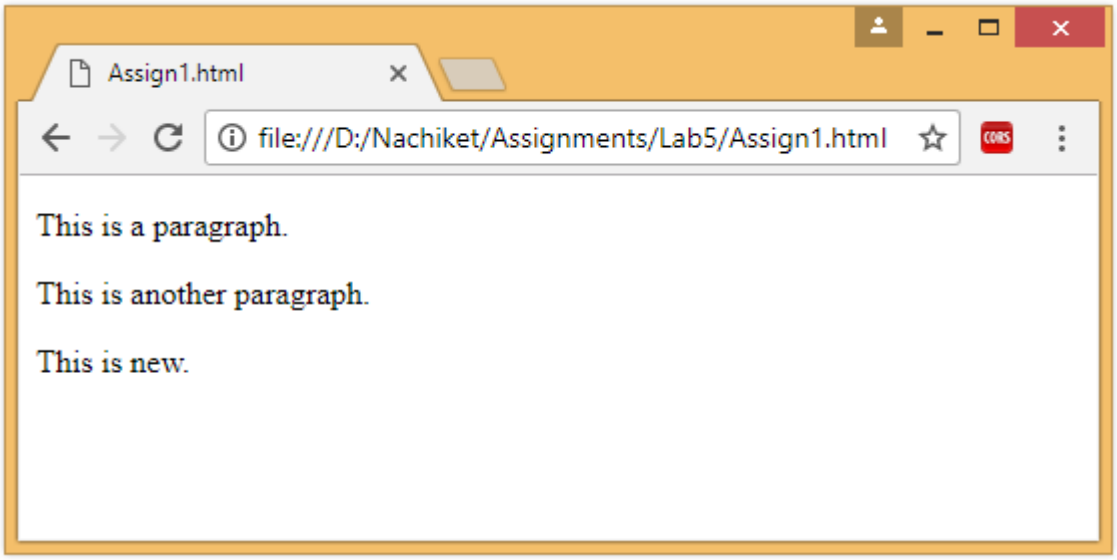


Figure 13: Adding element using DOM

6.1: Location Object

Create a web page which will display the properties **href**, **protocol**, and the **pathname** of the location object of your current file. The page should look as shown below:

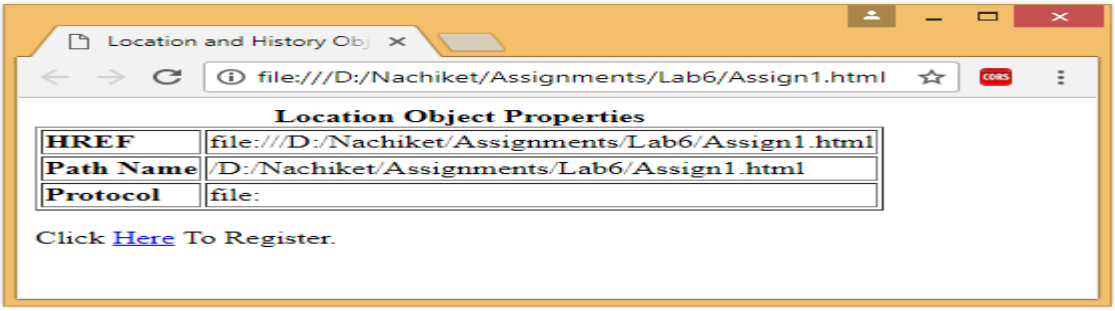


Figure 14: Location and History Object

When user clicks on the hyperlink, a new page named “Register.html” should open in the same window. “Register.html” page should have a Back button. When user clicks the Back button, “Assign1.html” page should open. The “Register.html” page should look as shown below:

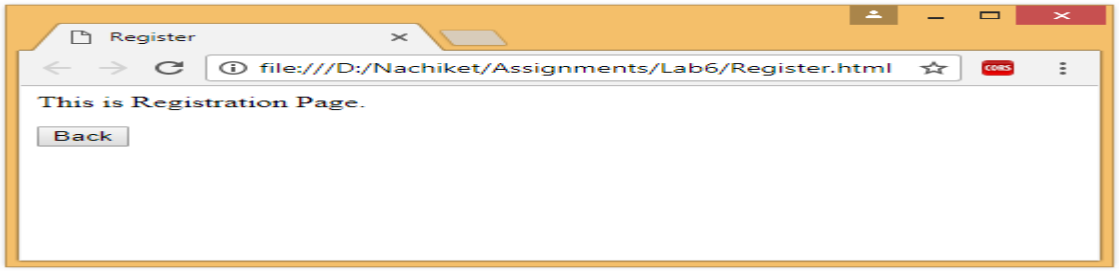


Figure 15: Location and History Object

Solution:

Step 1: Write the code and save it as **Assign1.html**.

