GUI BASICS

Documentation

Abstract

In this module I learned basic concept HTML, CSS and JavaScript and learned how we can developed frontend in web development and how we can integrate frontend with web API in web development.

Dhruvil Dobariya
dhruvildobariya21@gmail.com

INDEX

BASIC OF HTML

1.1 WHAT IS HTML?

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

1.2 HISTORY OF HTML

- The first version of HTML was written by Tim Berners-Lee in 1993.
- Since then, there have been many different versions of HTML.
- The most widely used version throughout the 2000's was HTML 4.01, which became an official standard in December 1999.
- Currently Html 5 is stable version of html.

1.3 STRUCTURE OF HTML

Example:

```
<!DOCTYPE html>
<html>
<head>
    <title>Page Title</title>
</head>
<body>

    <h1>My First Heading</h1>
    <p>My first paragraph.</p>

</body>
</html>
```

- The <!DOCTYPE html> declaration defines that this document is an HTML5 document.
- The <html> element is the root element of an HTML page.
- The <head> element contains meta information about the HTML page.

- The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab).
- The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The <h1> element defines a large heading.
- The <p> element defines a paragraph.

BASIC CONTROLS

2.1 FORM:

- The <form> tag is used to create an HTML form for user input.
- The <form> element can contain one or more of the following form elements,
  - <input>
  - <textarea>
  - <button>
  - <select>
  - <option>
  - <optgroup>
  - <fieldset>
  - <label>
  - <output>

2.1.1 ATTRIBUTES:

- The <form> tag also supports the Global Attributes in HTML.
- The <form> tag also supports the Event Attributes in HTML.
- The <form> tag has own attributes, like…

Attribute	Value	Description
accept-charset	character_set	Specifies the character encodings that are to be used for the form submission
action	URL	Specifies where to send the form-data when a form is submitted
autocomplete	on off	Specifies whether a form should have autocomplete on or off
enctype	application/x-www-form-urlencoded multipart/form-data text/plain	Specifies how the form-data should be encoded when submitting it to the server (only for method="post")
method	get post	Specifies the HTTP method to use when sending form-data
name	text	Specifies the name of a form
novalidate	novalidate	Specifies that the form should not be validated when submitted

rel	external help license next nofollow noopener noreferrer opener prev search	Specifies the relationship between a linked resource and the current document
target	_blank _self _parent _top	Specifies where to display the response that is received after submitting the form

2.2 File Controls

- The <input type="file"> defines a file-select field and a "Browse" button for file uploads.
- To define a file-select field that allows multiple files to be selected, add the multiple attribute.

Example:

```
<input type="file" name="" id="">
```

2.2.1 Attributes:

- It contains all common attribute which have input tag.
- Like, id, name, class…

Value:

- A file input's value attribute contains a string that represents the path to the selected file(s).
- If no file is selected yet, the value is an empty string ("").
- When the user selected multiple files, the value represents the first file in the list of files they selected.

Accept:

- The accept attribute value is a string that defines the file types the file input should accept.
- This string is a comma-separated list of unique file type specifiers.

- Because a given file type may be identified in more than one manner, it's useful to provide a thorough set of type specifiers when you need files of a given format.
- A valid case-insensitive filename extension, starting with a period (".") character.
- For example: .jpg, .pdf, or .doc.
- A valid MIME type string, with no extensions.
- The string audio/* meaning "any audio file".
- The string video/* meaning "any video file".
- The string image/* meaning "any image file".

Example:

```
<input type="file" accept="image/*,.pdf" />
```

Capture:

- The capture attribute value is a string that specifies which camera to use for capture of image or video data.
- if the accept attribute indicates that the input should be of one of those types.
- A value of user indicates that the user-facing camera and/or microphone should be used.
- A value of environment specifies that the outward-facing camera and/or microphone should be used.
- If this attribute is missing, the user agent is free to decide on its own what to do.
- If the requested facing mode isn't available, the user agent may fall back to its preferred default mode.

Multiple:

- When the multiple Boolean attribute is specified, the file input allows the user to select more than one file.

2.2.2 GETTING INFORMATION ON SELECTED FILES

- The selected files' are returned by the element's HTMLInputElement.files property, which is a FileList object containing a list of File objects.
- The FileList behaves like an array, so you can check its length property to get the number of selected files.
- Each File object contains the following proparties,

name:

- The file's name.

lastModified:

- A number specifying the date and time at which the file was last modified, in milliseconds since the UNIX epoch (January 1, 1970 at midnight).

size:

- The size of the file in bytes.

type:

- The file's MIME type.

CONTROL'S ATTRIBUTES

3.1 INTRODUCTION

- HTML have some basic control attributes like,
  - name
  - id
  - class
  - value

3.2 NAME

- The name attribute specifies the name of an <input> element.
- The name attribute is used to reference elements in a JavaScript, or to reference form data after a form is submitted.
- Only form elements with a name attribute will have their values passed when submitting a form.

Example:

```
<label for="name">Name : </label>
<input type="text" name="name" id="name">
```

3.3 VALUE:

- The value attribute specifies the value of an <input> element.

Example:

```
<label for="name">Name : </label>
<input type="text" name="name" id="name" value="Dhruvil Dobariya">
```

3.4 ID

- The HTML id attribute is used to specify a unique id for an HTML element.
- You cannot give more than one element with the same id in an HTML document.
- The id attribute is used to point to a specific style declaration in a style sheet.
- It is also used by JavaScript to access and manipulate the element with the specific id.
- The syntax for id is: write a hash character (#), followed by an id name.
- Id also used for bookmark.

- To use a bookmark, you must first create it, and then add a link to it.
- Then, when the link is clicked, the page will scroll to the location with the bookmark.
- It is a global attribute, means it can be used with all HTML elements.

3.5 CLASS

- The class attribute specifies one or more classnames for an element.
- The class attribute is mostly used to point to a class in a style sheet.
- It can also be used by a JavaScript (via the HTML DOM) to make changes to HTML elements with a specified class.
- It is a global attribute, means it can be used with all HTML elements.

BASIC TAG WITH ITS ATTRIBUTE

4.1 ACTION TAG

- The <a> tag defines a hyperlink, which is used to link from one page to another.
- The most important attribute of the <a> element is the href attribute, which indicates the link's destination.
- By default, links will appear as follows in all browsers:
  - An unvisited link is underlined and blue
  - A visited link is underlined and purple
  - An active link is underlined and red
- But we can change these color using CSS.
- The <a> tag also supports the Global Attributes and Event Attribute in HTML.

4.1.1 ATTRIBUTES:

Attribute	Value	Description
download	filename	Specifies that the target will be downloaded when a user clicks on the hyperlink
href	URL	Specifies the URL of the page the link goes to
hreflang	language_code	Specifies the language of the linked document
media	media_query	Specifies what media/device the linked document is optimized for
ping	list_of_URLs	Specifies a space-separated list of URLs to which, when the link is followed, post requests with the body ping will be sent by the browser (in the background). Typically used for tracking.
referrerpolicy	no-referrer no-referrer-when-downgrade origin origin-when-cross-origin same-origin strict-origin-when-cross-origin unsafe-url	Specifies which referrer information to send with the link

rel	alternate author bookmark external help license next nofollow noreferrer noopener prev search tag	Specifies the relationship between the current document and the linked document
target	_blank _parent _self _top	Specifies where to open the linked document
type	media_type	Specifies the media type of the linked document

4.2 IMAGE TAG

- The <img> tag is used to embed an image in an HTML page.
- Images are not technically inserted into a web page, images are linked to web pages.

4.2.1 ATTRIBUTES:

Attribute	Value	Description
alt	text	Specifies an alternate text for an image
crossorigin	anonymous use-credentials	Allow images from third-party sites that allow cross-origin access to be used with canvas
height	pixels	Specifies the height of an image
ismap	ismap	Specifies an image as a server-side image map
loading	eager lazy	Specifies whether a browser should load an image immediately or to defer loading of images until some conditions are met
longdesc	URL	Specifies a URL to a detailed description of an image
referrerpolicy	no-referrer no-referrer-when-downgrade origin	Specifies which referrer information to use when fetching an image

	origin-when-cross-origin unsafe-url	
sizes	sizes	Specifies image sizes for different page layouts
src	URL	Specifies the path to the image
srcset	URL-list	Specifies a list of image files to use in different situations
usemap	#mapname	Specifies an image as a client-side image map
width	pixels	Specifies the width of an image

4.3 META TAG

- The <meta> tag defines metadata about an HTML document. Metadata is data (information) about data.
- <meta> tags always go inside the <head> element.
- It is used to specify,
  - specify character set
  - page description
  - keywords
  - author of the document
  - viewport settings
- It will not be displayed on the page, but is machine parsable.
- It is used by browsers (how to display content or reload page), search engines (keywords), and other web services.
- There is a method to let web designers take control over the viewport (the user's visible area of a web page), using the <meta> tag.

4.3.1 ATTRIBUTES:

Attribute	Value	Description
charset	character_set	Specifies the character encoding for the HTML document
content	text	Specifies the value associated with the http-equiv or name attribute
http-equiv	content-security-policy content-type default-style refresh	Provides an HTTP header for the information/value of the content attribute

name	application-name author description generator keywords viewport	Specifies a name for the metadata

Example:

```
<head>
    <!-- Define keywords for search engines: -->
    <meta name="keywords" content="HTML, CSS, JavaScript">

    <!-- Define a description of your web page: -->
    <meta name="description" content="About meta tag">

    <!-- Define the author of a page: -->
    <meta name="author" content="Dhruvil A. Dobariya">

    <!-- Refresh document every 30 seconds: -->
    <meta http-equiv="refresh" content="30">

    <!-- Setting the viewport to make your website look good on all devices:
-->
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>
</head>
```

4.4 RESPONSIVENESS

- It is a web development approach that creates dynamic changes to the appearance of a website, depending on the screen size and orientation of the device being used to view it.
- For archive responsiveness we have three approach,
    - Media Query
    - Web Browser
    - Website Interface(HTML, CSS, Javascript)

Media Query:

- Media queries are supported by the latest web browsers and work by creating media queries.
- They are called “media queries” and can be used in various ways, such as in connection with the layout of a page or as part of the content of an application.
- They also permit designers to fabricate various formats utilizing similar HTML archives by specifically serving templates dependent on the client specialist’s highlights, like the browser’s window’s size.

Web Browser:

- Since websites can contain many images, it is essential to keep these images fluid too.
- The thought behind liquid pictures is that you convey concepts at the greatest size they will utilize.
- In responsive web pages, designers don’t announce the stature and width in your code instead, they let the browsers resize the pictures on a case by case basis while utilizing CSS to manage their relative size.
- It’s an extraordinary and straightforward procedure to resize images correctly.

Website Interface(HTML, CSS, Javascript):

- Many mobile devices are touchscreen nowadays, which requires mindfulness of the size of the interactive elements within interfaces.
- Aside from the flexibility of images, usage of native controls such as keyboards and drop-out menus should be optimized so it can provide a great experience to its users, whether it’s on mobile or desktop**.**

4.5 Other Tags

4.5.1 Basic Tags:

Tag	Description
<!DOCTYPE>	Defines the document type
<html>	Defines an HTML document
<head>	Contains metadata/information for the document
<title>	Defines a title for the document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings
<p>	Defines a paragraph
** **	Inserts a single line break
<hr>	Defines a thematic change in the content

<!--...-->	Defines a comment

4.5.2 Forms and Inputs Tags:

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control
<textarea>	Defines a multiline input control (text area)
<button>	Defines a clickable button
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<datalist>	Specifies a list of pre-defined options for input controls
<output>	Defines the result of a calculation

4.5.3 Images Tags:

Tag	Description
<img>	Defines an image
<svg>	Defines a container for SVG graphics

4.5.4 Audio / Video Tags:

Tag	Description
<audio>	Defines sound content
<source>	Defines multiple media resources for media elements (<video>, <audio> and <picture>)
<track>	Defines text tracks for media elements (<video> and <audio>)
<video>	Defines a video or movie

4.5.5 Links Tags:

Tag	Description
<a>	Defines a hyperlink
<link>	Defines the relationship between a document and an external resource (most used to link to style sheets)
<nav>	Defines navigation links

4.5.6 List Tags:

Tag	Description
** **	Defines an unordered list
** **	Defines an ordered list
** **	Defines a list item
<dir>	Not supported in HTML5. Use instead. Defines a directory list
<dl>	Defines a description list
<dt>	Defines a term/name in a description list
<dd>	Defines a description of a term/name in a description list

4.5.7 Table Tags:

Tag	Description
<table>	Defines a table
<caption>	Defines a table caption
<th>	Defines a header cell in a table
<tr>	Defines a row in a table
<td>	Defines a cell in a table
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table
<col>	Specifies column properties for each column within a <colgroup> element
<colgroup>	Specifies a group of one or more columns in a table for formatting

4.5.8 Styles and Semantics:

Tag	Description
<style>	Defines style information for a document
** **	Defines a section in a document
****	Defines a section in a document
<header>	Defines a header for a document or section
<footer>	Defines a footer for a document or section
<main>	Specifies the main content of a document
<section>	Defines a section in a document
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window
<summary>	Defines a visible heading for a <details> element

<data>	Adds a machine-readable translation of a given content

4.5.9 META INFO TAGS:

Tag	Description
<head>	Defines information about the document
<meta>	Defines metadata about an HTML document

BASIC OF CSS

5.1 INTRODUCTION

- CSS stands for Cascading Style Sheets.
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
- It can control the layout of multiple web pages all at once.

5.2 TYPES OF CSS

- We have three types of CSS,
  - Inline CSS
  - Internal or Embedded CSS
  - External CSS

5.2.1 INLINE CSS:

- Inline CSS contains the CSS property in the body section attached with element is known as inline CSS.
- This kind of style is specified within an HTML tag using the style attribute.

5.2.2 INTERNAL OR EMBEDDED CSS:

- This can be used when a single HTML document must be styled uniquely.
- The CSS rule set should be within the HTML file in the head section.

5.2.3 EXTERNAL CSS:

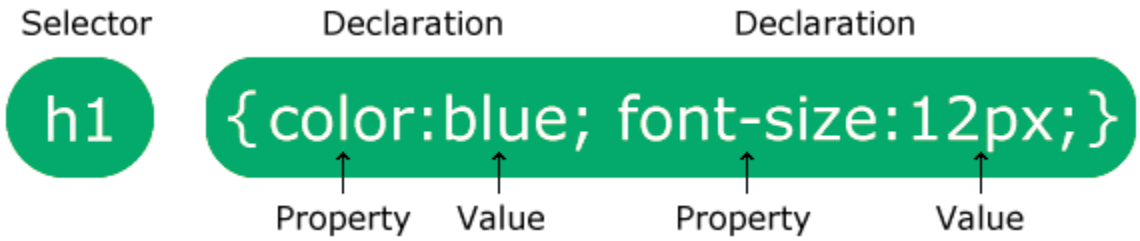
- External CSS contains separate CSS file which contains only style property with the help of tag attributes.
- CSS property written in a separate file with .css extension and should be linked to the HTML document using link tag.
- This means we can use CSS file in multiple HTML documents.

