Processing Islands in the Enterprise

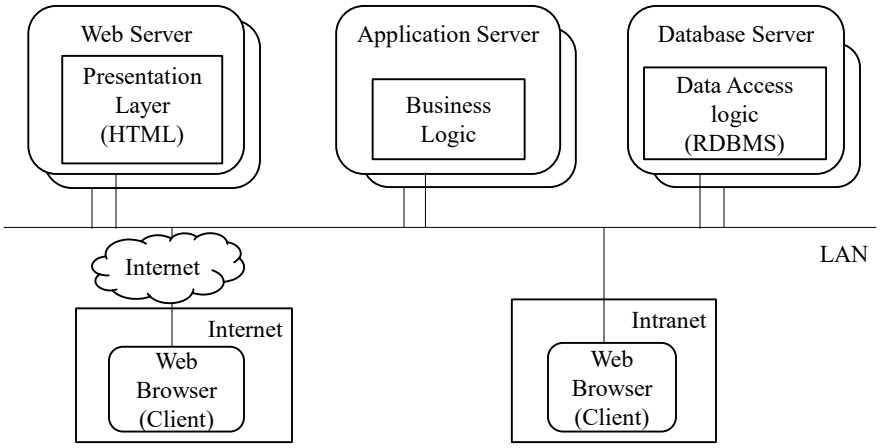
- **Data Processing** – the core IT systems that control the fundamental business processes in an organization.
- **Personal Productivity** – Employees works with information using various tools.
- **Collaboration** – the use of communications and GroupWare software that enables both organizations and individuals to work in a partnership and teams.

PC Networking: Moving Towards Decentralization

- **Client/Server Computing**
- Client/Server solutions involve the following independent layers:
  - **Presentation Logic** – how the user interacts with application
  - **Business Logic** – Everyday functions of the business
  - **Data Storge** – Handles the storage and retrieval of data

System Architecture

- **Web Technologies: Centralized Computing**

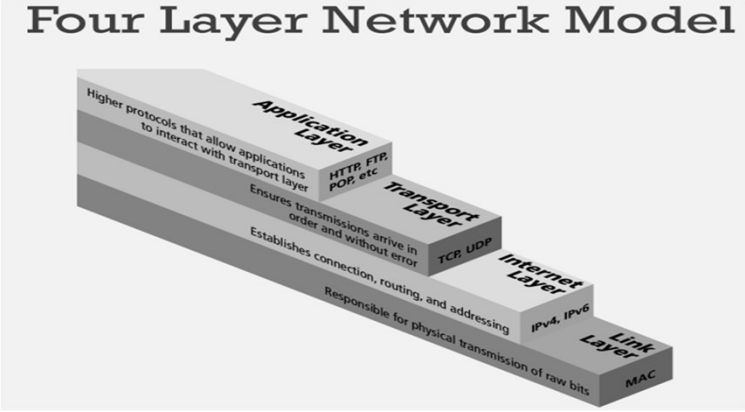


Web Technology Fundamentals

- **ARPANET**
  - ***A**dvanced **R**esearch **P**rojects **A**gency **N**etwork*
- **TCP/IP Networking**
  - **Internet Protocol** – It is used to route a packet of data from node to node across the network.
  - **Transmission Control Protocol** – It is used to create a point-to-point communication channel and ensures that the information is delivered error free and in correct order that it was originally transmitted.

Web Technology Fundamentals

- **TCP/IP Protocol Stack**



Web Technology Fundamentals

- **Port Numbers** – It is used to distinguish between the individual networking applications that are running simultaneously above the TCP/IP protocol stack.
  - Port Numbers for standard TCP/IP services examples:
    - 80 – Hypertext Transfer Protocol (HTTP) …Web/WWW
    - 21 - File Transfer Protocol (FTP) … File Transfer
- **Uniform Recourse Locator (URL)** – All information over the internet can be accessed, using a reference called URL.

Web Technology Fundamentals

- **Uniform Recourse Locator**
  - <protocol> : / / <machine id> / <local name>

    - *<protocol>* : identifies the application protocol used to retrieved the source.
    - <machine id> : identifies the server on which the resource is located.
    - <local name> identifies the resource, and that can include a directory structure.

Web Technology Fundamental

- **The World Wide Web (WWW)**

- Web is not a network but an application that operates over network using TCP/IP protocols.
- WWW is a service used world over for information sharing
- Introduced in 1990 by Tim Berners Lee
- One of the services available over Internet

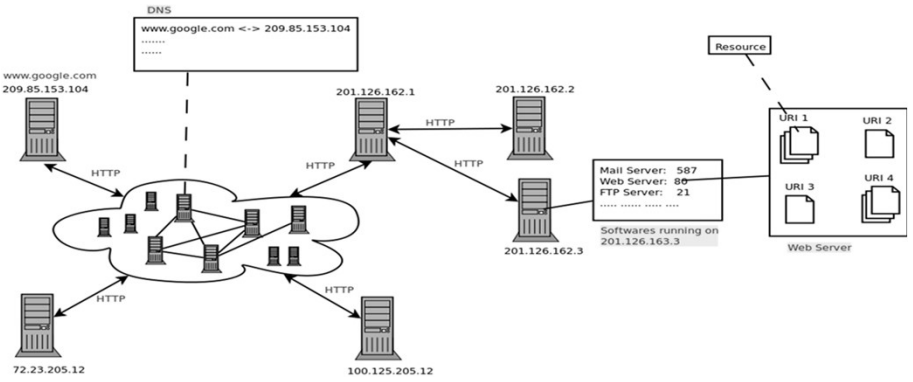
Tim Berners Lee

Web Technology Fundamental (WWW service over internet)

- Comprises of a client-server architecture
- Three major components
  - Web Server
  - Agent / Browser
  - HTTP Protocol
- Early contributing organization
  - CERN
- Current contributing organization
  - W3C

Web Technology Fundamental (WWW as Service internet)

- **Hyper Text Transfer Protocol (HTTP)** – protocol used to get the required document using HTTP request and response message.