Step 2: Open **Assign1.html** page in the browser, and verify that you get the same output as required.

Sr. No	Field Name	Type	Validations
1	First Name	Textbox	Required
2	Last Name	Textbox	Required
3	Date of Birth	Textbox	Required, should be a past date and format should be DD-MM-YYYY
4	Email Address	Textbox	Required
5	Hobby (3 checkboxes: Reading, Watching Movies, Photography)	Checkbox	At least one check box should be selected.
6	Favorite Writer	Textbox	Should be visible only if hobby is “Reading”
7	Submit Button	Submit	
8	Reset Button	Reset	

7.1: Form Validation (to be completed at end of – Core JS)

Create a form with the following attributes and validations -

Additionally -

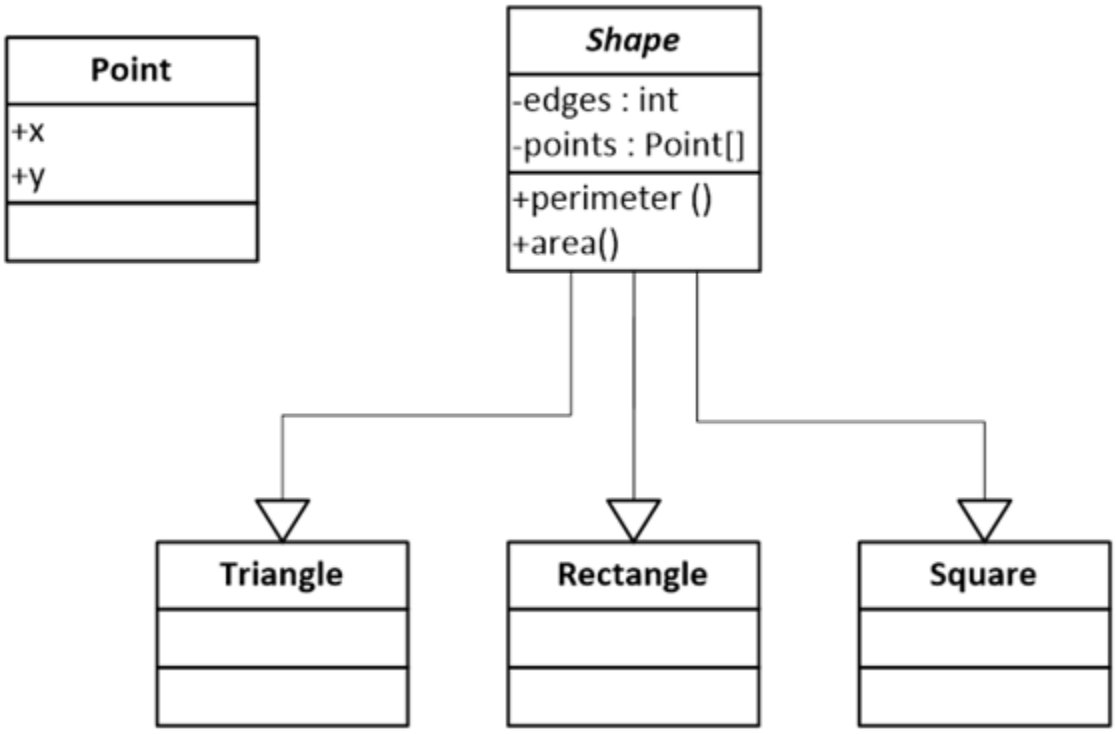
- Show a welcome message to user
  - Add a timer, which will keep counting the time for which page is open.
  - This should also contain the browser name that is being used. (Optional)
  - E.g. Welcome! You are using “Chrome” and you are on this page since 00:25 minutes.
- Date field
  - Only numbers and “-” should be allowed in the Date field.
  - If the user enters any other character -
    - Border color should be changed to red.
    - Invalid character should be cleared, but valid characters should not be cleared.
    - Display a message besides the textbox showing hint for the format.
- Show validation messages using alert boxes.
- On validation failure, focus the corresponding field and highlight it with red border
- Validation for email address should be done using regular expressions. (Optional)

When user clicks on Submit button, if all the validations are successful then a new paged named “Success.html” should be opened and it should display some message like “Congratulations!!! You have registered successfully!!!

