ITSE-2192 Fundamental of web Design and Development

Introduction to Git

Lab Zero

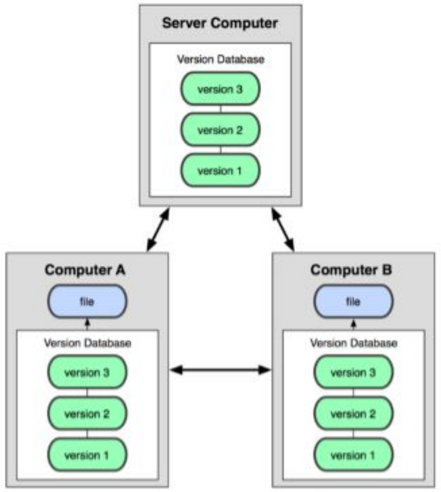
Outline

- Version Control System
- History about git
- How to Install Git
- Working With Git locally
- Working with Github

Version Control System

- Version control is the management of changes to documents, primarily computer programs.
- Also known as revision control or source control
    - Examples: git, mercurial, subversion
- Why version control?
    - Makes working in a team easy!
        - Code without interference
        - Go back to a previous version
        - Integrate code of multiple developer’s easily
        - Know who did what, when
- Why Git then ??
    - Distributed
    - Git keeps "snapshots" of the entire state of the project
    - Git generates a unique SHA-1 hash – 40 character string of hex digits, for every commit
        - Refer to commits by this ID rather than a version number

Distributed Model



About Git

- Created by Linus Torvalds, creator of Linux, in 2005
    - Came out of Linux development community
    - Designed to do version control on Linux kernel
- Goals of Git:
    - Speed
    - Support for non-linear development (thousands of parallel branches)
    - Fully distributed – Able to handle large projects efficiently

    - (A "git" is a cranky old man. Linus meant himself.)

How to Install Git

Things you’ll need:

1. You need Git installed on your system, and you can access it in a UNIX Terminal, either the Terminal on the Mac or Git Bash on Windows.
2. Download Git from the following link:
    a. Download Link [Link]
    b. Installation Process [Link]

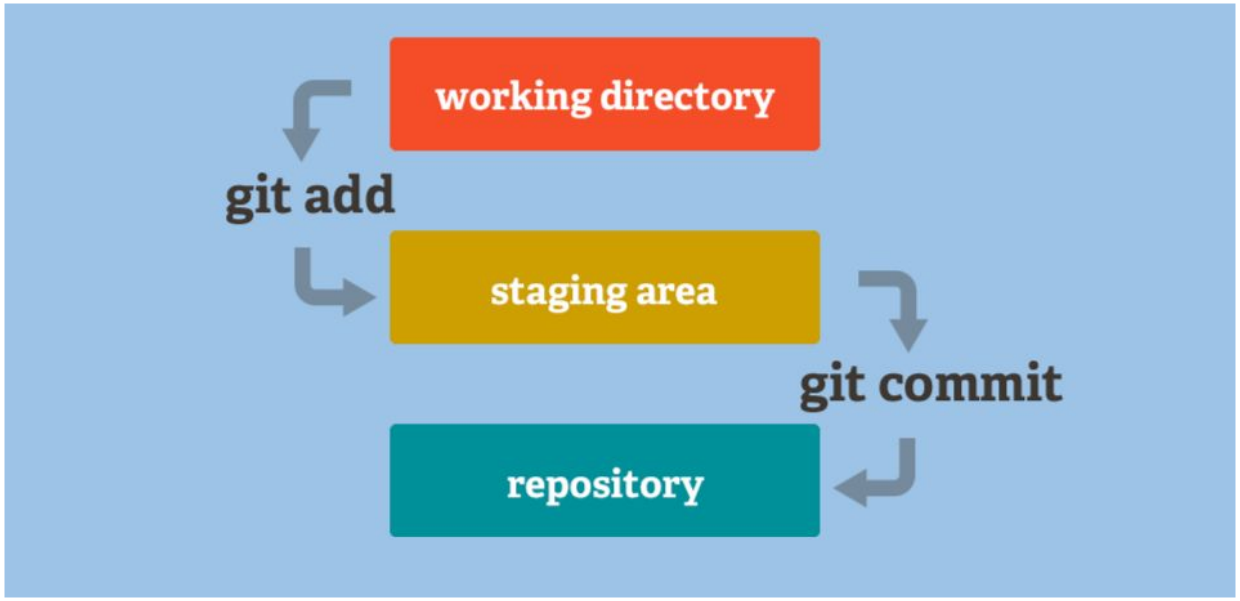
Basics Git Commands

1. Run the below command on the terminal to view about git commands
   ```
   git
   ```
2. Run the below command on the terminal to view git version
   ```
   git --version
   ```
3. To get help about the git command write the command
   `git help init` .... what do you get??
4. To clear git screen type
   `clear` or `ctrl+l`
5. Introduce yourself to git (git config)
   ```
   git config --global user.name "your name"
   git config --global user.email "your mail"
   ```
6. To view the configuration
   ```
   git config --list
   ```

Git Model(the work flow)

The Work Flow

- **Modify** files in your working directory
- **Stage files**, adding snapshots of them to your staging area
- **Commit**, which takes the files in the staging area and stores that snapshot permanently to your Git directory



Create and Maintain Local Repository

1. Create a folder named **project1** in Other local drive than C: and run the following commands below

   Type pwd //to view current directory

   Type ls or ls -a (hidden files) //to view content of the directory

   Type cd drive-letter: //to change drive (d: , e: ,.....)

   Type mkdir project1 //create the directory

   Type cd project1 //change to the directory

   Check the Status >>>>>Type git status ....what do you get??? Let go ahead.

Create and Maintain Local Repository(Cont..)

1. Create a new repository by running the command
   git init   //This will create a .git directory in your current directory
   git status   .... what do you get now???????
2. Create a new empty file inside project1 and name it README
   touch file1.txt   //you can create the file manually
   git status   .... something wrong???
3. Tell git repository track the file to, run the command below to add it.
   git add file1.txt (or *.txt , or . to track all files)
   git status   ....what do you get now???
4. To commit the file to the local repository, issue the commit command
   git commit -m "Your Commit message "
   git commit   //if you need to write a long message   :wq(to save and exit)
   git commit -a -m/-am "Your Message"   //to skip the staging area

View Commit History

1. To see the version history of your repository, run the command below

```
 git log
git log --oneline                  //to print on one line
git log  -p -2                      // the last two commits
git log  --since=1.weeks            //since last week
git log –since=”2014-05-05”         //since specific day
git log  --author=”name”            //change made by specific author
 git show 45678789e8789789
git log --graph --pretty=oneline
gitk(Windows)/gitx(Mac)
```

Status and Diff

To view status

git status or $ git status –s (-s shows a short one line version)

•To see what is modified but unstaged:

git diff

•To see staged changes:

git diff --cached

Undoing Commits and Changes

To return to any point of time

git checkout HasID(45678789e8789789) **or**

git checkout -- filename

To unstage a change on a file before you have committed it

git reset HEAD filename (unstages the file)

To Delete a untracked file

git rm <file>

To deleted staged file (after you add the file)

git rm -f <file >

To remove file from staging area (after you add the file)

git rm –cached <file >

Remote Repository hosted by GitHub

1. create an account on github.com
2. create a repository called fordemo on your account by clicking the “+” sign on your account
3. Now let’s push what we have on local to remote repository

```
git remote add origin url

git push -u origin master
```

4. If there is proxy, you need to set it like the following set

```
git config --global http.proxy 10.90.10.70:8080
```

5. To remove if you mistake account

```
git remote remove origin
```

Remote Repository hosted by GitHub(Cont..)

4. Synchronize the new cloned project and the old

```
git pull
```

5. To pull data without merging

```
git fetch origin
```

6. To pull data with merging

```
git remote -v//to view remote repositories
git pull “fetch url”
```

7. To Rename the name origin

```
git remote rename origin <anyname>
```

Resource

- Free online book: http://git-scm.com/book
- Git tutorial: http://schacon.github.com/git/gittutorial.html
- Reference page for Git: http://gitref.org/index.html
- Git website: http://git-scm.com/
- official Doc : Doc

ITSE-2192 Fundamental of web Design and Development

Hyper Text Markup Language(HTML)

Lab One

What we will Learn

- After completing this lab :
  - You will be able to work on basic html tags
  - Learn the major difference on HTML5 from earlier versions
  - You will be able to build you personal web app[CV]
  - Hosting in GIthub
- We will go through learning the html by building our personal cv with different pages

Required Software's

- Text Editor [Any one of them]

  VsCode [Recommended for this course] Bracket

  Atom Sublime

  Check for More

- Browsers
  - **Chrome [Recommended for this course]** , Mozila , Safari , IE
- Online Version [if you want to work on the cloud]
  - Check this list of editors online : Link

HTML

- **HTML : Hyper Text Markup Language**
    - It is a scripting language but not a programing language
    - Current Version : HTML5
    - Defines the Structure and Content of a document to be rendered in a browser
    - Html Tags
        - <tagName   attributes> Contents </tagName>
        - attributes ⇒ key = value pairs , the value can be “ ”
            - Tag Specific or Global
        - Some tags are **self closing tags** like <br>, <hr>

Note : HTML5 do not promote styling using the some tag attributes even though it do not prohibit you from using them , so you are not encouraged to use styling in html tags

HTML Boilerplate

1. Create a Folder Called MyCv
2. Open The Folder with VSCode
3. Create a file index.html //This file naming is Mandatory for hosting
4. Use The Emmet [will use more as we go on the course] >> in VsCode to generate Code
    1. Type ! And Enter //This will Generate the HTML5 boilerplate
    2. Change the title with your name CV

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Your Name Cv</title>
</head>
<body>
</body>
</html>
```

Working with Text Elements and Line

- Heads
    - <h#> #==> 1 – 6   [Block element with closing tag]
    - This element create a heading text with large font
    - Attributes : align(obsolete)
- Paragraphs
    - <p>   [Block element with closing tag]
    - This element Creates a paragraph
    - Attributes : align(obsolete)
- Horizontal line
    - <hr> [Block element and self closing tag]
    - Attributes : size , color, width, align

Working with Text Elements and Line(ctnd..)

- Inside the body tag add information Regarding yourself
  - Name : h1
  - Occupation , Place of Work : h3
  - Bio : p
  - Horizontal line for beauty and separation : hr

```
<h1>Your Name</h1>
 <h3>Student , Addis Ababa institute of Technology</h3>
 <p>Hi there , </p>
 <p>I am Your Name , i am expert <strong>Java</strong> Developer
who has been working on Java since <em>5 </em></em>years. I am
passionate about working in different types of programing languages
</p>
<hr>
```

Working with List

- Unordered List
    - <ul> <li>   [Block element with closing tag]
    - This element create a list with different kinds of bullets
    - Attributes : type(circle, disc, square, triangle)

- Ordered List
    - <ol> <li>   [Block element with closing tag]
    - This element create a list with different kinds of bullets
    - Attributes : type(a,A, i, I , 1(default)) , start, reversed
- We can create a complicated list with nesting

Working with List(ctnd..)

- Below the hr tag add the following list
- Hobbies and Interest : h3
    - List of Hobbies and Interest: ul [Feel free to change with your own]

```
<h3>Hobbies and Interest</h3>
<ul>
   <li>Sport : I like to watch and play
      <ul>
         <li>Football</li>
         <li>BasketBall</li>
         <li>VolleyBall</li>
      </ul>
   </li>
   <li>Reading Books
      <ul>
         <li>Educational Mostly Programing Books </li>
         <li>Fictions specially of literature
            <li>Books that impress me the most  :
               <ul>
                  <li>One Love by Mr. Hadis Almayhu</li>
                  <li>Black Magic by Mr. Samuale A.</li>
               </ul>
             </li>
         </li>
      </ul>
   </li>
<hr>
```

Working with Images

- Image
  - <img> [Inline-Block element with self closing tag]
  - This element allow as to display image
  - Attributes : src(mandatory) , hight, width, alt, more
- HTML5 Options
  - <figure> <img>   <figcaption> </figcaption></figure>
  - Attributes : Only Global attributes

Working with Images (ctnd..)

- Below the body tag/ above the heading information add your picture
    - Create folder asset > images
    - Add a picture : <img src = “url”>
    - Add additional info incase browser did not display it   :

        alt = “Your Name Profile image ”
    - **url**   local or http://   ?? Relative or absolute path

```
<img src="./assets/images/Imagename.png" alt=”Your Name Profile Image">
```

Working with Tables

- Tables
  - <table> <tr> <th> <td>[Block element with closing tag]
  - This element create a table
  - Attributes : cellspacing, cellpadding , border , bgcolor, align
- HTML5
  - <thead> <tbody> <tfooter> <caption>, <col>, <colgroup>
  - To show the structural components of a table

Working with Tables (ctnd..)

- Below Hobbies and interest lets add skill with the help of tables
    - Add horizontal line ; hr
    - Add the text **skills** : h3
    - Add your table with one row and two/three skill sets of your own
    - Keyboard shortcut for **emoji** : Window : Win + ./;   , Mac : Command + Control + Space

```
<hr>
<h3>Top Skills</h3>
<table cellspacing="10">
   <tr>
    <td>Java for Web and Android </td>
    <td>⭐⭐⭐⭐⭐</td>
   </tr>
   <tr>
      <td>HTML, CSS and Javascript</td>
      <td>⭐⭐⭐⭐⭐</td>
   </tr>
</table>
```

Working with Tables (ctnd..)

- Update the Header part with table
- Add table with two cell [td]
- Place the picture in one cell and the other content in other cell

```
<table cellspacing="10">
    <tr>
        <td>
            <img src="./assets/images/Tim.png"  border="1"  alt="Your Name Profile Image">
        </td>
        <td>
            <h1>Your Name/h1>
            <h3>Student , Addis Ababa institute of Technology</h3>
            <p>Hi there , </p>
            <p>I am Your Name , i am expert <strong>Java</strong> Developer who has been working
on Java since <em>5 </em></em>years. I am passionate about working in different types of
programing languages </p>
        </td>
    </tr>
</table>
```

Working with Tables (ctnd..)

- Add a new table at the top of skill [for you to work with]
- **Note :** Add this element above **Top Skills**
  - Add hr
  - Add h3 : Work Experience
  - Add a row with two cells with the header of “Date”   and   “Job Title”   : th
  - Add a row with two jobs you have worked on [just assumption]
  - Apply the HTML5 structural components : thead , tbody

Working with Hyperlink

- Hyperlink
  - <a>   [inline element with closing tag]
  - This element create a hyperlink between Documents
  - Attributes : href(mandatory), target, download, More
- Other Document relationship tags
  - <base>, <link>, <script>

Working with Hyperlink(ctnd..)

- Above the Head lets create navigation
  - Create Horizontal line below and above the header info: hr
  - Create an empty contact.html file on the same level as index
  - Create two links inside the horizontal line
  - Copy the code from index then remove everything in the body except the header part

```
<hr>
<a href="index.html">Home</a>
<a href="contact.html">Contact</a>
<hr>
```

Working with Hyperlink(ctnd..)

- Creating Bottom Navigation (for you to do)
  - Create a bottom navigation with hr surrounded for nice separation on the bottom : **a**
  - The links should go to your social media : Facebook , tweeter, YouTube , ...

Working with Form

- User Input forms
    - <form> <input> [inline element with self closing tag]
    - This element create a form with different fields
    - Types : text , password , ..
    - Attributes[Form]: , method, action , more
    - Attributes[input]: disable, name , value
- HTML5
    - New types : email , date , time , tel, number, url
    - Attributes[Input] : required ,autocomplete, , pattern, more

Working with Form(ctnd..)

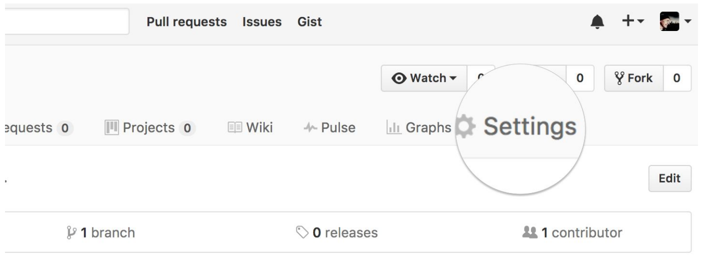
Inside the contact.html add information for receiving inputs from user

- Add Text " Contact Me With the following Information" : H3
    - Add the form ⇒ action: mailto:" your mail", method: post , enctype: utf-8
    - Add the form with **input** : text , email , submit , with **label**
    - add text area : <textarea>
    - add validation required

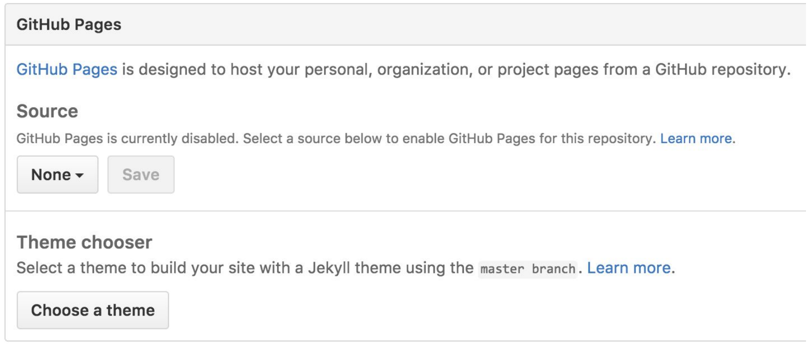
```
<form action="mailto:yourmail" method="POST" enctype="utf-8">
<label for="name">First Name: </label>
   <input type="text" name="fName" id="fName" required><br>