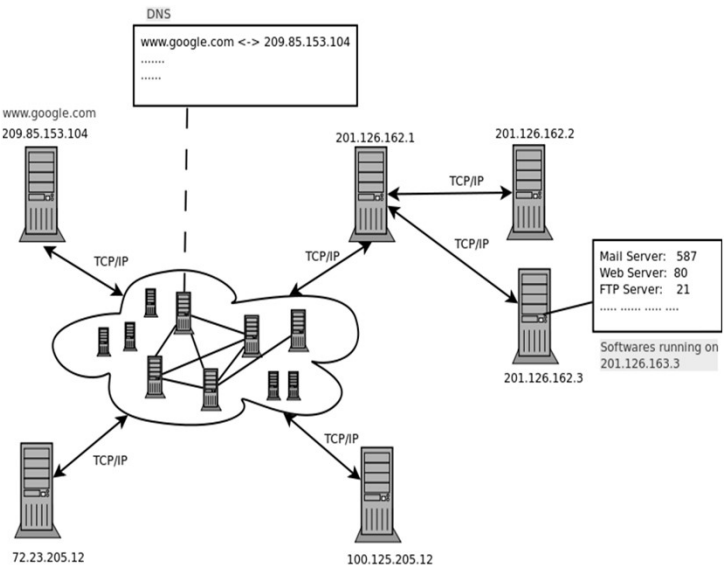


Internet

- **Internet** is a global network of computers communicating through **internet** over diverse underlying infrastructure
- The core enabler of internet is TCP/IP protocol suite worked out by Robert Kahn and Vint Cerf in 1973
- Early contributing organizations:
  - DARPA,
  - Stanford Research Institute
  - CERN
- Current organizations:
  - IETF
  - ICANN

Robert Kahn

Vint Cerf



Browser

- Browser function is to:
  - Fetch web content
  - Present web content
  - Efficient rendering
  - Secure communication
  - Support extensibility
- Early browsers
  - WorldWideWeb
  - Mosaic
  - Netscape Navigator
- Modern browsers
  - Google Chrome
  - Microsoft Internet Explorer

Mark Andreesen

HTML Document - Table

- An HTML table is defined with the **<table>** tag.
- Each table row is defined with the **<tr>** tag. A table header is defined with the **<th>** tag.
- By default, table headings are bold and centered. A table data/cell is defined with the **<td>** tag.

Inline Styles

Style rules placed within an HTML element via the style attribute

```
<h1>Share Your Travels</h1>
<h2>style="font-size: 24pt"Description</h2>
...
<h2>style="font-size: 24pt; font-weight: bold;">Reviews</h2>
```

LISTING 3.1 Internal styles example

An inline style only affects the element it is defined within and will override any other style definitions for the properties used in the inline style.

Using inline styles is generally discouraged since they increase bandwidth and decrease maintainability.

Inline styles can however be handy for quickly testing out a style change.

Embedded Style Sheet

Style rules placed within the <style> element inside the <head> element

```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <style>
      h1 { font-size: 24pt; }
      h2 {
       font-size: 18pt;
       font-weight: bold;
       }
   </style>
</head>
<body>
   <h1>Share Your Travels</h1>
   <h2>New York - Central Park</h2>
   ...
```

LISTING 3.2 Embedded styles example

While better than inline styles, using embedded styles is also by and large discouraged.

Since each HTML document has its own <style> element, it is more difficult to consistently style multiple documents when using embedded styles.

External Style Sheet

Style rules placed within a external text file with the .css extension

```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels -- New York - Central Park</title>
   <link rel="stylesheet" href="styles.css" />
</head>
```

LISTING 3.3 Referencing an external style sheet

This is by far the most common place to locate style rules because it provides the best maintainability.

- When you make a change to an external style sheet, all HTML documents that reference that style sheet will automatically use the updated version.
- The browser is able to cache the external style sheet which can improve the performance of the site

The History of the Kumquat

The origins of the kumquat are shrouded in mystery. Little is known of this wondrous fruit prior to its discovery by Spanish explorers in the early 15th century. Even then, those who attempted to trace the origins of the fruit met with resistance, misdirection, and even death at the hands of the North American natives who jealously guarded their "quom-te-cotl" (literally, the *fruit of life*).

although we must disregard reports that kumquat rinds were found amid the wreckage of the well-documented Roswell spacecraft crash. Still,

the fact that the kumquat has no seeds and must be propogated by hand leaves no other conclusion except that the

The Noble Fruit

Solution

```
<table border=0 cellspacing=7>
  <tr>
    <th colspan=5><h2>The History of the Kumquat</h2>
  <tr valign=top>
    <td rowspan=2>Copy for column 1...
    <td rowspan=2 width=24><br>
    <td >Copy for column 2...
    <td width=24><br>
    <td >Copy for column 3...
  <tr>
    <td colspan=3 align=center><img src="pics/fruit.gif">
    <p>
    <i>The Noble Fruit</i>
</table>
```

Two Main Categories of HTML 5 Form:

- **Search Forms**
  - A forms that helps the visitors to search something from the website.
- **Data-Collecting Forms**
  - A forms that collects some necessary data and information from the visitors.

