HTML

- Declaration of meta tags :
  - ➢ Most important meta attributes

Attribute	Value	Description
charset	character_set	Specifies the character encoding for the HTML document
name	application-name author description generator keywords Viewport robots	Specifies a name for the metadata
http-equiv	content-security-policy content-type default-style refresh	Provides an HTTP header for the information/value of the content attribute

Attribute	Value	Description
content	text	Specifies the value associated with the http-equiv or name attribute

  - ➢ Example:

```
<title>Meta Tags Example</title>
<meta charset = "utf-8">
<meta name = "keywords" content = "HTML, Meta Tags, Metadata" />
<meta name = "description" content = "Learning about Meta Tags." />
<meta name = "revised" content = "Mck, 5/8/2016" />
<meta http-equiv = "refresh" content = "5; url = https://google.com" />
```

- Character entities :

Description	Entity Name	Entity Number (decimal)	Entity Number (hexadecimal)	Result
non-breaking space				
less than	&lt;	&#60;	&#x3c;	<
greater than	&gt;	&#62;	&#x3e;	>
ampersand	&amp;	&#38;	&#x24;	&
double quotation mark	&quot;	&#34;	&#x22;	"
single quotation mark (apostrophe)	&apos;	&#39;	&#x27;	'
cent	&cent;	&#162;	&#xa2;	¢
pound	&pound;	&#163;	&#xa3;	£
yen	&yen;	&#165;	&#xa5;	¥
euro	&euro;	&#8364;	&#x20ac;	€
copyright	&copy;	&#169;	&#xa9;	©
registered trademark	&reg;	&#174;	&#xae;	®

- Creating list
  - **Syntax:**
    <ul> or **<ol>**
    **
content
**
    **
....
**
    **
** or **
**
- Element attributes :
  - Id
  - Title - used as tool tip
  - Class
  - Style

- Image creation
  - ❑ Use img tag to include images in html page

    Syntax:
    <img attributes **>**

    ❑ Most import attributes:
    ❑ alt is mainly used for accessibility
    ❑ alt attribute provides alternate text for image:

Attribute	Value	Description
alt	text	Specifies an alternate text for an image
height	pixels	Specifies the height of an image
loading	eager, lazy	Specifies whether a browser should load an image immediately or to defer loading of images until some conditions are met
src	URL	Specifies the path to the image
width	pixels	Specifies the width of an image

- Anchor Link :
  - ❑ Most important attribute: href → hyper reference:
    - o URL that hyperlink points to
    - o Links are not restricted to HTTP-based URLs – they can use any URL scheme supported by browsers:
      - Sections of a page with fragment URLs `<a href="#section2">Go to Section 2</a>`
      - Pieces of media files with media fragments (parts of images, video files, not fully supported yet)
      - Telephone numbers with tel: URLs `<a href="tel:+4733378901">+47 333 78 901</a>`
      - Email addresses with mailto: URLs `<a href="mailto:someone@example.com">Send email</a>`
  - ➢ Anchor (a) tag
    - ❑ Other important attribute is target
    - ❑ Where to display linked URL, as name for browsing context (tab, window, or <iframe>):
      - o _self: current browsing context (Default)
      - o _blank: usually new tab, but users can configure browsers to open new window instead
      - o _parent: parent browsing context of current one (If no parent, behaves as _self)
      - o _top: topmost browsing context ("highest" context that is ancestor of current one, if no ancestors, behaves as _self)

    Syntax:
    *****text to display*****
- Table : colspan to add column, rowspan = "2"

```
<table border="1">
    <tr>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Email Address</th>
    </tr>
    <tr>
        <td>John</td>
        <td>Doe</td>
        <td>john.doe@hotmail.com</td>
    </tr>
    <tr>
        <td>Mary</td>
        <td>Evans</td>
        <td>mevans@gmail.com</td>
    </tr>
</table>
```

First Name	Last Name	Email Address
John	Doe	john.doe@hotmail.com
Mary	Evans	mevans@gmail.com

- Audio
  - **Audio Element**

    - ❑ Part of new HTML5 web components standard
    - ❑ Fully-equipped audio player with controls
    - ❑ Most important attributes:
      - Can use src (one source only) or source attribute (multiple sources)

    Syntax:
    <audio attributes**>**
    fall back options if browser does not support element
    </audio>