   <label for="name">Last Name: </label>

   <input type="text" name="lName" id="fName" required><br>

<label for="email">Email: </label>

<input type="email" name="email" id="email" required><br>

Leave Your Comment here ..<br>

<textarea name="comment" id="comment" cols="30" rows="10"></textarea>

<br>

<input type="submit" value="Submit">

</form>
```

Hosting in Github

1. Make sure you initialize , and perform a commit operation locally
2. Perform a git push to a repo called "**MyCv**" or any name you like
3. Go Head to the repo setting tab



4. Scroll down to the GitHub Pages section. Press source and choose the master branch.



GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Source

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. Learn more.

Theme chooser

Select a theme to build your site with a Jekyll theme using the `master branch`. Learn more.

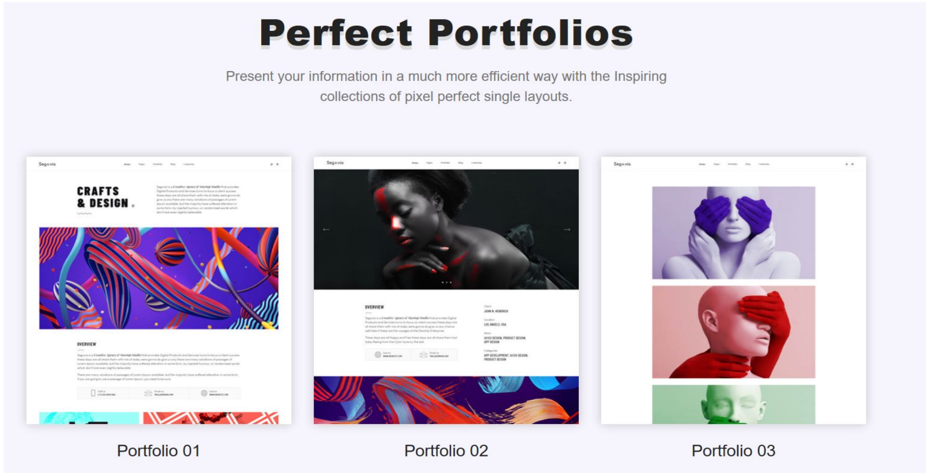
Miscellaneous Changes

- To make the website more interesting :
  - Adding | between the Navigation Items for good view
  - Adding a rounded image [Online Link] and updating the the image
  - Adding one br in each form element for adding more space between the form elements
  - Adding space between label and the element using &nbsp; element
  - Adding footer that shows the developer , below the last table
  - To Generate Copy Symbol :   &copy;
    - For More : Link

```
<hr>
<p align="center">Developed By Your Name. &copy;2020</p>
<hr>
```

Exercise (10pt)

1. Add icon on the head part using link tag inside the head tag : Link
2. Update the contact form to be displayed inside a table where each form take a new row [and **Label : Form** Element with two cells in a row]
3. Add more html5 input elements to the contact page
4. Use the pattern attribute to validate first and last name name , password
5. Add Gallery page [Update the navigation on the top to have **Gallery**]
   a. Three images in a row
   b. Min of 3 rows [You can use category to show what the images are about at the top of each row]
   c. use hr, figure with caption , and table [with table attributes to make it good looking]


Perfect Portfolios
Present your information in a much more efficient way with the Inspiring collections of pixel perfect single layouts.
CRAFTS & DESIGN
Portfolio 01
Portfolio 02
Portfolio 03

6. **Bonus+** : Embed Map on the contact page

Resource for Further Working

1. MDN Downloadable Cheat Sheet
2. HTML5
3. Client Side Form validation

ITSE-2192 Fundamental of web Design and Development

Cascading Style Sheet(CSS)

Lab Two

What we will Learn

- After completing this lab :
    - You will be able to work on basic css properties and values
    - Learn the major difference on CSS3 from earlier versions
    - Continues in improving the personal web app[CV] started on previous lab
- We will go through learning the css by continuing on the building of personal cv with different pages

CSS

- **CSS : Cascading Style Sheet**
  - It is a styling language but not a programing language
  - Current Version : CSS3
  - Defines the style and layout of a markup document to be rendered in a browser
  - CSS Syntax
    - Selector { property : value ;}
    - Selector can be element , class , id , more
- How to Embed :
  - **Inline :** when it is only one single element
  - **Internal :** applying the same style for multiple elements
  - **External**[recommended] : applying the same style on multiple pages

CSS Basics Setup

1. On the root folder[MyCv] create a file called **blog.html**
2. Create the HTML boilerplate using Emmet
3. Add the reference to external css in <head> section

   <link rel="stylesheet" href="./assets/css/style.css">

4. On **assets/css** folder create a file **style.css** **//naming can be any**

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" href="./assets/css/style.css">
   <title>Your Name / Blog</title>
</head>
<body>

</body>
</html>
```

Working with Color and Image

- Color
    - background-color : colorType
    - color : colorType
    - Color type => green(text) , #000000(hex), rgb(0,0,0) , rgba(0,244,233,0.1)
- Will helps as to apply change on background and foreground color of an an element respectively
- Image
    - background-images : url(“image path”)
    - background-repeat: no-repeat
    - background-size: cover/auto/contain
- Will helps as to apply change on background image of an element

Working with Color and Image(ctnd..)

- Inside the body tag lets re-do the navigation again
  - Add HTML5 Navigation : <header> <nav> <ul> <li>
  - Adding the three navigations with hyperlink (Home, Blog, Contact): <a>

```
<header>
    <nav>
     <ul>
         <li>
             <a href="index.html">Home</a>
         </li>
         <li>
              <a href="blog.html">Blog</a>
        </li>
         <li>
             <a href="contact.html">Contact</a>
        </li>
     </ul>
    </nav>
</header>
```

Working with Color and Image(ctnd..)

- Let's add background color for the whole page and the navigation
  - Change the body background : background-color : skyblue
  - Change the nav background: background-color : rgba(190,90,90,0.7)

```
/* applying background color with element selectors .. */
body{

   background-color: skyblue;
}
header
{
    background-color: rgba(119, 90, 90, 0.6);
}
```

Working with Color and Image (ctnd..)

- Changing Background image (for you to do)
    - Create a style tag(internal style) inside head of the contact.html
    - Change the background image of the page with no repeat

Working with Font

- Font
  - font-family: font1, font2,font3;
  - font-size : #Number
    - Measures : px . cm, mm, in , %, em, rem
  - font-style : italic / normal
  - font-weight : bold / [100 -900]

- Only fonts that are on the machine will be used by the browser

Working with Font(ctnd..)

- Inside the body tag lets re-do the navigation again
  - Add HTML5 Section with main div surrounded : <div id="content"><section><p class="title"><article><p class="date">
  - Add title , some lorem ipsum/ bacon Ipsum content and date

```
<div id="content">
<section>
<p class="title">

</p>
<article>

</article>
<p class="date">

</p>
</section>
</div>
```

Working with Font(ctnd..)

- Lets apply font(family, size and weight) on blogs that are posted
  - Apply serif font family on the section : font-family
  - Apply font size of 23px , 15px and 14px in the title , article and the date : font-size
  - Apply font weight of bold on the title : font-weight
  - Apply font style of italic on date : font-style

```
/*  applying font with element, class  selectors..*/
section {
   font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;

}
article {
   font-size: 15px;
}
.title {
   font-size: 23px;
   font-weight: bold;
}
.date {
   font-style: italic;
   font-size: 15px; }
```

Working with Text Styles

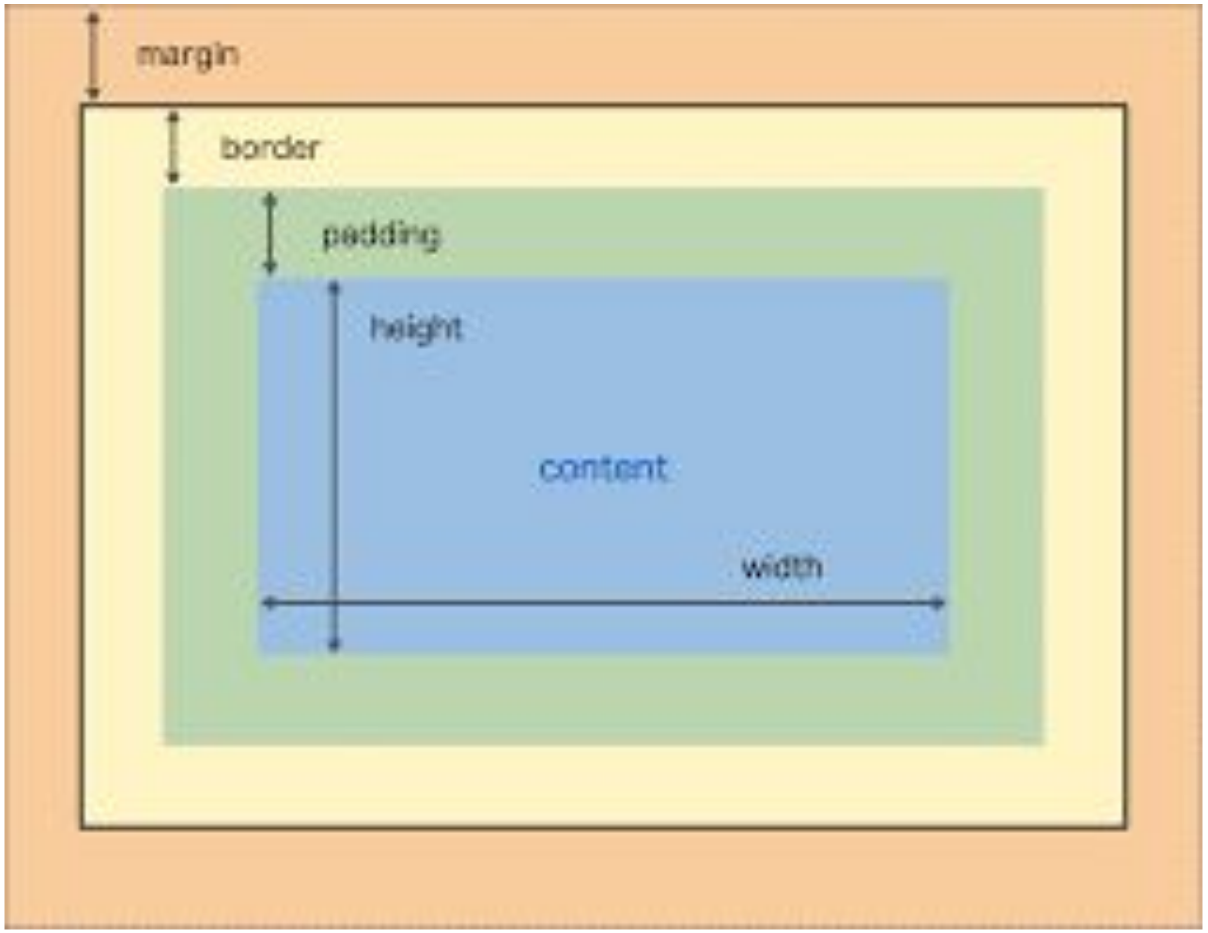
- Text
    - text-align: justify /center /right / left /
        - Deals with text alignment
- Line
    - line-height : #number
        - Deals with line spacing
- Character Case
    - text-transform : uppercase / lowercase / capitalize
        - Deals with the case of characters

Working with Text Styles(ctnd..)

- Lets apply text style on the font(alignment, case and line space) on blogs that are posted
  - Update the article alignment with justified style : text-align
  - Update the article line spacing with 25px : line-space
  - Update the date with uppercase : text-transform

```
/*  updating the alignment, height and case..*/
article
{
 ….
text-align: justify;;
line-height: 25px;
}
.date
{
 ….
text-transform: uppercase;
}
```

Review : The Box Model



Working with Border

- Border
    - border-width: justify /center /right / left /
    - border-color: colorType
    - border-style : solid / dotted / dashed
    - Shortcut
        - border: type size color ;
    - To be Be Specific
        - Border-top/bottom/left/right : type size color ;

Working with Border(ctnd..)

- Let's apply border on the body
  - Update the body with the border of type solid , with light gray color and size of 10px : border

```
/*  updating the body border.*/

body{

   ....
   border: 10px lightgrey solid ;
}
```

- Do you observe any problem on the content ??

Working with Margin and Padding

- Margin
  - margin: #number //all sides
  - margin: #number #number //top/bottom left/right
  - To be Be Specific
    - margin-top/bottom/left/right : #number ;
- Padding
  - padding: #number //all sides
  - padding: #number #number //top/bottom left/right
  - To be Be Specific
    - padding-top/bottom/left/right : #number ;

Working with Margin and Padding(ctnd..)

- Let's apply margin and padding on body to make the content somehow center , on the section for creating space for the content
    - Update the body with 10px margin : margin
    - Update the section with 2px of tip/bottom and 30px left/right: padding

```
/*  updating the body margin and section padding.*/

body{
…..
margin: 10px;
}
section
{
    …..
    padding: 5px 30px;

}
```

- Feel free to apply padding and margin of your own

Working with Links Style

- Display
  - display: inline/ block / inline-block
    - To Change the Block or inline nature of the tag
- Anchor tag style Pseudo – Class
  - text-decoration : none
  - To manipulation of color on this actions from user
    - a:link - a normal, unvisited link
    - a:visited - a link the user has visited
    - a:hover - a link when the user mouses over it
    - a:active - a link the moment it is clicked

some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

Working with Links Style(ctnd..)

- Lets apply style on the navigation
  - Change the lis in the ul inside navigation to display as inline : display
  - Remove the links underline using none value : text-decoration
  - Changing the pusdo-class of the a : active , hover , visited

```
/* changing the nav ul to inline */
nav > ul > li{

    display: inline-block;
    padding: 10px 3px;
    font-size: 20px;
}
/* removing the underline  */
nav > ul > li > a
{
    text-decoration: none;
}
```

```
/* changing the hover on the li */

nav > ul > li:hover
{
    background-color: #ffffff;

}
```

- Can you apply the other properties to <a>?

Working with Position

- Position
    - position : static / relative / absolute / fixed
    - top/bottom/left/right : #no px ;
        - The default layout is **static**
        - **Relative :** based on the relative to where it is supposed to be when it is rendered by the HTML
        - **Absolute:** relative to the container
        - **Fixed :** no movement Even the page scrolls

Working with Position(ctnd..)

- Lets apply style on the navigation
  - Change the the navigation to stick on the head even on scroll : position
  - Add margin top on the **content div** : margin-top
  - remove the border from body //just for the look

```
/* changing the nav to stick at the top */
header
{

    position: fixed;
    top: 0;
   width: 98%;
   background-color : rgba(125,125,125,0.7);

}
```

Working with Floating of text

- Used to float a text around an element mostly image
  - float : right / left / top / bottom
  - clear : clear
- Don’t use this property unless you want to float a text around an element

Working with Floating of text(ctnd..)

- Lets apply floating in the heading information
  - Add the external link in the **contact.html**
  - Remove the table from the heading content
  - Add a div after the the bio with id of clear : <div id = “clear”>

```
/* Image floating  and clearing*/
img
{
    float: left;
    border-radius: 100%;          // making circular border around
    border-style: hidden;
    width: 250px;
    height: 250px;
    margin: 0px 15px 0px 3px;
}
#clear
{
    clear: left;
}
```

Working with Floating of text (ctnd..)

- Making update on index and contact (for you to do)
  - Perform an update on the remaining part of the contact.html : the form using box model
  - Replace the hr with border bottom : border-bottom [update all pages]
  - Update the form layout with : relative position , margin-bottom , width and height[only for the submit] , try to use attribute selector
  - Perform an update on the index.html Using box model

Exercise(10pt)

1. **Creating a 3*3 image gallery [To be Done Here]**

Re-Develop the image gallery with 3*3[**three images in a row and with three rows**] with the following steps

a. Create the file gallery.html

b. The CSS Rules

  i. **One Text**

    1. Create a heading(h3) “Yout Name / Gallery” with all caps [This must be done by css not the text inside the html]
    2. Making use of appropriate font metrics on the body[**size, style, family**]
    3. Use the width of the text to be relative to root element
    4. bottom border and having a good bottom padding

  ii. **On The Image** [**You Must Choose Good Image**]

    1. Use the flot for making the images flot one after the other with no gap
    2. Use the width property with % to divide equal for three image on one row [30%]
    3. Apply margin property to share the remaining 10% width equally by the images [10/6 %]
    4. Make sure the images does not inherit style from the body

Exercise(10pt)