Components of form

<input type="text">	Displays a single-line text input field
<input type="radio">	Displays a radio button (for selecting one of many choices)
<input type="checkbox">	Displays a checkbox (for selecting zero or more of many choices)
<input type="submit">	Displays a submit button (for submitting the form)
<input type="button">	Displays a clickable button

Data Collecting Forms – Example 1

- **Options:**
  - <select>
  - <option>Selection an option</option>
    - <option>Option 1</option>
    - <option>Option 2</option>
    - <option>Option 3</option>
  - </select>

Data Collection Forms - Example 3

```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset = “utf-8”>
    <title>Creating Forms</title>
<head>
<body>
    <h1>How to Create Forms in HTML 5</h1>
    <form action=sunflower.jpg>
    <label for="fname">First name:</label><br>
     <input type="text" id="fname" name="fname" value=Ali><br>
     <label for="lname">Last name:</label><br>
     <input type="text" id="lname" name="lname" value=Ahmad><br><br>
     <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Sample Document - Table

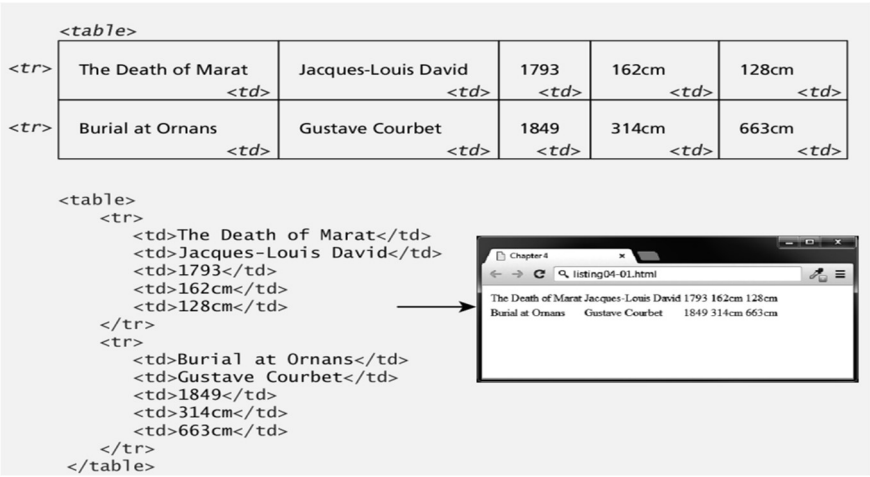


Table with Heading

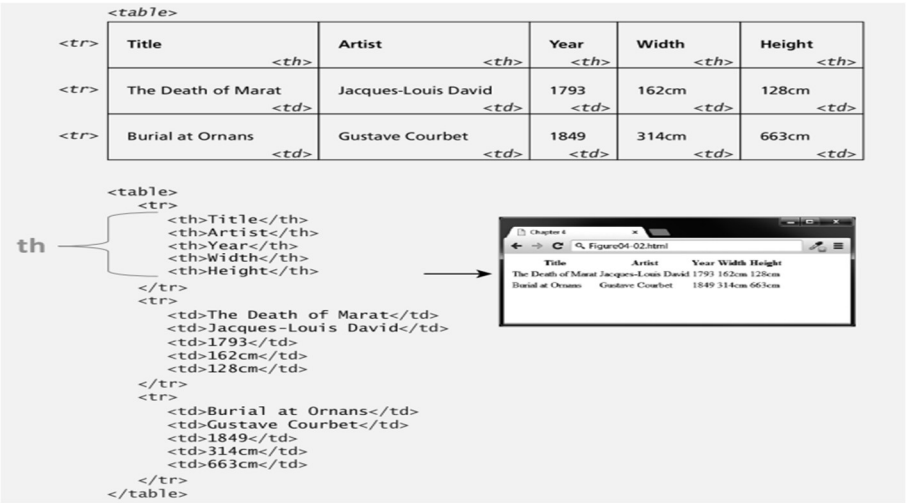
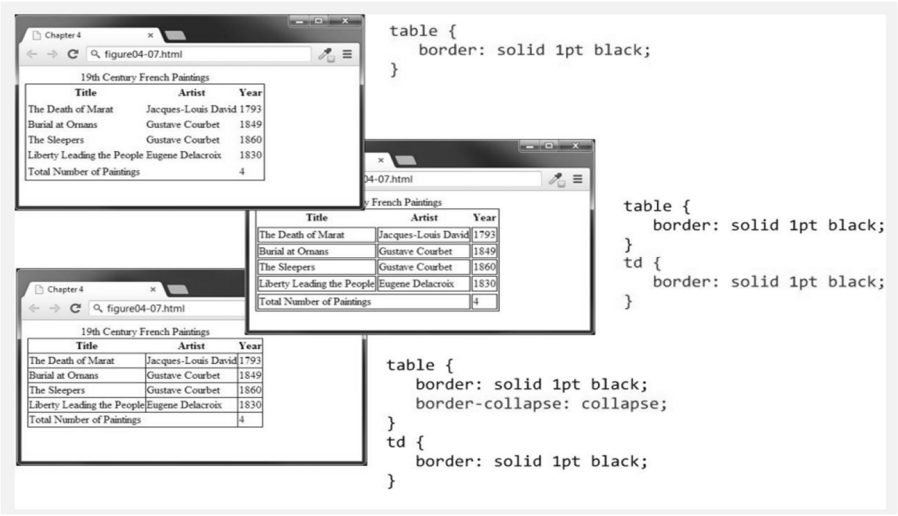
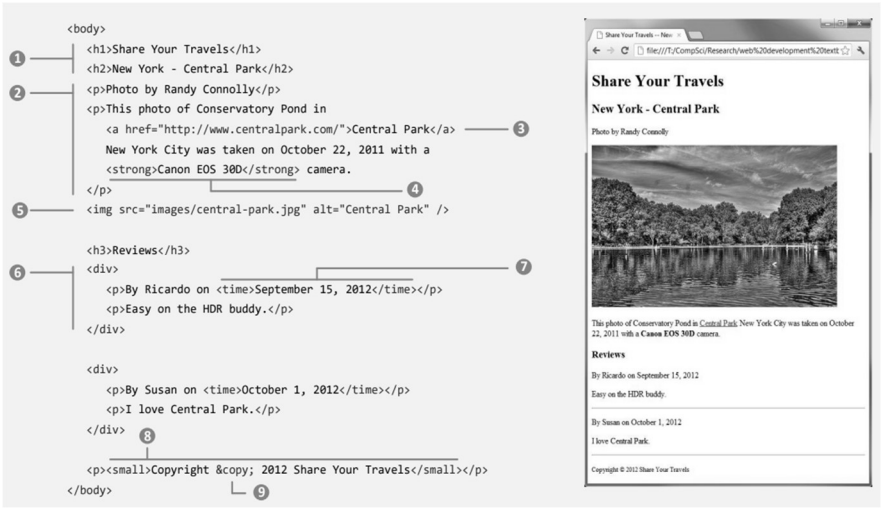


Table – Styling Borders



Sample Document



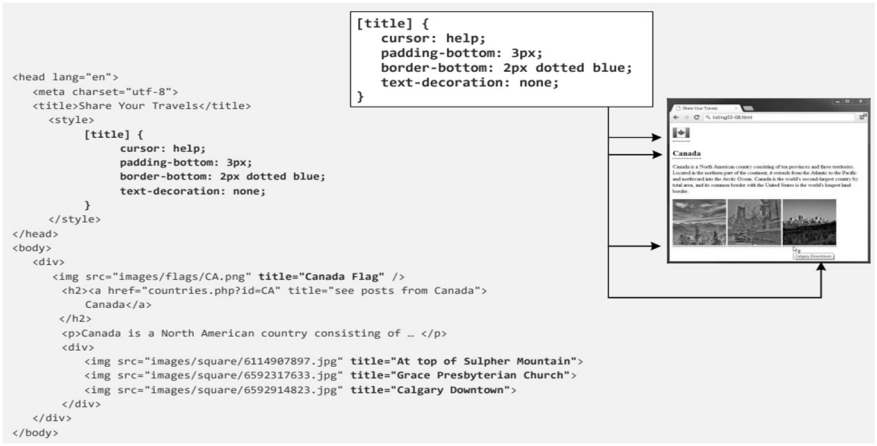
TASK

Artist	Title	Year
Jacques-Louis David <td rowspan=3>	The Death of Marat	1793
	The Intervention of the Sabine Women	1799
	Napoleon Crossing the Alps	1800

Solution