Attribute	Description
autoplay	Boolean attribute: if specified, the audio will automatically begin playback as soon as it can do so, without waiting for the entire audio file to finish downloading
controls	If this attribute is present, the browser will offer controls to allow the user to control audio playback, including volume, seeking, and pause/resume playback
loop	Boolean attribute: if specified, the audio player will automatically seek back to the start upon reaching the end of the audio
mute	Boolean attribute that indicates whether the audio will be initially silenced. Its default value is false.
preload	Intended to provide a hint to the browser about what the author thinks will lead to the best user experience. It may have one of the following values: • none: Indicates that the audio should not be preloaded. • metadata: Indicates that only audio metadata (e.g. length) is fetched. • auto: Indicates that the whole audio file can be downloaded, even if the user is not expected to use it.
src	URL of the audio to embed
source	HTML element specifying multiple media resources for the <picture>, the <audio> element, or the <video> element Allows to specify alternative video files which the browser may choose from, browser will use the first recognized format

- Picture tag to add multiple elements depending on the condition

```
<picture>
    <source media="(min-width: 1300px)" srcset="cat-1280.jpg">
    <source media="(min-width: 800px)"  srcset="cat-800.jpg">
    <source media="(min-width: 600px)"  srcset="cat-600.jpg">
    <img src="cat-400.jpg" >
</picture>
```

- Video :

```
<video controls width="500" height="400" muted>
    <source src="flower.webm" type="video/webm" />
    <source src="flower.mp4" type="video/mp4" />
    <!--Fallback option-->
    Download the <a href="flower.webm">WEBM</a>
    or <a href="flower.mp4">MP4</a> video.
</video>
```

```
src="flower.webm#t=2,4"
src="flower.mp4#t=2,4"
```

- Content setting IFrame

```
<h1>iframe Element</h1>
<a href="Example4-4a.html" target="content">Content A</a><br />
<a href="Example4-4b.html" target="content">Content B</a>
<hr />
<iframe name="content">
</iframe>
```

- Form
  - Input types :
    - Radio Single - Checkbox multiple
    - Text Input Controls
      Checkboxes Controls
      Radio Box Controls
      Select Box Controls
      File Select boxes
      Hidden Controls
      Clickable Buttons
      Submit and Reset Button
  - Input scripting :
    - ➢ Text input controls
      - ❑ name attribute:
        - o Only used within form tag
        - o Gives control name when submitting data to server
        - o Is not necessarily unique (multiple radio buttons have same name)
      - ❑ id attribute:
        - o Used within entire html page
        - o Must be unique within page
        - o Used by other controls, CSS and JavaScript

      Syntax:
      <input type= "text" **name=** "*name*" **value =** "initial value" **/>**

Attribute	Description
type	Indicates the type of input control and for text input control it will be set to text
name	Used to give a name to the control which is sent to the server to be recognized and get the value
id	Used to give a unique ID to the control which is used by other controls and JavaScript
value	This can be used to provide an initial value inside the control
size	Allows to specify the width of the text-input control in terms of characters
maxlength	Allows to specify the maximum number of characters a user can enter into the text box
placeholder	Defines a non-selectable placeholder text that only appears when the input is empty
required	Used for form validation, requires a non-empty value to submit form
disabled	Disables the control, cannot be used by the user but is visible and grayed out
  - Fieldset

```
<form>
    <fieldset>
        <legend>Billing Address</legend>
        Address<br /><input type="text" placeholder="Enter address" /><br />
        City<br /><input type="text" placeholder="Enter city" /><br />
        State<br /><input type="text" placeholder="Enter state" /><br />
        Zip Code<br /><input type="text" placeholder="Enter zip code"/><br />
    </fieldset>
```

  - Labelling is based on ID :

```
<form>
    <fieldset>
        <legend>Billing Address</legend>
        <label for="txtAddress">Address</label><br />
        <input type="text" id="txtAddress" placeholder="Enter address" /><br />
        <label for="txtCity">City</label><br />
        <input type="text" id="txtCity" placeholder="Enter city" /><br />
        <label for="txtState">State</label><br />
        <input type="text" id="txtState" placeholder="Enter state" /><br />
        <label for="txtZipcode">Zip Code</label><br />
        <input type="text" id="txtZipCode" placeholder="Enter zip code"/><br />
    </fieldset>
</form>
```