2. On the CV App you started , demonstrate the following concepts by identifying the necessary areas [To be Done Home]

Try to follow the steps

a. Read the concepts of Flexible Box Layout , and Media Queries that are used for creating a responsive layout
   i. Links for Flex : Link 1, LInk 2 , Links for Media Queries : Link 1 , Link 2

b. Try to figure out how you can integrate this concepts in the cv application we have been working so far
   i. **Hint :** On the navigation , body of the text

c. Demonstrate the usage by applying on specific areas you have chosen

ITSE-2192 Fundamental of web Design and Development

Bootstrap 4 Basics

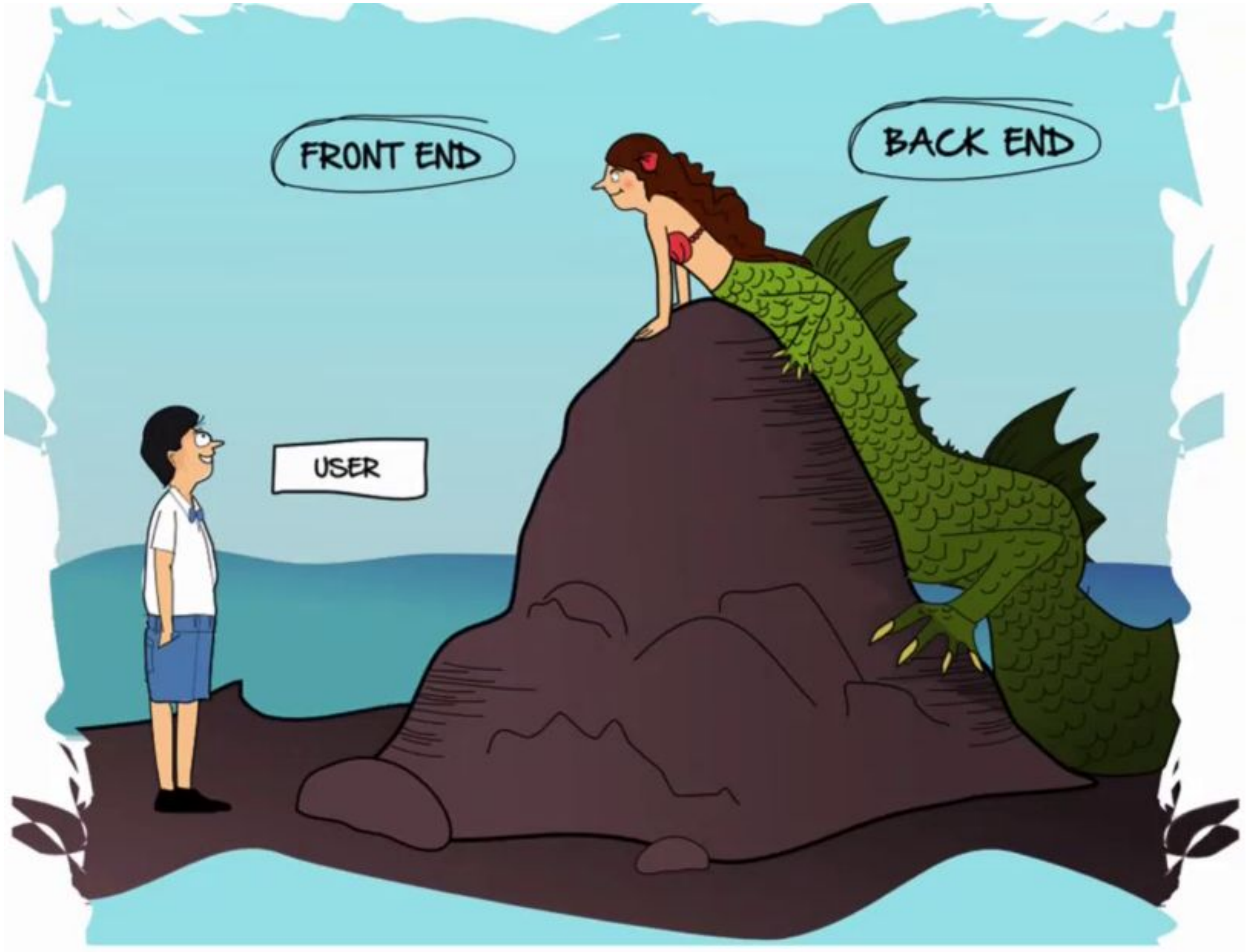
Lab Three

What we will Learn

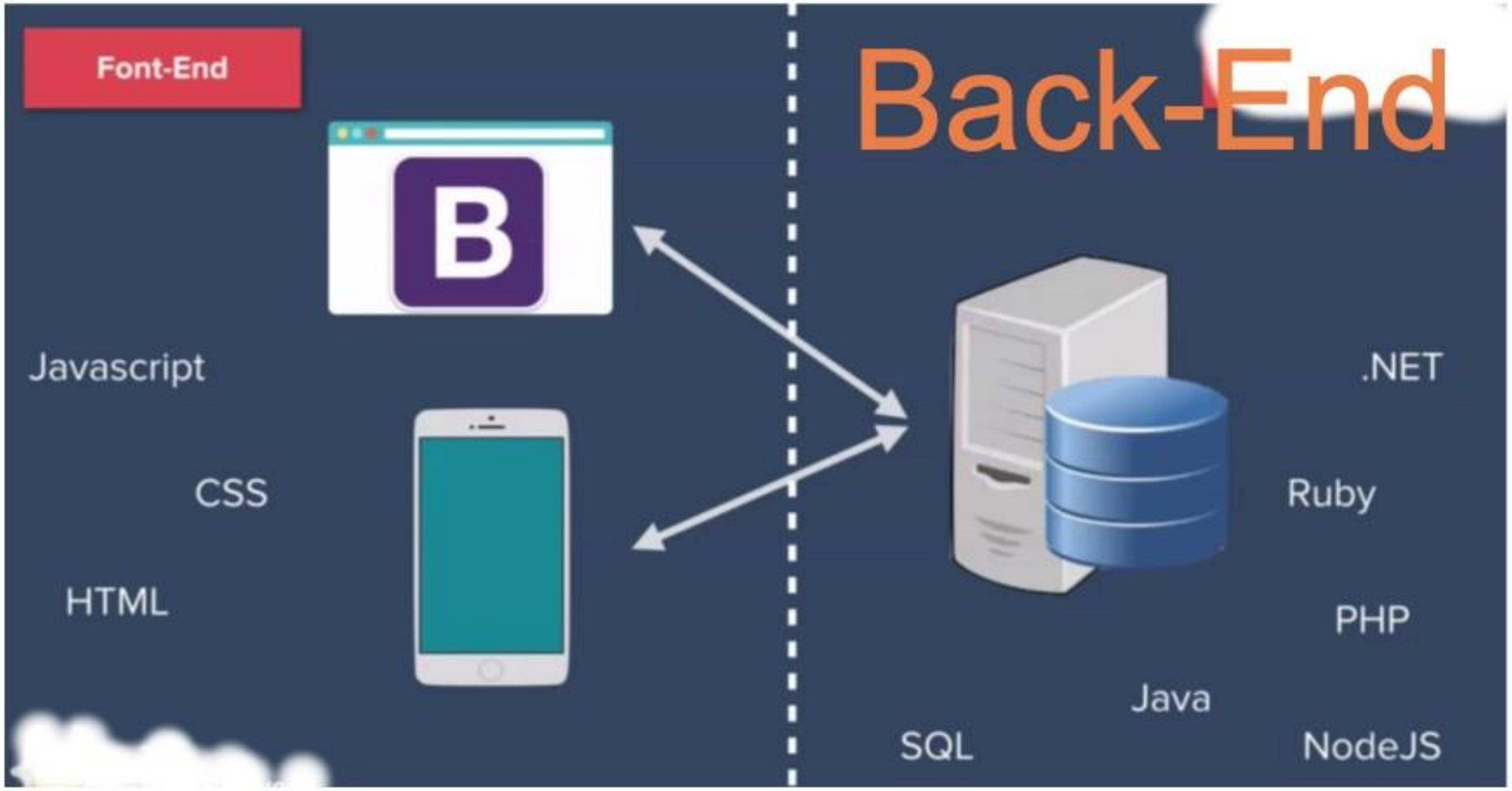
- After completing this lab :
  - Grasp the concept of Front End/Back End Basics
  - You will be able to learn basics of Bootstrap[client side library]
  - Learn the major changes on Bootstrap 4
  - Start a new project that will be used for showcase of shoes
- We will go through learning bootstrap by building a shoe selling website

- **Full Source Code :** Link

Front End Basics



Front End Basics(ctnd..)



Basics of Bootstrap

- Bootstrap is the most popular HTML , CSS and JS Framework for developing responsive , mobile first project on the web

“From Bootstrap Official Site ”

- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins

- Bootstrap also gives you the ability to easily create responsive designs

About Bootstrap

- Bootstrap was developed by **Mark Otto and Jacob Thornton** at Twitter, and released as an open source product in August 2011 on GitHub
- In June 2014 Bootstrap was the No.1 project on GitHub!

- How to incorporate bootstrap in a Website
  1. Bootstrap Compiled Download : Link
  2. Bootstrap CDN : Including it from a CDN (Content Delivery Network).
     a. Required File : Jquery

Quiz : Which one do you think better ?? why ?

Why BootStrap ?

1. **Easy to use:** Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
2. **Pre Styled elements** , unlike using generators for styling different element like button (Link)
3. **Open source** and support fast development
4. **Grid System** : for creating a mobile responsive website
5. Creating Responsive **Navigation**
6. **Very popular** : The Second top(⭐) project in github in 2018 now it is Seventh :) but still popular
    - **Some Competitors** : Semantic UI , Bulma, Pure.css, Tailwind CSS...More

What is new in Bootstrap 4

1. Bootstrap v3 uses **Less** for source CSS files. With Bootstrap v4, **Sass** is now used for source CSS files
2. The primary CSS unit is now **rem** rather than **px**. However, pixels are widely used for media queries.
3. An improvement has been made to make it a 5 grid tier system, ***xs*, *sm***, ***md***, ***lg***, and ***xl***. The new grid tier, *xl*, extends the media query range all the way down to 544px.
4. With Bootstrap v4, **Flexbox** is enabled out of the box
5. Bootstrap v4 dropped support for **panels**, **thumbnails**, and **wells** in favor of the new **card** component built with Flexbox
6. Bootstrap includes a plethora of shorthand responsive margin and padding utility classes to modify an element’s appearance
7. The global font-size has been increased from **14px** to **16px**
8. Bootstrap 4 dropped the **Glyphicons icon font.** Suggested options are **fontAwesome** and **Octicons**.

For More Check : LInk

Setup Bootstrap

1. Download bootstrap 4 from : Link
2. Create a folder called **ShoePurchaseWebsite**
3. Create the following folder structure in the root folder:
    - assets/images //where will put images
    - assets/css //where will put the bootstrap and other css files
    - assets/js //where will put the required javascript file

4. Add the reference to external bootstrap css file in <head> section
5. Adding javascript files inside body(at the last line) according to the order [will do that in a moment]

Let's Create Navigation

First we need to create a navigation for our website with the following links : Services , About , Login and Register

1. Create the nav bar with default layout

   <nav class="navbar navbar-expand-md navbar-dark bg-dark sticky-top"> </nav> // view the output

   "navbar-expand-md" : means when does the nav becomes vertical aligned or collapse [sm/md/lg/xl]

   can try different background colors and style

2. Add a Logo on the left corner [Inside the nav]

   <a class="navbar-brand" href="index.html">AdShoe</a>

Note we will revisit with icon later on

More Ref : Link

Let's Create Navigation (ctnd..)

3. Create two showcases navigations in the right side [after the logo]

4. Create login and register link at the left using margin left: **ml auto ;**

```
<ul class="navbar-nav ">

  <li class="nav-item">
    <a class="nav-link" href="">Service</a>
   </li>
   <li class="nav-item">
      <a class="nav-link" href="">About</a>
    </li>

 </ul>
```

```
<<!-- To the left side side  -->
<ul class="navbar-nav ml-auto">

<li class="nav-item">
            <a class="nav-link" href="">Login</a>
    </li>
    <li class="nav-item">
         <a class="nav-link" href="">Register</a>
     </li>

 </ul>
```

Let's Create Navigation (ctnd..)

Let's make the navbar to be responsive to the viewport

1. Surround the ul with a div that has collapse property

   <div class="collapse navbar-collapse" id="colNav">

2. Let's add the collapse button inside the navbar-header [after the logo]

```
<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#colNav"
aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">

      <span class="navbar-toggler-icon"></span>

 </button>
```

Note : This togler requires bootstrap.js , popper js [*with jquery dependency*]

The order matters: Jquery js , bootstrap.bundle.js(popper), bootstrap js

Creating a Header [jumbotron]

Let's create a **heading /Hero** for our shopping website using jumbotron element

1. Create a divs with class jumbotron
2. Put some content with **h4** and **p**
3. Add two link buttons that point to about and service page

```
<div class="jumbotron" >

     <h4>Welcome to Addis Shoe</h4>

     <p>We Provide various types of shoes around the city , enjoy viewing our website
      Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident at inventore ,</p>

       <hr class="my-4">

       <a  class="btn btn-outline-info btn-lg" href="about.html">About Us</a>
       <a  class="btn btn-info btn-lg" href="service.html">Service</a>

  </div>
```

More Ref : Link

The Container Class

For making the contain in the center we can use a container class

1. Add a dive with a class **container** surrounding all the containte below the navigation.
2. use the container fluid for making it full width with some padding and margin
3. You can revisit the jumbotron to be fluid by using jumbotron-fluid

```
<div class="container-fluid" >

//The content here


  </div>
```

The Grid System

- In the most basic terms, a grid is a structure comprising a series of lines (vertical or intersecting) that divide a page into columns or modules
- This structure helps designers to arrange content on the page : Read More
- Bootstrap's grid system uses a series of containers, rows, and columns to layout and align content.
- It's built with flexbox and is fully responsive
- Bootstrap's grid system allows up to **12 columns** across the page
- If you do not want to use all 12 column individually, you can group the columns together to create wider columns

The Bootstrap 4 grid system has five classes:

- **.col-** for small mobile phones (devices with resolutions **<576px**);
- **.col-sm** for larger mobile phones (devices with resolutions **≥ 576px**);
- **.col-md** for tablets (**≥768px**);
- **.col-lg** for laptops (**≥992px)**;
- **.col-xl** for desktops (**≥1200px**)

The classes above can be combined to create more dynamic and flexible layouts

The Grid System(ctnd...)

Lets create a features section below our heading jumbotron of one row with three columns that takes ⅓ of the screen when only on large screen (**lg**)

1. Add emojis that represents the heading [Beautiful, Comfort and Guarantee]
   a. **Note:** we will replace with icon on later stages
2. Put two <p> one for feature Term , the other for explaining the feature
   a. apply class of font-weight-bold [Term p]
3. Puting image with class of img-thumbnail [Later Stages]

For Emoji : Install vs code extension :emjiisense

More Ref on Grid : Link

The Grid System (ctnd..)

```
<!-- Features of Our Shoe -->
 <div class="row text-center px-4">

   <div class="col-lg-4">
             <p>✅✅</p>
             <p class="font-weight-bold"> Beauty In your Eye</p>
             <p>Our Products have the best aesthetics in the world that can attract any body</p>
     </div>

      <div class="col-lg-4">
             <p>💁‍♀️💁‍♀️</p>
             <p class="font-weight-bold"> Comfortable for your Foot</p>
             <p>Our Products have the the most comfort for you and your health</p>
        </div>

         <div class="col-lg-4">
             <p>🦍🦍</p>
             <p class="font-weight-bold"> Guarantee for Years </p>
             <p>Our Products have one year warranty if you want to change of your shopping</p>
          </div>
</div>
```

Working with Slideshow

Carousel : is a slideshow component for cycling through elements—images or slides of text—like a carousel

1. Below the *feature Component* add the following slideshow

```
<!-- Some of the products from store -->
<div id="MySlide" class="carousel slide my-4" data-ride="carousel">
      <div class="carousel-inner">
            <div class="carousel-item active">
               <img src="assets/img/Shoe1.jpg" class="d-block w-100" alt="...">
            </div>
            <div class="carousel-item">
               <img src="assets/img/Shoe2.jpg" class="d-block w-100" alt="...">
            </div>
            <div class="carousel-item">
               <img src="assets/img/Shoe3.jpg" class="d-block w-100" alt="...">
            </div>
       </div>
</div>
```

More Ref : Link

Working with Slideshow(ctnd...)

To add indicator and controls [Prev , Next]

```
<!-- Some of the products from store -->
<div id="MySlide" class="carousel slide my-4" data-ride="carousel">
<ol class="carousel-indicators">
      <li data-target="#MySlide" data-slide-to="0" class="active"></li>
      <li data-target="#MySlide" data-slide-to="1"></li>
      <li data-target="#MySlide" data-slide-to="2"></li>
    </ol>

  <div class="carousel-inner">   …. . ...</div>

<a class="carousel-control-prev" href="#MySlide" role="button" data-slide="prev">
   <span class="carousel-control-prev-icon bg-info" aria-hidden="true"></span>
   <span class="sr-only">Previous</span>
 </a>
 <a class="carousel-control-next" href="#MySlide" role="button" data-slide="next">
   <span class="carousel-control-next-icon bg-info" aria-hidden="true"></span>
   <span class="sr-only">Next</span>
 </a>

</div>
```

Working with Card

Card : is a flexible and extensible content container. It includes options for headers and footers, a wide variety of content, contextual background colors, and powerful display options

1. Create a file called services.html that will contain the same code as index except the content
2. Add a text with display of 4 : “Shoe Expo” below the navigation
3. Put a horizontal rule : <hr>
4. Create a div with class container
5. Create a grid of three row by three col
6. Lets Create a card in each row that is responsive [on mobile that stack one col]

More Card Ref : Link

Working with Card(ctnd...)

```
<p class="display-4">Shoe Expo</p>
<hr class="mx-auto">

<div class="container-fliud ml-auto mt-2"">

 <div class="row text-center” >

  <div class="col-md-4">

   <div class="card">
    <img class="card-img-top img-fluid" src="./assets/images/shoe/pic1.png" alt="">
    <div class="card-body">
     <h5 class="card-title">Snekear Shoe</h5>
     <p class="card-text">Lorem ipsun repellendus sit, tenetur quod
      tione hic ad totam quo sed magni saepe? Quos obcaecati dolor doloribus magnam
      illo.</p>
     <a href="" class="btn btn-primary btn-lg">Buy</a>
    </div>
   </div>
  </div>

   </div>
 …….(contiue here )
</div>
```

Working with form

Form : Form element is style by default in bootstrap but we can use different class to maintain the layout

Let's create a registration form

1. Create a file called register.html that will contain the same code as index except the content
2. Place a header text : h3 “Create Account”
3. Place a horizontal ruler : hr
4. Place a form with fname , lname , email , password , confirm password , and submit button
5. Surround with a div of form-group class on each element
6. Use a form-control class on each input
7. Surround the form with div of class container to create some margins and padding

More Form Ref : Link

Working with form(ctnd...)

```
<!-- Registration -->
   <h3>Create an Account</h3>     <hr>
   <div class="container">
      <form action="">
         <div class="form-group">
            <label for="fname">First Name : </label>
            <input class="form-control" type="text" name="fname" id="fname">
         </div>
         <div class="form-group">
            <label for="sname">Last Name : </label>
            <input class="form-control" type="text" name="lname" id="fname">
         </div>
         <div class="form-group">
            <label for="memail">Email : </label>
            <input class="form-control" type="email" name="memail" id="memail">
         </div>
         <div class="form-group">
            <label for="pass">Password : </label>
            <input class="form-control" type="password" name="pass" id="pass">
         </div>
         <div class="form-group">
            <label for="pass2">Confirm Password : </label>
            <input class="form-control" type="password" name="pass2" id="pass2">
         </div>
         <button type="submit" class="btn btn-primary btn-lg">Submit</button>
      </form>
   </div>
```