5.2.4 PRIORITY OF CSS:

- Inline CSS has the highest priority, then comes Internal/Embedded followed by External CSS which has the least priority.
- Multiple style sheets can be defined on one page.
- If for an HTML tag, styles are defined in multiple style sheets then the below order will be followed.

5.3 CSS SYNTAX

- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.



Example:

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

- Here, p is a selector in CSS.
- color is a property, and red is the property value.
- text-align is a property, and center is the property value.

5.4 SELECTORS

- CSS have different types of selectors like,
  - Basic Selector
  - Grouping Selector
  - Combinator Selector
  - Pseudo Selector

5.4.1 BASIC SELECTOR:

- Basic selector contains five different types like,
  - Universal selector
  - Class selector

- ID selector
- Attribute selector
- Type selector

Universal Selector:

- Selects all elements.
- Optionally, it may be restricted to a specific namespace or to all namespaces.

Syntax: * ns|* *|*

Example:

```
*{
    background-color: rgb(209, 209, 209);
    font-size: 25px;
}
```

Class Selector:

- Selects all elements that have the given class attribute.

Syntax: .classname

Example:

```
.class-selector{
    color: white;
}
```

Id Selector:

- Selects an element based on the value of its id attribute.

Syntax: #idname

Example:

```
#Id-Selector{
    color: red;
}
```

Type(Tag) Selector:

- Selects all elements that have the given node name.

Syntax: elementname

Example:

```
span{
    color: green;
}
```

Attribute Selector:

- Selects all elements that have the given attribute.

Syntax: [attr], [attr=value], [attr~=value], [attr|=value], [attr^=value], [attr$=value], [attr*=value]

Example:

```
[type=text]{
    color: aqua;
}
```

5.4.2 Grouping Selector:

- Grouping selector contains five different types like,
    - Selector list

Select list Selector:

- Selects all the matching nodes.
- All node separated by ",".

Syntax: elementname1, elementname2

Example:

```
i,.select-list-with-class{
    color: blue;
}
```

5.4.3 Combinator Selector

- Combinator selector contains five different types like,
    - Descendant combinator
    - Child combinator
    - General sibling combinator

- Adjacent sibling combinator

Descendant Combinator:

- It select elements which is exist inside the particular element.
- First we write element which contain our selector node and give space and we write second element which is our selector node.

Syntax: div span

- So here, “div span” will match all elements that are exist a element.

Example:

```
div b{
    color: blueviolet;
}
```

Child Combinator:

- It select elements which is directly children of the particular element.
- First we write element which is **directly parent node** of our selector node and write “>” and we write second element which is our selector node.

Syntax: ui > li

- So here, “ui > li” will match all <li> elements that are directly child of the <ui> element.

Example:

```
p>b{
    color: brown;
}
```

General Sibling Combinator:

- It select elements which is sibling of the particular element.
- First we write element which is sibling of our selector node and write “~” and we write second element which is our selector node.

Syntax: p ~ span

- So here, “p ~ span” will match all elements that are sibling of the <p> element.

Example:

```
h3~h4,h3~.sibling-combinator-with-class{
    color: cadetblue;
}
```

Adjacent Combinator:

- It select elements which **first and immediately follows** the particular element.
- First we write element which is followed by our selector node and write "+" and we write second element which is our selector node.

Syntax: h2 + p

- So here, "h2 + p" will match all <p> element that is first and followed the <h2> element.

Example:

```
header+footer{
    color: yellowgreen;
}
```

5.4.4 Pseudo Selector

- Pseudo selector contains five different types like,
    - Pseudo classes selector
    - Pseudo elements selector

Pseudo Classes Selector:

- It select element based on their state.
- First we write element which is our selector node and write ":" and we write state of that element when we want to select it in that state.

Syntax: a:visited

- So, here <a> is selected when it visited.

Example:

```
a:visited{
    color: coral;
}
input:invalid{
    background-color: red;
}
```

Pseudo Element Selector:

- It is used to select particular parts of selector.
- First we write element which is our selector node and write "::" and we write part of that element which we want to select it.

Syntax: p::first-line

- So, here first line of <p> is selected.

Example:

```
p::first-line{
    color: rgb(73, 82, 27);
}
```

5.5 Properties

- CSS have many properties, like background, text, box-model…

5.5.1 Background Properties:

Property	Description
background	Defines a variety of background properties within one declaration.
background-attachment	Specify whether the background image is fixed in the viewport or scrolls.
background-clip	Specifies the painting area of the background.
background-color	Defines an element's background color.
background-image	Defines an element's background image.
background-origin	Specifies the positioning area of the background images.
background-position	Defines the origin of a background image.
background-repeat	Specify whether/how the background image is tiled.
background-size	Specifies the size of the background images.

5.5.2 Font Properties:

Property	Description
font	Defines a variety of font properties within one declaration.
font-family	Defines a list of fonts for element.
font-size	Defines the font size for the text.
font-size-adjust	Preserves the readability of text when font fallback occurs.

font-stretch	Selects a normal, condensed, or expanded face from a font.
font-style	Defines the font style for the text.
font-variant	Specify the font variant.
font-weight	Specify the font weight of the text.
color	Specify the color of the text of an element.

5.5.3 Text Properties:

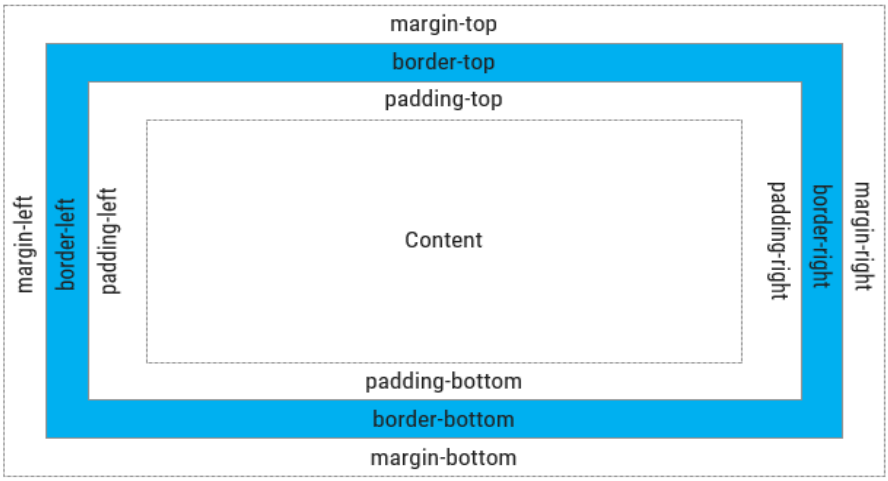
Property	Description
text-align	Sets the horizontal alignment of inline content.
text-align-last	Specifies how the last line of a block or a line right before a forced line break is aligned when text-align is justify.
text-decoration	Specifies the decoration added to text.
text-decoration-color	Specifies the color of the text-decoration-line.
text-decoration-line	Specifies what kind of line decorations are added to the element.
text-decoration-style	Specifies the style of the lines specified by the text-decoration-line property
text-indent	Indent the first line of text.
text-justify	Specifies the justification method to use when the text-align property is set to justify.
text-overflow	Specifies how the text content will be displayed, when it overflows the block containers.
text-shadow	Applies one or more shadows to the text content of an element.
text-transform	Transforms the case of the text.

5.5.4 Box Model Properties:

- All HTML elements can be considered as boxes.
- In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around HTML elements, and it consists of margins, borders, padding, and the actual content.

GUI Basic

- The box model allows us to place a border around elements and space elements in relation to other elements.



Property	Description
padding-bottom	Sets the padding to the bottom side of an element.
padding-left	Sets the padding to the left side of an element.
padding-right	Sets the padding to the right side of an element.
padding-top	
margin	Sets the margin on all four sides of the element.
margin-bottom	Sets the bottom margin of the element.
margin-left	Sets the left margin of the element.
margin-right	Sets the right margin of the element.
margin-top	Sets the top margin of the element.
border	Sets the width, style, and color for all four sides of an element's border.
border-bottom	Sets the width, style, and color of the bottom border of an element.
border-bottom-color	Sets the color of the bottom border of an element.
border-bottom-left-radius	Defines the shape of the bottom-left border corner of an element.
border-bottom-right-radius	Defines the shape of the bottom-right border corner of an element.
border-bottom-style	Sets the style of the bottom border of an element.
border-bottom-width	Sets the width of the bottom border of an element.
border-collapse	Specifies whether table cell borders are connected or separated.

border-color	Sets the color of the border on all the four sides of an element.
border-image	Specifies how an image is to be used in place of the border styles.
border-image-outset	Specifies the amount by which the border image area extends beyond the border box.
border-image-repeat	Specifies whether the image-border should be repeated, rounded or stretched.
border-image-slice	Specifies the inward offsets of the image-border.
border-image-source	Specifies the location of the image to be used as a border.
border-image-width	Specifies the width of the image-border.
border-left	Sets the width, style, and color of the left border of an element.
border-left-color	Sets the color of the left border of an element.
border-left-style	Sets the style of the left border of an element.
border-left-width	Sets the width of the left border of an element.
border-radius	Defines the shape of the border corners of an element.
border-right	Sets the width, style, and color of the right border of an element.
border-right-color	Sets the color of the right border of an element.
border-right-style	Sets the style of the right border of an element.
border-right-width	Sets the width of the right border of an element.
border-spacing	Sets the spacing between the borders of adjacent table cells.
border-style	Sets the style of the border on all the four sides of an element.
border-top	Sets the width, style, and color of the top border of an element.
border-top-color	Sets the color of the top border of an element.
border-top-left-radius	Defines the shape of the top-left border corner of an element.
border-top-right-radius	Defines the shape of the top-right border corner of an element.
border-top-style	Sets the style of the top border of an element.
border-top-width	Sets the width of the top border of an element.
border-width	Sets the width of the border on all the four sides of an element.

5.5.5 List Properties:

Property	Description
list-style	Defines the display style for a list and list elements.
list-style-image	Specifies the image to be used as a list-item marker.
list-style-position	Specifies the position of the list-item marker.
list-style-type	Specifies the marker style for a list-item.

5.5.6 Overflow Properties:

Property	Description
overflow	Specifies the treatment of content that overflows the element's box.
overflow-x	Specifies the treatment of content that overflows the element's box horizontally.
overflow-y	Specifies the treatment of content that overflows the element's box vertically.

5.5.7 Flexbox Properties:

Property	Description
flex	Specifies the components of a flexible length.
flex-basis	Specifies the initial main size of the flex item.
flex-direction	Specifies the direction of the flexible items.
flex-flow	A shorthand property for the flex-direction and the flex-wrap properties.
flex-grow	Specifies how the flex item will grow relative to the other items inside the flex container.
flex-shrink	Specifies how the flex item will shrink relative to the other items inside the flex container.
flex-wrap	Specifies whether the flexible items should wrap or not.
order	Specifies the order in which a flex items are displayed and laid out within a flex container.

5.5.8 Grid Properties:

Property	Description
column-gap	Specifies the gap between the columns
gap	A shorthand property for the row-gap and the column-gap properties
grid	A shorthand property for the grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns, and the grid-auto-flow properties
grid-area	Either specifies a name for the grid item, or this property is a shorthand property for the grid-row-start, grid-column-start, grid-row-end, and grid-column-end properties
grid-auto-columns	Specifies a default column size
grid-auto-flow	Specifies how auto-placed items are inserted in the grid
grid-auto-rows	Specifies a default row size
grid-column	A shorthand property for the grid-column-start and the grid-column-end properties
grid-column-end	Specifies where to end the grid item
grid-column-gap	Specifies the size of the gap between columns

grid-column-start	Specifies where to start the grid item
grid-gap	A shorthand property for the grid-row-gap and grid-column-gap properties
grid-row	A shorthand property for the grid-row-start and the grid-row-end properties
grid-row-end	Specifies where to end the grid item
grid-row-gap	Specifies the size of the gap between rows
grid-row-start	Specifies where to start the grid item
grid-template	A shorthand property for the grid-template-rows, grid-template-columns and grid-areas properties
grid-template-areas	Specifies how to display columns and rows, using named grid items
grid-template-columns	Specifies the size of the columns, and how many columns in a grid layout
grid-template-rows	Specifies the size of the rows in a grid layout
row-gap	Specifies the gap between the grid rows

5.5.9 Animation Properties:

Property	Description
animation	Specifies the keyframe-based animations.
animation-delay	Specifies when the animation will start.
animation-direction	Specifies whether the animation should play in reverse on alternate cycles or not.
animation-duration	Specifies the number of seconds or milliseconds an animation should take to complete one cycle.
animation-fill-mode	Specifies how a CSS animation should apply styles to its target before and after it is executing.
animation-iteration-count	Specifies the number of times an animation cycle should be played before stopping.
animation-name	Specifies the name of @keyframes defined animations that should be applied to the selected element.
animation-play-state	Specifies whether the animation is running or paused.
animation-timing-function	Specifies how a CSS animation should progress over the duration of each cycle.

5.5.10 Transition Properties:

Property	Description
transition	Defines the transition between two states of an element.

transition-delay	Specifies when the transition effect will start.
transition-duration	Specifies the number of seconds or milliseconds a transition effect should take to complete.
transition-property	Specifies the names of the CSS properties to which a transition effect should be applied.
transition-timing-function	Specifies the speed curve of the transition effect.

5.5.11 TRANSFORM PROPERTIES:

Property	Description
transform	Applies a 2D or 3D transformation to an element.
transform-origin	Defines the origin of transformation for an element.
transform-style	Specifies how nested elements are rendered in 3D space.

5.5.12 OTHER PROPERTIES:

Property	Description
cursor	Specify the type of cursor.
direction	Define the text direction/writing direction.
display	Specifies how an element is displayed onscreen.
position	Specifies how an element is positioned.
opacity	Specifies the transparency of an element.
float	Specifies whether or not a box should float.
height	Specify the height of an element.
width	Specify the width of an element.
left	Specify the location of the left edge of the positioned element.
right	Specify the location of the right edge of the positioned element.
top	Specify the location of the top edge of the positioned element.
bottom	Specify the location of the bottom edge of the positioned element.
z-index	Specifies a layering or stacking order for positioned elements.
max-height	Specify the maximum height of an element.
max-width	Specify the maximum width of an element.
min-height	Specify the minimum height of an element.
min-width	Specify the minimum width of an element.
word-break	Specifies how to break lines within words.
word-spacing	Sets the spacing between words.
word-wrap	Specifies whether to break words when the content overflows the boundaries of its container.

BASIC OF BOOTSTRAP

6.1 INTRODUCTION:

- Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.
- Bootstrap is completely free to download and use!
- It provides faster and easier web development.
- It contains HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins.
- It also gives you the ability to easily create responsive designs.
- Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter, and released as an open source product in August 2011 on GitHub.
- Bootstrap 5 is a latest version of Bootstrap.
- Until Bootstrap 4, it use jQuery with Javascript.
- But in Bootstrap 5, it removed jQuery and it totally use vanilla Javascript for DOM manipulation.