- Checkbox
  - ```
    <label for="chkShipBill">Same as shipping address</label>
    <input type="checkbox" id="chkShipBill"><br />
    ```

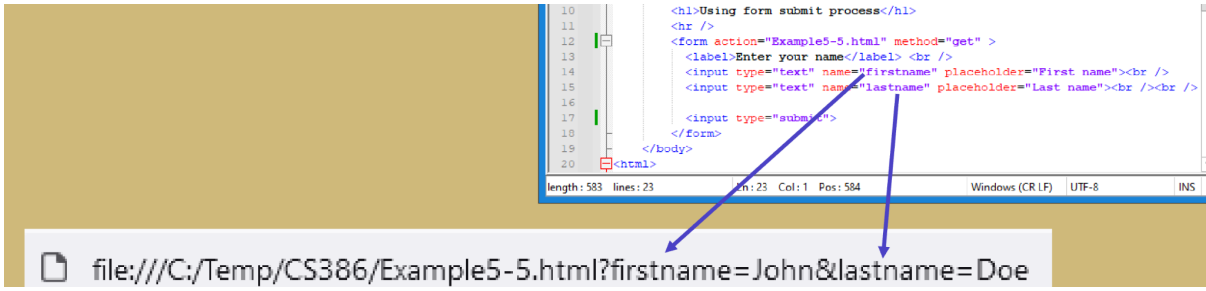
- Radio - only one is selected for one name value
  - ```
    <p>Please select your gender:</p>
      <input type="radio" id="male" name="gender" value="male">
      <label for="male">Male</label><br>
      <input type="radio" id="female" name="gender" value="female">
      <label for="female">Female</label><br>
      <input type="radio" id="diverse" name="gender" value="diverse">
      <label for="diverse">Diverse</label>
    ```

- Select Elements :
  - Drop down list - only one value is should and button click to open. Single select , size is one
    - ```
      <form>
          <select id="country" name="country">
              <option>Select country.....</option>
              <option value="AF">Afghanistan</option>
              <option value="AD">Andorra</option>
              <option value="AR">Argentina</option>
              <option value="BE">Belgium</option>
              <option value="BO">Bolivia</option>
              <option value="CA">Canada</option>
              <option value="CN">China</option>
              <option value="DK">Denmark</option>
              <option value="FI">Finland</option>
              <option value="GU">Guam</option>
              <option value="IS">Iceland</option>
              <option value="KE">Kenya</option>
              <option value="MX">Mexico</option>
              <option value="NG">Nigeria</option>
              <option value="PT">Portugal</option>
              <option value="RS">Serbia</option>
              <option value="SE">Sweden</option>
              <option value="TN">Tunisia</option>
              <option value="GB">United Kingdom</option>
              <option value="US">United States</option>
              <option value="ZW">Zimbabwe</option>
          </select>
      </form>
      ```
  - Single select list box - multiple value , size for init display
    - ```
      <select id="country" name="country" size = "5">
      ```
  - Multiselect list box - multiple
    - ```
      <select id="country" name="country" size = "5" multiple>
      ```

- Range :
  - ```
    <form>
        <h4>Select temperature</h4>
        <label for="rng">30</label>
        <input type="range" id="rng" min="30" max="120" step="0.1" value="70" oninput="out.value=val
        <label for="rng">120</label>
        <br />
        <output id="out">70</output>
    </form>
    ```

- Additional Types :

```
<table>
    <tr><th>Element</th><th>Apperance</th></tr>
    <tr><td><label for="type-text">Text (text)</label></td><td><input type="text" name="type-text" id="type-text"></td></tr>
    <tr><td><label for="type-search">Search (search)</label></td><td><input type="search" name="type-search" id="type-search"></td></tr>
    <tr><td><label for="type-tel">Telephone (tel)</label></td><td><input type="tel" name="type-tel" id="type-tel"></td></tr>
    <tr><td><label for="type-url">URL (url)</label></td><td><input type="url" name="type-url" id="type-url"></td></tr>
    <tr><td><label for="type-email">E-mail (email)</label></td><td><input type="email" name="type-email" id="type-email"></td></tr>
    <tr><td><label for="type-datetime">Date and Time (datetime)</label></td><td><input type="datetime" name="type-datetime" id="type-datetime"></td></tr>
    <tr><td><label for="type-date">Date (date)</label></td><td><input type="date" name="type-date" id="type-date"></td></tr>
    <tr><td><label for="type-month">Month (month)</label></td><td><input type="month" name="type-month" id="type-month"></td></tr>
    <tr><td><label for="type-week">Week (week)</label></td><td><input type="week" name="type-week" id="type-week"></td></tr>
    <tr><td><label for="type-time">Time (time)</label></td><td><input type="time" name="type-time" id="type-time"></td></tr>
    <tr><td><label for="type-dt-local">Local Date and Time (datetime-local)</label></td><td><input type="datetime-local" name="type-dt-local" id="type-dt-local"></td></tr>
    <tr><td><label for="type-number">Number (number)</label></td><td><input type="number" name="type-number" id="type-number" min="0" max="20"></td></tr>
    <tr><td><label for="type-range">Range (range)</label></td><td><input type="range" name="type-range" id="type-range" min="0" max="100"></td></tr>
    <tr><td><label for="type-color">Color (color)</label></td><td><input type="color" name="type-color" id="type-color"></td></tr>
</table>
```

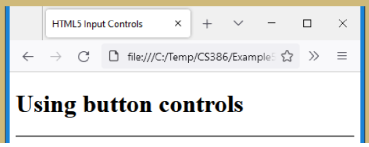
- Form Submit :
  - To post back to same page, use:
    - Empty string
    - Question mark (action="?")
    - Actual file name of current page
  - 