Working with External Styles

Now we can use font awesome to replace our icons

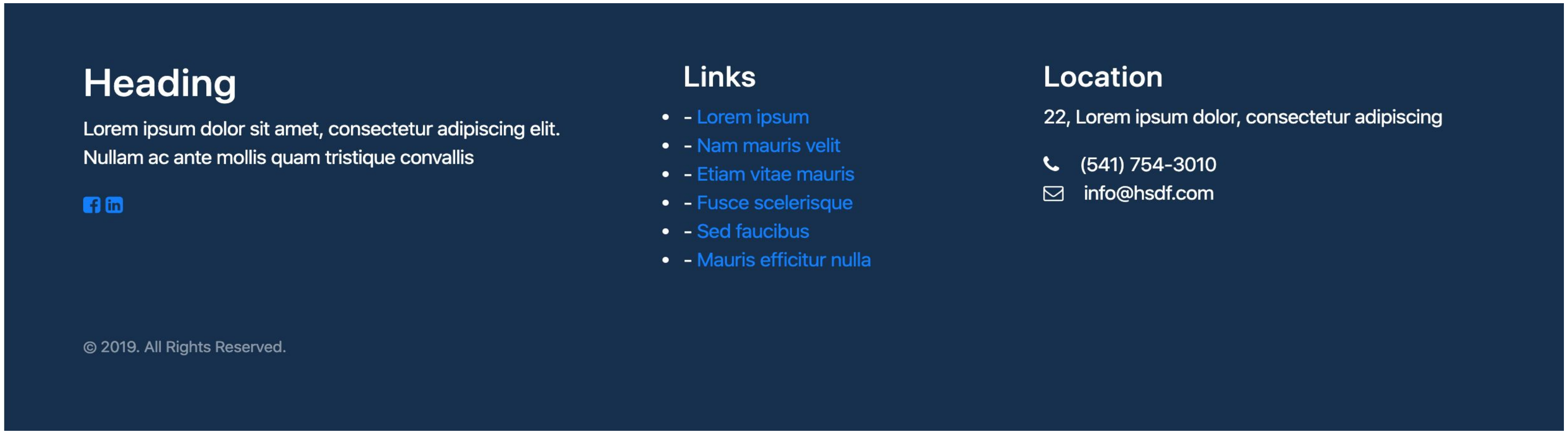
1. Go to font awesome page : LInk
2. Search for free icons of your choice
3. Place the icon with i tag
4. Make sure you incorporate the font awesome CDN

Working with Google Fonts

1. Go to Google Font page : LInk
2. Search for font of your choice
3. Make sure you incorporate the font CDN in the page you want to use
4. write your own css to use that font family

Exercise

1. Create a footer with social icons , with a grid system of three **cols** from ***medium screen on and use symbol/icons from font awesome***



Heading

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam ac ante mollis quam tristique convallis

Links

- - Lorem ipsum
- - Nam mauris velit
- - Etiam vitae mauris
- - Fusce scelerisque
- - Sed faucibus
- - Mauris efficitur nulla

Location

22, Lorem ipsum dolor, consectetur adipiscing

(541) 754-3010

info@hsdf.com

© 2019. All Rights Reserved.

2. Create a login form in **login.html** [*keep the header and footer*]
3. Demonstrate the usage of flex box on the navigation to move the navigation items to move to left when it collapse
4. Create a **buy.html** that allows the user purchase the shoe by passing the information to the page and choosing payment method , with checkout

Add On [Reading Assignment]

- Bootstrap has Released new Version few months ago : few things has been changed on Version 5 , so try to explore what has changed , apply it on the exercise .
- **Reference to Read**
  - Link 1
  - Link 2
  - Link 3
  - Documentation Link

ITSE-2192 Fundamental of web Design and Development

Javascript Basics

Lab Four

What we will Learn

- After completing this lab :
  - You will be able to work on basic of javascript
  - Learn the major concepts with javascript
  - Work on a sample personal database and do some staff
- We will go through learning basic javascript working on a sample project on the console

- **Full Source Code :** Link (*Will Use this Repo for All Upcoming Javascript Lab Sessions)*

Setup Javascript

1. Create a folder called **PersonalDatabase**
2. Create the following folder structure in the root folder:
   - assets/images //where will put images
   - assets/css //where will put css files
   - assets/js //where will put javascript files
   - index.html //html file for incorporating and testing js

3. Create **app.js** inside the js folder //This is where we write js
4. Add the reference to external js file in <body> section at last line
   <script src="./assets/js/app.js"></script>

Quiz : *Why do you think we add JS in the body not the Head section ??*

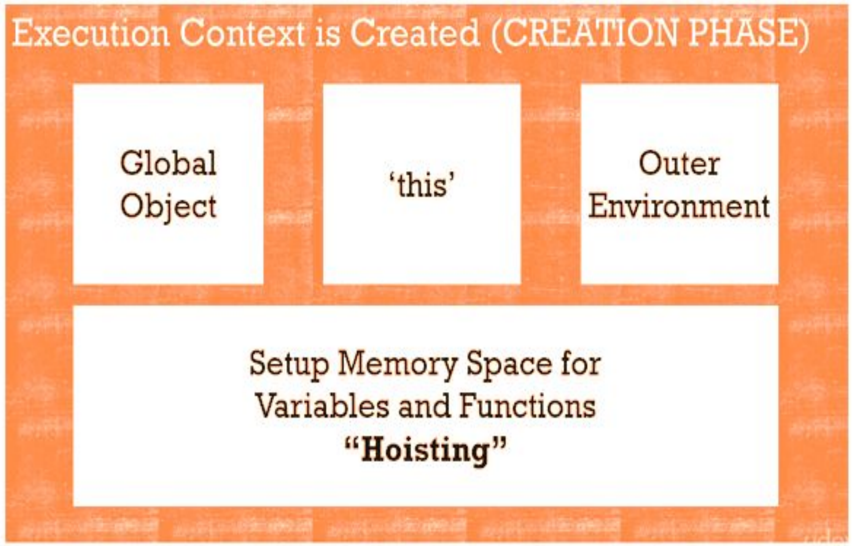
Setup Javascript (ctnd...)

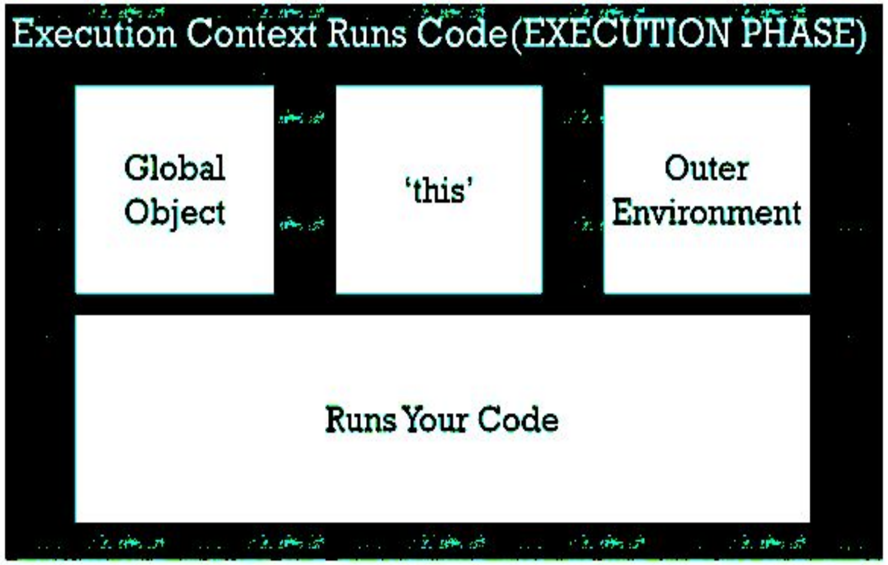
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Personal Database</title>
</head>

<body>

    <h1>Javascript Demo on Console OutPut !!!</h1>

    <h3>The Program Does the Following : </h3>
    <ul>
        <li>Take Input From User </li>
        <li>Display User Data on the Console </li>
    </ul>

    <!-- Adding External Js  -->
    <script src="assets/js/app.js"></script>

</body>

</html>
```

How Javascript Work

Once you set up the files run the program and lets see what happens in the javascript Engine

- Global Execution Context is created **[Creation and Execution Phase**]
- Function Execution Context is created **[Creation and Execution] Phase**]


Execution Context is Created (CREATION PHASE)
Global Object
'this'
Outer Environment
Setup Memory Space for Variables and Functions "Hoisting"



Reference Article : Link 1 , Link 2

Visualizer : Link

How Javascript Work (ctnd..)

Let's view the Global Object

1. **window [this]**
2. This object has several objects attached to it as key value pair
3. check the following methods

```
window.document                    // to view the DOM

window.location                    //to view the path
```

Will return to this in depth on Next Lab [BOM and DOM]

Javascript Input/Output

In javascript we can to interact with the html document as input/output

Input :

- **window.prompt(“String Message”)** 👈👈👈👈
- Form Elements [**document.forms["myForm"]["fname"].value**]

We can "display" data in different ways :

- **window.alert()** 👈👈👈👈
- **document.write()**
- **innerHTML [document.getElementById("demo").innerHTML]**
- **console.log()** 👈👈👈👈

Will use few of them on this lab (👈👈👈👈)

Working with Variables and Data Type

Let's create variables with **var , let and const [Let's use var here]**

1. Create four variables firstName , last Name, profession and age
2. Take the values from user by **prompt** and assign them to the variables declared
3. View it in the console the detail of the individual

```
// Declaring variables
// You can check this Variables are Available or not under the window object
var firstName;
var lastName;
var age;
var job;
// Receive the values from input
firstName = prompt("Enter Your First Name");
lastName = prompt("Enter Your Last Name");
job = prompt("What is Your Profession ?")
age = prompt("Enter Your Age");
// Display  the result on console from input
console.log("Here is your Profile ")
console.log("Full Name: " + firstName + " "+lastName);
console.log("Profession : " + job);
console.log("Age : " + age + " " + "years old");
```

Working with Operators and Conditional Statement

Let's check if the user is eligible to participate in county level election

1. Change the age to number : Number, parseInt, (-,/,*)
2. Check is the person is above 18 , place the value of isEligibleToVote to true otherwise false
3. Add the display for Eligibility to vote

```
...
//Age variable for Holding Number Value
let tempAge;
//Eligible to vote
let isEligibleToVote;
...
tempAge = parseInt(age);
// check Eligibility
if (tempAge >= 18) {
    isEligibleToVote = true;
}
else {
    isEligibleToVote = false;
}
....
console.log("Is Eligible to Vote : " + isEligibleToVote);
```

Working with Array and Loop

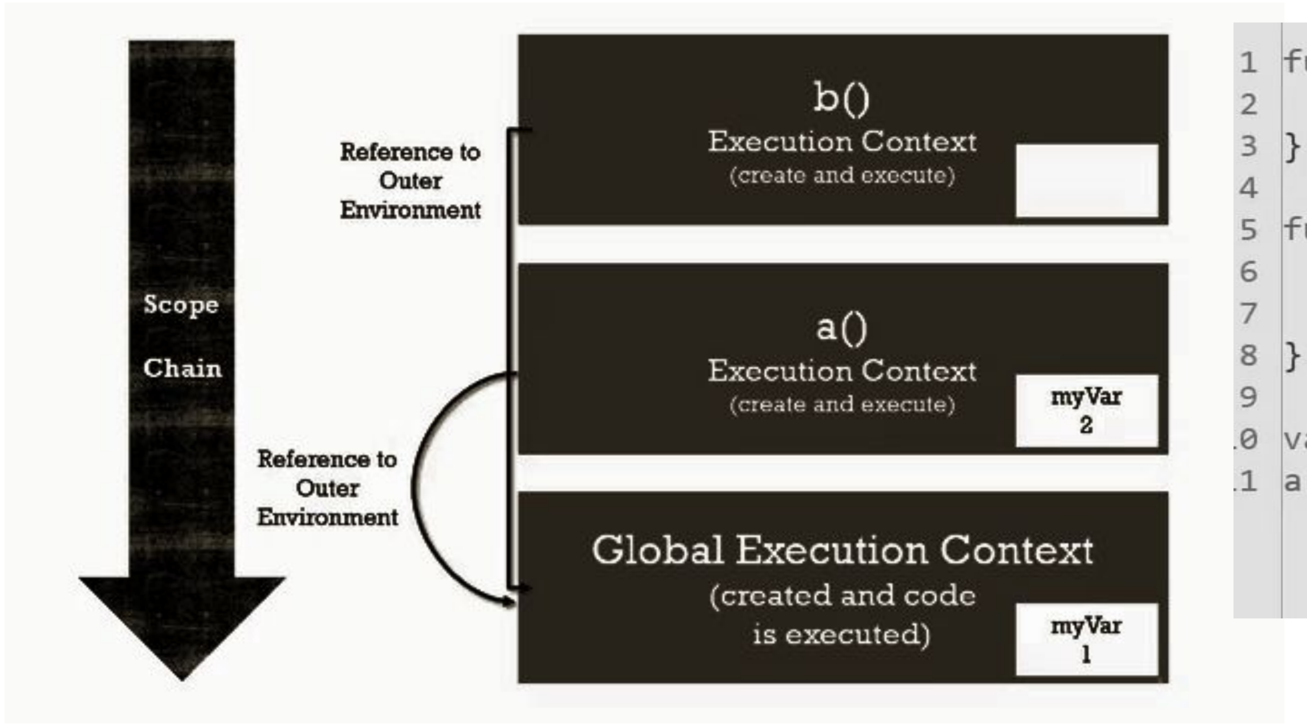
Let's receive and display family member

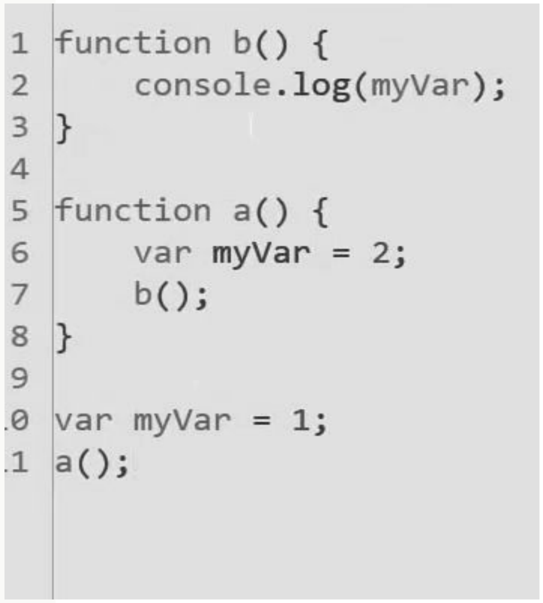
1. Create the empty array : **new Array() ;** and numberOfFamily variable
2. Receive the family number and the family List
3. Loop over the array to receive family members using **for** loop
4. Display the family members with **foreach** below the eligibility

```
….
//family storage array
let familyMember = new Array();
//number of family
let numberOfFamily;
….
numberOfFamily = prompt("Number of Family  ? ");
//Receiving the family number
for (let i = 0; i < parseInt(numberOfFamily); i++) {
    familyMember[i] = prompt("Your Family  Members " + (i + 1));
}
….
console.log("Family Members ");
//Displaying the family member with foreach
familyMember.forEach(function(member) {
   console.log("Family Member  " + (index + 1) + " : " + member);
});
```

Working with Functions

Lets review the Lexical Environment , execution context and Scope Chain





```
function b() {
    console.log(myVar);
}

function a() {
    var myVar = 2;
    b();
}

var myVar = 1;
a();
```

Working with Functions (ctnd...)

Lets Create a function that calculates the age

1. Change the variable age to birthYear
2. Create a function that take the birth date and return that age : **new Date().getFullYear();**

```
....
var birthYear;          //Changing the var age to birth Year
...
//Some changes
let tempAge = ageCalc(birthYear);
birthYear = prompt("Enter Your Birth Year");

console.log("Age : " + tempAge + " " + "years old");

//age calculator function
function ageCalc(birthYear)
{
    return new Date().getFullYear() - birthYear;

}
```

Working with Functions (ctnd...)

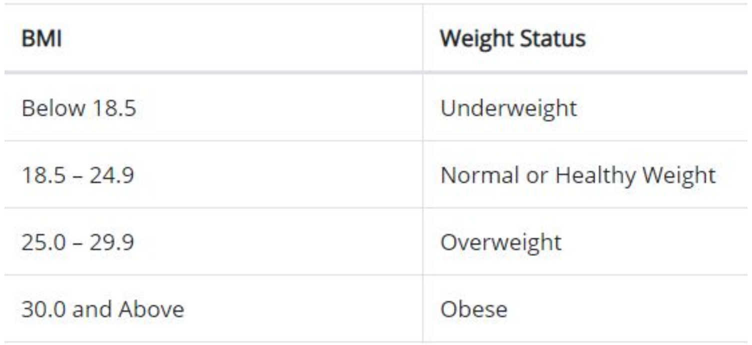
Lets Create a another function that calculates the BMI (for you to do)

1. Create two variables called **weight and Height**
2. Create a **function expression** called **calcBmi**
3. Use the following formula

$$BMI = \frac{weight\,(kg)}{height^2\,(m^2)}$$

```
//bmi calculator
let calcBmi = function(weight, height)
{
}
```

4. Round the Bmi value to to **number.toFixed(**after decimal) , since the value is change to string change it to Number
5. Make a decision for **bmi values** :



BMI	Weight Status
Below 18.5	Underweight
18.5 – 24.9	Normal or Healthy Weight
25.0 – 29.9	Overweight
30.0 and Above	Obese

To calculate the square : you can use Math.pow(height, 2) or just multiply

Working with Functions (ctnd...)

Let's put the console outputs inside self invoking function [IIFEs]

```
// Adding Self Invoking Function Expression
(function() {

    console.log("Here is your Profile ")
    console.log("Full Name: " + firstName + " " + lastName);
    console.log("Profession : " + job);
    console.log("Age : " + tempAge + " " + "years old");
    console.log("Is Eligible to Vote : " + isEligibleToVote);

    console.log("Family Members ");