6.2 HOW TO USE BOOTSTRAP

- We have two different way to use Bootstrap,
  - Install Bootstrap
  - Using CDN
- We should include four file in HTML file for use Bootstrap.
  - bootstrap.css or bootstrap.min.css
  - bootstrap.js or bootstrap.min.js
  - popper.js or popper.min.js
  - jQuery (if use Bootstrap 5 then doesn't need to include)
- jQuery must include above the all JS files.
- We can also include the bundle.js or it's minified file.
- bundle.js is a combination of bootstrap.js and popper.js file, So these file don't need to include when we include bundle.js or it's minified file.

6.3 STRUCTURE OF BOOTSTRAP

- A web page using twitter bootstrap has a basic html structure which should start from type of document declaration, navigation and container in body, adding style sheets, java scripts declaration and Meta tags in header.
- For uniform rendering of its components and controls across all browsers, a HTML 5 doctype is needed by twitter bootstrap.
- This is due to the fact that HTML 5 elements are understood by the bootstrap which makes it necessary to include an appropriate doctype tag to the web page.

6.3.1 HTML Header Structure:

- In order for a responsive working of the bootstrap with mobile devices and latest browsers, the minimum requirements in the html header part should be to include a bootstrap style sheet and declare a viewport meta tag, which is crucial for a mobile device and bootstrap to be able to respond in accordance with the zoom level and the width of display.
- Adjusting viewport's width enables browsers to appropriately respond for the display size.
- During the earlier versions of bootstrap, a separate style sheet referred to as bootstrap-responsive existed.
- But latter, an inbuilt responsive in nature base style sheet known as bootstrap.css which is a single style sheet that handles web page responsiveness was developed.
- This bootstrap.css proves to be very useful for debugging during development.

6.3.2 HTML Body Structure:

- Before you close the body tag in the html body, the bootstrap java script file and jQuery file must be included.
- The bootstrap java script file should be rendered after the jQuery file.

6.3.3 Layout:

Fluid layout:

- No extra style sheet or additional step is required to obtain a responsive fluid layout in the bootstrap.
- It is designed to implement the responsive layout by default.
- This layout is highly recommended for public websites.

Fixed layout:

- Creating a website with a fixed layout is not generally recommended.

- This is mainly due to the increasing amount of internet website traffic that is arising from small devices such as smart phones and tablets.
- A fixed layout is however needed in some cases such as intranet applications.
- If this is the case, steps followed to disable the fluid responsiveness of the layout like,
    - Removing viewport mega tag
    - Override width property to a fixed size
    - If you are using NavBar, get rid of expanding and collapsing behavior
    - Instead of .col-md-* and .col-lg-*., use col-xs-* class in grid layouts

6.3.4 COMPONENTS:

Menu section:

- In bootstrap, to design a menu is the easiest thing to do in the web designing world.
- This is because it is designed to be responsive by default and even has the ability to get new appearances in smaller devices.

Header area:

- Jumbotron, which can display large headers and contents is a highly usable class offered by bootstrap.
- It is largely utilized by product based websites.

The content area:

- This content should be divided into three equal parts placed side by side.
- With the bootstrap's flex box-based grid, doing this is one easy and fast task.
- A12-column grid system is provided, so dividing the screen into equal parts and all one needs to do is to specify which HTML is occupied by each part.

Footer area:

- Footer area uses the same principle as the content area.

BASIC OF JAVASCRIPT

7.1 INTRODUCTION

- JavaScript (JS) is a lightweight, interpreted(implementations execute instructions directly without earlier compiling a program into machine language), or just-in-time(run time compilation) compiled programming language with first-class functions(function treat like variable).
- is a prototype-based(classes are not explicitly defined), multi-paradigm, single-threaded(run on main thread), dynamic language(interpreter assigns variables a type at runtime based on the variable's value).
- It is used for client-side scripting as well as server side using Node.js.

7.2 USE OF JAVASCRIPT

- JavaScript helps the users to build modern web applications to interact directly without reloading the page every time.
- JavaScript is commonly used to dynamically modify HTML and CSS to update a user interface by the DOM API.
- It is mainly used in web applications.
    - Web Development
    - Mobile Development
    - Game Development
    - Presentation

7.3 WAY TO USE JAVASCRIPT

- We should use JavaScript two different way,
    - Internal scripting
    - External scripting

7.3.1 INTERNAL SCRIPTING:

- We wrap script inside <script>.
- We use <script> inside <head> or <body>.

7.3.2 External SCRIPTING:

- We write script in separate JavaScript file, which have ".js" extension.
- We use <script> inside <head> or <body>.

7.4 Syntax of JavaScript

Basic Syntax:

- JavaScript is case-sensitive.
- Statements should end in a semicolon (;), but it is not compulsory.

Variable Syntax:

- Must be defined before being used.
- The variable name can contain A – Z, a – z, underscore or digits and must start with a letter or an underscore ("_").
- The data type does not have to be explicitly defined.

Comment Syntax:

- For single line comment we use "//".
- For multiline comment we use "*/…/*"

7.5 Events in JavaScript

- Events are very important part of DOM.
- Using JavaScript we should manipulate element of DOM based on various events.
- We have four types of events in JavaScript,
  - Mouse events
  - Keyboard events
  - Frame or Object events
  - Form events

7.5.1 Mouse events:

- These events base on mouse.
- Here we should manipulate DOM on mouse events.

Event	Attribute	Description
click	onclick	The event occurs when the user clicks on an element
dblclick	ondblclick	The event occurs when the user double-clicks on an element
mousedown	onmousedown	The event occurs when a user presses a mouse button over an element
mousemove	onmousemove	The event occurs when a user moves the mouse pointer over an element

mouseover	onmouseover	The event occurs when a user mouse over an element
mouseout	onmouseout	The event occurs when a user moves the mouse pointer out of an element
mouseup	onmouseup	The event occurs when a user releases a mouse button over an element

7.5.2 Keyboard Events:

- These events based on keyboard.
- Here, we should manipulate DOM based on keyboard event.

Event	Attribute	Description
keydown	onkeydown	The event occurs when the user is pressing a key or holding down a key
keypress	onkeypress	The event occurs when the user is pressing a key or holding down a key
keyup	onkeyup	The event occurs when a keyboard key is released

7.5.3 Frame or Object Events:

- These events based on object.
- Here, we should manipulate DOM based on object event.

Event	Attribute	Description
abort	onabort	The event occurs when an image is stopped from loading before completely loaded (for <object>)
error	onerror	The event occurs when an image does not load properly (for <object>, <body> and <frameset>)
load	onload	The event occurs when a document, frameset, or <object> has been loaded
resize	onresize	The event occurs when a document view is resized
scroll	onscroll	The event occurs when a document view is scrolled

7.5.4 Form Events:

- These events based on form.
- Here, we should manipulate DOM based on form event.

Event	Attribute	Description
blur	onblur	The event occurs when a form element loses focus
change	onchange	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)

focus	onfocus	The event occurs when an element gets focus (for <label>, <input>, <select>, textarea>, and <button>)
reset	onreset	The event occurs when a form is reset
select	onselect	The event occurs when a user selects some text (for <input> and <textarea>)
submit	onsubmit	The event occurs when a form is submitted

7.6 EVENTLISTENER

- Event listener is a method which used to attached event handler with particular element.
- We can attach many event handler on a element.
- Here, we are attach event handler with the DOM object not HTML element.
- It provide more reliability and flexibility, because we are not deal with HTML markup, we can directly control any element without HTML markup control.
- We have two method in event listener.

Syntax:

```
element.addEventListener(event, function, useCapture); // Add event listener
element.removeEventListener(); // remove elevent listener
```

- “event” parameter is use to specify event.
- "function” parameter is used to specify function that execute when event triggered.
- “useCapture” is optional parameter which by default is false.
- “useCapture” is used for specify propagation mode.
- We have two type of propagation,
  - Bubbling
  - Capturing
- Bubbling is use when we want that child element handle event before parent element when event of child element triggered.
- Let say, we have one and inside that we have <p>, now we bind event listener on both element for same event, So when event of **<p>** triggered at that time <p> handle event first and after handle event second.
- Bubbling is use when we want that parent element handle event before child element when event of child element triggered.
- Let say, we have one and inside that we have <p>, now we bind event listener on both element for same event, So when event of **<p>** triggered at that time handle event first and after <p> handle event second.

7.7 VALIDATION

- It is a technique that used to ensure that user fill data correctly.
- In traditional way validation, we perform server-side validation.
- If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information.
- This was really a lengthy process which used to put a lot of burden on the server.
- JavaScript provides a way to validate form's data on the client's computer before sending it to the web server.
- We have two type of validation which we should do for archive better validation,
    - Basic Validation
        - Emptiness
        - Confirm Password
        - Length Validation…
    - Data Format Validation
        - Email Validation
        - Mobile Number Validation
        - Enrollment Number Validation…
- We are use regular expression for validate data.

7.7.1 REGEXP:

- A regular expression is an object that describes a pattern of characters.
- Regular expressions are used to perform pattern-matching and "search-and-replace" functions on text.

Syntax:

```
var pattern = "^[\\w]+$";   // will allow only words in the string
var regex = new RegExp(pattern);
if (regex.test(testString)){
    //Valid
} else {
    //Invalid
}
```

Regular Expression Syntax:

- To find word characters in the string we can use \w
- We can also use [a-zA-Z0-9_] for the same

- To find non-word characters in the string we can use \W
- to find digit characters in the string we can use \d
- We can also use [0-9] for the same
- To find non-digit characters in the string we can use \D
- We can use \n for new line and \t for tab

BASIC OF JQUERY

8.1 USE OF JQUERY

- jQuery is a lightweight, "write less, do more", JavaScript library.
- The purpose of jQuery is to make it much easier to use JavaScript on your website.
- jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
- The jQuery library contains the following features,
  - HTML/DOM manipulation
  - CSS manipulation
  - HTML event methods
  - Effects and animations
  - AJAX
  - Utilities
- In addition, jQuery has plugins for almost any task out there.

8.2 DIFFERENCE BETWEEN JQUERY AND JAVASCRIPT

Features	JavaScript	jQuery
Existence	JavaScript is an independent language and can exist on its own.	jQuery is a JavaScript library. It would not have been invented had JavaScript was not there. jQuery is still dependent on JavaScript as it has to be converted to JavaScript for the browser in-built JavaScript engine to interpret and run it.
Language	It is a high-level interpreted client-side scripting language. This is a combination of ECMA script and DOM (Document Object Model)	It is a light-weight JavaScript library. It has only the DOM
Coding	JavaScript uses more lines of code as we have to write our own code	jQuery uses fewer lines of code than JavaScript for the same functionality as the code is already written in its library, we just have to import the library and use the relevant function/method of the library in our code.

Usage	JavaScript code is written inside the script tag in a HTML page <script> </script>	We need to import the jQuery from CDN or from a location where the jQuery library is downloaded in order to use it. jQuery code is also written inside the script tag on the HTML page.
Animations	We can make animations in JavaScript with many lines of code. Animations are mainly done by manipulating the style of an Html page.	In jQuery, we can add animation effects easily with fewer lines of code.
User-friendliness	JavaScript can be cumbersome for the developer as it can take a number of lines of code to attain a functionality	jQuery is more user-friendly than JavaScript with few lines of code
Cross-browser compatibility	In JavaScript, we may have to deal with cross-browser compatibility by writing extra code or a workaround.	jQuery is cross-browser compatible. We don't need to worry about writing any workaround or extra code to make our code compatible to a browser.
Performance	Pure JavaScript can be faster for DOM selection/manipulation than jQuery as JavaScript is directly processed by the browser.	jQuery has to be converted into JavaScript to make it run in a browser.
Event handling, Interactivity, and Ajax calls	All these can be done in JavaScript but we may have to write many lines of code.	All these can be done easily with jQuery with fewer lines of code. It is easier in jQuery to add interactivity, animations and also make ajax calls as the functions are already pre-defined in the library. We just use those functions in our code at the required places.
Verbosity	JavaScript is verbose as one has to write many lines of code for a functionality	jQuery is concise and uses fewer lines of code, sometimes only one line of code.
Size and Weight	Being a language, it is heavier than jQuery	Being a library, it is lightweight. It has a minified version of its code which makes it light-weight.
Reusability and Maintainability	JavaScript code can be verbose and therefore can be difficult to maintain and reuse.	With fewer lines of code, jQuery is more maintainable as we just have to call the predefined functions in the jQuery library in our code. This

		also makes us to easily reuse jQuery functions at different places in the code.

8.3 HTML/CSS METHOD OF JQUERY

- jQuery Contains many methods for HTML/CSS DOM manipulation.

Method	Description
addClass()	Adds one or more class names to selected elements
after()	Inserts content after selected elements
append()	Inserts content at the end of selected elements
appendTo()	Inserts HTML elements at the end of selected elements
attr()	Sets or returns attributes/values of selected elements
before()	Inserts content before selected elements
clone()	Makes a copy of selected elements
css()	Sets or returns one or more style properties for selected elements
detach()	Removes selected elements (keeps data and events)
empty()	Removes all child nodes and content from selected elements
hasClass()	Checks if any of the selected elements have a specified class name
height()	Sets or returns the height of selected elements
html()	Sets or returns the content of selected elements
innerHeight()	Returns the height of an element (includes padding, but not border)
innerWidth()	Returns the width of an element (includes padding, but not border)
insertAfter()	Inserts HTML elements after selected elements
insertBefore()	Inserts HTML elements before selected elements
offset()	Sets or returns the offset coordinates for selected elements (relative to the document)
offsetParent()	Returns the first positioned parent element
outerHeight()	Returns the height of an element (includes padding and border) If We pass true as a parameter in then it also include margin.
outerWidth()	Returns the width of an element (includes padding and border) If We pass true as a parameter in then it also include margin.
position()	Returns the position (relative to the parent element) of an element
prepend()	Inserts content at the beginning of selected elements
prependTo()	Inserts HTML elements at the beginning of selected elements
prop()	Sets or returns properties/values of selected elements
remove()	Removes the selected elements (including data and events)
removeAttr()	Removes one or more attributes from selected elements
removeClass()	Removes one or more classes from selected elements

removeProp()	Removes a property set by the prop() method
replaceAll()	Replaces selected elements with new HTML elements
replaceWith()	Replaces selected elements with new content
scrollLeft()	Sets or returns the horizontal scrollbar position of selected elements
scrollTop()	Sets or returns the vertical scrollbar position of selected elements
text()	Sets or returns the text content of selected elements
toggleClass()	Toggles between adding/removing one or more classes from selected elements
unwrap()	Removes the parent element of the selected elements
val()	Sets or returns the value attribute of the selected elements (for form elements)
width()	Sets or returns the width of selected elements
wrap()	Wraps HTML element(s) around each selected element
wrapAll()	Wraps HTML element(s) around all selected elements
wrapInner()	Wraps HTML element(s) around the content of each selected element

8.4 jQuery Selectors

- jQuery Selectors are used to select and manipulate HTML elements.
- They are very important part of jQuery library.
- With jQuery selectors, you can find or select HTML elements based on their id, classes, attributes, types and much more from a DOM.
- Selectors are used to select one or more HTML elements using jQuery and once the element is selected then you can perform various operation on that.
- All jQuery selectors start with a dollor sign and parenthesis e.g. $(). It is known as the factory function.