  - ```
    <form action="Example5-5.html" method="post" >
      <label>Enter your name</label> <br />
      <input type="text" name="firstname" placeholder="First name" required><br />
      <input type="text" name="lastname" placeholder="Last name" required><br /><br />
    ```
  - **Syntax:**
    <input type="button" value="*Click here***">**
  - Image Button :

Form buttons

- Also separate button tag
- Allows for more formatting options including image on button
- <input> is void element, whereas <button> is not
- Place text or image between opening and closing tag
- **WARNING:** Without type attribute, button element is submit button by default!

Syntax:
<button type="button" >
text or image
</button>


    ```

CSS

- Linking
  - **<link rel="**stylesheet**" type="**text/css**" href="***file location***">**
- Setting property

```
p {
  color: red;
  width: 500px;
  border: 1px solid black;
}
```

- Selector

Selector name	What does it select	Example
Element selector (sometimes called a tag or type selector)	All HTML elements of the specified type.	p selects <p>
ID selector	The element on the page with the specified ID. On a given HTML page, each id value should be unique.	#my-id selects <p id="my-id"> or <a id="my-id">
Class selector	The element(s) on the page with the specified class. Multiple instances of the same class can appear on a page.	.my-class selects <p class="my-class"> and <a class="my-class">
Attribute selector	The element(s) on the page with the specified attribute.	img[src] selects <img src="myimage.png"> but not <img>
Universal selector	Selects all elements in a web page	* selects all elements

- Selector Combinators

Combinator name	What does it select	Example
Descendant combinator	The " " (space) combinator selects elements that are descendants of the first element.	p a selects <a> within <p>
Child combinator	The > combinator selects nodes that are direct children of the first element.	ul > li selects nested directly inside element
General sibling combinator	The ~ combinator selects siblings. This means that the second element follows the first (though not necessarily immediately), and both share the same parent.	p ~ span selects elements that follow <p>, immediately or not
Adjacent sibling combinator	The +/- combinator matches either the next (+) or the previous (-) element based on the first element.	h2 + p Selects first <p> element that immediately follows <h2> element

```
abbr[title] {
  text-decoration: underline;
  text-decoration-style: dotted;
  text-decoration-color: red;
}
```

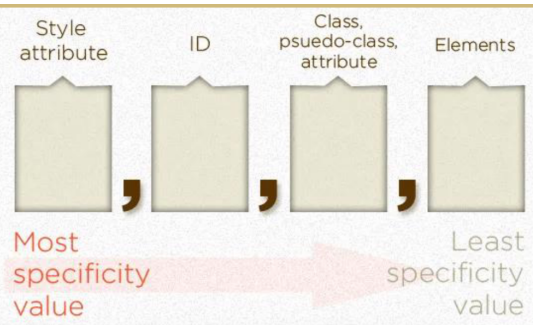
- Pseudo elements

```
a:hover {
  text-decoration: underline;
}

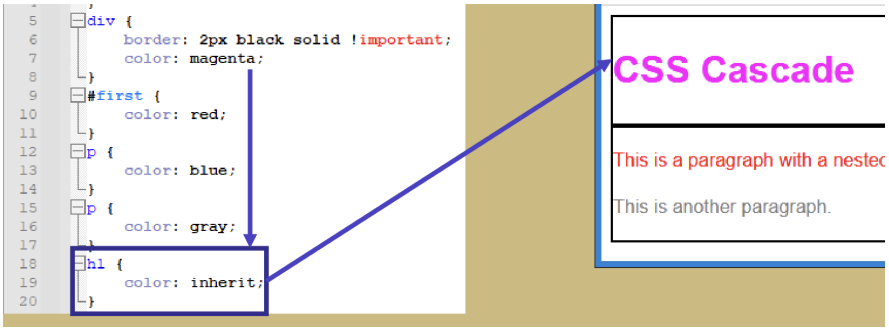
a:active {
  color: black;
}

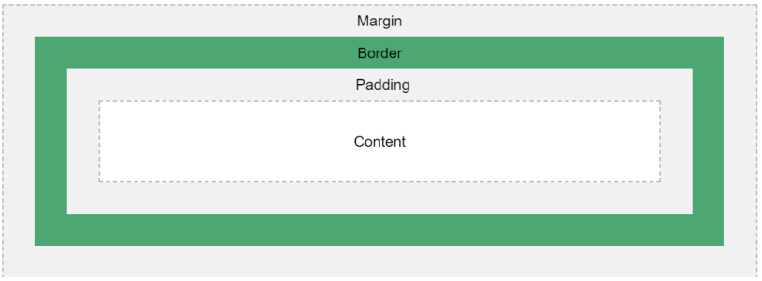
a:visited {
  color: purple;
}
```