    //Displaying the family member with foreach
    familyMember.forEach(function(member, index) {
        console.log("Family Member  " + (index + 1) + " : " + member);
    });

    // call to bmi calculator
    calcBmi(weight, height);
})();
```

Working with Objects

Object : Key value pair in simple terms

1. To Create Object
    a. **const name = {key : value, key : value }**
    b. **const name = new Object();**
2. To Access Values
    a. **objecname.property/key**
    b. **objectName["property/key "]**

Review Setup

1. Create **withObj.js** inside the **js folder** **//to put the object based code**
2. Replace the script link on index.html to **withObj.js**

Working with Objects (ctnd...)

Let's review our program by making the profile to be an object called person

1. Create a personProfile object with the properties listed earlier
2. Attache the properties with empty value , and the two functions

```
// Declaring Object +  Remove the variables and modify with the object
let personProfile = {
    firstName: "",
    lastName: "",
    birthYear: "",
    job: "",
    age: "",
    isEligibleToVote: false,
    familyMembers: [],
    weight: "",
    height: "",
    ageCalc: function() { return this.age = new Date().getFullYear() - this.birthYear; },
    checkVote: function() {
        let tempAge = this.ageCalc();
        if (tempAge >= 18) { this.isEligibleToVote = true; } else { this.isEligibleToVote = false; }
    },
    calcBmi: function() {
    }
};
```

Working with Objects (ctnd...)

Update the input and the loop with the object property access method :
personProfile.property

```
// Receive the values from input and assign to object properties
personProfile.firstName = prompt("Enter Your First Name");
personProfile.lastName = prompt("Enter Your Last Name");
personProfile.job = prompt("What is Your Profession ?");
personProfile.birthYear = prompt("Enter Your Birth Date");
personProfile.weight = prompt("Your Weight in Kg  ? ");
personProfile.height = prompt("Your Height in M  ? ");
let numberOfFamily = prompt("Number of Family  ? ");

//Receiving the family number
for (let i = 0; i < parseInt(numberOfFamily); i++) {
    personProfile.familyMembers[i] = prompt("Your Family  Members " + (i + 1));
}
```

Working with Objects (ctnd...)

Add the two functions to run and the variables with the object property access method **personProfile.Property**

```
// call age and check vote fun
personProfile.ageCalc();
personProfile.checkVote();
// Adding Self Invoking Function Expression
(function() {

    console.log("************************************************************")
    console.log("Here is your Profile ")
    console.log("Full Name: " + personProfile.firstName + " " + personProfile.lastName);
    console.log("Profession : " + personProfile.job);
    console.log("Age : " + personProfile.age + " " + "years old");
    console.log("Is Eligible to Vote : " + personProfile.isEligibleToVote);
    console.log("Family Members ");
    //Displaying the family member with foreach
    personProfile.familyMembers.forEach(function(member, index) {
        console.log("Family Member  " + (index + 1) + " : " + member);
    });
    // call bmi calculator
    personProfile.calcBmi();
    console.log("************************************************************")

})();
```

Working with Objects (ctnd...)

Let's attache the BMI function with the personProfile object (for you to do)

1. Add the bmi method in the object
2. Invoke the method outside the object
3. Change the name you used to input and output using the object access method : **personProfile[‘bmi’]**

Exercise(5pt)

Creating a calculator that works with input from user

1. Create four functions [add, mult, div , sub] ..can add more [max, min , average , square,….]
2. Create an init **Immediately Invoking function expression(IIFEs)** that contain all code apart from the 4 functions
3. Allow user to input of the service option and the values for each service
4. Make the addition and multiplication to work on more than two parameters [**use array**]
5. For Division service check if the denominator input is different from zero

Extra Work (5pt)

Creating a Banking System that works with input from user

1. Create four functions [Deposit, Withdraw, Balance, Transfer] ..can add more
2. Create an init **Immediately Invoking function expression(IIFEs)** that contain all code apart from the 4 functions
3. Allow user to input of the service option and the values for each service
4. Make a rule on Withdrawal where a user can not withdraw if he/she reaches certain amount [Max and Min]

ITSE-2192 Fundamental of web Design and Development

DOM and BOM

Lab Five

What we will Learn

- After completing this lab :
    - You will be able to understand how the DOM Works
    - Learn how to manipulate the DOM
    - Learn how to use the BOM Methods
    - Work on a sample Task Manager App
- We will go through learning DOM and BOM by working on a Task Manager app

- **Full Source Code :** Link (*Lesson 02 [Lab 05]*)

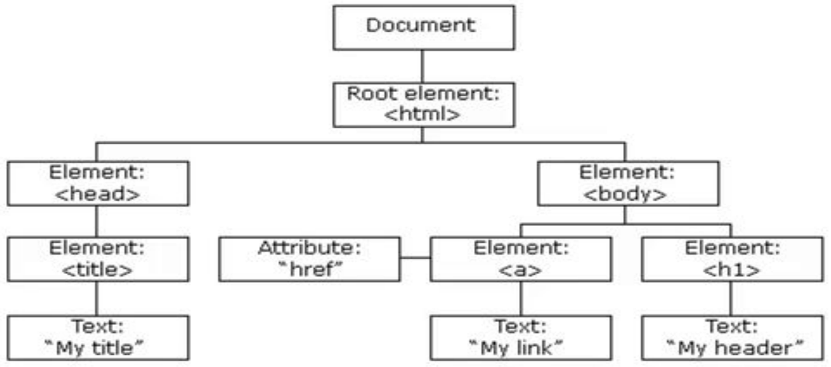
What is DOM ?

- **Document Object Model**
- Tree of **Nodes/Elements** created by the browser
- Javascript can be used to **read/write/manipulate** to the DOM
  - DOM **methods** are actions you can perform (on HTML Elements).
  - DOM **properties** are values (of HTML Elements) that you can set or change.

- **Object Oriented Representation**

👇 👇

```
document.getElementById("demo").innerHTML = "Hello World!";
```



What Can We Do ??

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- Change all the HTML elements in the page
- Change all the HTML attributes in the page
- Change all the CSS styles in the page
- Remove existing HTML elements and attributes
- Add new HTML elements and attributes
- React to all existing HTML events in the page
- Create new HTML events in the page

Setup Project

1. Create a folder called **TaskManager**
2. Download **Starter** folder from the Repository shared earlier (***Lesson 02 [Lab 05]***)
3. Copy the content it on your TaskManager Folder
4. Your Folder Should Look LIke this :
   - assets/images //where will put images
   - assets/css //where will put css files
   - assets/js //where will put javascript files
   - **index.html** //The Task Manager App UI
   - **bom.html** //For BOM Manipulation

Note : *The UI is Built based on Material UI : Material UI*

DOM Selection

To select the DOM element we can use methods of **document** object

Single Element Selectors

getElementById()

querySelector() (👈👈👈)

Multiple Element Selectors

getElementsByTagName()	**=> HTML Collection**
getElementsByName()	**=> Node List [Same as Array]**
getElementsByClassName()	**=> HTML Collection**
querySelectorAll()	**=> Node List [Same as Array]**

Note : => HTML Collection is treated as an Array , but not exactly an array

Can Be Converted to array = > Array.from(htmlCollection)

DOM Selection

Let's Select various elements we need for manipulation

1. Select the #task , #task-form, #filter , .collection and .clear-tasks using querySelector
2. If you want to check the **selection** you can console the variables

```
// Define UI Variables

const taskInput = document.querySelector('#task');          //the task input text field

const form = document.querySelector('#task-form');          //The form at the top

const filter = document.querySelector('#filter');           //the task filter text field

const taskList = document.querySelector('.collection');     //The ul

const clearBtn = document.querySelector('.clear-tasks');    //the all task clear button
```

Event Handling

In the DOM there are various types of events that we can listen for performing **DOM** interaction

Mouse Based

- click(👈👈), dblclick , mousedown , mouseup ,mouseenter, mouseleave, mouseover, mouseout, mousemove

Keyboard Based

- keydown , keyup(👈👈) , keypress ,focus , blur , cut , paste , input , change

Other

- **submit**(👈👈) : form submit
- **DOMContentLoaded**(👈👈) : On Document Load

More Ref : Link

Event Handling

Let's Add event listeners on some of the selected elements

1. For Now form , clearBtn and filter
2. Syntax : **varElemnt.addEventListener('EventName', function/Handler)**

```
// Add Event Listener  [Form , clearBtn and filter search input ]

// form submit
form.addEventListener('submit', addNewTask);

// Clear All Tasks
clearBtn.addEventListener('click', clearAllTasks);

//   Filter Task
filter.addEventListener('keyup', filterTasks);
```

Event Handling

Let's Define the three function and test them

1. Add three function declarations **addNewTask** , **clearAllTasks** and **filterTasks**
2. Add an **alert box** inside each function *//will replace it with actual code later*

```
// Add New  Task Function definition
function addNewTask(e) {

    alert("Add New Task ....");

    e.preventDefault(); //disable form submission
}

// Clear Task Function definition
function clearAllTasks() {

    alert("Clear tasks ....");

}
```

```
// Filter tasks function definition
function filterTasks(e) {

    console.log("Task Filter ...");

}
```

DOM Read/Write

The DOM allow as to read and write in any DOM element via a few methods

Read From [Form Input]

value E.g document.getElementById("firstName").value; *//always string*

Write To

Let say we have const cardTitle = document.querySelector('.card-title');

write	**E.g** document.write(5 + 6);
innerHTML	**E.g** **cardTitle**.innerHTML = “New Text 1”;
innerText	**E.g** **cardTitle**.innerText = “New Text 2”;
textContent	**E.g** **cardTitle**.textContent = “New Text 3”;

(innerHTML/innerText/textContent : We can also use them to Read from html elements)

DOM Read/Write

Let's Update **addNewTask** Function

1. Read the task value from the input and check if it empty
2. Return if it is empty **// This is a technique for avoiding else statement**

```
// Add New  Task Function definition
function addNewTask(e) {

     if (taskInput.value === '')
     {
        alert('Enter New Task');
        return;      //avoiding else statement


     }

    e.preventDefault();    //disable form submission
}
```

Alternative : if(taskInput.value.length === 0) or if(!taskInput.value) //Type coercion

Adding and Removing Nodes (HTML Elements)

In DOM one of the ability we are given is to create element from scratch and also deleting them .

This are achieved by DOM methods

Create Element : `document.createElement('element');`

E.g `const li = document.createElement('li');`

Add Class : `SelectedElement.className = 'ClassName';`

E.g `li.className = “collection-item”;`

Adding and Removing Nodes (HTML Elements)

In DOM one of the ability we are given is to create element from scratch and also deleting them .

This are achieved by DOM methods

Add Id : SelectedElement.id = 'AnyId';

E.g li.id = 'new-item';

Add attribute : SelectedElement.setAttribute('title', 'New Item');

E.g li.setAttribute('title', 'New Item');

Adding and Removing Nodes (HTML Elements)

Text Element(text inside HTML) is created in different way than HTML element nodes

DOM methods ....

Create text node : `document.createTextNode('Text Here...')`

E.g `const txt = document.createTextNode('Hello World')`

Append Element : `ParentElement.appendChild(childElement);`

E.g `li.appendChild(txt);`

Adding and Removing Nodes (HTML Elements)

Removing an element from the DOM tree can be achieved in different ways

DOM methods ....

remove : Node.remove()

E.g var el = document.getElementById('div-02');

el.remove();

removeChild : ParentElement.removeChild(childElement);

E.g var elem = document.querySelector('#some-element');

elem.parentNode.removeChild(elem);

Note : If you just want to hide the element with CSS (useful if you may bring it back at some point), you can use the style property. **elem.style.display = 'none';**

Adding and Removing Nodes (HTML Elements)

Let's Update **addNewTask** Function by applying node manipulation

1. Create an **li** element

   const li = document.createElement('li');

   a. **Add Class** : **li.className = 'collection-item';**
   b. **Append Text Element from Input**
      i. **li.appendChild(document.createTextNode(taskInput.value));**
2. Create an **a** element

   const link = document.createElement('a');

   a. innerHTML with icon : **link.innerHTML = '<i class="fa fa-remove"></i>';**
   b. Add class : **link.className = 'delete-item secondary-content';**
3. Append **a** to the **li**

   li.appendChild(link);

4. Add the **li** to the **ul**

   taskList.appendChild(li);

Adding and Removing Nodes (HTML Elements)

```
// Add New  Task Function definition
function addNewTask(e) {

      …..

    // Create an li element when the user adds a task
    const li = document.createElement('li');
    // Adding a class
    li.className = 'collection-item';
    // Create text node and append it
    li.appendChild(document.createTextNode(taskInput.value));
    // Create new element for the link
    const link = document.createElement('a');
    // Add class and the x marker for a
    link.className = 'delete-item secondary-content';
    link.innerHTML = '<i class="fa fa-remove"></i>';
    // Append link to li
    li.appendChild(link);
    // Append to ul
    taskList.appendChild(li);

    e.preventDefault(); //disable form submission
}
```

Applying Css

In order to apply css in the DOM , we can use style object

Syntax **selectedElement**.style.StyleProperty = “Value”

E.g : **document.getElementById("myH1")**.style.color = "green";

Note : We can also use the class List to add/remove css as an alternative

```
var element = document.getElementById("myDIV");

element.classList.add("mystyle");

element.classList.remove("mystyle");
```

Applying Css

Let's Update **addNewTask** empty case by adding red border around the input text

1. Remove the **alert**
2. Replace it with :   style.borderColor = "red";

```
// Add New  Task Function definition
function addNewTask(e) {

    if (taskInput.value === '')
    {
      taskInput.style.borderColor = "red";
      return;
    }
    ……..

}
```

DOM Traversing

With the HTML DOM, you can navigate the node tree using node relationships

According to the W3C HTML DOM standard, everything in an HTML document is a node:

- The entire document is a document node
- Every HTML element is an element node
- The text inside HTML elements are text nodes
- Every HTML attribute is an attribute node (deprecated)
- All comments are comment nodes

There are various methods that allow us to achieve navigation on DOM tree

DOM Traversing

DOM traversing methods :

Let say we have two elements [**ul** and **li** respectively]

```
const list = document.querySelector('ul.collection');

const listItem = document.querySelector('li.collection-item:first-child');
```

Child Nodes [Space is Counted] : SelectedElement.childNodes; => **NodeList**

E.g list.childNodes;

list.childNodes[0].nodeName;

list.childNodes[3].nodeType; //1,2,3,8,9,10,11

Other : list.firstChild; // First child

list.lastChild; // Last child

DOM Traversing

Children Nodes **[Space is not Counted]** : SelectedElement.children; => **HTMLCollection**

E.g list.children;

```
list.children[1].textContent = 'Hello';

list.children[3].children[0].id = 'test-link'; // Children of children
```

Other :

```
list.firstElementChild;    // First child

list.lastElementChild;     // Last child
```

DOM Traversing

Parent Nodes : SelectedElement.parentNode/parentElement; => **Node/Element**

E.g listItem.parentNode;

listItem.parentElement;

listItem.parentElement.parentElement; //parent of parent

Sibling : SelectedElement.nextSibling/nextElementSibling; => **Node/Element**

SelectedElement.previousSibling/previousElementSibling; => **Node/Element**

E.g listItem.nextSibling;

listItem.nextElementSibling.nextElementSibling;

listItem.previousSibling;

listItem.previousElementSibling;

DOM Traversing

Let's Update **clearAllTasks** to clear task list [two ways to do it !!]

1. First Way is to assign empty string to innerHtml of **taskList**
2. Second way is to traverse the first child of tasklist and removeChild (👌👌)

```
// Clear Task Function definition
function clearAllTasks() {

   //This is the first way
   // taskList.innerHTML = '';

   //  Second Way
   while (taskList.firstChild) {
      taskList.removeChild(taskList.firstChild);
   }

}
```

Event Bubbling and Delegation

- **Event bubbling** is the propagation of an event from its origin towards the root element.
  - In other words, if an event occurs on a given element, it will be triggered on its parent as well and on its parent's parent and all the way up, until the html element.

```
<header id="header">
   <div id="account_links">
      <a href="#" id="signup">Sign up</a>
   </div>
</header>
```

What will happen

- When you click on the header?
- And the account_links div?
- How about the link?

```
document.querySelector('#signup').addEventListener('click',
function() {
  console.log('Sign up button click');
});

document.querySelector('#account_links').addEventListener('c
lick', function()  {
  console.log('Account links click');
});

document.querySelector('#header').addEventListener('click',fu
nction () {
  console.log('Header click');
});
```

Event Bubbling and Delegation

- **Event delegation** is a technique for listening to events where you delegate a parent element as the listener for all of the events that happen inside it.