Selector	Example	Description
*****	$("*")	It is used to select all elements.
#id	$("#firstname")	It will select the element with id="firstname"
.class	$(".primary")	It will select all elements with class="primary"
class,.class	$(".primary,.secondary")	It will select all elements with the class "primary" or "secondary"
element	$("p")	It will select all p elements.
el1,el2,el3	$("h1,div,p")	It will select all h1, div, and p elements.
:first	$("p:first")	This will select the first p element
:last	$("p:last")	This will select he last p element

:even	$("tr:even")	This will select all even tr elements
:odd	$("tr:odd")	This will select all odd tr elements
:first-child	$("p:first-child")	It will select all p elements that are the first child of their parent
:first-of-type	$("p:first-of-type")	It will select all p elements that are the first p element of their parent
:last-child	$("p:last-child")	It will select all p elements that are the last child of their parent
:last-of-type	$("p:last-of-type")	It will select all p elements that are the last p element of their parent
:nth-child(n)	$("p:nth-child(2)")	This will select all p elements that are the 2nd child of their parent
:nth-last-child(n)	$("p:nth-last-child(2)")	This will select all p elements that are the 2nd child of their parent, counting from the last child
:nth-of-type(n)	$("p:nth-of-type(2)")	It will select all p elements that are the 2nd p element of their parent
:nth-last-of-type(n)	$("p:nth-last-of-type(2)")	This will select all p elements that are the 2nd p element of their parent, counting from the last child
:only-child	$("p:only-child")	It will select all p elements that are the only child of their parent
:only-of-type	$("p:only-of-type")	It will select all p elements that are the only child, of its type, of their parent
parent > child	$("div > p")	It will select all p elements that are a direct child of a div element
parent descendant	$("div p")	It will select all p elements that are descendants of a div element
element + next	$("div + p")	It selects the p element that are next to each div elements
element ~ siblings	$("div ~ p")	It selects all p elements that are siblings of a div element
:eq(index)	$("ul li:eq(3)")	It will select the fourth element in a list (index starts at 0)
:gt(no)	$("ul li:gt(3)")	Select the list elements with an index greater than 3
:lt(no)	$("ul li:lt(3)")	Select the list elements with an index less than 3
:not(selector)	$("input:not(:empty)")	Select all input elements that are not empty
:header	$(":header")	Select all header elements h1, h2 ...

:animated	$(":animated")	Select all animated elements
:focus	$(":focus")	Select the element that currently has focus
:contains(text)	$(":contains('Hello')")	Select all elements which contains the text "Hello"
:has(selector)	$("div:has(p)")	Select all div elements that have a p element
:empty	$(":empty")	Select all elements that are empty
:parent	$(":parent")	Select all elements that are a parent of another element
:hidden	$("p:hidden")	Select all hidden p elements
:visible	$("table:visible")	Select all visible tables
:root	$(":root")	It will select the document's root element
:lang(language)	$("p:lang(de)")	Select all p elements with a lang attribute value starting with "de"
[attribute]	$("[href]")	Select all elements with a href attribute
[attribute=value]	$("[href='default.htm']")	Select all elements with a href attribute value equal to "default.htm"
[attribute!=value]	$("[href!='default.htm']")	It will select all elements with a href attribute value not equal to "default.htm"
[attribute$=value]	$("[href$='.jpg']")	It will select all elements with a href attribute value ending with ".jpg"
[attribute|=value]	$("[title|='Tomorrow']")	Select all elements with a title attribute value equal to 'Tomorrow', or starting with 'Tomorrow' followed by a hyphen
[attribute^=value]	$("[title^='Tom']")	Select all elements with a title attribute value starting with "Tom"
[attribute~=value]	$("[title~='hello']")	Select all elements with a title attribute value containing the specific word "hello"
[attribute*=value]	$("[title*='hello']")	Select all elements with a title attribute value containing the word "hello"
:input	$(":input")	It will select all input elements
:text	$(":text")	It will select all input elements with type="text"
:password	$(":password")	It will select all input elements with type="password"
:radio	$(":radio")	It will select all input elements with type="radio"
:checkbox	$(":checkbox")	It will select all input elements with type="checkbox"
:submit	$(":submit")	It will select all input elements with type="submit"

:reset	$(":reset")	It will select all input elements with type="reset"
:button	$(":button")	It will select all input elements with type="button"
:image	$(":image")	It will select all input elements with type="image"
:file	$(":file")	It will select all input elements with type="file"
:enabled	$(":enabled")	Select all enabled input elements
:disabled	$(":disabled")	It will select all disabled input elements
:selected	$(":selected")	It will select all selected input elements
:checked	$(":checked")	It will select all checked input elements

8.5 jQuery Events

- jQuery have set of events which is use to react on movement of DOM.

8.5.1 Event Properties:

Property	Description
event.currentTarget	The current DOM element within the event bubbling phase
event.data	Contains the optional data passed to an event method when the current executing handler is bound
event.delegateTarget	Returns the element where the currently-called jQuery event handler was attached
event.target	Returns which DOM element triggered the event
event.timeStamp	Returns the number of milliseconds since January 1, 1970, when the event is triggered
event.type	Returns which event type was triggered
event.which	Returns which keyboard key or mouse button was pressed for the event
event.pageX	Returns the mouse position relative to the left edge of the document
event.pageY	Returns the mouse position relative to the top edge of the document
event.namespace	Returns the namespace specified when the event was triggered
event.result	Contains the last/previous value returned by an event handler triggered by the specified event
event.relatedTarget	Returns which element being entered or exited on mouse movement

event.preventDefault()	Prevents the default action of the event
event.isDefaultPrevented()	Returns whether event.preventDefault() was called for the event object
event.stopPropagation()	Prevents the event from bubbling up the DOM tree, preventing any parent handlers from being notified of the event
event.isPropagationStopped()	Returns whether event.stopPropagation() was called for the event object
event.stopImmediatePropagation()	Prevents other event handlers from being called
event.isImmediatePropagationStopped()	Returns whether event.stopImmediatePropagation() was called for the event object

8.5.2 Basic Events:

Method	Description
click()	Attaches/Triggers the click event
dblclick()	Attaches/Triggers the double click event
mousedown()	Attaches/Triggers the mousedown event
mouseenter()	Attaches/Triggers the mouseenter event
mouseleave()	Attaches/Triggers the mouseleave event
mousemove()	Attaches/Triggers the mousemove event
mouseout()	Attaches/Triggers the mouseout event
mouseover()	Attaches/Triggers the mouseover event
mouseup()	Attaches/Triggers the mouseup event
hover()	Attaches two event handlers to the hover event
keydown()	Attaches/Triggers the keydown event
keypress()	Attaches/Triggers the keypress event
keyup()	Attaches/Triggers the keyup event
blur()	Attaches/Triggers the blur event
change()	Attaches/Triggers the change event
focus()	Attaches/Triggers the focus event
focusin()	Attaches an event handler to the focusin event
focusout()	Attaches an event handler to the focusout event
off()	Removes event handlers attached with the on() method
on()	Attaches event handlers to elements
one()	Adds one or more event handlers to selected elements. This handler can only be triggered once per element
$.proxy()	Takes an existing function and returns a new one with a particular context

ready()	Specifies a function to execute when the DOM is fully loaded
resize()	Attaches/Triggers the resize event
scroll()	Attaches/Triggers the scroll event
select()	Attaches/Triggers the select event
submit()	Attaches/Triggers the submit event
trigger()	Triggers all events bound to the selected elements
triggerHandler()	Triggers all functions bound to a specified event for the selected elements

8.5.3 Fire event programmatically:

- We can trigger event using two different way.

```
$(document).ready(function(){
    $("#btn").click(function(){
        alert("clicked");
    });

    // Method 1
    $("#btn").click();

    // Method 2
    $("#btn").trigger("click");
})
```

8.5.4 Custom Event:

- We can create custom event using “on” and “toggle event”.

```
$(document).ready(function(){
    $("#btn").click(function(){
        let x = $("#num1").val();
        let y = $("#num2").val();
        sum(x,y);
    });

    function sum(x,y){
        ans = parseFloat(x) + parseFloat(y);
        $(document).trigger("display");
    }

    $(document).on("display",function(){
        alert(ans);
    })
```

```
});
```

8.6 jQuery Validation

- We have three way to done validation using jQuery.
  - Basic Validation
  - Using jQuery Validator
  - Using Regular Expression

8.6.1 Basic Validator:

- In this method we just bind event with element and check form is correctly fill by the user or not.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        div{
            margin: 1em;
        }
    </style>
    <script src="../../assets/lib/jQuery/jquery-3.6.3.min.js"></script>
    <script>
        $(document).ready(function(){
            $("#submit").click(function(){
                if($("#name").val() == ""){
                    $("#name-validation").text("Please eneter name");
                }
                else if($("#name").val().length < 2){
                    $("#name-validation").text("Name must be contain minimum
two character");
                }
                else{
                    alert("form submited successfully");
                    $("#name-validation").text("");
```

```
                    $("#form").trigger("reset");
                }
            });
        });
    </script>
</head>
<body>
    <div>
        <form id="form">
            <div>
                <label for="name">Name : </label>
                <input type="text" name="name" id="name">
                <span id="name-validation"></span>
            </div>
            <div>
                <input type="button" id="submit" value="submit">
            </div>
        </form>
    </div>
</body>
</html>
```

8.6.2 Using jQuery Validator:

- jQuery provides plugin for validate forms.
- We are just include that script and use for validation of form.
- We have two types of script for validation,
    - Core validation using basic jQuery plugin
    - Additional validation using additional jQuery plugin

Attribute	Value	Description
required	true/false	Set required field validation
minlength	number	Set minlength validation
maxlength	number	Set maxlength validation
lettersonly	true/false	Accept letter only
email	true/false	Accept email only
nowhitespace	true/false	Accept string without whitespaces
alphanumeric	true/false	Accept alphanumeric only
letterswithbasicpunc	true/false	Accept string which contains only letter with basic punctuation
step	number	It accept number on specific staps
number	true/false	Accept number only include point numbers

digits	true/false	Accept digit only
date	true/false	Accept date only in “yyyy-mm-dd” or “yyyy/mm/dd”
equalTo	Id of element which contains comparison value	Compare two elements value
greaterThanEqual	Id of element which contains comparison value	Compare two elements value
lessThanEqual	Id of element which contains comparison value	Compare two elements value
greaterThan	Id of element which contains comparison value	Compare two elements value
lessThan	Id of element which contains comparison value	Compare two elements value
range	Take range of number like, [10,20]	Accept number between range
min	number	Set minimum number
max	number	Set maximum number

- We can change validation error colour by inheriting error class.
- We can also change message of validation error using “message” key of validation method.
- We can also add custom validation using “addMthod()” of validator.

Example:

```
<style>
    .error{
        color: red;
    }
</style>
<script>
    (document).ready(function(){

        // Create custom method for custome validation
        $.validator.addMethod("StrongPassword", function(value){
            return /(?=^.{8,}$)((?=.*\d)|(?=.*\W+))(?![.\n])(?=.*[A-
Z])(?=.*[a-z]).*$/.test(value);
            // wrap regular expression inside "/.../".
```

```
        }, "Please enter strong password");

        $("#form").validate({
            rules:{
                name:{
                    required: true,
                    minlength: 3,
                    maxlength: 50,
                    lettersonly: true
                },
            },
            messages:{
                name:{
                    required: "Please enter name",
                    lettersonly: "Name can only contain letter"
                }
            },
            highlight:function(element){
                $(element).removeClass("is-valid");
                $(element).addClass("is-invalid");
            },
            unhighlight:function(element){
                $(element).removeClass("is-invalid");
                $(element).addClass("is-valid");
            },
            invalidHandler:function(element){
                var validator = $("#form").validate();
                $("#summary").text("number of invalid : " +
validator.numberOfInvalids());
            }
        })
    });
</script>
```

REGULAR EXPRESSION IN JQUERY

- We have various type of Regular expression.

9.1 NUMBER VALIDATION

```
//select integers only
var intRegex = /[0-9 -()+]+$/;
//match any ip address
var ipRegex = 'bd{1,3}.d{1,3}.d{1,3}.d{1,3}b';
//match number in range 0-255
var num0to255Regex = '^([01][0-9][0-9]|2[0-4][0-9]|25[0-5])$';
//match number in range 0-999
var num0to999Regex = '^([0-9]|[1-9][0-9]|[1-9][0-9][0-9])$';
//match ints and floats/decimals
var floatRegex = '[-+]?([0-9]*.[0-9]+|[0-9]+)';
//Match Any number from 1 to 50 inclusive
var number1to50Regex = /(^[1-9]{1}$|^[1-4]{1}[0-9]{1}$|^50$)/gm;
```

9.2 GENERAL VALIDATION

```
//match email address
var emailRegex = '^[A-Z0-9._%+-]+@[A-Z0-9.-]+.[A-Z]{2,4}$';
//match credit card numbers
var creditCardRegex = '^(?:4[0-9]{12}(?:[0-9]{3})?|5[1-5][0-
9]{14}|6(?:011|5[0-9][0-9])[0-9]{12}|3[47][0-9]{13}|3(?:0[0-5]|[68][0-9])[0-
9]{11}|(?:2131|1800|35d{3})d{11})$';
//match username
var usernameRegex = '/^[a-z0-9_-]{3,16}$/';
//match password
var passwordRegex = '/^[a-z0-9_-]{6,18}$/';
//Match 8 to 15 character string with at least one upper case letter, one
lower case letter, and one digit (useful for passwords).
var passwordStrengthRegex = /((?=.*d)(?=.*[a-z])(?=.*[A-Z]).{8,15})/gm;
//match elements that could contain a phone number
var phoneNumber = /[0-9-()+]{3,20}/;
```

9.3 DATE VALIDATION

```
//MatchDate (e.g. 21/3/2006)
var dateRegex = /(d{1,2}/d{1,2}/d{4})/gm;
//match date in format MM/DD/YYYY
```

```
var dateMMDDYYYRegex = '^(0[1-9]|1[012])[- /.](0[1-9]|[12][0-9]|3[01])[-
/.](19|20)dd$';
//match date in format DD/MM/YYYY
var dateDDMMYYYRegex = '^(0[1-9]|[12][0-9]|3[01])[- /.](0[1-9]|1[012])[-
/.](19|20)dd$';