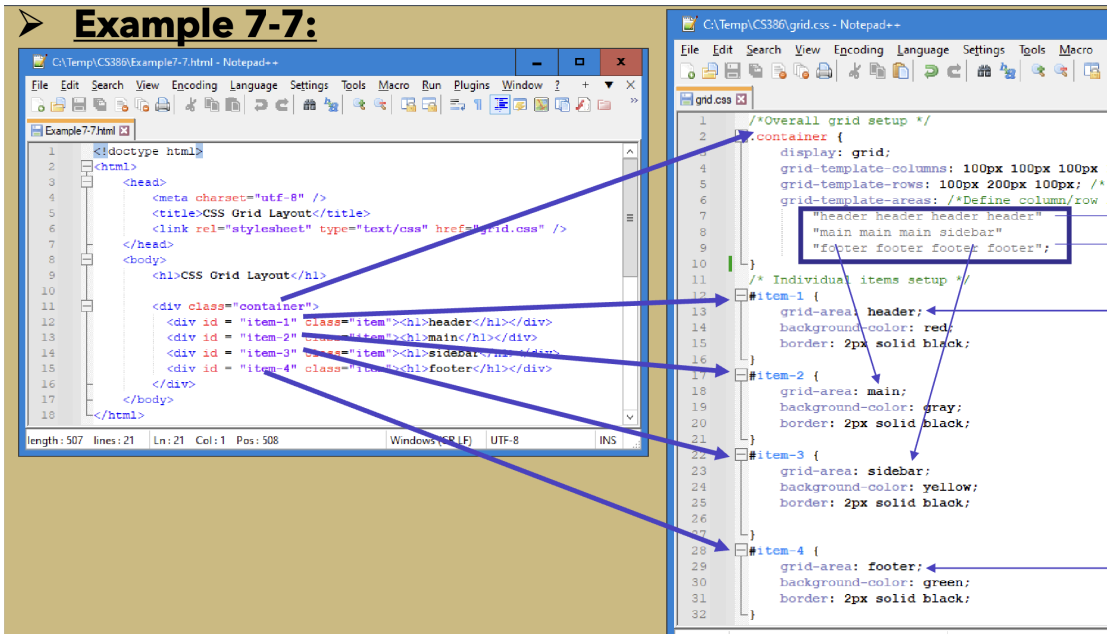
- Nested CSS
  - ```
    a {
        color: red;
        text-decoration: none;
        text-transform: uppercase;
        &:hover {
            text-decoration: underline;
        }
        &:active {
            color: black;
        }
        &:visited {
            color: purple;
        }
    }
    ```
- Unit Measure
Unit	Description
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px *	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)
- Relative Length
Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to the width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element
- Specificity
  - ❑ **Notes:**
    - Generally read values as if they were just numbers, like 1,0,0,0 is "1000", and so clearly wins over specificity of 0,1,0,0 or "100"
    - Commas are there to remind us that this is not really "base 10" system

    ```

- At all declared - last one wins
- Inheritance
    - Child copy father
    - 

    - border-width: 1em 2em 3em 4em
    - 

- Declaration of grid styling :
    - 

    - ```
/*Overall grid setup */
.container {
    display: grid;
    grid-template-columns: 100px 100px 100px 100px; /* 4 columns, widths evenly spaced */
    grid-template-rows: 100px 200px 100px; /* 3 rows, varying heights */
    grid-template-areas: /*Define column/row layout*/
        "header header header header"
        "main main main sidebar"
        "footer footer footer footer";
    column-gap: 5px;
    row-gap: 20px;
}
```
```

JAVASCRIPT

- Simple JAVAScript syntax
  - ```
    let num1, num2, sum; //Declare variables
    try {
        num1 = +prompt('Enter number1 ', 0); //Prompt returns string
        num2 = +prompt('Enter number2 ', 0); //Prompt returns string
    }
    catch {
        num1 = +process.argv[2]; //3rd element of argument vector
        num2 = +process.argv[3]; //4th argument of argument vector
    }
    sum = num1 + num2; //Calculate sum
    console.log('The sum =' + sum); //Display sum
    ```
  - ```
    alert(`Using interpolation string
    Sum = ${sum}`); //Display sum
    ```
- Maths
  - ```
    let hypo = Math.sqrt(Math.pow(a,2) + Math.pow(b,2));
    ```
- Date
  - ```
    let dt = new Date(); //Initialize variable to current date/time
    //Initialize with first day of current month
    let firstDayMonth = new Date(dt.getFullYear(), dt.getMonth());
    //Display today's date in ISO format
    console.log(`Today's date in ISO format is ${dt.toISOString()}`);
    //Display today's date in localized date string only
    console.log(`Today's date in local format is ${dt.toLocaleDateString()}`);
    //Display first day of current month in localized date string only
    console.log(`First day of current month is ${firstDayMonth.toLocaleDateString()}`);
    //Calculate difference between these two dates
    let diffDates = dt - firstDayMonth; //This is expressed in milliseconds
    console.log(`Difference between those two dates is ${diffDates}`);
    //Calculate number of whole hours
    let hours = Math.round(diffDates/1000/3600,0);
    console.log(`Difference in hours is ${hours}`);
    //Calculate number of days
    let days = dt.getDate() - firstDayMonth.getDate();
    console.log(`Difference in days is ${days}`);
    ```
- Boolean
  - Following values are false:
    - undefined
    - null
    - 0
    - -0
    - NaN
    - "" // empty string
- Conversion
  - **Syntax:**
    num.**toFixed**(*d*)
    d = number of decimals (optional, default 0)

    Syntax:
    num.**toExponential**(*d*)
    d = number of decimals (optional, sets as many as possible)

    Syntax:
    num.**toPrecision**(*d*)
    d = number of decimals (optional, number is returned without any formatting)

Syntax:
parseInt(*string [, radix]***)**

nber

Syntax:
parseFloat(*string***)**

- Declarations
  - **var**: Declares variable, optionally initializing it to value (older way, hoisted to top)
  - **let**: Declares block-scoped, local variable, optionally initializing it to value (modern declaration)
  - **const**: Declares block-scoped, read-only named constant, must initialize
  - Variables in block only exist inside block
  - Only the fail safe is evaluvated