```html
<!-- Step 1 -->
<div id="buttons">
  <!-- buttons... -->
  <button class="buttonClass">Click me</button>
  <button class="buttonClass">Click me</button>
  <button class="buttonClass">Click me</button>
</div>
```

```javascript
document.getElementById('buttons')
  .addEventListener('click', function(e) {
      // Step 2
    if (e.target.className === 'buttonClass') {
     // Step 3
     console.log('Click!');
    }
  });
```

More Ref : Link 1 , Link 2 , Link 3 , Link 4 , Link 5

Event Bubbling and Delegation

Let's Update our code base **to include remove functionality**

1. First , Let's Add Event Listener on the ul : **taskList**
    a. This is event delegation since we are targeting the (**x**) icon

```
………..
  // Remove task event [event delegation]
  taskList.addEventListener('click', removeTask);


…..
```

Event Bubbling and Delegation

Let's Update our code base **to include remove functionality**

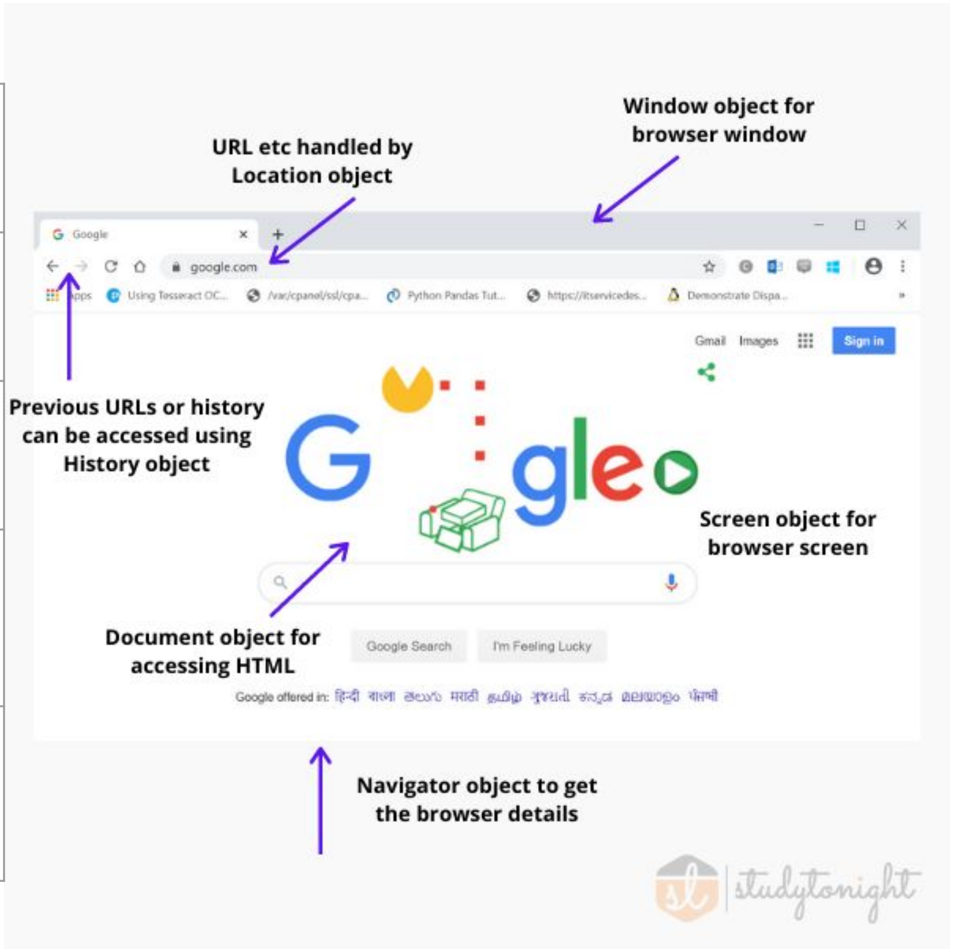
1. Second , Let's Define removeTask
   a. The (**x**) icon has a class of delete item , so let's check if the target contains that class
      i. **e.target.parentElement.classList.contains('delete-item')**

```
………..
// Remove Task function definition
function removeTask(e) {

    if (e.target.parentElement.classList.contains('delete-item'))
        {
        if (confirm('Are You Sure about that ?'))
        {
            e.target.parentElement.parentElement.remove();
        }


    }

}
```

BOM

The BOM provides you with objects that expose the web browser's functionality

window : innerHeight , innerWidth , `alert() ,` `prompt() ,` Read More
history :length , `go()` ,`back() , forward() ,` Read More
navigator :appname , platform, language, `share()`, Read More
location :href ,protocol, hostname,port, `reload()`,`assign(),` Read More
screen :height,width, availHeight, availWidth Read More



URL etc handled by Location object
Window object for browser window
Previous URLs or history can be accessed using History object
Screen object for browser screen
Document object for accessing HTML
Navigator object to get the browser details
studytonight

BOM

Let Add an event listener to reload the page when the reload button clicked

1. Select the reload icon : document.querySelector('.fa');
2. Add click Event Listener on it : reloadIcon.addEventListener('click', reloadPage);
3. Define **reloadPage** function that reload the page by using location object **reload** method

```
…..
//the reload button at the top right of navigation
const reloadIcon = document.querySelector('.fa');

….

// Event Listener for reload
reloadIcon.addEventListener('click', reloadPage);

…..
// Reload Page Function
function reloadPage() {
    //using the reload fun on location object
    location.reload();
}
```

Exercise(5pt)

1. Work on the **bom.html** according to the instruction that is written on **bom.js**
2. Add **search/filter** functionality on the **Task Manager App** based on your own logic or the instruction logic that is written below
3. Add a drop down that is going to have options for sorting the list in ascending and descending manner [***Based On Date***]
   a. User can select sort order to view the list
   b. Default Sorting should be ascending

```
/*
 Instruction for Handling the Search/filter

 1. Receive the user input from the text input
 2. Assign it to a variable so the you can reuse it
 3. Use the querySelectorAll() in order to get the collection of li which have  .collection-item class
 4. Iterate over the collection item Node List using forEach
 5. On each element check if the textContent of the li contains the text from User Input  [can use indexOf]
 6 . If it contains , change the display property of the element as block , else none

 */
```

Exercise(5pt)

1. Recreate the calculator application that you have developed on the previous lab with DOM
   a. Functions : ADD , Subtraction , Division and Multiplication
   b. Exception Handling on division using if else
   c. Allowing n numbers [In Case of Addition and Multiplication]
   d. Supporting Some Other Func : Pow , Sqrt , ……etc

Reading Assignment (Demo)

Working with DOM Functionalities

- Insert a Node - insertBefore()
- The insertAdjacentElement() method
- The ParentNode.prepend()
- Copy a Node - The cloneNode()
- Add Text to a Text Node - insertData()
- The **nodeValue** Property
- Get an Attribute Value - getAttributeNode()

- Try to work on them

ITSE-2192 Fundamental of web Design and Development

Javascript Storage API

Lab Six

What we will Learn

- After completing this lab :
  - You will be able to overview Storage API's in the Browser
  - Learn how to manipulate Local/Session Storage
  - Learn how to use the Index DB
  - Continue on the Task Manager App
- We will go through learning by working on a Task Manager app we have started on the previous lab to support persistent data storage

- **Full Source Code :** Link (***Lesson 03 [Lab 06]***)

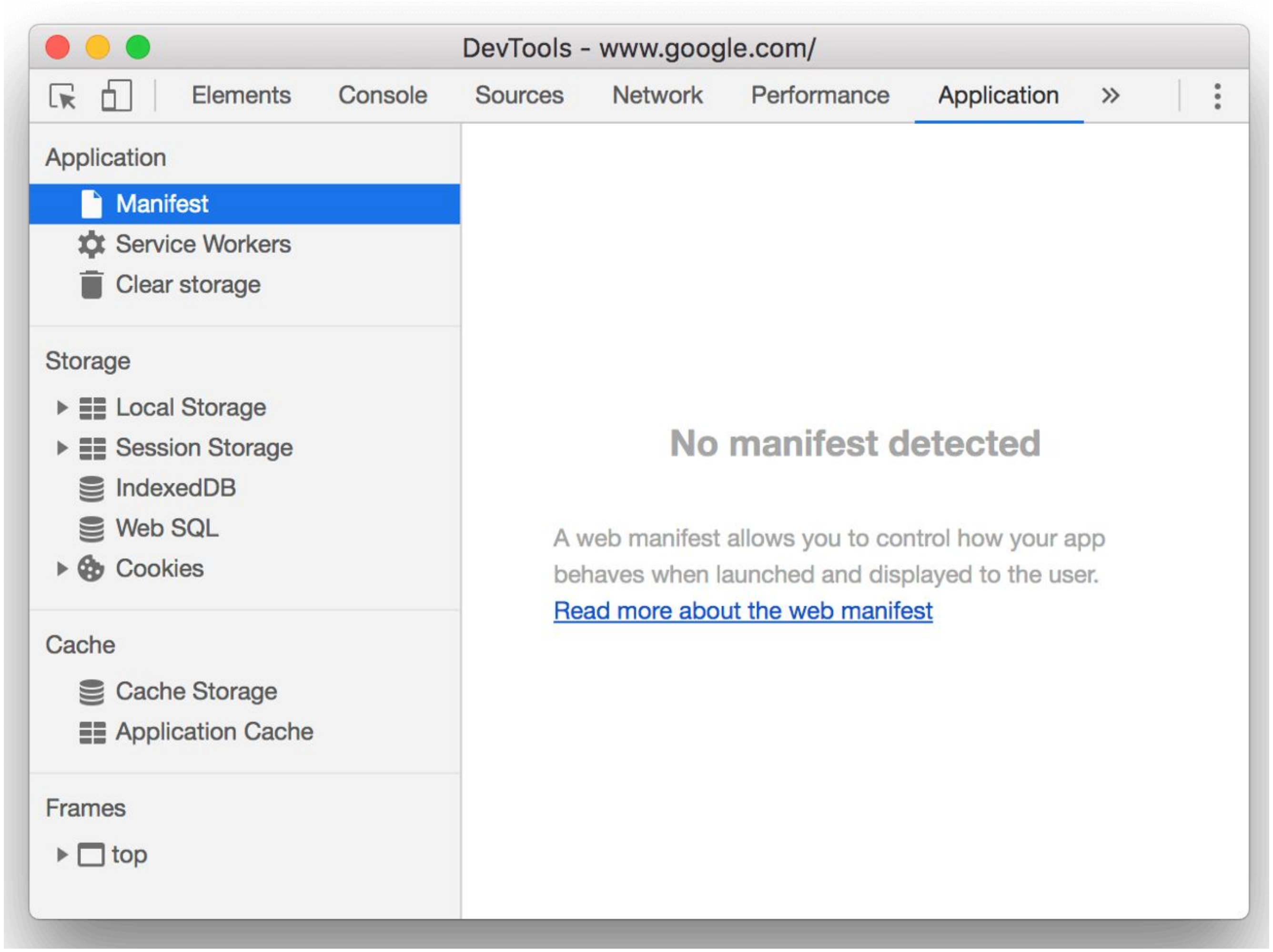
What is HTML Web Storage?

- The Web Storage API provides a way to store data in the browser
- HTML web storage provides two objects for storing data on the client:
  - window.localStorage - stores data with no expiration date
  - window.sessionStorage - stores data for one session
    - data is lost when the browser tab is closed

More Ref : Link 1 , Link 2

Where to View them ?

Open your DevTools > Application > Storage



No manifest detected

A web manifest allows you to control how your app behaves when launched and displayed to the user.
Read more about the web manifest

Storage Size Limits

Desktop

- Chrome, IE, Firefox: **10MB**
- Safari: **5MB** for local storage, unlimited session storage

Mobile

- Chrome, Firefox: 1**0MB**
- iOS Safari and WebView: **5MB** for local storage, session storage unlimited unless in iOS6 and iOS7 where it’s 5MB
- Android Browser: **2MB** local storage, unlimited session storage

Working on Local/Session Storage

They Share the same methods : **setItem(key, value)** , **getItem(key), removeItem(key), key(n) ,clear() , length**

setItem() : adds an item to the storage. Accepts a string as key, and a string as a value

E.g **localStorage.setItem('name', 'Kebede')**

If you pass any value that's not a string, it will be converted to string:

```
localStorage.setItem('id', 123)           //stored as the '123' string
localStorage.setItem('test', { test: 1 })    //stored as "[object Object]"
```

Working on Local/Session Storage

getItem() is the way to retrieve a string value from the storage, by using the key string that was used to store it

```
E.g   localStorage.getItem('name')        // 'Kebede'
      localStorage.getItem('id')          // '123'
```

removeItem() removes the item identified by key from the storage, returning nothing (an undefined value)

E.g `localStorage.removeItem('id')`

clear() removes everything from the storage object you are manipulating

E.g `localStorage.clear()`

Working on Local/Session Storage

To Store Object or an array we should use

- JSON.stringify() : to make it a string
- JSON.parse() : to parse object from the string

Example

```
// Storing data:
let person = { name: "John", age: 31, city: "New York" };
let myJSON = JSON.stringify(person);
localStorage.setItem("person", myJSON);

// Retrieving data:
let text = localStorage.getItem("person");
let obj = JSON.parse(text);
console.log(obj.name);        //John
```

Setup Project

1. Open the folder called **TaskManager from Previous Lab**
2. Create a file called **storage.js** on assets/js folder
3. Let's Incorporate the external js file to index page
4. Make sure the storage js is at the top of app.js
    - **Note :** This is due to the variables and functions defined on storage js will be used by app.js

      If the functions are regular functions and due to hoisting , even though you put storage.js below it will work

You Can use the starter folder on (Lesson 03 [Lab 06])

Local Storage -Add

Let's Define a function called **addToDatabase** that add text input to the storage as an array

1. First Declare a variable the stores the array
    a. Check if the **tasks key** is empty :
        1. if(**localStorage.getItem('tasks') == null)**
    b. If Empty : create an empty array , else add the parsed array from local storage to the array

    listofTasks = JSON.parse(localStorage.getItem('tasks'));

    c. Push The Input to the Array
        i. **listofTasks.push(newTask);**
    d. Add the array to local storage by making it a string
        1. **localStorage.setItem('tasks', JSON.stringify(listofTasks));**

Local Storage -Add

Let's Define a function called **addToDatabase** that add text input to the storage as an array

```
// Add to LocalStorage function declaration
function addToDatabase(newTask)
{
   let listofTasks;
   if(localStorage.getItem('tasks') == null)
   {
       listofTasks = [];
   }
   else
   {
       listofTasks = JSON.parse(localStorage.getItem('tasks'));
   }
    listofTasks.push(newTask);
    localStorage.setItem('tasks', JSON.stringify(listofTasks));
}
```

Local Storage -Add

2. Second, Invoke it inside **addNewTask**

```
// Add New  Task Function definition
function addNewTask(e) {
      …….
   addToDatabase(taskInput.value);

}
```

3. Third , Go to DevTools > Application > Storage Tab and check if the data is added

Local Storage -Load

Now , Let's Define a function called **loadfromDB** that retrieves the data from local storage as an array

1. First Declare a variable the stores the array : **let listofTasks ;**
   a. Check if the **tasks key** is empty :
      1. if(**localStorage.getItem('tasks') == null)**
   b. If Empty : create an empty array , else add the parsed array from local storage to the array

      listofTasks = JSON.parse(localStorage.getItem('tasks'));

   c. return the array : **listofTasks**

Local Storage -Load

Let's Define a function called **loadfromDB** that retrieves the data from local storage as an array

```
// Load task from local storage function declaration
function loadfromDB()
{
    let listofTasks;
    if(localStorage.getItem('tasks') == null)
    {
        listofTasks = [];
    }
    else
    {
        listofTasks = JSON.parse(localStorage.getItem('tasks'));
    }
     return listofTasks;    //return array
}
```

Local Storage -Load

Let's Add **DOMContentLoaded** Event Listener on index , to load our task list from local storage

1. Add DOMContentLoaded Event Listener on document
2. Define the function loadTaskFromDB

```
…..
// DOM load event
document.addEventListener('DOMContentLoaded', loadTasksfromDB);



…..

function loadTasksfromDB()
{




}
```

Local Storage -Load

1. Let's Invoke **loadfromDB** and assign it to a variable
2. Iterate over the array using **forEach**
3. for each task , Create the DOM [**Same as addNewTask**]

```
// Load from Storage Database
function loadTasksfromDB() {
    let listofTasks = loadfromDB();
    if (listofTasks.length != 0) {

        listofTasks.forEach(function(eachTask) {

          const li = document.createElement('li');          // Create an li element when the user adds a task
          li.className = 'collection-item';                              // Adding a class
          li.appendChild(document.createTextNode(eachTask));           // Create text node and append it
          const link = document.createElement('a');                  // Create new element for the link
          link.className = 'delete-item secondary-content';          // Add class and the x marker for a
          link.innerHTML = '<i class="fa fa-remove"> </i>';
          li.appendChild(link);                                   // Append link to li
          taskList.appendChild(li);                               // Append to UL
        });
    }
}
```

Local Storage -Clear

Let's Define a function called **clearAllTasksfromDB** that clears the local storage

1. First function Declaration that uses clear method of local storage
2. Invoke it **clearAllTasks** function in app.js
3. Check the Local Storage [if it is cleared]

```
// Clear from Local Storage
function clearAllTasksfromDB()
{
   localStorage.clear();
}
```

```
// Clear Task Function definition
function clearAllTasks() {

   ……..

    clearAllTasksfromDB();
}
```

Local Storage -Delete

Let's Define a function called **removefromDB** that removes an item from local storage when user click (**x**)

1. First Declare a variable the stores the array : **let listofTasks ;**
   a. Check if the **tasks key** is empty :
      1. if(**localStorage.getItem('tasks') == null)**
   b. If Empty : create an empty array , else add the parsed array from local storage to the array
      i. listofTasks = JSON.parse(localStorage.getItem('tasks'));
   c. Iterate over the task list : **listofTasks.forEach(function(task,index)**
   d. Lets use splice method of an array to remove the task if it matches with the arraylist item

```
if(taskItem.textContent === task)

    listofTasks.splice(index,1);

  });
```

   e. Let's Set the localstorage with the Updated array list :

      localStorage.setItem('tasks', JSON.stringify(listofTasks));

Local Storage -Delete

2. Invoke it **removeTask(Inside the if)** in app.js

```
// Remove from Local storage function declaration
function removefromDB(taskItem) {

    // console.log(taskItem.textContent);
    let listofTasks;
    if (localStorage.getItem('tasks') == null) {
        listofTasks = [];
    } else {
     listofTasks = JSON.parse(localStorage.getItem('tasks'));
    }
    listofTasks.forEach(function(task, index) {
        if (taskItem.textContent.trim() === task.trim())
            listofTasks.splice(index, 1);
    });
    localStorage.setItem('tasks', JSON.stringify(listofTasks));

}
```

```
// Remove Task function definition
function removeTask(e) {


   // Remove from DB [Local Storage ...]