```
<table>
   <tr>
      <th>Artist</th>
      <th>Title</th>
      <th>Year</th>
   </tr>
   <tr>
      <td rowspan="3">Jacques-Louis David</td>
      <td>The Death of Marat</td>
      <td>1793</td>
   </tr>
   <tr>
      <td>The Intervention of the Sabine Women</td>
      <td>1799</td>
   </tr>
   <tr>
      <td>Napoleon Crossing the Alps</td>
      <td>1800</td>
   </tr>
   ...
</table>
```

Attribute Selector



```
[title] {
   cursor: help;
   padding-bottom: 3px;
   border-bottom: 2px dotted blue;
   text-decoration: none;
}
```

```
<head lang="en">
   <meta charset="utf-8">
   <title>Share Your Travels</title>
      <style>
            [title] {
                  cursor: help;
                  padding-bottom: 3px;
                  border-bottom: 2px dotted blue;
                  text-decoration: none;
            }
      </style>
</head>
<body>
   <div>
      <img src="images/flags/CA.png" title="Canada Flag" />
       <h2><a href="countries.php?id=CA" title="see posts from Canada">
           Canada</a>
       </h2>
       <p>Canada is a North American country consisting of … </p>
       <div>
           <img src="images/square/6114907897.jpg" title="At top of Sulpher Mountain">
           <img src="images/square/6592317633.jpg" title="Grace Presbyterian Church">
           <img src="images/square/6592914823.jpg" title="Calgary Downtown">
       </div>
   </div>
</body>
```

Pseudo Selector