```
console.log(`5 && 0 && null --> ${5 && 0 && null}`);
console.log(`0 && 5 && null --> ${0 && 5 && null}`);
console.log(`5 || 0 || null --> ${5 || 0 || null}`);
console.log(`0 || 5 || null --> ${0 || 5 || null}`);
```

```
5 && 0 && null --> 0
0 && 5 && null --> 0
5 || 0 || null --> 5
0 || 5 || null --> 5
```

- Arrays

```
let carBrands = ["GM", "Ford", "Honda"]; //Creating Array
alert("First element = " + carBrands[0]);//Accessing first element
alert("Last element = " + carBrands[2]);//Accessing last element
//Better to use length property to access last element
alert("Last element = " + carBrands[carBrands.length-1]);
```

  - Array Manipulation
    - Adding/Deleting Array Elements
      - Method unshift
        - Allows for adding one or more elements to beginning of array
        - Shifts existing elements to the "right"
        - Automatically updates indices
      - Method pop
        - Use pop method to delete last element at end of array
      - Method shift
        - Use shift method to delete first element at beginning of array
        - Shifting existing elements to "left"
        - Automatically updates indices
      - Method splice
        - Deletes or adds elements at specified position
      - Operator delete
        - Deletes array elements, no shifting takes place
        - **Warning:** Leaves holes in array (sparse array), elements are undefined

Syntax:
*var_array***.unshift(** *element[s]* **)**

Syntax:
*var_array***.pop()**

Syntax:
*var_array***.shift()**

Syntax:
*var_array***.splice(***index, howmany, item1, item2 ..itemn***)**

Syntax:
delete *var_array***[***index***]**

- Looping over

Syntax:
for (let *var_index* **in** *var_array*) **{**
statements
}

Syntax:
for (let *var_element* **of** *var_array*) **{**
statements
}

- 2D Arrays

```
//Two dimensional array
let arrActivities = [
    ['Work', 9],
    ['Eat', 1],
    ['Commute', 2],
    ['Play Game', 1],
    ['Sleep', 7]
];
//console.table(arrActivities); //Use table method of console
let strOutput = "Array of activities \n";
for (let row = 0; row < arrActivities.length; row++) { //Loop over rows
    strOutput += "Row " + row + "--> "; //Initialize row string
    for (let col = 0; col < arrActivities[row].length; col++) { //Loop over columns
        strOutput += "Col " + col + ": " + arrActivities[row][col] + " / "; //Output for columns
    }
    strOutput = strOutput.slice(0, strOutput.length - 3); //Remove last forward slash by cutting last 3 chars
    strOutput += "\n" //Add line break after each row
}
console.log(strOutput); //Output final string
```

- Functions