removefromDB(e.target.parentElement.parentElement);
}
```

Index DB

- A complete **NoSQL** database in Browser
- You can save any type of data such us an object , array , files and images
- You can access data with key -> value pairs
- Asynchronous by default
- Enables applications to work both online and offline

Issues

- You can not sync data with backend
- Can not be used in private mode , since all data will be removed once you close a private tab
- if the visitor clears caches , cookies and navigation history the data will be removed too

Ref : Link 1 , Link 2 , LInk 3 , Link 4

Working with IndexedDB

Open Database	**Create Object Store [Table]**

let opReq = indexedDB.open(name, version);
 	
return : **openRequest = > listen to events**
● **success**
● **error**
● **upgradeneeded**
	db.createObjectStore(name[, keyOptions]);
	
	E.g db.createObjectStore('books', {keyPath: 'id'});
	
	An object store can only be created/modified while updating the DB version, on **upgradeneeded** handler.

Delete Database	**Delete Object Store**
// deleteReq.onsuccess/onerror tracks the result let deleteReq = indexedDB.deleteDatabase(name)	db.deleteObjectStore(name) **E.g** db.deleteObjectStore('books')

Working with IndexedDB

Create Transaction

```
let tran = db.transaction(store[, type]);
```

type

- **readonly**
- **readwrite**

Create Index[Table Schema]

```
objectStore.createIndex('name', 'name', {
unique: false });
```

Perform Operation

```
let obj = tran.objectStore(storeName);
```

Methods on Obj

- add () : takes the object
- get () : value of the primary key
- put() : taks an object
- delete() : value of the primary key

Example

```
var  db ;

 request.onupgradeneeded = (event) => {
     db = event.target.result;

   let store = db.createObjectStore('Contacts', {
         keyPath : 'id' , autoIncrement: true  });

  let index = store.createIndex('email', 'email', {
         unique: true  });
 };
```

Setup Project

1. Open the folder called **TaskManager** from Previous Lab
2. Remove All the Code Apart from UI Declaration in the app.js [*Look Like the code below*]

Note : we are doing this b/c indexedDB is Asynchronous

```
// Define UI Variables [Except this code remove the rest]
const taskInput = document.querySelector('#task');        //the task input text field
const form = document.querySelector('#task-form');      //The form at the top
const filter = document.querySelector('#filter');        //the task filter text field
const taskList = document.querySelector('.collection');    //The UL
const clearBtn = document.querySelector('.clear-tasks');     //the all task clear button

const reloadIcon = document.querySelector('.fa'); //the reload button at the top navigation

//Rest of Code for Indexeddb will continue …..
```

You Can use the starter folder on (**Lesson 04 [Lab 06])**

The DOM Load

Let’s add a **'DOMContentLoaded' that load the Database , event listeners**

1. Create a variable the hold the Database : **let DB;**
2. Create DOM load event listener with arrow function :

document.addEventListener('DOMContentLoaded', () => { //body});

```
……
//DB variable
let DB;

// Add Event Listener [on Load]
document.addEventListener('DOMContentLoaded', () => {

    //all code will reside here

});
```

Creating the Database

Let’s Create the Database and handle **onsuccess** and **onerror**

1. Open the Database with DB name “tasks” and version(1 default)

```
let TasksDB = indexedDB.open("tasks", 1);
```

   a. Display Success message and invoke the display all task method **on success event** occured

```
TasksDB.onsuccess = function(event) { //code here  };
```

   b. Display error message **on error event** happens

```
TasksDB.onerror = function(event) {  // code here  };
```

Creating the Database

Let’s Create the Database and handle **onsuccess** and **onerror**

```
// create the database
 let TasksDB = indexedDB.open('tasks', 1);

 // if there's an error
 TasksDB.onerror = function() {
        console.log('There was an error');
    }
    // if everything is fine, assign the result to the instance
    TasksDB.onsuccess = function() {

     console.log('Database Ready');

    // save the result
    DB = TasksDB.result;

    // display the Task List
    displayTaskList();
 }
```

Creating the Objectstore[Table]

Let's create a table called **‘tasks’ with id attribute of autoincrement**

Note : This shall be done once so we are going to use onupgrade event to handle the creation of tables

1. Define onupgrade event :
   a. **TasksDB.onupgradeneeded = function(e) {//code here }**

Inside onupgradeneeded

2. Create a variable to store the database : **let db = e.target.result;**
3. Create the object store :
   a. **let objectStore = db.createObjectStore('tasks', { keyPath: 'id', autoIncrement: true });**
4. Create a field called task name
   a. **objectStore.createIndex('taskname', 'taskname', { unique: false });**
5. Display a console message with database ready text

Creating the Objectstore[Table]

Let's create a table called **'tasks' with id attribute of autoincrement**

```
// This method runs once (great for creating the schema)
  TasksDB.onupgradeneeded = function(e) {
     // the event will be the database
     let db = e.target.result;

     // create an object store,
     // keypath is going to be the Indexes
     let objectStore = db.createObjectStore('tasks', { keyPath: 'id', autoIncrement: true });

     // createindex: 1) field name 2) keypath 3) options
     objectStore.createIndex('taskname', 'taskname', { unique: false });

     console.log('Database ready and fields created!');
  }
```

Add Task to ObjectStore

Let's redefine our **addNewTask** function to handle creating an entry in the database and invoking the **displayTaskList** [will define it later]

1. Add event listener for the from
   a. **form.addEventListener('submit', addNewTask);**
2. Define the **addNewtask** function :
   a. **function addNewTask(e) { e.preventDefault(); //the rest of code }**
3. Create the task object :
   a. **let newTask = { taskname: taskInput.value}**
4. Create the transaction[with read and write] and object store
   a. **let transaction = DB.transaction(['tasks'], 'readwrite');**
   b. **let objectStore = transaction.objectStore('tasks');**
5. Add the new task to table
   a. **let request = objectStore.add(newTask);**
6. call onsuccess , oncomplete and onerror events respectively
   a. **form.reset();**
   b. **displayTaskList();** and **Completer Message**
   c. **Error Message**

Add Task to ObjectStore

```
function addNewTask(e) {
          ……
        // create a new object with the form info
        let newTask = {
            taskname: taskInput.value
        }
        // Insert the object into the database
        let transaction = DB.transaction(['tasks'], 'readwrite');
        let objectStore = transaction.objectStore('tasks');

        let request = objectStore.add(newTask);
        // on success
        request.onsuccess = () => {
            form.reset();
        }
        transaction.oncomplete = () => {
            console.log('New Task added');
            displayTaskList();
        }
        transaction.onerror = () => { console.log('There was an error, try again!'); }
    }
```

Display from Database

Lets display all task from database and update the ui . we need to create **displayTaskList function**

1. Define the displayTaskList function : **function displayTaskList() { //code here }**
2. The Logic Behind this function
   a. Remove the current list from the UI so that we can replace it with new dataset **Note :** done it earlier for the clearAllTasks
   b. Read the object store which will return the data as cursor
   c. On reading the data from cursor object create the li with a [x] **//we have done it earlier**
   d. Read the **task name** from cursor value object and make it as the text of text Node
   e. Add **data-taks-id** custom attribute on each **li** element [**Only Difference**]
      i. This will be used for deletion
      ii. Appending the li to the UL

Display from Database

Lets display all task from database and update the ui . we need to create **displayTaskList function [... means same as previous code]**

```
function displayTaskList() {
        // clear the previous task list
        while (taskList.firstChild) {   taskList.removeChild(taskList.firstChild);}

        // create the object store
        let objectStore = DB.transaction('tasks').objectStore('tasks');

        objectStore.openCursor().onsuccess = function(e) {
            // assign the current cursor
            let cursor = e.target.result;

            if (cursor) {
                      ……..
                li.setAttribute('data-task-id', cursor.value.id);
                // Create text node and append it
                li.appendChild(document.createTextNode(cursor.value.taskname));
                 ……...
                cursor.continue();
            }
        }
    }
```

Clear The Database[Delete All]

Let's Define a function called **clearAllTasks** that delete all entries from the database

1. Add event listener for the clear task button
    a. **clearBtn.addEventListener('click', clearAllTasks);**
2. Define the clearAllTasks function :
    a. **function clearAllTasks() { //code here }**
3. Create the transaction and object store
    a. **let transaction = DB.transaction("tasks", "readwrite"); // (1)**
    b. **let tasks = transaction.objectStore("tasks");**
4. clear the database using clear method
    a. **tasks.clear();**
5. Invoke displayTaskList
    a. **displayTaskList();**

Clear The Database[Delete All]

Let's Define a function called **clearAllTasks** that delete all entries from the database

```
//clear button event listener
clearBtn.addEventListener('click', clearAllTasks);
   //clear tasks
   function clearAllTasks() {
      //Create the transaction and object store
      let transaction = DB.transaction("tasks", "readwrite");
      let tasks = transaction.objectStore("tasks");

      // clear the the table
      tasks.clear();
      //repaint the UI
      displayTaskList();

      console.log("Tasks Cleared !!!");
   }
```

Delete a Task [From Database]

Let's Define a function called **removeTask** that delete a single entries from the database [code is similar to previous lab ...]

1. Add event listener for the x icon using event delegation
   a. **taskList.addEventListener('click', removeTask);**
2. The logic to read from DB
3. Read the li **data-task-id** attribute and convert it to Number:
   a. **Number(e.target.parentElement.parentElement.getAttribute('data-task-id'));**
4. Create the Object store **[same as previous]**
5. Use delete function by passing the id
   a. **objectStore.delete(taskID);**
6. Remove the li from the UI [same logic as before]

Delete a Task [From Database]

Let's Define a function called **removeTask** that delete a single entries from the database [code is similar to previous lab ...]

```
// Remove task event [event delegation]
 taskList.addEventListener('click', removeTask);

 function removeTask(e) {

     if (e.target.parentElement.classList.contains('delete-item')) {
         if (confirm('Are You Sure about that ?')) {
             // get the task id
             let taskID = Number(e.target.parentElement.parentElement.getAttribute('data-task-id'));
             // use a transaction
             let transaction = DB.transaction(['tasks'], 'readwrite');
             let objectStore = transaction.objectStore('tasks');
             objectStore.delete(taskID);

             transaction.oncomplete = () => {
                 e.target.parentElement.parentElement.remove();
             }

         }
     }
 }
```

Exercise(5pt)

Add the following features [continued from previous lab]

1. Sorting by data [ascending and descending]
   a. Update the table to include date column
   b. add a drop down that allows Sorting options
   c. trigger an event that update the UI when user selects a specific option
2. Update a single Task [update by using query string]
   a. Go to **edit.js** on the starter file
   b. View the instruction set on updateTask Method
   c. Update **link.innerHTML** line of the `displayTaskList` method on app.js as below
   d. Make sure to point the **edit.html** path appropriately

```
 link.innerHTML = `
 <i class="fa fa-remove"></i>  &nbsp;
<a href="/.../edit.html?id=${cursor.value.id}"><i class="fa fa-edit"></i> </a>
`;
```

ITSE-2192 Fundamental of web Design and Development

Async Programing

Lab Seven

What we will Learn

- After completing this lab :
    - You will be able to learn major concepts in Async Programing
    - Learn how to work with JSON
    - Learn how to manipulate Third Party API
- We will go through learning by working on a Simple Blog Application that we read from JSON and Third Party API (JSONPlaceholder)

- **Full Source Code :** Link (***Lesson 05 [Lab 07]***)

Required Extra Application

API Testing

Postman : Download Link (👉👉)

JSON Formatter

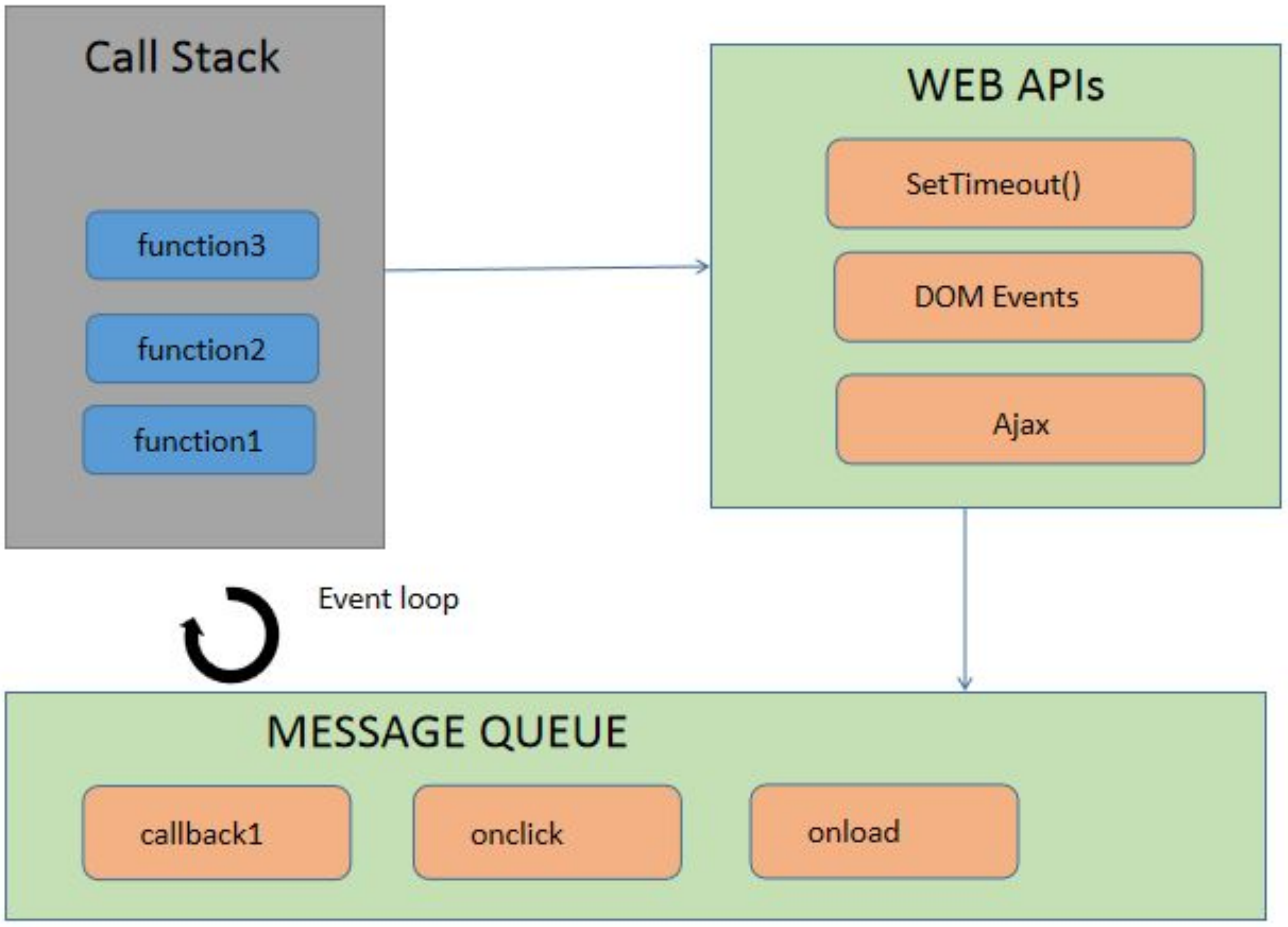
Chrome Extension : LInk (👉👉)

Online Options

JSON Formatter : LInk

JSON Validator : Link

Async Overview



From Callback ...Promise...Async/Await[Evolution]

- To handle async code we can either use callback , promise or async/await
- Let's take an example where a user will **login** , then get his/her **recent works** , then view the **detail** of single work
  - Will use **setTimeout** to imitate a delay where data is coming from remote server
  - First will look at it as synchronous [what will be the result]
  - Then will work with callback ...Promise ...Async/Await

```
// Sync
let login = function(username, password) {
   setTimeout(() => {    return { username,  password  }
}, 2000)
}
console.log("Start of Program !!!");
let user = login("user 1", "password");
//what will be the result and why??
console.log(user);
console.log("End of Program !!!");
```

Lets Define Functions `[with setTimeout]`

`userWorks :`

`Return array of works`

`Accept User Name`

`workDetail :`

`Display a single work description`

`Accept : A work Name from works array`

Full Source Code : LInk

Setup Project

1. Create a folder called **AsyncDemo**
2. Download **Starter** folder from the Repository shared earlier (***Lesson 05 [Lab 07]***)
3. Copy the content it on your AsyncDemo Folder
4. Your Folder Should Look LIke this :
   - assets/images   //where will put images
   - assets/css   //where will put css files
   - assets/js   //where will put javascript files
   - **index.html**   //The Async Demo App UI
   - **api.html**   //The Third Party API Data Display UI

Note : *The UI is Built based on Semantic UI : Semantic UI*

Timer Event API

One of the Most used async methods on the web is Timers , lets see some of them

- `setTimeout(`*`function, milliseconds`*`)`
  Executes a function, after waiting a specified number of milliseconds.
- `setInterval(`*`function, milliseconds`*`)`
  Same as setTimeout(), but repeats the execution of the function continuously.
- The `clearTimeout()` method stops the execution of the function specified in setTimeout().

Working With Timer

Let's add setTimeout timer event in ordered to show the current time

1. Let's use the startTime function on **timer.js** that will display the current time
   a. Logic Behind the code
      i. Get the values of each data component from Date object
      ii. Change the hour to 12
      iii. Decide whether am/pm [based on the hour value]
      iv. use the **setTimeout** to invoke the startTime function repeatedly [**recursive**]
2. On addZero function, lets attaches zero if the number is less than 10 so that a number is displayed as two digits
   a. **addZero(i)** **//attach zero of the number is less than 10**

Working With Timer API

```
//timer.js
function startTime() {
    //retrieve date
    var today = new Date();
    var h = today.getHours();
    var m = today.getMinutes();
    var s = today.getSeconds();
    //get the AM / PM value
    let am_pm = h > 12 ? 'PM' : 'AM';
    // Convert the hour to 12 format
    h = h % 12 || 12;
    // add zero
    m = addZero(m);
    s = addZero(s);
    // Assign to the UI [p]
    timerDemo.innerHTML =
     `${h} : ${addZero(m)} : ${addZero(s)} ${am_pm }`;
   //run the timer recursively
   setTimeout(startTime, 500);
}
```