```
<head>
   <title>Share Your Travels</title>
   <style>
         a:link {
         text-decoration: underline;
         color: blue;
      }
         a:visited {
         text-decoration: underline;
         color: purple;
      }
         a:hover {
         text-decoration: none;
         font-weight: bold;
      }
         a:active {
         background-color: yellow;
      }
   </style>
</head>
<body>
      <p>Links are an important part of any web page. To learn more about
         links visit the <a href="#">W3C</a> website.</p>
    <nav>
      <ul>
         <li><a href="#">Canada</a></li>
         <li><a href="#">Germany</a></li>
         <li><a href="#">United States</a></li>
      </ul>
    </nav>
</body>
```

Bootstrap 5

- Adding bootstrap via **CDN** (Content Delivery Network).
- It reduces loading time, as they are hosting files on multiple servers spread across the globe.
- When a user request for file, it will be served from the nearest server around them.

Bootstrap 5

- **Integrity**
- **Crossorigin**

- The attributes integrity and crossorigin have been added to CDN links to implement **Subresource Integrity** (SRI).

Bootstrap 5

- **Bootstrap files**
  - We can also download the bootstrap files from their official website.
  - The complied minified version of CSS and JavaScript files are used for faster and easier web development.
  - The minified version of CSS and JS files will improve performance of the website.
  - It will also save the bandwidth due to less HTTP request and response.

CSS Layout – Position

- The position is CSS sets how an element is positioned in a document.

- **Static Position**
- **Relative Position**
- **Absolute Position**
- **Fixed Position**

Static Position

- It is by default position for HTML elements.
- Positions the element according to the normal flow of the page.
- Not effected by top, bottom, left and right properties.

```
#rose
    {
        position: static;
    }
```

Relative Position

- In relative position the element is positioned relative to its normal.

```
#rose
    {
        position: relative;
    }
```

Absolute Position

- It is used to position an element relative to the first element that has a position other than the static.
- In absolute position the element position is changed relative to its parent.

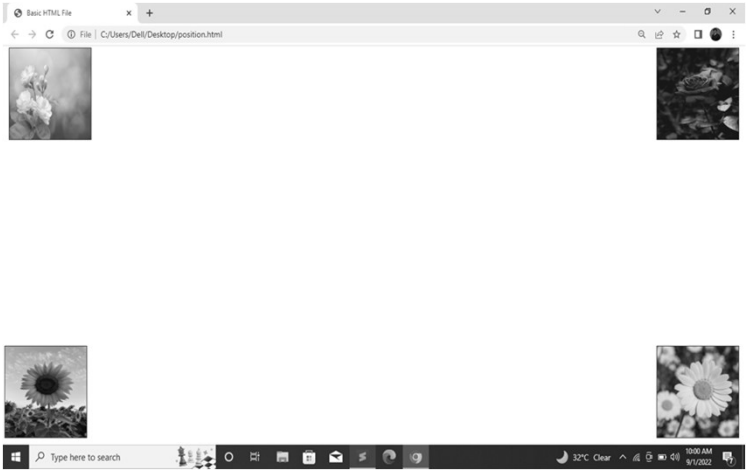
```
#rose
    {
        position: absolute;
    }
```

Fixed Position

- The element will stay in same place, even the page is scrolled

```
#rose
    {
        position: fixed;
            top: 6px;
    }
```

Task - write a program to display similar output on a browser.



Solution

```
#rose
      {
            position: fixed;
            right: 0px;
      }

#jasmine
      {
         position: fixed;
         left: 0px;
      }
```

```
#daisy
      {
         position: fixed;
         bottom: 0px;
         right: 0px;
      }

#sunflower
      {
         position: fixed;
         bottom: 0px;
      }
```

Inline JavaScript

```
<a href="#" onClick="(function()
        {
            alert('Hey I am Inline JavaScript');
        return false;
        }
    )();return false;">click here</a>
```

External JavaScript

```
<script type="text/javascript“
src="greeting.js"></script>


<form>
<input type="button" value="click"
onclick="msg()"/>
</form>
```

JavaScript's Reputation

- Everything is type sensitive, including function, class, and variable names.
- The scope of variables in blocks is not supported. This means variables declared inside a loop may be accessible outside of the loop, counter to what one would expect.
- There is a === operator, which tests not only for equality but type equivalence.
- **Null** and **undefined** are two distinctly different states for a variable.
- Semicolons are not required, but are permitted (and encouraged).
- There is no integer type, only number, which means floating-point rounding errors are prevalent even with values intended to be integers.

Comparison Operator

Operator	Description	Matches (x=9)
==	Equals	(x==9) is true (x=="9") is true
===	Exactly equals, including type	(x==="9") is false (x===9) is true
<, >	Less than, Greater Than	(x<5) is false
<=, >=	Less than or equal, greater than or equal	(x<=9) is true
!=	Not equal	(4!=x) is true
!==	Not equal in either value or type	(x!=="9") is true (x!==9) is false

Functions