```
//Function declaration
function fAdd(val1, val2) {
    let sum = val1 + val2;
    return sum;
}
```

- CRUD Operations: Create read update and delete
  - Find

```
//Storing table like data in array with object elements
let arrUsers = [
    {id: 1, name: 'Bob', privs: 'admin', active: true },
    {id: 2, name: 'Mary', privs: 'user', active: true },
    {id: 3, name: 'Mia', privs: 'user', active: true },
]
console.log('CREATE operation--------------------------');
//CREATE: Add element at the end of array
arrUsers[arrUsers.length] = {id: 4, name: 'Tome', privs: 'readonly', active: true};
//READ: Loop over array and display records
console.log('READ operation--------------------------');
for (let el of arrUsers) {
    console.log(JSON.stringify(el)); //Show each row/object
}
//Finding record within Array having id value of 2, return index
const idx = arrUsers.findIndex(function(e) {return e.id === 2});
```

  - Set and delete

```
arrUsers[idx].name = "Susan";
console.log(JSON.stringify(arrUsers[idx]));//Show record
console.log('DELETE operation----------------------------
//DELETE: Using splice method
arrUsers.splice(idx,1); //At index, how many elements
```

  -

- Objects - Javascript
  - Objects are initialised to var instead of pointing values

```
let objX = {x:5, y:12}; //Declare object objX
let objZ = {x:5, y:12}; //Declare object objY, same values, but different object
alert('objX is equal to objZ = ' + (objX === objZ));
let objY = objX; //Create objY, points to same object as objX
objY.y = 10; //Reassigning property y to new value
alert("Property y of objX = " + objX.y);//Property y of objX also shows new value
```

  - Looping

```
let book = {author: "Shakespeare", title: "MacBeth", edition: 6}; //Book Object
for (let prop in book) { //Loop over each property
    alert('Property: ' + prop + '  Value: ' + book[prop]); //Show property and value
}
```

  - Static variables of objects are initialised and can be used across functions

```
main(); //Call function main
//Function declarations
function main() {
    fCount.count = 0; //Create object property
    for (let i=0; i < 3; i++) {//Loop 3 times
        fCount.count++; //Increment object property count
        console.log('Iteration = ' + i); //Display iteration variable
        fCount(); //Call function fCount()
    }
}
function fCount() {
    let counter = 0; //Initialize local variable
    console.log('----Static count = ' + fCount.count); //Display property count
    counter++; //Increment local variable counter
    console.log('----Local variable counter = ' + counter);//Display variable counter
}
```

  -

- Json
  - Function, RegExp, and Error objects - are not defined in json
  - Stringify and parse
  - Debugging by typing debugger

```
let object = {a:1, b: false, c: null, d: "", f: undefined, z: new Date()}; //Initialize
console.log("Initial object: \n" + fObjLoop(object)); //Display object in string version
let serial = JSON.stringify(object); //Create serialized string
console.log("Serialized string: \n" + serial); //Display serialized string
let restore = JSON.parse(serial); //Restore object from serialized string
console.log("Restored object: \n" + fObjLoop(restore));
```

- Adding javascript
  - **Syntax:**
    <script src = "*js file*" **type=**"text/javascript"**></script>**
  - To listen :

```
window.addEventListener("load", fLoad ); //Event registration

function fLoad() {
    let btn = document.getElementById("btnClick"); //Reference to button value
    console.log("Button value attribute = " + btn.value); //Display value attribute
}
```

  - Navigator-window and document
  - Accessing nodes :

```
len = b.childNodes.length; i < len; i++){
(i + ": " + b.childNodes[i].nodeName); //C
```

Properties	Description
parentNode	Node that is the parent of this one, or null for nodes like the Document object that have no parent
childNodes	Read-only array-like object (NodeList) that is a live representation of a Node's child nodes
firstChild, lastChild	First and last child nodes of a node, or null if the node has no children
nextSibling, previousSibling	The next and previous sibling node of a node (Two nodes having same parent are siblings, their order reflects the order in which they appear in the document)
nodeType	Document nodes = 9, Element nodes = 1, Text nodes =3, Comments nodes =8
nodeValue	The textual content of a Text or Comment node
nodeName	The tag name of an Element, converted to uppercase

childElementCount	Returns an elements's number of child elements
firstElementChild	Returns the first child element of an element
lastElementChild	Returns the last child element of an element
nextElementSibling	Returns the next element at the same node tree level
parentElement	Returns the parent element node of an element
previousElementSibling	Returns the previous element at the same node tree level

- Tracing elements by tag
  - **let** *els* **= document.getElementsByName(“***name value***”)**
  - **let** *el* **= document.getElementById(“***id value***”)**
  - **let** *els* **= document.getElementsByTagName(“**tag**”)**
    Or
  - **let** *els* **= *element*.getElementsByTagName(“**tag**”)**

    Syntax:
    let *els* **= document.getElementsByClassName(“***class name***”)**
    Or
  - **let** *els* **= *element*.getElementsByClassName(“***class name***”)**
  - CSS Selector :

    Syntax:
    let *els* **= document.querySelectorAll(“***CSS selector syntax***”)**
    Or
  - **let** *els* **= *element*.querySelectorAll(“***CSS selector syntax***”)**

    Syntax:
    let *el* **= document.querySelector(“***CSS selector syntax***”)**
    Or
  - **let** *el* **= *element*.querySelector(“***CSS selector syntax***”)**

- Setting style - either ele.setattr or ele.style.font