Object Oriented JavaScript Lab Assignments

Assignment – 1 (to be given at end of – OOJS)

- When a set of co-ordinates is supplied, the area and perimeter of a square, rectangle and triangle needs to be determined.



- From the above diagram –
  - Point, Shape, Triangle, Rectangle and Square are all classes
  - x and y are properties of Point
  - edges and points are properties of Shape
  - perimeter and area are methods of Shape
- Hint – Use the co-ordinates to determine the distance between two points.

Assignment – 2

Build out a calculator on a web page that mimics the look and feel of the Calculator that you get on the Windows system. At the bare minimum it should have –

- An input control that allows you to toggle between a standard and scientific calculator.
- An output display.
- Controls to feed in numbers.
- Controls to feed in operations.
- A control that removes one digit at a time (the <- button)
- A control that resets the entire state (the C button)

Standard operations include -

- Addition
- Subtraction
- Multiplication
- Division

Scientific operations include –

- Square
- Cube
- Square root

JQuery

1. Selectors
2. DOM Manipulation
3. Events Handling
4. jQuery Lookups
5. Ajax

SASS

- Introduction to SASS
- Features of SASS
- How to use SASS
- Variables
- Nesting
- Comments
- Mixins
- Selector inheritance
- Include one sass into another
- Demo

Assignment

To create a SASS project:

Once sass and compass are installed successfully,

1. Create a new folder to save your project (say 'Test Project' in D drive)
2. From the path D:/Text Project in the command prompt, issue the command 'compass create'
3. This shall create some default folders like 'stylesheets' and 'sass' with a config.rb file.
4. Delete all the default files created under folders sass and stylesheets folder under 'Test Project ' folder.

Now we are all set to create our own SASS structure.

1. Create folders 'base' and 'modules' under D:/test Project/sass folder
2. Create a file called 'main.scss' under D:/test Project/sass folder
3. Add a file called '_buttons'scss' under modules folder.
4. Add a file called '_variables.scss' under 'base' folder.
5. Declare some variable in _variable.scss file, say

   $btn-bg-color: #cccccc;
6. Now, lets us add some class under _buttons.scss file.

   Eg:

```
.button-class{
        background-color:$btn-bg-color
        color: #000000;
        border-radius: 3px;
        border: 1px solid #cccccc;
  }
```

7. Now import the sass files in main.scss file. Open main.scss file add following lines

   @import 'base/_variables.scss'

   @import 'modules/_buttons.scss'

8. Go to the command prompt where you issues the compass create command.
9. And issue the command , 'compass watch' (This command has to be issued from the path where your config.rb exists)
10. If everything is working fine, then you should be able to see compiled main.css file under stylesheets folder.

Try create an html file and refer this css file in your html file.

Assignment 1:
Write SASS structure for an input control with following specifiations:
Normal Input specs:
Border radius: 3px;
Width:300px;
height:36px;
//border:1px solid #ccc;
font-style:SegoUI
font size:13px
padding-left:10px
color:#666666
background-color:#ffffff;

Input control when Disabled:
//Border: 1px solid #cccccc;
background-clor:#eeeeee;
colro:#999999;
cursor: not-allowed;

Input control when error:
Background - color:#fde7e1
border:1px solid #f44336
color:#f44336

Assignment 2:
Button control specifications:
We need to create primary and secondary button styles, where following are the properties in common.

Border Radius: 3px;
padding: 12px 15px 12px 15px;
Background color: #8bc34a;
border: 0;
min-width: 110px;
font-family: "SegoeUI SemiBold";
font-size: 16px;
color: #FFFFFF;

Primary button shall have following specifications:
Background Color: #8bc34a;
Color: #FFFFFF;
Border: 1px solid #8bc34a;

Primary button on Hover shall have following classes:
background-color: #9dce5d;
color: #FFFFFF;
border: 1px solid #9dce5d;
Primary button when disabled shall have following classes:
background-color: #eeeeee;
color: #999999;
border: 1px solid #eeeeee;

Secondary Button shall have following classes in addition to base properties:
background-color: #FFFFFF;
color: #2095f2;
border: 1px solid #2095f2;