```
function power(x,y)
{
    var pow=1;
    for (var i=0;i<y;i++)
    {
        pow = pow*x;
    }
return pow;
}
```

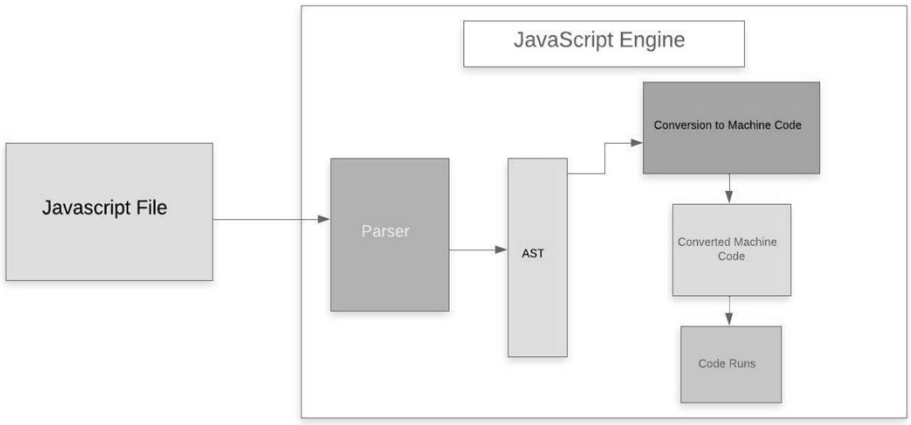
Alert

- The alert() function makes the browser show a pop-up to the user, with whatever is passed being the message displayed. The following JavaScript code displays a simple hello world message in a pop-up:
- **alert ("Good Morning");**
- Using alerts can get tedious fast. When using debugger tools in your browser you can write output to a log with:
- **console.log("Put Messages Here");**
- And then use the debugger to access those logs.

How JavaScript runs behind the browser?

- **Parser:** A Parser or Syntax Parser is a program that reads your code line-by-line. It understands how the code fits the syntax defined by the Programming Language and what it (the code) is expected to do.
- **JavaScript Engine:** A JavaScript engine is simply a computer program that receives JavaScript source code and compiles it to the binary instructions (machine code) that a CPU can understand. JavaScript engines are typically developed by web browser vendors, and each major browser has one. Examples include the V8 engine for Google chrome, SpiderMonkey for Firefox, and Chakra for Internet Explorer.

How JavaScript Works?



Execution Context

- The **Execution Context** contains the code that's currently running, and everything that aids in its execution.
- Execution Context is an environment in which our code is executed.
- **Global Execution Context**
  - By Default
- JavaScript engine creates the global execution context before it starts to execute any code
- Variables and function that is not inside any function.

Execution Context

- A **New execution context** gets created every time a function is executed.
- A **Global Object** is also known as window object.
- For Example
  - name == window.name
  - This === window

Execution Stack

- Execution Stack is also known as **calling stack**
- It is used to store all the **execution** context created during the code execution.
- https://www.freecodecamp.org/news/execution-context-how-javascript-works-behind-the-scenes/#:~:text=First%2C%20the%20script%20is%20loaded,%2C%20second%20%2C%20and%20third%20functions.

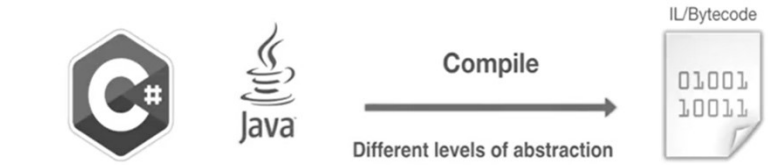
Execution Context - Properties

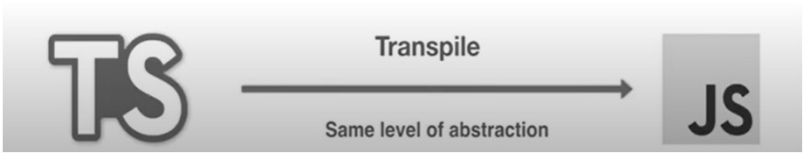
- **Creation Phase**
  - Variable Object are created
  - Scope Chain are created
  - 'this' variable
- **Execution Phase**
  - Execution context run line by line

Variable Object

- Property created in variable object
  - **Which is pointing to functions.(for function declarations)**
  - **Property created for variables but set to undefined. .(for variable declarations)**
  - **Argument object** are created to pass as argument in a function

Compiler Vs Transpiler





External Script

- **window.onload = writeMessage;**
- **window.onload** is an event handler. Events are actions that a users performs while visiting a web page. Submitting a form or moving a mouse over a image are examples of events
- After equal sign there is the name of a function, **writeMessage.**

External Script

- **Document.getElementById(“hMsg”).innerHTML = “Hello World!!”;**
- Taking the string “Hello World!!”, and put it into the document, inside the element on the page named **hMsg.**

Task

- Write a program that redirects the user to a page by clicking on a href. But before the browsers loads a page, it generates an alert.

Solution

```
function initRedirect()
{
   alert("The content of other web
page");
   window.location = this;
   return false;
}
```

What Is “this”?

- **window.location = this;**

This line allows us to set the browser window to the location specified by the keyword **this,** which contains the link. The JavaScript keyword **this** allows the script to pass a value to a function, based on the factor where the keyword is used.

In our case, **this** is used inside a function triggered by an event attached to a tag, so here, **this** is a link object.

Task

- Write a program having three buttons named by some famous personalities. Whenever the user click any of the button, an alert appear with quote of that person.

Solution

```
<body>
    <form action="#">
        <input type="button" id="Jinnah" value="Jinnah">
        .
        .
        .
    </form>
</body>
```

Solution