//match a url
var urlRegex = '/^(https?://)?([da-z.-]+).([a-z.]{2,6})([/w .-]*)*/?$/';
//match a url slug (letters/numbers/hypens)
var urlslugRegex = '/^[a-z0-9-]+$/';
//match a url string (Fixes spaces and querystrings)
var urlRegex = '/(https?://)?([da-z.-]+).([a-z.]{2,6})([/w.-=?]*)*/?/';
```

9.4 URL Validation

```
//match a url
var urlRegex = '/^(https?://)?([da-z.-]+).([a-z.]{2,6})([/w .-]*)*/?$/';
//match a url slug (letters/numbers/hypens)
var urlslugRegex = '/^[a-z0-9-]+$/';
//match a url string (Fixes spaces and querystrings)
var urlRegex = '/(https?://)?([da-z.-]+).([a-z.]{2,6})([/w.-=?]*)*/?/';
```

9.5 Vowel Validation

```
//select vowels only
var vowelRegex = /^[aeiou]/;
```

9.6 Whitespace Validation

```
//select whitespace
var whiteSpaceRegex = '^[ t]+';
//select whitespace and tabs
var whiteSpaceRegex = '^[ t]+|[ t]+$';
//select whitespace and linebreaks
var whiteSpaceRegex = '[ trn]';
//replace newline characters with
 tags
newLineToBr = function(str) { return str.replace(/(rn|[rn])/g, '
'); }
```

9.7 Domain Name Validation

```
//match domain name (with HTTP)
```

```
var domainRegex = '/(.*?)[^w{3}.]([a-zA-Z0-9]([a-zA-Z0-9-]{0,65}[a-zA-Z0-
9])?.)+[a-zA-Z]{2,6}/igm';
//match domain name (www. only)
var domainRegex = '/[^w{3}.]([a-zA-Z0-9]([a-zA-Z0-9-]{0,65}[a-zA-Z0-
9])?.)+[a-zA-Z]{2,6}/igm';
//match domain name (alternative)
var domainRegex =
'/(.*?).(com|net|org|info|coop|int|com.au|co.uk|org.uk|ac.uk|)/igm';
//match sub domains: www, dev, int, stage, int.travel, stage.travel
var subDomainRegex =
'/(http://|https://)?(www.|dev.)?(int.|stage.)?(travel.)?(.*)+?/igm';
```

9.8 Image

```
//Match jpg, gif or png image
var imageRegex = /([^s]+(?=.(jpg|gif|png)).2)/gm;
//match all images
var imgTagsRegex = /
/ig;
//match just .png images
```

9.9 Other Regular Expression

```
//match RGB (color) string
var rgbRegex = /^rgb((d+),s*(d+),s*(d+))$/;
//match hex (color) string
var hexRegex = '/^#?([a-f0-9]{6}|[a-f0-9]{3})$/';
//Match Valid hexadecimal colour code
var hexRegex = /(#?([A-Fa-f0-9]){3}(([A-Fa-f0-9]){3})?)/gm;
//match a HTML tag (v1)
var htmlTagRegex = '/^(.*)|s+/>)$/';
//match HTML Tags (v2)
var htmlTagRegex = /(]+)>)/gm;
//match /product/123456789
var productUrlRegex = '(/product/)?+[0-9]+';
//Match Letters, numbers and hyphens
var lnhRegex = /([A-Za-z0-9-]+)/gm;
//match all .js includes
var jsTagsRegex = /
```

CALLBACK FUNCTIONS

10.1 INTRODUCTION

- It is a technique which is used to call next function after 100% execute current function.
- Using this we should wait for current function result and after we execute next function.

Example:

```
$(document).ready(function(){
    // without callback
    $("#btn1").click(function(){
        $("#content1").hide("slow");
        alert("Content 1 hide");
    });

    // with callback
    $("#btn2").click(function(){
        $("#content2").hide("slow", function(){
            alert("Content 2 hide");
        });
    });
});
```

DEFERRED & PROMISE OBJECT

11.1 PROMISE

- The creation of a deferred object is 'automatic' but deferred objects can also be created manually by doing something like,

```
var myPromise = new jQuery.Deferred();
```

- The manually created deferred object should be controlled manually.
- Promises object have tree state,
  - Pending: The call hasn't been made, or it has and we're waiting for a response
  - Resolved: The call has been made and it was successful
  - Rejected: The call has been made and it was successful
- A deferred object's state can be manually changed by calling resolve() and reject() on it.

11.2 HANDLING STATES(ATTACHING CALLBACK)

- We can add handler with the object of deferred by using attaching carious methods.
- This attached method have different states like,
  - Success
  - Failure
  - In progress

Example:

```
var mySearch = jQuery.ajax('/api/items');
mySearch.then(
    function success (response) {
        console.log('Search succeeded!', response)
    },
    function failure (response) {
        console.log('Search failed!', response)
    }
);
```

- We can also split handler like,

```
mySearch.done(function success (response) {
    console.log('Search succeeded!', response)
});
```

```
mySearch.fail(function fail (response) {
    console.log('Search failed!', response)
});
```

11.3 Chaining

- It is a technique to create a chain of the method instant or call method one by one.

Example:

```
function search (keyword, api) {
    var search = jQuery.ajax('/api/' + api + '?q=' + keyword)

    // This will be called if the call succeeds
    .done(function done (response) {
        console.log('Success!')
    })

    // This will be called if the call fails
    .fail(function fail (response)
    {
        console.log('Failure!')
    })

    // And this will always be called when we get a response, whether the
call succeeds or fails
    .always(function always (response)
    {
        console.log('Call finished');
    });

    return search
};
```

11.4 Multiple Promises

- We can use multiple promises at a one time by using “when()” method.

Example:

```
/ Define the search function
function search (keyword) {
```

```
    var search = jQuery.ajax('/api/items?q=' + keyword);

    search.done(function done () {
        console.log('Search for ' + keyword + ' complete!')
    });

    return search
}

// Create an artificial delay (eg a slow API response)
function sleep (time) {
    var promise = jQuery.Deferred();

    setTimeout(promise.resolve, 3000);
    promise.done(function done () {
        console.log('Artificial delay complete!')
    });

    return promise
}

// Example calls
jQuery.when(search('hat'), search('jacket'), sleep(3000))

.done(function (s1, s2) {
    console.log('Search 1:', s1[0]);
    console.log('Search 2:', s2[0]);
});
```

Output:

```
Search for jacket complete!
Search for hat complete!
Artificial delay complete!
Search 1: {total: 15, items: Array(15), corrections: Array(0), locale: {…},
backend: "E", …}
Search 2: {total: 37, items: Array(37), corrections: Array(0), locale: {…},
backend: "E", …}
```

AJAX

12.1 WHAT IS AJAX?

- Asynchronous Javascript and XML.
- Ajax load a data in background and show on webpage without reload.
- Using Ajax we should get data like text, XML and JSON using http request.

12.2 AJAX

- We have different function and methods in ajax to manipulate data.

Methods:

Sr.No.	Methods & Description
1	**jQuery.ajax(options)** Load a remote page using an HTTP request.
2	**jQuery.ajaxSetup(options)** Setup global settings for AJAX requests.
3	**jQuery.get(url, [data], [callback], [type])** Load a remote page using an HTTP GET request.
4	**jQuery.getJSON(url, [data], [callback])** Load JSON data using an HTTP GET request.
5	**jQuery.getScript(url, [callback])** Loads and executes a JavaScript file using an HTTP GET request.
6	**jQuery.post(url, [data], [callback], [type])** Load a remote page using an HTTP POST request.
7	**load(url, [data], [callback])** Load HTML from a remote file and inject it into the DOM.
8	**serialize( )** Serializes a set of input elements into a string of data.
9	**serializeArray( )** Serializes all forms and form elements like the .serialize() method but returns a JSON data structure for you to work with.

Events:

Sr.No.	Methods & Description
1	**ajaxComplete(callback)** Attach a function to be executed whenever an AJAX request completes.
2	**ajaxStart(callback)**

	Attach a function to be executed whenever an AJAX request begins and there is none already active.
3	**ajaxError(callback)** Attach a function to be executed whenever an AJAX request fails.
4	**ajaxSend(callback)** Attach a function to be executed before an AJAX request is sent.
5	**ajaxStop(callback)** Attach a function to be executed whenever all AJAX requests have ended.
6	**ajaxSuccess(callback)** Attach a function to be executed whenever an AJAX request completes successfully.

12.3 UNDERSTANDING HTTP VERBS

- We have mainly use four types of verb which mostly use like,
  - Get
  - Post
  - Put
  - Delete

12.3.1 GET:

- This method is used to retrieve a representation of a resource.
- A GET request is considered safe because as HTTP specifies, these requests are used only to read data and not change it.
- So in short there are no side effects and GET requests can be re-issued without worrying about the consequences.
- The disadvantage of GET requests is that they can only supply data in the form of parameters encoded in the URI or as cookies in the cookie request header.

Example:

- Displaying your registered account details on any website has no effect on the account.
- If you are using Internet Explorer you can refresh a page and the page will show information that resulted from a GET, without displaying any kind of warning.
- We have also other HTTP components like PROXIES that automatically retry GET requests if they encounter a temporary network connection problem.

Tasks:

- You can cache the requests.
- It can remain in the browser history.
- You can bookmark the requests.

- You can apply length restrictions with GET requests.
- It is used only to retrieve data.

12.3.2 POST:

- POST is used when the processing you wish to do on the server should be repeated or when creating a new resource or for a POST to the parent when the service associates the new resource with the parent or for assigning an ID, etcetera.
- It is used to create new resources.
- For some resources, it may be used to alter the internal state.
- For others, its behavior may be that of a remote procedure call.
- If you try to refresh a page while using a POST request, then you will get an error like page can't be refreshed.
- Why? Because the request cannot be repeated without explicit approval by the user.

Example:

- The best example of a POST request I must say is when the user submits a change and then uses a 302 redirection (Code redirection details you will find later on this article) to change to a GET that displays the result of the action as the new updated value.

Tasks:

- Data will be re-submitted.
- It cannot be bookmarked.
- It cannot be cached.
- Parameters are not saved in browser history.
- No restrictions. Binary data is also allowed.
- Data is not displayed in the URL.

12.3.3 PUT:

- PUT is used to create a resource, not in a general case but when the resource ID is chosen by the client instead of by the server, then only PUT is used.
- Simply PUT updates data in the repository, replacing any existing data with the supplied data.

Example:

- Suppose we wanted to change the image that we have something already on the server, So, we just upload a new one using the PUT verb.

Tasks:

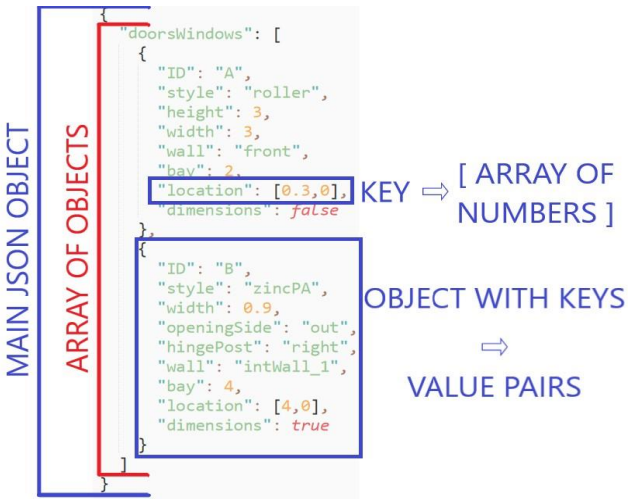
- It completes as possible.
- It's as seamless as possible.
- Easy to use via HTML.
- It should integrate well with servers that already support PUT.

12.3.4 DELETE:

- A DELETE request is as simple as its name implies it just deletes, it is used to delete a resource identified by a URI and on successful deletion, it returns HTTP status 200 (OK) along with a response body.
- The important and unique thing about a DELETE request is that the client cannot be guaranteed that the operation has been carried out, even if the status code returned from the origin server indicates that the action has been completed successfully.
- Some HTTP infrastructures do not support the DELETE method. In such cases, the request should be submitted as a POST with an additional X-HTTP-Method-Override header set to DELETE.

12.4 UNDERSTANDING JSON STRUCTURE

- JSON stands for JavaScript Object Notation.
- JSON is a text format for storing and transporting data.
- JSON is "self-describing" and easy to understand.
- JSON is a lightweight data-interchange format.
- JSON is plain text written in JavaScript object notation.
- JSON is used to send data between computers.
- JSON is language independent *.



12.5 Serialization & De-serialization

- We are used ajax method for doing serialization and deserialization of data.
- We have some method,
  - jQuery.parseJSON
  - jQuery.parseXML
  - jQuery.parseHTML

12.6 Functions

12.6.1 Map:

- The jQuery map() function translates all items in an object or in array to a new array of items.
- It applies a function to each item of the object or array and maps the results into a new array.

Syntax:

```
jQuery.map( array/object, callback );
// return new array/object
```

Example:

```
$(document).ready(function(){
    var arr = [2, 4, 5, 20, 21];
    var newArr = $.map(arr, function(val, index){
        return val*val;
    });
    console.log(newArr);
});
```

12.6.2 Grep:

- The grep() method in jQuery finds the array elements that satisfy the given filter function.

Syntax:

```
jQuery.grep(array, function(element, index), invert)
// return new array
```

Example:

```
$(document).ready(function(){
    var arr = [2, 4, 5, 20, 21];
```

```
    var newArr1 = $.grep(arr, function(val, index){
        return (val%2 == 0); // condition
    });
    console.log(newArr1);
    var newArr2 = $.grep(arr, function(val, index){
        return (val%2 == 0);
    }, true);
    // by default invert is false, if we specify true the it return opposite
value of condition.
    console.log(newArr2);
});
```

SESSION

13.1 INTRODUCTION

- The session way let you store key/value pairs in the browser.
- We have two types of session.
  - sessionStorage
  - localStorage
- We have various method in session.

Sr.No.	Methods & Description
1	**getItem(key)** For get value of particular key.
2	**setItem(key, value)** For set new key value pair or change value of existing key.
3	**removeItem(key)** For remove particular key value pair.
4	**clear( )** For clear whole session.

13.1.1 SESSIONSTORAGE:

- It store data for only one session.
- When browser is close, session should be cleared.

13.1.2 LOCALSTORAGE:

- It store data for more then one session.
- When browser is close, session shouldn't be cleared.

COOKIES

14.1 INTRODUCTION

- It is used to store information in web page.
- After server send response, server don't remember information.
- So we must resend all information when we send new request.
- But cookies solve this problem, when a user visits a web page, information can be stored in a cookie.
- Next time the user visits the page, the cookie "remembers" information of user and it automatically send information with request.

14.2 HOW TO USE

- JavaScript can create, read, and delete cookies with the "document.cookie" **property**.
- Cookies stored data in form of string with is contains key value pairs and separate by semicolon.
- We have some predefine key in cookie for manage cookie.

Key	Value Format	Description
expires	date-in-GMTString-format	Set expire time of cookie
max-age	max-age-in-seconds	Set max age of cookie
partitioned		Indicates that the cookie should be stored using partitioned storage.
path		Indicates the path that must exist in the requested URL for the browser to send the Cookie header. If not specified, it defaults to the current path of the current document location.
samesite	lax, strict or none	SameSite prevents the browser from sending this cookie along with cross-site requests.
secure	true/false	Specifies that the cookie should only be transmitted over a secure protocol.
domain	true/false	Set domain which can access cookie, sub-domain can also access this cookie.

14.2.1 SAMESITE KEY:

- We have three value,

lax:

- The lax value will send the cookie for all same-site requests and top-level navigation GET requests.
- This is sufficient for user tracking, but it will prevent many Cross-Site Request Forgery (CSRF) attacks.
- This is the default value in modern browsers.

strict:

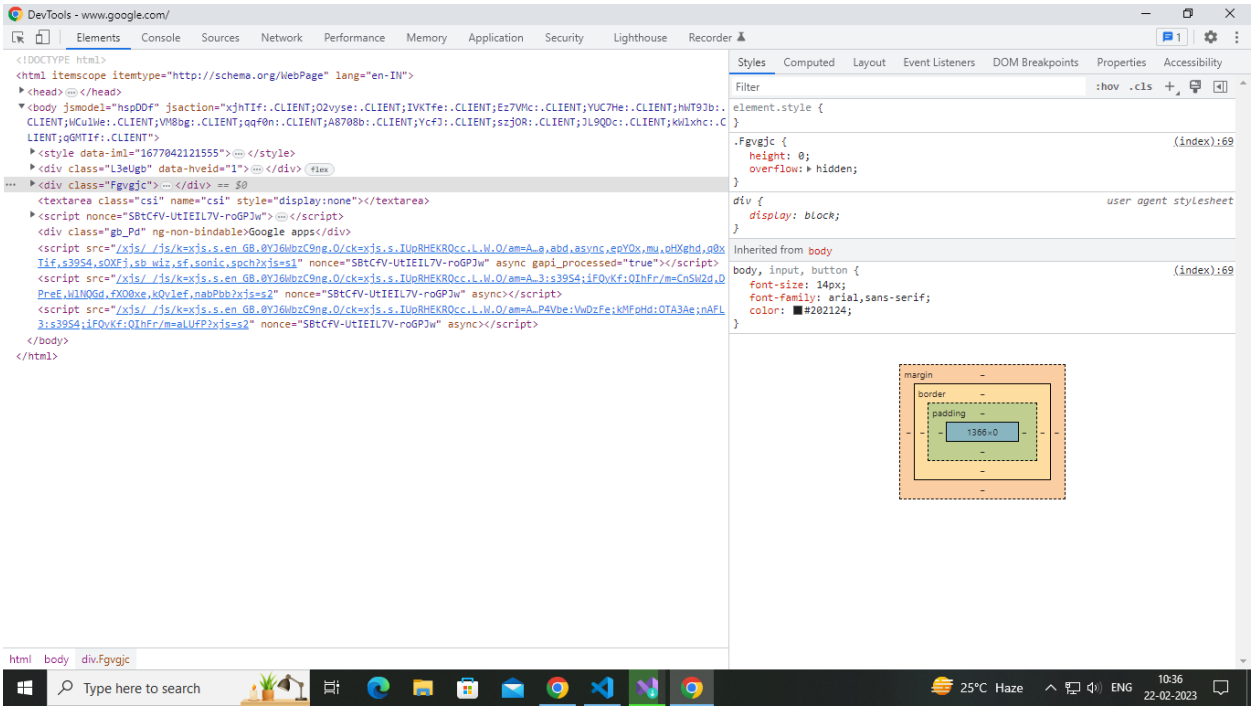
- The strict value will prevent the cookie from being sent by the browser to the target site in all cross-site browsing contexts, even when following a regular link.

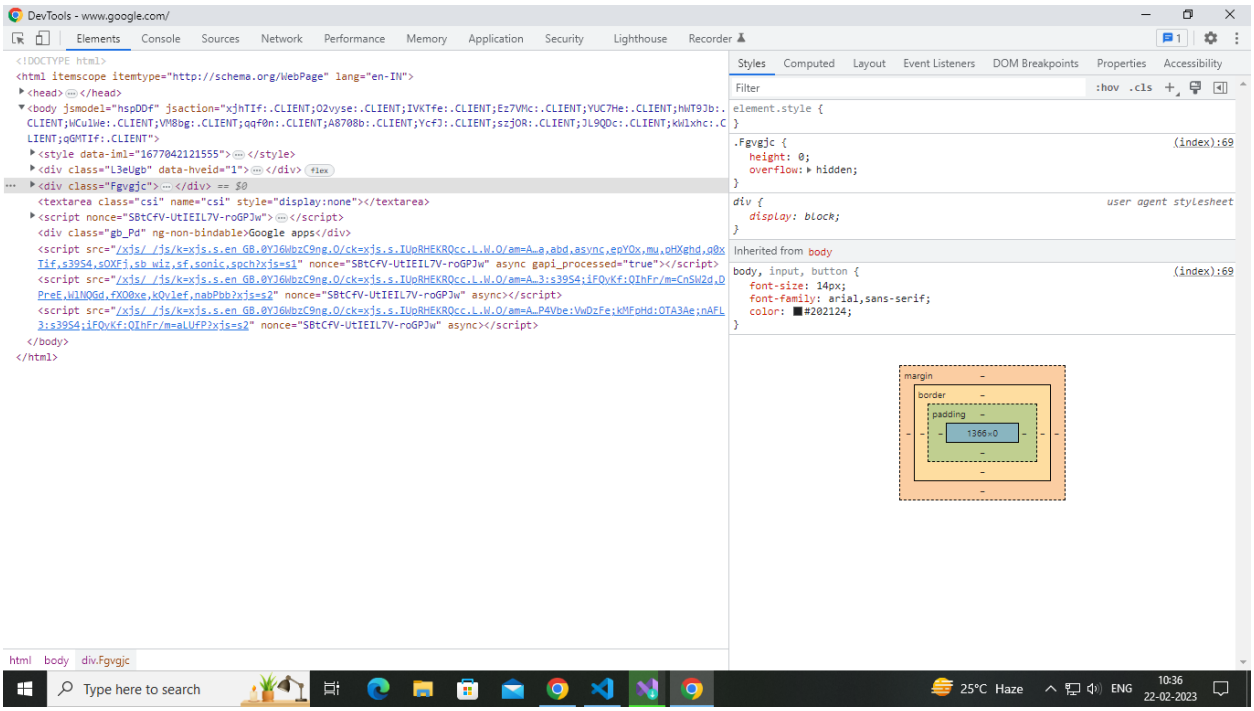
none:

- The none value explicitly states no restrictions will be applied.
- The cookie will be sent in all requests—both cross-site and same-site.

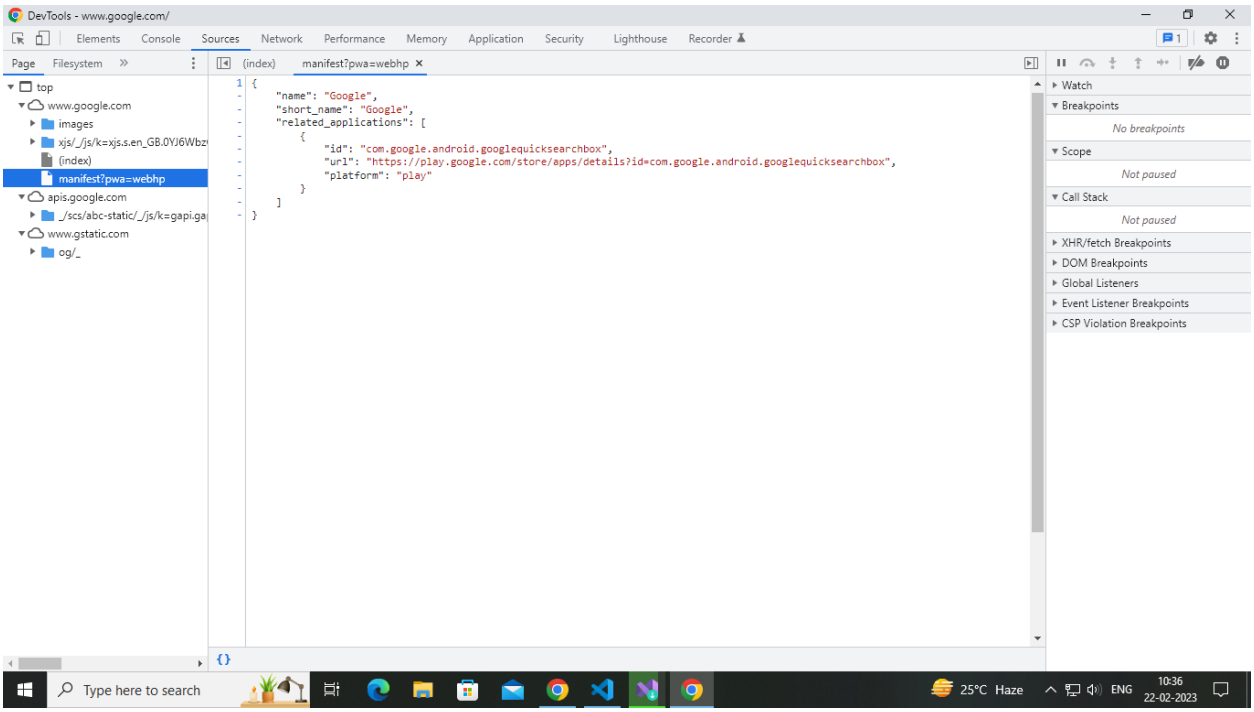
Note: Understanding of same-site and cross-site : https://web.dev/same-site-same-origin/

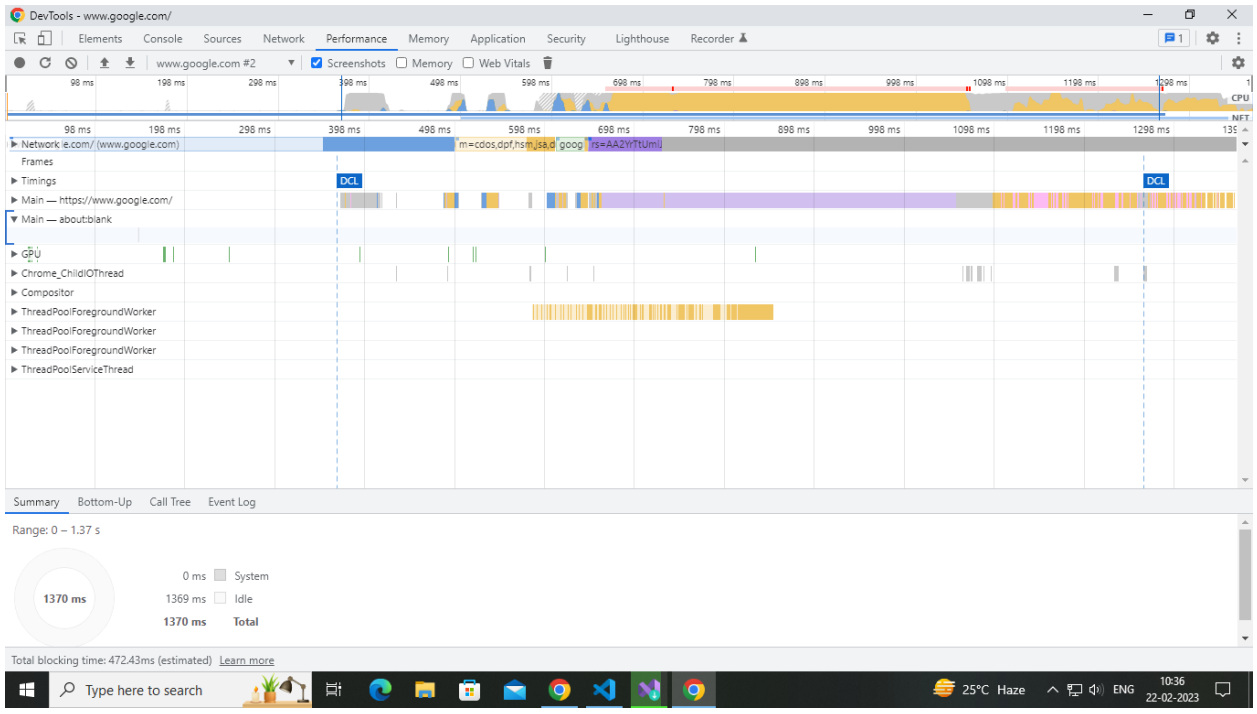
Google Dev Tool



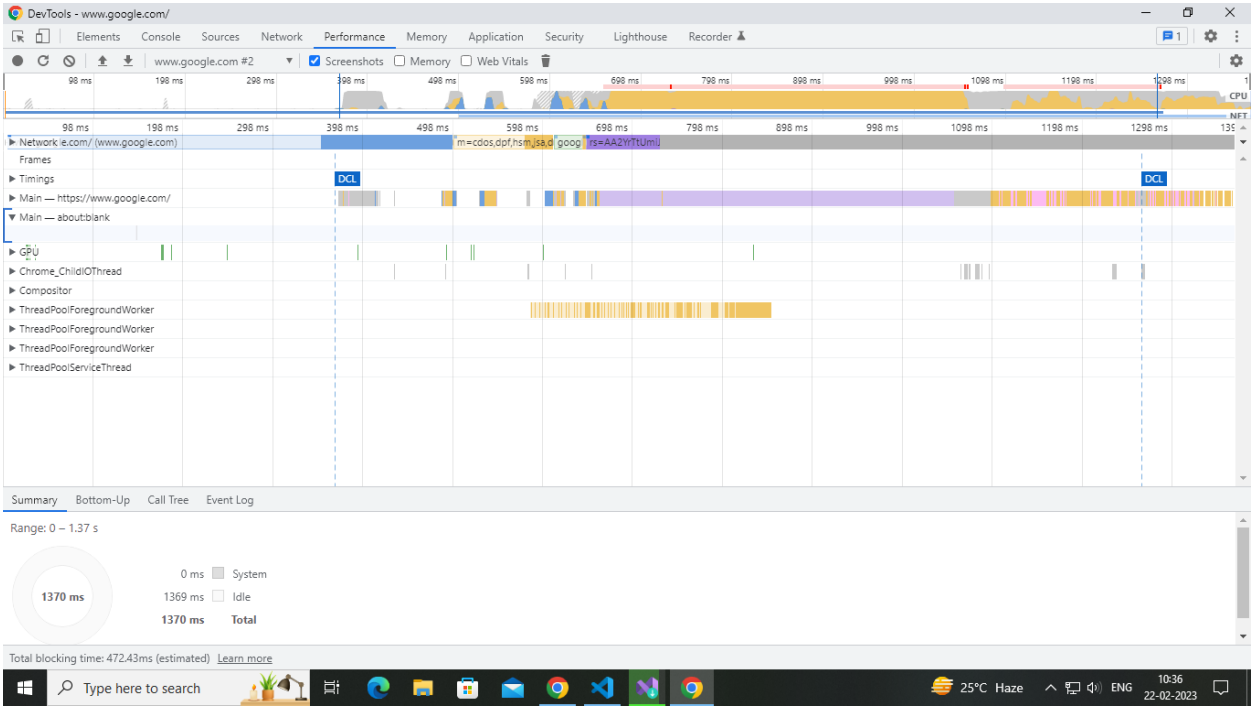


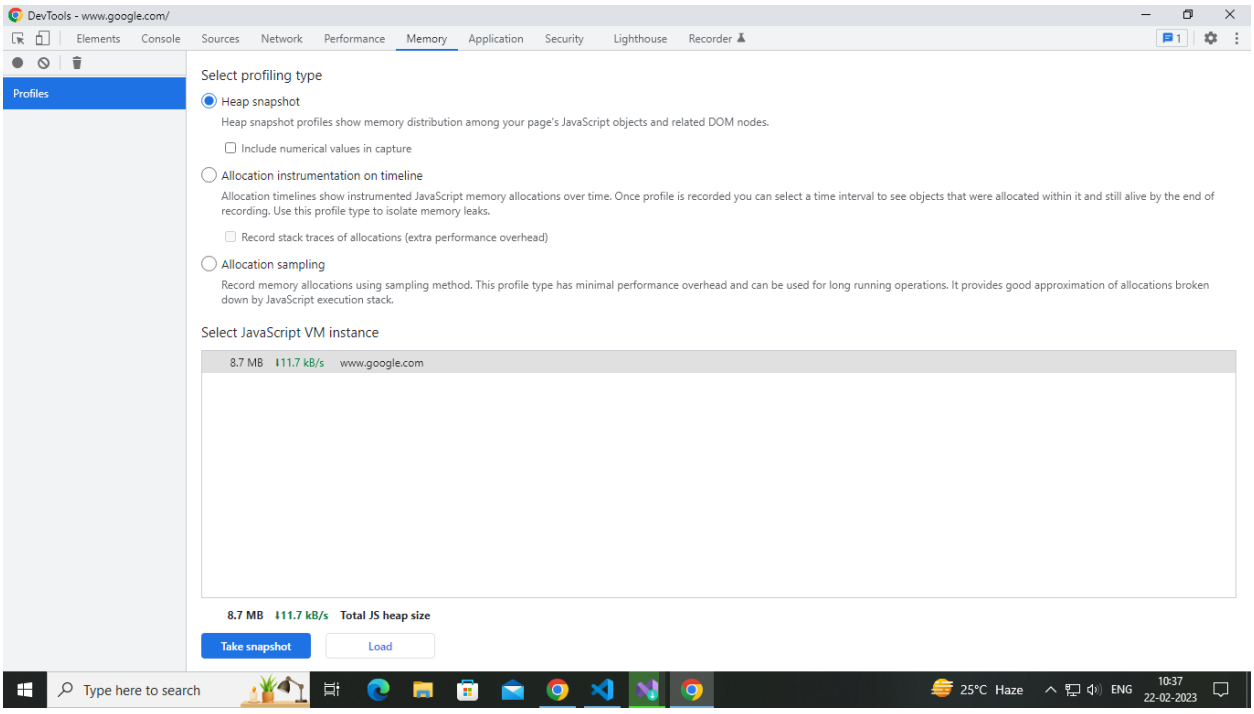
GUI Basic


DevTools - www.google.com/
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Page
Filesystem
(index)
manifest?pwa=webhp
top
www.google.com
images
(index)
manifest?pwa=webhp
apis.google.com
www.gstatic.com
og/_
{
"name": "Google",
"short_name": "Google",
"related_applications": [
{
"id": "com.google.android.googlequicksearchbox",
"url": "https://play.google.com/store/apps/details?id=com.google.android.googlequicksearchbox",
"platform": "play"
}
]
}
Watch
Breakpoints
No breakpoints
Scope
Not paused
Call Stack
Not paused
XHR/fetch Breakpoints
DOM Breakpoints
Global Listeners
Event Listener Breakpoints
CSP Violation Breakpoints
Type here to search
25°C Haze
ENG
10:36
22-02-2023

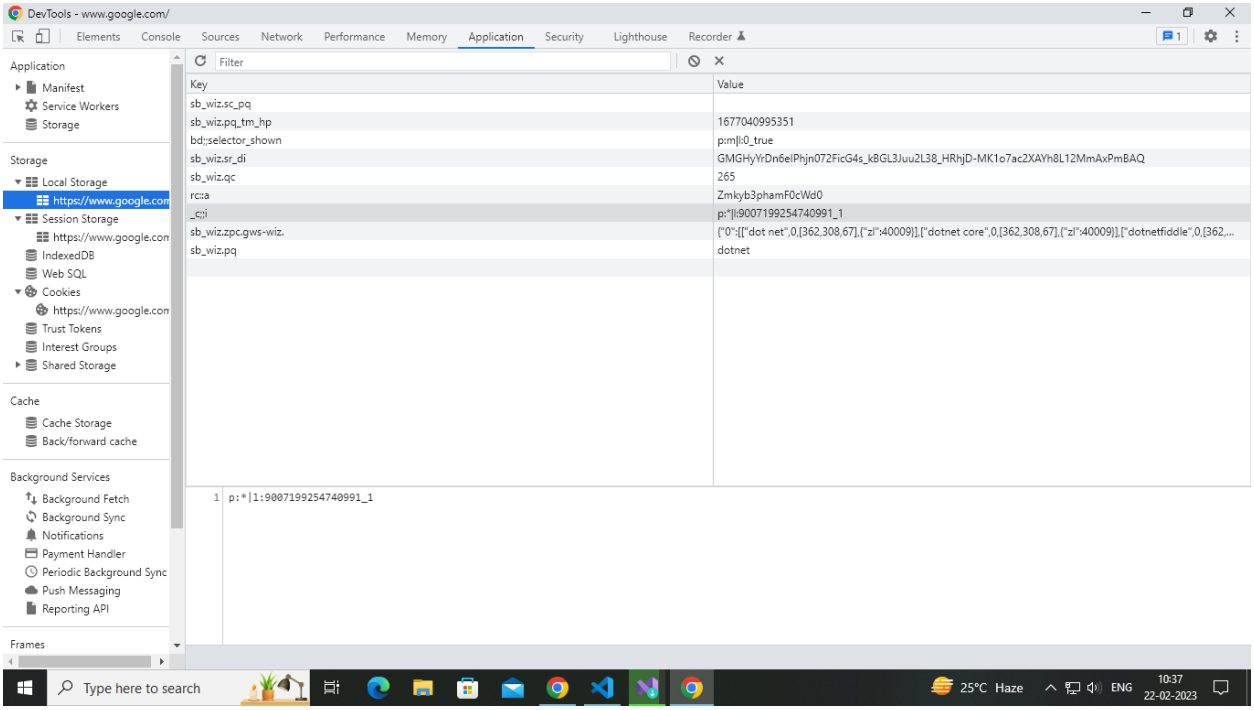

DevTools - www.google.com/
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
www.google.com #2
Screenshots
Memory
Web Vitals
CPU
NET
Frames
Timings
DCL
DCL
Main — https://www.google.com/
Main — about:blank
GPU
Chrome_ChildIOThread
Compositor
ThreadPoolForegroundWorker
ThreadPoolForegroundWorker
ThreadPoolForegroundWorker
ThreadPoolServiceThread
Summary
Bottom-Up
Call Tree
Event Log
Range: 0 – 1.37 s
1370 ms
0 ms System
1369 ms Idle
1370 ms Total
Total blocking time: 472.43ms (estimated) Learn more
Type here to search
25°C Haze
ENG
10:36
22-02-2023

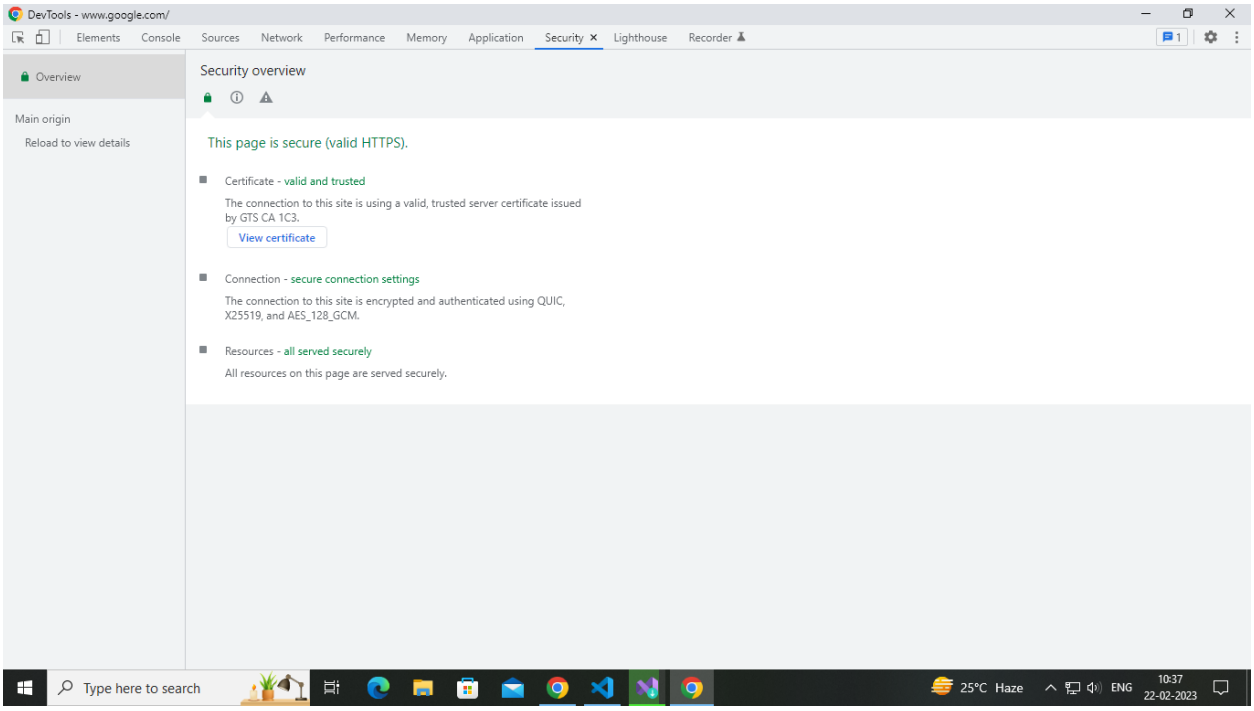
GUI Basic

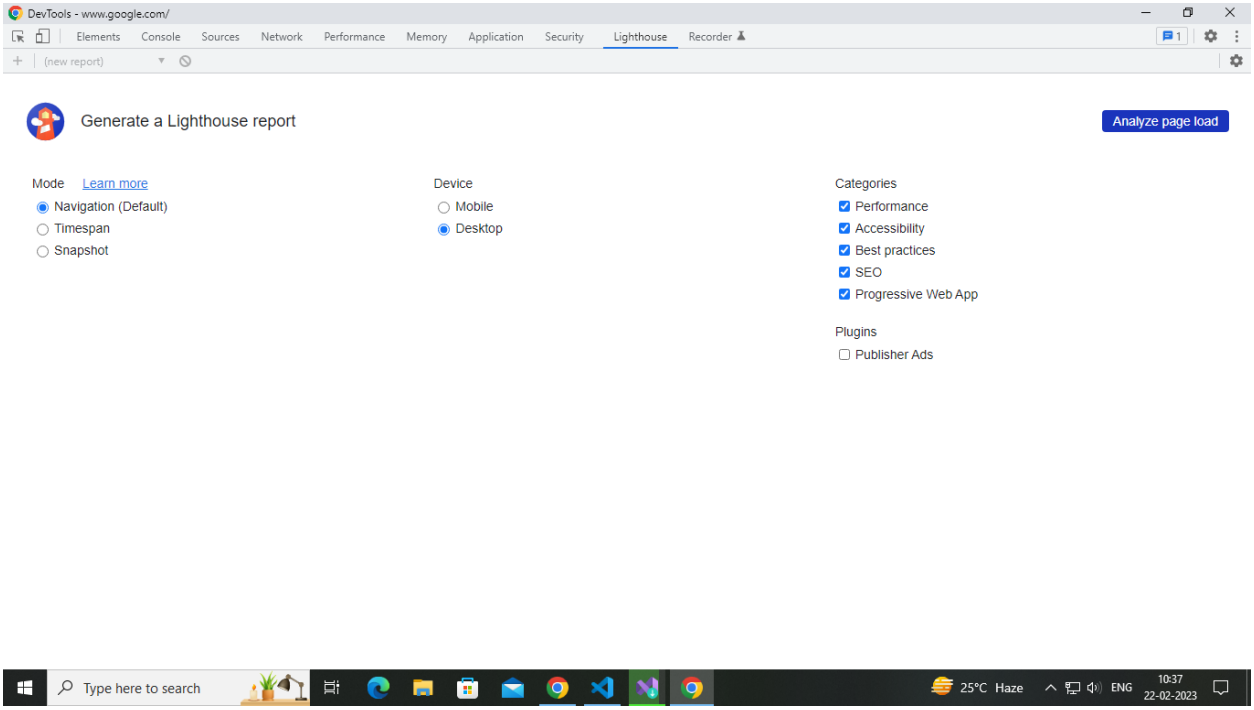

DevTools - www.google.com/
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Screenshots
Memory
Web Vitals
Frames
Timings
DCL
Main — https://www.google.com/
Main — about:blank
GPU