Secondary button on hover shall have following classes:
background-color: #2095f2;
color: #FFFFFF;
border: 1px solid #2095f2;

Secondary button when disabled shall have following classes:
background-color: #ffffff;
color: #999999;
border: 1px solid #cccccc;

use the compiled CSS generated in an sample html and sample page shall look like below and shall have all added effects on the control (effect on hover, disabled etc)

Inputs

Input control with placeholder, maxlength and size

Normal input

Input disabled

Disabled input

Input in error stage

Error

Buttons

Customised buttons, button width shall depend on the caption (min width = 110px)

Primary Button

Secondary Button

Primary Button

Secondary Button

Hints:

1. Use variables wherever you can.
2. Identify the common variables and move them to _variables.scss
3. Write mixins for border radius and add a file called _mixins.scss under base folder
4. Do not forget to import this file before using in main.scss file
5. Use inheritance wherever possible.
6. Variables which are not used across files can be kept in the module file (like _buttons.scss) itself.
7. For writing input control related SCSS, add another file called _inputs.scss file under modules.

RWD

Agenda

- Responsive Web Design
- Viewport
- Media queries
- Responsive vs Adaptive
- Mobile first vs Desktop first
- Relative units vs Static units
- Orientation
- How to handle images and elements
- Flexible Grid
- Key features
- Frameworks

Bootstrap

Agenda

- Quick recap of CSS
- Quick recap of RWD
- Bootstrap Grid
- Bootstrap Content classes
- Bootstrap Components
- Bootstrap Utilities classes

Photoshop

Agenda

- Introduction
- Photoshop Tools

Angular 4

- Introduction to Components
- Templates, Interpolation, and Directives
- Data Binding & Pipes
- More on Components
- Building Nested Components
- Services and Dependency Injection
- Retrieving Data Using Http
- Navigation and Routing
- Building, Testing and Deploying with the CLI

Assignments

Assignment 1:

1) Create an Angular JS 4 application.

2) Add a variable named message in AppComponent class and assign it some value. Your AppComponent class should look like this:

```
export class AppComponent
{
    title = 'App';
    message= 'Single Page Applications have become a way of website development';
}
```

3) Print the value of message variable on app.component.html (Use paragraph element).

4) Add a new component named FooComponent to the application. Observe the changes made in app.module.ts file after adding this component.

5) Display the contents of foo.component.html on app.component.html. Note the usage of element selector defined in foo.component.ts file.

Assignment 2:

1) Create an Angular JS 4 application.

2) Add a class named Actor in this application. Actor class should look like this:

```
export class Actor {
    public name : string;
    public characterPlayed : string;
    public isHandsome : boolean;
}
```

3) Create an instance of Actor class in AppComponent class and assign value to name and characterPlayed using that instance.
   E.g.- Actor actor=new Actor();
   actor.Name = 'Amrish Puri';
   actor. characterPlayed = 'Mogambo';
   actor. isHandsome = true;
4) Display name of the actor in app.component.html (Do it once using paragraph element and string interpolation and once using paragraph element and property binding). Apply style to actor name using ngStyle directive. Display actor name in bold in Verdana font with font size 14 and in royal blue color depending on value of isHandsome variable (Use ngClass directive).

5) Add a function in AppComponent class. The function should display an alert.
   E.g. –

```
function displayMessage() {
    alert('Welcome to the world of Angular JS 4');
}
```

   Add a button in app.component.html and call this function on click event of a button.

6) Add a textbox in app.component.html. Display the name of the actor (created in Step 3) in this textbox. Enter the name of the actor in this textbox and display the newly entered actor name in app.component.html (Use two way data binding).

7) Display the name and characterPlayed by an actor in upper case in app.component.html.

8) Display the name and characterPlayed by an actor in lower case in app.component.html.