```
window.onload = initAll;
function initAll()
{
    document.getElementById("Jinnah").onclick = quotes;
    .
    .
    .
}
```

Solution

```
function quotes()
{
    switch (this.id)
    {
        case "Jinnah":
            alert("Failure is a word unknown to me");
            break;
            .
            .
            .
    }
}
```

Handling Events

- Events are actions that the users performs while visiting your page. When the browser detects an event, such as mouse click or a key press, it can trigger JavaScript objects associated with the event called event handler.

Handling Window Events

- **The onload event**
  - What to do, when you need to do multiple things to happen when the pages gets uploaded.
  - We want two more things to happen when the first page gets loaded.
  - Setting the **window.onload** wont work, because it will override the previous settings.
  - Instead we will call a new function **addOnLoad(),** which handles the onload handler for us.

The onload event

```
<html>
    <head>
        <title>welcome !</title>
        <script src="script01.js"></script>
    <head>
        <body id="pagebody">
            <h1>Welcome to our Web Site</h1>
        </body>
</html>
```

The onload event

```
addOnLoad(initOne);
addOnLoad(initTwo);

function addOnLoad(newFunction)

    var oldOnLoad = window.onload;
    if (typeof oldOnLoad == "function")
    {
        window.onload = function()
        {
            oldOnload();
            newFunction();
        }
    }
    else
    {
    window.onload = newFunction;
    }
}
```

The onload event

```
function  initOne()
{
    Document.getElementById("pageBody").style.backgroundC
    olor = "#00F";
}


function  initTwo()
{
    Document.getElementById("pageBody").style.color =
    "#F00";
}
```

The onload event

- **addOnLoad(initOne);**

- It handles the **onload** handler for us.
- For each call, we are passing one parameter: the name of the function we want to run when an onload event is triggered.

The onload event

- **function addOnLoad(newFunction)**

Instead of calling:

window.onload = myNewFunction;

We will call:

addOnload (myNewFuncytion);

- if (typeof oldOnLoad = "function")

We check to see what kind of variable **oldOnLoad** is. If we have set **window.onload,** it will be a function call.

The onload event

```
window.onload = function()
        {
            oldOnload();
            newFunction();
        }
```

These lines of code reset the value of **window.onload** to do two things:

Whatever it was doing before, and our new function. The **window.onload** event handler is set to be anonymous function. Then, we tell **window.onload** to do what it was already doing. But before the function ends, we add that it needs to also do our newFunction() as well.

The onload event

```
else
    {
    window.onload = newFunction;
    }
```

If the **oldOnload** wasn't a function (that is, it was undefined). We tell it to do our new function when the page complete loading.

We call addOnload() multiple times: the first time it assigns its function to **window.onload;** the second time it creates that anonymous function, telling the javaScript to do everything tis been told to do previously and the new things as well.

Task

- Write a program to find the frequency of an int type data from an array, The onload event.

More Event Handlers

- **The onbeforeunload event**
  - **Onbeforeunload –** triggered after the user has left the page
- **The unload event**
  - **Onunload** – when user leaves the page

Introduction & Prerequisites

- A JavaScript library
- Easy to learn and implement
- jQuery is a fast and concise JavaScript Library that simplifies
  - HTML document traversing
  - Event handling
  - Animation
  - AJAX Interaction for rapid web development
- Prerequisites
  - HTML
  - CSS
  - JavaScript

What is jQuery?

- A library of JavaScript Functions.
- A lightweight "write less, do more" JavaScript library.
- The jQuery library contains the following features:
  - HTML element selections
  - HTML element manipulation
  - CSS manipulation
  - HTML event functions
  - JavaScript Effects and animations
  - HTML DOM traversal and modification
  - AJAX
  - Utilities
- An open source project, maintained by group of developers with active support base and well written documentation

What is available with jQuery

- Cross Browser support
- AJAX Functions
- CSS functions
- DOM Manipulation
- DOM Traversing
- Attribute Manipulation
- Event detection and handling
- JavaScript Animation
- Hundreds of plug-ins for pre-built user interfaces, advanced animation and form validation etc ...
- Expandable using custom plug-ins
- Small footprint

How to use jQuery?

- A single JavaScript file, containing all the jQuery methods.
- Add the following code into your html/jsp page and call the jQuery APIs.

```
<head>
<script type="text/javascript" src="jquery.js"></script>
</head>
```

- For Example

```
<html>
<head>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript">
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
</script>
</head>