```
function addZero(i) {
// add zero in front of numbers < 10
    if (i < 10) { i = "0" + i }    return i;
}
```

Can you convert the code to work with `setInterval` **??**

Working with JSON

What is JSON (**Javascript Object Notation**)?

- It is a lightweight data-interchange format [replacement for XML]
- It is easy for humans to read/write and easy for machines to parse and generate
- It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999
    - Keys and Values , where Values can be Array , Boolean , Number , Object, String

```
{
  "firstName": "Jonathan",
  "lastName": "Freeman",
  "loginCount": 4,
  "isWriter": true,
  "worksWith": ["Spantree Technology Group", "InfoWorld"],
  "pets": {  "name": "Lilly",  "type": "Raccoon" }
}
//object
```

```
[
  {
    "name": "Molecule Man",
    "age": 29,
    "secretIdentity": "Dan Jukes",
  },  …..
]
//array
```

JSON Creation

1. Create a folder called **jsonData on assets folder**
2. Create a file called **post.json [for a single post]**
3. Create a single post Item that will be displayed on the Index page
4. Feel free to change the values [name , image , postTitle, date and postText]

Note : trailing comma(last comma) is not allowed in json unlike JS Object

```
{
   "name": "John",
   "image": "Any Image URL ",
   "postTitle": "New Technology Arrives",
   "date": "January 21 ,2021",
   "postText": "Lorem Ipsum ………"
}
```

XmlHttpRequest(XHR) object

- To send an HTTP request, we can create an XMLHttpRequest object, open a URL, and send the request
- The XMLHttpRequest object is a developers dream, because you can update a web page without reloading the page

Create the Object	**Events to Handle**

`var oReq = new XMLHttpRequest();`	`oReq.onload = function(e) {}`(👈)
	
	`oReq.onerror = function(e) {}`(👈)
	
	`oReq.onprogress = function(e) {}`
	
	`oReq.onreadystatechange = function(e) {}  //deprecated`
	
	`vars : status , readyState , response`

Open Request	**Send The request**

`oReq.open("GET", url);`
 	
`Type of Req : GET (👈), POST , PUT , DELETE`
	`oReq.send();`

XmlHttpRequest(XHR) object

Let's create our XHR object to read the Post from **post.json** and display it on the UI

Steps [Will Work on **loadPost** function in **loadPost.js**]

1. Create a new XMLHttpRequest object
2. Configure it: GET-request for the URL
3. Send the request over the network
4. Call Onload to load the data when the status is **200**

XmlHttpRequest(XHR) object

```
function loadCustomer() {
   const xhr = new XMLHttpRequest();

  //change file URL up on your location
   xhr.open('GET', '/Lesson 05[Lab 07]/Finished/asset/jsonData/post.json', true);

   xhr.onload = function() {
      if (this.status === 200)     //check the status
        {
         const post = JSON.parse(this.responseText);
         let output = `
            <div class="item">
               <div class="image"> <img src="${post.image}"> </div>
               <div class="content">
                   <a class="header" href="#" id="bTitle"> ${post.postTitle} </a>
                   <div class="meta">
                      <span id="bDate">${post.date} </span>
                      <span>By: <a href="#" id="bAuthor"> ${post.name}</a></span>
                   </div>
                   <div class="description">  <p id="bDesc">  ${post.postText} </p>  </div>
                   <div class="extra"> <a class="ui floated basic violet button" href="#">Read Mores</a> </div>
               </div>
            </div>         `;
         postDiv.innerHTML = output;
      }
   }

   xhr.send();
}
```

Handling Multiple Data(JSON Creation)

1. Create a file called **posts.json [for a multiple post] on jsonData folder**
2. Create an array of posts that will be displayed on the Index page
3. Feel free to change the values [name , image , postTitle, date and postText]

```
[{
    "name": "John 1",
    "image": "Any Image URL ",
    "postTitle": "Post One",
    "date": "January 21 ,2021",
    "postText": "Lorem Ipsum ........"
},
{
    "name": "John 2",
    "image": "Any Image URL ",
    "postTitle": "Post Two",
    "date": "January 21 ,2021",
    "postText": "Lorem Ipsum ........"
}

]
```

Handling Multiple Data(JSON Creation)

Let's create our XHR object to read the All Posts from **posts.json** and display it on the UI

Steps [Will Work on **loadPosts** function in **loadPosts.js**]

1. Create a new XMLHttpRequest object
2. Configure it: GET-request for the URL
3. Send the request over the network
4. Call onload to load the data when the status is **200**
5. Use forEach to iterate over the response Array

Handling Multiple Data(JSON Creation)

```
function loadCustomers() {

   const xhr = new XMLHttpRequest();

   //change file URL up on your location
   xhr.open('GET', '/Lesson 05[Lab 07]/Finished/asset/jsonData/posts.json', true);

   xhr.onload = function() {
      if (this.status === 200) {
         const posts = JSON.parse(this.responseText);
         let output = '';
     posts.forEach(post => {

            //same as previous code
             let output  += `     `;

         }});

         postDiv1.innerHTML = output;
      }
   }

   xhr.send();
}
```

Fetch API and Promise

Fetch API, a new standard to make server requests with promises

Fetch API Syntax

```
fetch(url, {options[method, headers...]})
.then(function(response) {
  //response.json()/text()
})
.catch(function() {
});
```

common response types : json , text , ...etc

Working with fetch on Public API

There are various public AP’s that you can manipulate

API Repositories

API Directory : LINK

Rapid API : LINK

Testing APIs for Demo

JSON PlaceHolder : LINK (👈👈👈)

Lorem Picsum : Link

First Let’s Use **Postman and the Browser** to test the APIS

Handling Public API Data with fetch

Let's use the post route(Link) and display on api.html

Steps [Will Work on **loadPostsAPI** function in **loadPostsAPI.js**]

1. Open GET Request with fetch
2. On then , let's retrieve the JSON format using reponse.json()[since the api returns json]
3. On then we can fetch the data and iterate over it [chaining]
4. Use forEach to iterate over the Array

Handling Public API Data with fetch

```
function load_fromPlaceHolder() {
  //open the request
   fetch('https://jsonplaceholder.typicode.com/posts')
    .then(function(res) {  return res.json(); //return the JSON Promise
       })
       .then(function(posts) {
           //iterate over each post [100 posts]
           let output = '';
           posts.forEach(function(post) {
               output += ` `;  // same code as previous with few update
           });
           postDiv3.innerHTML = output;
       })
       .catch(function(err) {     console.log(err);
       });   }
```

Async/Await

Special syntax to work with promises in a more comfortable fashion

Syntax

async : `async function f() {}`

async makes a function return a Promise

await : `let value = await promise;`

await makes a function wait for a Promise

Async/Await

Lets convert our fetch api request with async/await [on load_fromPlaceHolder_new() fun]

1. Add async keyword in front of function definition
2. Lets use await on each call [fetch , returned value]
3. Consume the load_fromPlaceHolder_new() function on loadDataNew() function and call it on DOMContentLoaded

Async/Await

```
async function load_fromPlaceHolder() {
  //open the request
 let res = await fetch('https://jsonplaceholder.typicode.com/posts');


 let data = await res.json();


 return data;


}
```

Async/Wait

```
function loadDataNew() {
   load_fromPlaceHolder_new().then(function(posts)
         {
               //iterate over each post [100 posts]
              //same code as previous


         }



}
//call this function instead
```

Exercise (15pt)

1. Add a spinner on the api.html that will be displayed until the data is loaded from server: Link
   a. Hint Use display property
2. Develop analog Clock (Example Link)
   a. Hint : Use Analog Clock Image as background
   b. Make the UI Look Good using UI Library
3. Develop an image Gallery with slider using
   a. Lorem Picsum : Link
   b. Make the UI Look Good using UI Library

Reading Assignment(Demo)

1. What is Promise.all (): LInk
2. What is Promise.race() . Link

ITSE-2192 Fundamental of web Design and Development

Advanced Javascript

Lab Eight

What we will Learn

- After completing this lab :
    - You will be able to overview Advanced Concepts on Javascript
        - Symbol , Template Literal , Destructuring , spread/rest syntax , New Loops [for in / for of] , function borrowing and Arrow Function
        - Object Oriented Javascript[Mainly ES6 Class]
        - Exception and ES6 Modules
    - Work on the each concepts
- This will be different setup from previous labs where students will practice the topics by following instructions on source code

- **Starter File :** Link (***Lesson 06 [Lab 08]***)

Symbol

Symbol : New Data Type introduced in ES6

- It represents a unique identifier

Syntax : const var = Symbol("Debuging_String")

E.g const id = Symbol('id')

Mostly Used as Hidden property to an Object

Syntax : obj[var] = value

E.g person[id] = 123456 //let's assume we have person object

Note : Every time we create symbol it is unique and the property is not visible on for in loop

*****Follow the instruction on **01[Symbol].js** and work on symbol

Template Literal

Template Literal : Is a new way of working in string concatenation .

- It allow to use strings or embedded expressions in the form of a string

Syntax

`  ${expression} `

E.g
```
const string = `something ${myVariable}`
const string = `something ${1 + 2 + 3}`
```

Note : We Can write multiple line string with template literal

Expression : Variable/Value , Function Call , Operation

***Follow the instruction on **01(TemplateLiteral).js** and work on template literal

For Loop(Newer flavors)

for of : a loop iterating over iterable objects(arrays, sets, maps, strings etc).

Syntax :

```
for (element of iterable) {   // body of for...of   }
```

E.g

```
const iterable = ['a', 'b'];
for (const x of iterable) {  console.log(x) }
```

for in : a loop iterating over all property keys of an object

Syntax :

```
for (key in object) {   // body of for...in   }
```

E.g

```
const st = { name: 'Monica', class: 7, age: 12 }
for ( let key in st ) { console.log(`${key} =>${st[key]}`); }
```

***Follow the instruction on **02(newLoops).js** and work on for loops

Arrays and Objects Destructuring

Destructuring Assignment : is a special syntax that allows us to “unpack” arrays or objects into a bunch of variables, as sometimes that’s more convenient

Syntax

Array : [var1,var2] = array

Object: {var1,var2} = object;

To change var name = > varName : newName

E.g
```
let [firstName, surname] = ["John", "Smith"];   //Array
 let {title : t, height = 0} = { title: "Menu", height: 20 }; //Object
```

***Follow the instruction on **03(Destructuring).js** and work on destructuring

Spread Syntax and Rest Parameter

Spread Syntax : used to take an array and spread to list of variables

Syntax : ...array [the three dots]

Rest Parameters : Allows to store multiple arguments in a single array . Mostly used in creating a flexible method argument passing

Syntax : functionName(p1,p2 , ...rest)

E.g

```
const no = [1,2,3,4]

  //Rest Parameter

 function add(x, y, ...rest) {  console.log(  x + y + rest[0] + rest[1] )  }

add(...no)       //Spread Syntax
```

***Follow the instruction on **03(Destructuring).js** and work on spread syntax and rest parameter

Arrow Function

Arrow Function : very simple and concise syntax for creating functions

- this is not associated with arrow functions
- We can leave the parenthesis if there is only one parameters
- We can leave the curly brace and return statement if the the body has one line

Syntax : let myFunction = (arg1, arg2, ...argN) => { statement(s) }

E.g

```
//full syntax

let sum = (a, b) => {  let result = a + b;  return result;   }
let sum = (a, b) => a + b;          //minimized version 1
let greet = x => console.log(x);   //minimized version 2
```

***Follow the instruction on **04(arrow_function).js** and work on arrow function

Call, Apply and Bind(Function Borrowing)

To call a function and change the value of this we can use call, apply and bind

Syntax : `oldObject.method.call(newObject, par1, par2)`

`oldObject.method.apply(newObject, [par1, par2])`

`const newFun = oldObject.method.bind(newObject, par1, par2)`

`newFun()`

E.g

```
let oldCar = {

   carId : 123,
   getId : function (pref)
   {
       console.log( pref + this.carId) ;
   }
};

let newCar = {carId : 456};
```

```
//call
oldCar.getId.call(newCar, “My Id : ”);

//bind takes argument as an array
oldCar.getId.apply(newCar, ['Id : ']);

//bind creates clone of the function
let newFun = oldCar.getId.bind(newCar, “My Id : ”);

 newFun()
```

***Follow the instruction on **04(function_borrowing).js** and work on call, apply and bind

Object Oriented Javascript(Object Creation)

Object Creation : In Javascript to create a blueprint for an object we can use three techniques : Constructor Function , Object.create(), and ES6 Class

Syntax :

```
//constructor function
function NameFun(par1, par2, ..)
{

 this.par1 = par1 ;
 this.par2 = part2;
  ...
 this.method = function() {  //method body };
}

//Object.create [Kind of inheritance ]
var obj = Object.create(templet_Object, new_prop);
//new prop can be added obj.newProp = value  or
// {prop :  {value : “value”}}
```

```
//ES6 Class => Syntactic Sugar
class ClassName
{
  constracture(pro1,pro2, ..)
  {
     this.pro1 = pro1 ;
     this.pro2 = pro2 ;
     ...
  }

method() { //method body }
}
```

```
const newOBJ = new NameFun(“par1”, “par2”)       //create Object
const newOBJ = new ClassName(“par1”, “par2”)   //create Object
newOBJ.method()
```

Object Oriented Javascript(Object Creation)

E.g

//constructor function	//ES6 Class => Syntactic Sugar

```
function Person (person_name, person_age)
{
    this.name = person_name,
    this.age = person_age,

    this.greet = function () {
        return ('Hi' + ' ' + this.name);
    }
  }
```
	```
	class Person
	{
	constructor (person_name, person_age)
	    this.name = person_name,
	    this.age = person_age,

	    greet() {
	        return ('Hi' + ' ' + this.name);
	    }
	  }
	```

```
const person1 = new Person('John',23);    //create Object

person1.greet()         //call member method
```

Object Oriented Javascript(Object Creation)

E.g

```
//Object.create
const person = {

  isHuman: false,

  printIntroduction: function() {

 console.log(`Hi ,  ${this.name}. Human? ${this.isHuman}`);

  }

};
```

```
const me = Object.create(people);

// "name" is a property set on "me", but not on "person"

me.name = "Marry";

 // inherited properties can be overwritten

me.isHuman = true;

me.printIntroduction();
```

*****Follow the instruction on **05(OOP1).js, 05(OOP2).js,05(OOP3).js** and work on object creation

Object Oriented Javascript(Inheritance)

Javascript is prototype based language . It uses a prototype object in ordered to make inheritance possible . prototype , ES6 extend

Syntax :

1. **Create Inherited Property**

```
ParentObject.prototype.PropName = property/method
```

2. **Call Parent Constructor On Child Constructor function**

```
Paretent.call( this, par1, par2); //list of parent properties
```

3. **Inherit the Property**

```
ChildObject.prototype = Object.create(ParetentObject.prototype);
```

4. **Set the constructor function to be Child**

```
ChildObject.prototype.constructor = ChildObject;
```

Other : ES6 class extend [Syntactic Sugar]

childClass extends ParentClass

Object Oriented Javascript(Inheritance)

E.g

```
function Person(firstName, lastName) {
 this.firstName = firstName || "unknown";
 this.lastName = lastName || "unknown";
}
Person.prototype.getFullName = function ()
{
   return this.firstName + " " + this.lastName;
}
function Student(firstName, lastName, schoolName)
{
   Person.call(this, firstName, lastName);
   this.schoolName = schoolName || "unknown";
}

//⇒ One of the Three ways
//Student.prototype = Person.prototype;
//Student.prototype = Object.create(Person.prototype);
 Student.prototype = new Person();

//Set the constructor
Student.prototype.constructor = Student;
```

```
var std = new Student("James","Bond", "XYZ");

alert(std.getFullName()); // James
```

Object Oriented Javascript(Inheritance)

E.g

```
class Person
{
constructor(firstName, lastName) {
 this.firstName = firstName || "unknown";
 this.lastName = lastName || "unknown"; }
 getFullName  ()  {
    return this.firstName + " " + this.lastName;
}
}
class Student extends Person
{
constructure(firstName, lastName, schoolName) {
    super(firstName, lastName);
    this.schoolName = schoolName || "unknown";
}}
```

```
var std = new Student("James","Bond", "XYZ");

alert(std.getFullName()); // James
```

***Follow the instruction on **05(OOP4).js, 05(OOP5).js** and work on Inheritance

Exception Handling in Javascript

The try...catch statement marks a block of statements to try and specifies a response should an exception be thrown

Syntax :

```
try {    // Code to run   }
catch ( e ) {  // Code to run if an exception occurs }
```

E.G

```
try {  nonExistentFunction(); }
catch (error) { console.error(error);}
   // expected output: ReferenceError: nonExistentFunction is not defined
   // Note - error messages will vary depending on browser
```

***Follow the instruction on **06(exception).js** and work on division by zero exception

ES6 Modules

An ES6 module is a file containing JS code. You can use import and export in modules

Syntax : export {var , fun , class }   //as last line

import {var,fun,class} from 'fileName.js'

Export Single Element : export fun, var , class   //in front of the element

Other Ways(Aliasing) :

export {var as var 2, fun as fun2, ….. }   //export with alias name

import {var as var 2} from 'fileName.js'   //import with alias name

import * as Util from 'fileName.js'   //import all as alias

Note : On js import keyword exist, you need to add type="module" in the html

<script src='app.js' type="module"> </script>

E.g export function add(x,y){return x+y}; import {add} from "app.js";

*******Follow the instruction on **07(Modules_Export).js, 07(Modules_Import).js** and work on exporting and importing modules

Read Assignment (Demo)

1. What is 'strict mode' ? the rules ? Demo them : Link
2. What is Javascript Linting ? Link
   a. Check JSLint
   b. Tools
3. Important array methods : Link
4. What are configurable , enumerable and writable properties? Link
5. How to add and read properties of object with (Link)
   a. Object.defineProperty
   b. Object.getOwnPropertyNames
   c. More ....
6. What is Proxy API ? Link
7. What is Transpiling ? Link , LInk 1, Link 2

THE END !!!