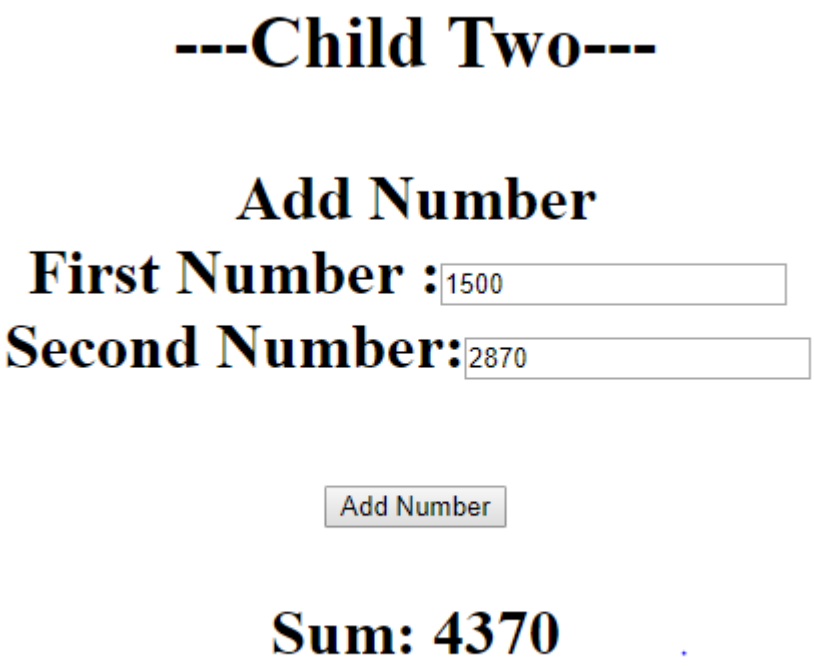
Component Communication:

9) Create three components namely Parent, ChildOne and ChildTwo in your angular application.
10) Declare an array of type string in the Parent component. The array should contain city names.
11) Display contents of this array in ChildOne component. Display the contents of ChildOne component in Parent component.

Please refer to the following screenshot for your reference:

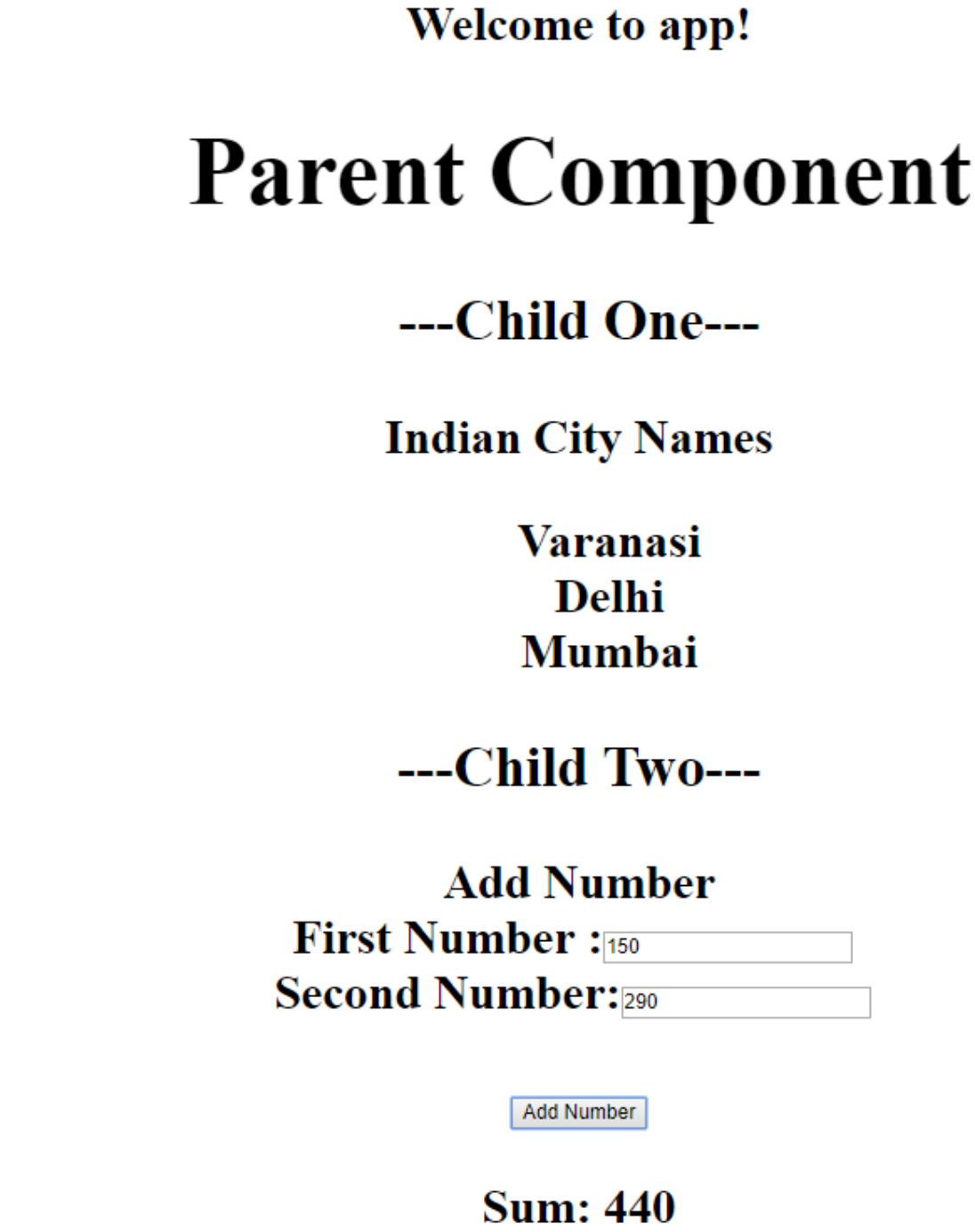
12) Create two textboxes and a button on ChildTwo component. Accept two integers from these textboxes and add those integers on click event of a button. Display the sum of these integers on Parent component.