Chrome_ChildIOThread
Compositor
ThreadPoolForegroundWorker
ThreadPoolForegroundWorker
ThreadPoolForegroundWorker
ThreadPoolServiceThread
Summary
Bottom-Up
Call Tree
Event Log
Range: 0 – 1.37 s
1370 ms
0 ms System
1369 ms Idle
1370 ms Total
Total blocking time: 472.43ms (estimated) Learn more
Type here to search
25°C Haze
ENG
10:36
22-02-2023

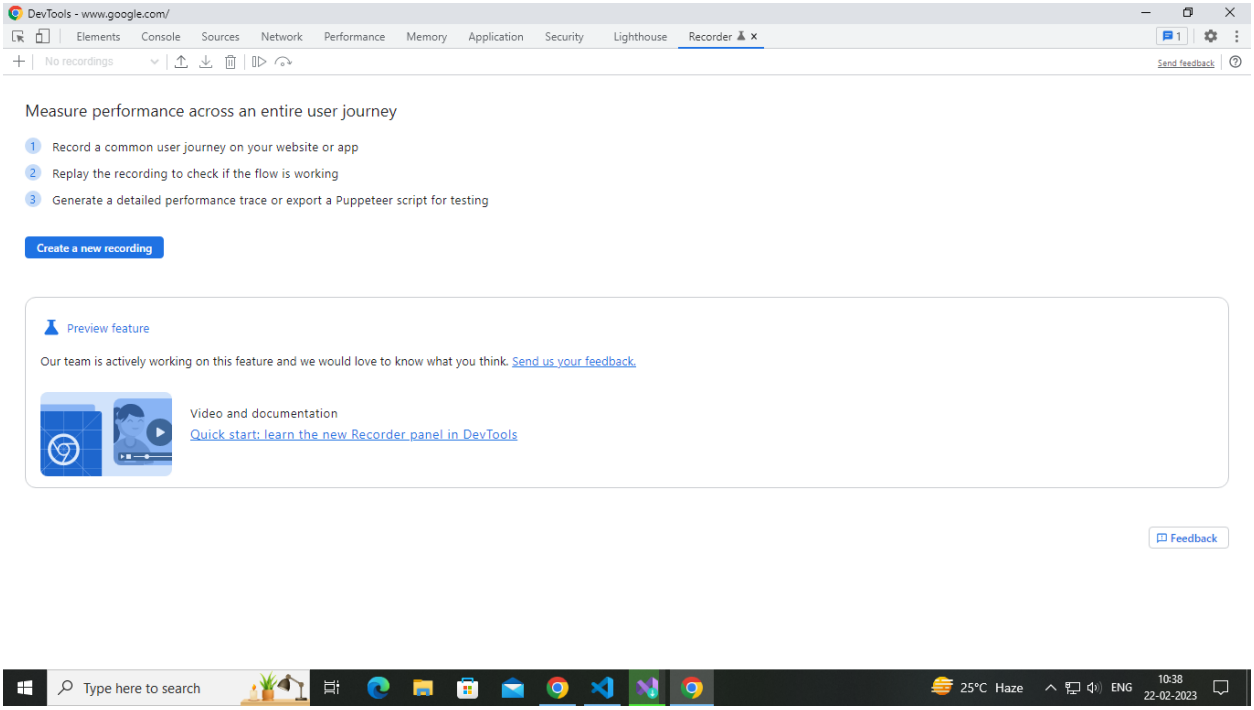

DevTools - www.google.com/
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Profiles
Select profiling type
Heap snapshot
Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.
Include numerical values in capture
Allocation instrumentation on timeline
Allocation timelines show instrumented JavaScript memory allocations over time. Once profile is recorded you can select a time interval to see objects that were allocated within it and still alive by the end of recording. Use this profile type to isolate memory leaks.
Record stack traces of allocations (extra performance overhead)
Allocation sampling
Record memory allocations using sampling method. This profile type has minimal performance overhead and can be used for long running operations. It provides good approximation of allocations broken down by JavaScript execution stack.
Select JavaScript VM instance
8.7 MB 11.7 kB/s www.google.com
8.7 MB 11.7 kB/s Total JS heap size
Take snapshot
Load
Type here to search
25°C Haze
ENG
10:37
22-02-2023

GUI Basic

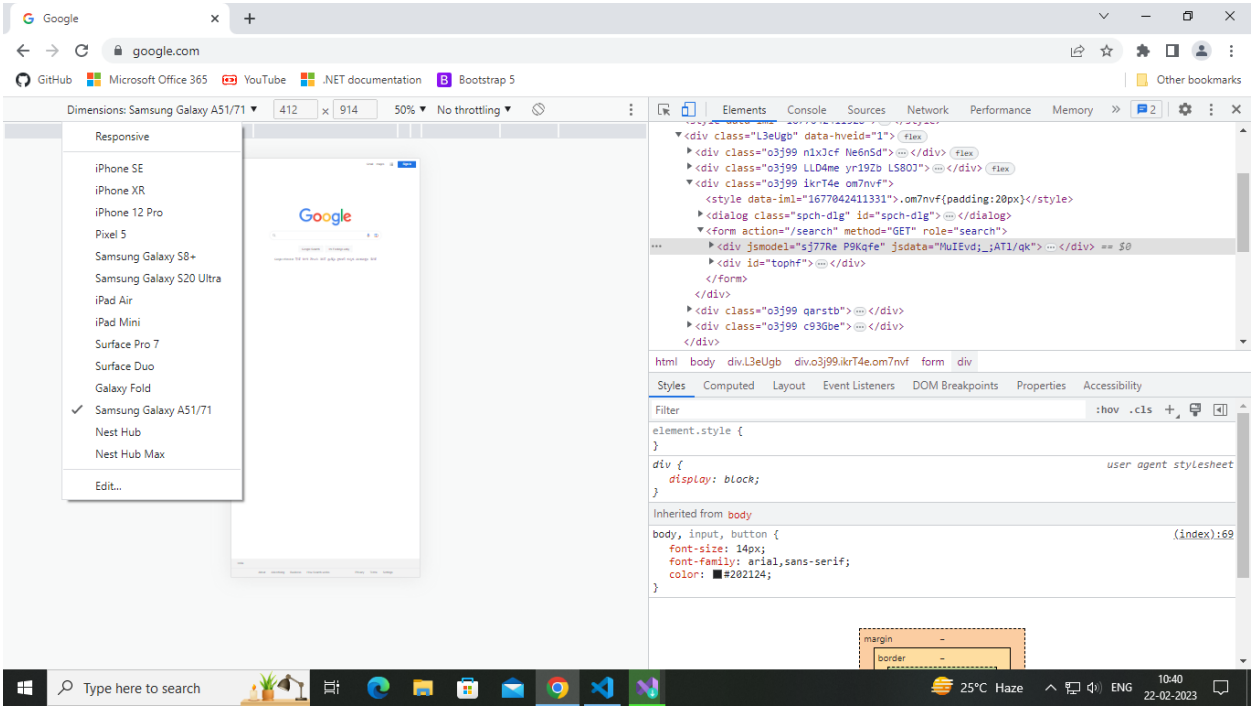

DevTools - www.google.com/
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Application
Manifest
Service Workers
Storage
Storage
Local Storage
https://www.google.com
Session Storage
https://www.google.com
IndexedDB
Web SQL
Cookies
https://www.google.com
Trust Tokens
Interest Groups
Shared Storage
Cache
Cache Storage
Back/forward cache
Background Services
Background Fetch
Background Sync
Notifications
Payment Handler
Periodic Background Sync
Push Messaging
Reporting API
Frames
Filter
Key
Value
sb_wiz.sc_pq
sb_wiz.pq_tm_hp
1677040995351
bd::selector_shown
p:m|i:0_true
sb_wiz.sr_di
GMGHyYrDn6elPhjn072FicG4s_k8GL3Juu2L38_HRhjD-MK1o7ac2XAYh8L12MmAxPmBAQ
sb_wiz.qc
265
rc::a
Zmkyb3phamF0cWd0
_c;;i
p:*|i:9007199254740991_1
sb_wiz.zpc.gws-wiz.
{"0":[["dot net",0,[362,308,67],{"zl":40009}],["dotnet core",0,[362,308,67],{"zl":40009}],["dotnetfiddle",0,[362,...
sb_wiz.pq
dotnet
1 p:*|1:9007199254740991_1
Type here to search
25°C Haze
ENG
10:37
22-02-2023


DevTools - www.google.com/
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Overview
Main origin
Reload to view details
Security overview
This page is secure (valid HTTPS).
Certificate - valid and trusted
The connection to this site is using a valid, trusted server certificate issued by GTS CA 1C3.
View certificate
Connection - secure connection settings
The connection to this site is encrypted and authenticated using QUIC, X25519, and AES_128_GCM.
Resources - all served securely
All resources on this page are served securely.
Type here to search
25°C Haze
ENG
10:37
22-02-2023


DevTools - www.google.com/
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
(new report)
Generate a Lighthouse report
Analyze page load
Mode
Learn more
Navigation (Default)
Timespan
Snapshot
Device
Mobile
Desktop
Categories
Performance
Accessibility
Best practices
SEO
Progressive Web App
Plugins
Publisher Ads
Type here to search
25°C Haze
ENG
10:37
22-02-2023


DevTools - www.google.com/
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
No recordings
Send feedback
Measure performance across an entire user journey
1 Record a common user journey on your website or app
2 Replay the recording to check if the flow is working
3 Generate a detailed performance trace or export a Puppeteer script for testing
Create a new recording
Preview feature
Our team is actively working on this feature and we would love to know what you think. Send us your feedback.
Video and documentation
Quick start: learn the new Recorder panel in DevTools
Feedback
Type here to search
25°C Haze
ENG
10:38
22-02-2023

GUI Basic


Google
google.com
GitHub
Microsoft Office 365
YouTube
.NET documentation
Bootstrap 5
Other bookmarks
Dimensions: Samsung Galaxy A51/71
412
914
50%
No throttling
Responsive
iPhone SE
iPhone XR
iPhone 12 Pro
Pixel 5
Samsung Galaxy S8+
Samsung Galaxy S20 Ultra
iPad Air
iPad Mini
Surface Pro 7
Surface Duo
Galaxy Fold
Samsung Galaxy A51/71
Nest Hub
Nest Hub Max
Edit...
Elements
Console
Sources
Network
Performance
Memory
Styles
Computed
Layout
Event Listeners
DOM Breakpoints
Properties
Accessibility
Filter
element.style {
}
div {
display: block;
}
user agent stylesheet
Inherited from body
body, input, button {
font-size: 14px;
font-family: arial,sans-serif;
color: #202124;
}
(index):69
Type here to search
25°C Haze
ENG
10:40
22-02-2023

CACHING

16.1 INTRODUCTION

- In catching we are store output of repetitive calculation.
- So we don't calculate these calculation which we are already calculate in past.
- Using this technique we can archive more performance.
- We user memorization for caching.

16.2 MEMOIZATION

- Here we use closures for memorization.
- Closure is a function which return function.

Example:

```
const addWith50WithClosure = () => {
    // outer part of function
    let lookup = {};
    return function(num){
        // inner part of function
        if(num in lookup){
            return lookup[num];
        }
        console.log("Not found, It will take time...");
        for(i = 0; i < 100000000; i++){

        }
        lookup[num] = num+50;
        return lookup[num];
    }
}
```

OOJS

17.1 Introduction

- OOJS means Object Oriented JavaScript.
- We have four main pillars.
  - Encapsulation
  - Abstraction
  - Inheritance
  - Polymorphism

Encapsulation:

- Encapsulation means wrapping up data and member function (Method) together into a single unit.
- Encapsulation automatically achieve the concept of data hiding providing security to data by making the variable as private and the property to access the private data which would be public.

Abstraction:

- Abstraction is the process of showing only essential/necessary features of an entity/object to the outside world and hide the other irrelevant information.

Inheritance:

- The ability of creating a new class from an existing class.
- It allows a child class to acquire the properties and behaviour of parent class.
- It helps to reuse, customize and enhance the existing code.

Polymorphism:

- Polymorphism is derived from 2 Greek words: poly and morphs.
- The word "poly" means many and "morphs" means forms.
- Here we can define same property with different behaviour.

17.2 Class

- Class represent real world entity which have different properties and behaviours.
- JavaScript provides class key word to create class.
- Class have constructor, properties and methods.

Example:

```
class Person{
    personId;
    firstName;
    lastName;

    constructor(personId, firstName, lastName){
        this.personId = personId;
        this.firstName = firstName;
        this.lastName = lastName;
    }

    getPerson(){
        console.log(this.personId + " " + this.firstName + " " +
this.lastName);
    }
}
```

17.3 Static Keyword

- Static key word help us to create class level properties.
- Static properties is shared by all object in class.
- In JavaScript we can create static variable, static method and static block.
- We can access static member using class name.

Example:

```
class Student{
    // we can't use static keyword directly with class, we only use static
key word with propart, method amd block

    id;
    name;
    static collage = "Collage Name";

    constructor(id, name){
        this.id = id;
        this.name = name;
    }

    static getCollage(){
        return this.collage;
```

```
    }
    static setCollage(collage){
        this.collage = collage;
    }
    static{
        console.log(this.collage);
    }
}

let student = new Student(1, "Dhruvil Dobariya");
Student.setCollage("Darshan");

console.log(student.id);
console.log(student.name);
console.log(Student.getCollage());

let student2 = new Student(2, "Dhaval Dobariya");

console.log(student2.id);
console.log(student2.name);
console.log(Student.getCollage());
```

ASYNC AND AWAIT

18.1 INTRODUCTION

- JavaScript is always synchronous and single-threaded that provides the event loops.
- The event loops enable us to queue up an activity.
- This activity will not happen until the loops become available after the program that queued the action has completed the execution.
- However, our program contains a large number of functionalities, which causes our code to be asynchronous.

18.2 ASYNC

- An async function is a function that is declared with the async keyword and allows the await keyword inside it.
- The async and await keywords allow asynchronous, promise-based behaviour to be written more easily and avoid configured promise chains.
- The async keyword may be used with any of the methods for creating a function.

18.3 AWAIT

- JavaScript Await function is used to wait for the promise.
- It could only be used inside the async block.
- It instructs the code to wait until the promise returns a response.
- It only delays the async block.
- Await is a simple command that instructs JavaScript to wait for an asynchronous action to complete before continuing with the feature.
- It's similar to a "pause until done" keyword.
- The await keyword is used to retrieve a value from a function where we will usually be used the then() function.
- Instead of calling after the asynchronous function, we'd use await to allocate a variable to the result and then use the result in the code as we will in the synchronous code.

Example:

```
async function myDisplay() {
    let myPromise = new Promise(function(myResolve, myReject) {
        myResolve("Hello World!!");
    });
    document.getElementById("main").innerHTML = await myPromise;
```

```
}
myDisplay();
```

IMPORT/EXPORT

19.1 REQUIRED

- The module.exports in Node.js is used to export any literal, function or object as a module.
- Here we can import any export element using required.

Example:

```
// Export.js
// Variable
module.exports = "Hello World!";

// Function
module.exports = function(a,b){
    return a+b;
}

// Class
module.exports = function(){
    this.name = "Dhruvil Dobariya";
    this.getName = () => {
        return this.name;
    }
}

// Import.js
let element = require("./Export");

// variable
console.log(element);

// Function
console.log(element(2,5));

// Class
const e = new element()
console.log(e.name);
console.log(e.getName());
```

19.2 ES6

- ES6 provides two ways to export a module from a file.
  - Named export
  - Default export

19.2.1 DEFAULT EXPORT:

- We can use only one default export in one file.
- Here we can change name of import component without using alias.

```
// Default
// export
const Student = () => {}
export default Student;

// import
import StudentExport from "./Export.js";
```

19.2.2 NAMED EXPORT:

- We can export multiple element in one file.
- Then import the specific exports they want surrounded in braces.
- The name of imported module has to be the same as the name of the exported module.

```
// Nameed export:
// exports from ./Export.js.js file
export const Student = () => {}
export const User = () => {}

// imports:
// import a single named export
import { Student } from "./Export.js";

// multiple named exports
import { Student, User } from "./Export.js";

// import a different name by using "as":
import { User as NormalUser } from "./Export.js";

// import all the named exports onto an object
import * as PersonManager from "./Export.js";
// PersonManager.User and PersonManager.Student
```