```
let par = document.getElementsByTagName("p")[0]; //Reference to first paragrap
par.style.fontSize = "18pt"; //CSSDeclarationObject: Set font size to 18 point
let strCSS = "color: blue; background-color: lightgray;"; //CSS string having
let hl = document.getElementsByTagName("h1")[0]; //Reference to h1 header
hl.setAttribute("style", strCSS); //Set CSS string to style attribute of h1 he
```

  - To get current config :

```
let par = document.getElementsByTagName("p")[0]; //Reference to first parag
par.style.fontSize = "18pt"; //CSSDeclarationObject: Set font size to 18 po
par.style.width = "60%"; //Set width of paragraph to 60%
console.log("CSS Computed width = " + window.getComputedStyle(par).width);
console.log("CSS computed font size = " + window.getComputedStyle(par).font
```

  - To manipulate class styles of a element

Syntax:

element.**classList.add(***class1, class2***, ...***classn***)**
element.**classList.remove(***class1, class2***, ...***classn***)**
element.**classList.toggle(***classname***)**
element.**classList.contains(***classname***)**

- Timeout and animation

```
window.setInterval(fDisplaytime,1000); //Call function ever
//fDisplaytime(); //Does not work, html page not parsed yet
function fDisplaytime() {
    let clock = document.getElementById("clock");//Referenc
    let now = new Date(); //Initialize current date/time
    clock.innerHTML = now.toLocaleTimeString(); //Display c
}
```

  - In setinterval give function
  - Request animation frame for efficiency

```
let requestID = requestAnimationFrame(fMoveSquare); //Calli
function fMoveSquare() { //Nested function
    leftPos += 2; //Increment position value by 2
    console.log("Pos = " + leftPos); //Output position valu
    //Defining end condition, when square left position rea
    if (leftPos > intPathWidth - intSquareWidth) { //This i
        window.cancelAnimationFrame(requestID); //Cance
    } else { //Did not reach end yet
        square.style.left = leftPos + "px"; //Set left attr
        requestID = requestAnimationFrame(fMoveSquare); //C
```

  - To utilise event element

```
window.addEventListener("load", function() {
    output = document.getElementById("objEvent"); //Reference to textarea
    //Register event handler function on entire html page using body tag
    document.body.addEventListener("click", function(evt) {fEvent(evt, output)} );
})
function fEvent(evt, pElt) { //Event listener
    pElt.cols = "50"; //Set number of columns of textarea
    pElt.rows = "4"; //Set number of rows of textarea
    strOutput = "Element passed = " + pElt.tagName + "\n"; //Display element passed
    strOutput += "Coordinates x and y = " + evt.clientX + ', ' + evt.clientY + "\n"; //Display mouse coordinates
    strOutput += "Target (element clicked) = " + evt.target.tagName + "\n"; //Display target tag
    strOutput += "Current target (event handler registered) = " + evt.currentTarget.tagName; //Display current target tag
    pElt.value = strOutput;
}
```

  - Anonymous function

```
let name = prompt('Enter your name ');
btnAddEvt.addEventListener("click", function() {fAddEvt(name);}); /
```

- Prevent default :

```
window.addEventListener("load", function() { //DOM is ready
    //Register key down event handler input element
    document.getElementsByTagName('input')[0].addEventListener('keydown', fKeyDown)
})

function fKeyDown(evt) {
    //ASCII codes for numeric characters: 48 - 57
    console.log(evt.keyCode + ' = ' + String.fromCharCode(evt.keyCode))
    if (48 <= evt.keyCode && evt.keyCode <= 57) { //Prevent numbers fro
    //if (48 <= evt.keyCode <= 57) --> does not work properly, yet no e
        console.log('numeric');
        evt.preventDefault(); //Prevent default action, subsequent even
    }
```