Please refer to the following screenshot for your reference:



Write the appropriate methods in Parent and Child components. Use appropriate directives. Display the contents of Parent component in App component.

Please refer to the following screenshot for final result:



1)

Assignment 3:

1) Create an Angular JS 4 application.

2) Add Animal class in the application. Animal class should look like this:

```
export class Animal {
        public id : number;
        public name : string;
        public weight : number;
}
```

3) Create following two components in the application:
   a) AnimalList
   b) AnimalDetails

4) Add a file named app.routes.ts in the application. This file will be used for defining routing information. Configure the routing as follows:
   a) Define a route for AnimalList component.
   b) Request should be routed to AnimalList component if we don't specify any route.

5) Include app.routes.ts file in app.module.ts file.

6) Run the application and make sure the routing is working as defined.

7) Add the AnimalService to the application. AnimalService should contain the following:
   a) Array of type Animal. Populate this array with some Animal objects.
   b) Method returning array of type Animal.

8) Call the above method of AnimalService in AnimalList component. Display the names of all animals in HTML of AnimalList component.

9) Add a new method in AnimalService class. The method should accept id as a parameter and return the Animal object corresponding to that id.

10) Make animal name displayed in AnimalList.component.html a hyperlink. When the user clicks on animal name, it should pass the id associated with that name as a parameter to AnimalDetails component. To make functionality work, add the route for AnimalDetails component to app.routes.ts file.

11) Display the id, name and weight of selected animal on AnimalDetails.component.html file.

Assignment 4:

1) Find a service from internet and invoke the functionality defined in that service by making HttpGet, HttpPost, HttpPut and HttpDelete requests.

React JS agenda

React components
Props and state
Components lifecycle
Events
Routing

Assignment 1

- Create a React app with component to display
  Hello welcome to React
- Create component employee with Name ,age,Desg,salary,company_name as properties
- Add default values to property(company Name).
- Add validations to employee id for accepting integer values.
- Convert your app with UI design to enter properties of employee
- Update employee salary with state by taking help of events
- Display different lifecycle methods of components on console
- Create links to view Employee Information(id,name,sal,company name),Company Information, Careers.

Assignment 2

- Create a login page with react app
- Once login is done try to find employee names starting from initials from table created and display information of employee in the form of table.

Node Js

Agenda

- What is Node.js & History
- Architecture & Design
- Benefits & Drawbacks
- Hosting companies & Tools
- How to Install and Run JavaScript in
- npm – Intro
- What is npm?
- What is package.json?
- How to install, uninstall npm?
- Using npm parameters
- Global vs Local npm
- Choose your npm
- Node.js environment
- Building Blocks of Node.js
- How Node.js Works
- Modules
- Events
- Streams and Buffers
- Database connectivity
- Advantages & limitations

Declaration by the Participant

My Understanding

Name:
Employee ID:

Sr. No	Topic	Theoretical		Practical	
		Yes	No	Yes	No
1	HTML5				
2	CSS3				
3	JavaScript				
4	jQuery				
5	SASS				
6	RWD				
7	Bootstrap				
8	Photoshop				
9	Angular4				
10	Node Js				
11	React Js				