<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me</button>
</body>
</html>
```

How to use jQuery?Cont…

- In order to use jQuery you need to load it.
- You can include it locally on your own server:
  - <script src="/js/jquery.js">
- Or use one of the CDN's made available:
  - ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js
  - ajax.microsoft.com/ajax/jquery/jquery-1.4.2.js
  - CDN's are Gzipped and minified

jQuery Syntax

- With jQuery you select (query) HTML elements and perform "actions" on them.
- **jQuery Syntax Examples**
  - $(this).hide()
  - $("#test").hide()
  - $("p").hide()
  - $(".test").hide()
- Basic syntax is: **$(selector).action()**
  - A dollar sign to define jQuery
  - A (selector) to "query (or find)" HTML elements
  - A jQuery action() to be performed on the element(s)

jQuery Selectors

- To select HTML elements (or groups of elements) by element name, attribute name or by content.

- **jQuery Element Selectors**
  - Uses CSS selectors to select HTML elements.
    - $("p“)
    - $("p.intro“)
    - $("p#demo“)

jQuery Selectors

- **jQuery Attribute Selectors**
  - jQuery uses XPath expressions to select elements with given attributes.
  - $("[href]“)
  - $("[href='#']")
  - $("[href!='#']“).
  - $("[href$='.jpg']“)
- **jQuery CSS Selectors**
  - jQuery CSS selectors can be used to change CSS properties for HTML elements.
  - For Ex :
    - $("p").css("background-color","yellow");

jQuery Selectors

- Few samples for the selectors

Syntax	Description
$(this)	Current HTML element
$("p")	All <p> elements
$("p.intro")	All <p> elements with class="intro"
$(".intro")	All elements with class="intro"
$("#intro")	The first element with id="intro"
$("ul li:first")	The first element of each
$("[href$='.jpg']")	All elements with an href attribute that ends with ".jpg"
$("div#intro .head")	All elements with class="head" inside a element with id="intro"

jQuery Filters

- First paragraph – p:first
- Last list item – li:last
- Fourth link – a:nth(3)
- Fourth Div – div:eq(3)
- Every other Paragraph – p:odd or p:even
- Every link after/upto 4th – a:gt(3) or a:lt(4)
- Links that contain word like click – a:contains("click")
- All radio inputs with in first form - $("input:radio", documents.forms[0])

jQuery Attributes

- Read
  - $(#image).attr("src");
- Set
  - $(#image).attr("src" , "images/jquery1.jpg");
- Multiple Set
  - $(#image).attr({src: "images/jquery1.jpg" , alt: "jQuery"});
- Set Class
  - $(p:last).addClass("selected");
- Read/Set Html
  - $(#id).html() & $(#id).html("value");

jQuery Events

- The jQuery event handling methods are core functions in jQuery.
- For Example
  - <html>
    <head>
    <script type="text/javascript" src="jquery.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){
     $("button").click(function(){
      $("p").hide();
     });
    });
    </script>
    </head>

    <body>
    <h2>This is a heading</h2>
    This is a paragraph.
    This is another paragraph.
    <button>Click me</button>
    </body></html>

jQuery Callback Functions

- A callback function is executed after the current animation (effect) is finished.
- Syntax
  - $(selector).hide(speed,callback)
- For Ex:
  - $("p").hide(1000,function(){
    alert("The paragraph is now hidden");
    });

jQuery HTML Manipulation

- jQuery can manipulate the HTML elements and attributes
- **Changing HTML Content**
  - $(selector).html(content)
- **Adding HTML content**
  - **$(selector).append(content)**
  - **$(selector).prepend(content)**
  - **$(selector).after(content)**
  - **$(selector).before(content)**

jQuery HTML Manipulation

- jQuery HTML Manipulation Methods

Function	Description
$(selector).html(content)	Changes the (inner) HTML of selected elements
$(selector).append(content)	Appends content to the (inner) HTML of selected elements
$(selector).after(content)	Adds HTML after selected elements

jQuery CSS Manipulation

- **jQuery css() Method**
  - css(name) - Return CSS property value
  - css(name,value) - Set CSS property and value
  - css({properties}) - Set multiple CSS properties and values
- Examples
  - Return CSS Property
    - $(this).css("background-color");
  - Set CSS Property & value
    - $("p").css("background-color"," yellow");
  - Set Multiple values in CSS
    - $("p").css({"background-color":"yellow","font-size":"200%"});
  - Size Manipulation
    - $("#div1").height("200px"); $("#div2").width("300px");

jQuery CSS Manipulation

- jQuery CSS Manipulation Methods

CSS Properties	Description
$(selector).css(name)	Get the style property value of the first matched element
$(selector).css(name,value)	Set the value of one style property for matched elements
$(selector).css({properties})	Set multiple style properties for matched elements
$(selector).height(value)	Set the height of matched elements
$(selector).width(value)	Set the width of matched elements

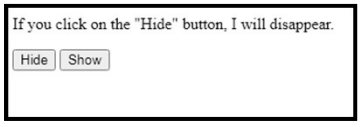
Hide/Show Example

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#hide").click(function(){
    $("p").hide();
  });
  $("#show").click(function(){
    $("p").show();
  });
});
</script>
</head>
<body>

<p>If you click on the "Hide" button, I will disappear.</p>

<button id="hide">Hide</button>
<button id="show">Show</button>

</body>
</html>
```



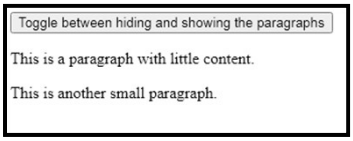
Toggle Example

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").toggle();
  });
});
</script>
</head>
<body>

<button>Toggle between hiding and showing the paragraphs</button>

<p>This is a paragraph with little content.</p>
<p>This is another small paragraph.</p>

</body>
</html>
```



jQuery Fading Effects

- With jQuery you can fade elements in and out of visibility.
- Click to fade in/out panel

- Examples:
- jQuery fadeIn()
  Demonstrates the jQuery fadeIn() method.
- jQuery fadeOut()
  Demonstrates the jQuery fadeOut() method.
- jQuery fadeToggle()
  Demonstrates the jQuery fadeToggle() method.
- jQuery fadeTo()
  Demonstrates the jQuery fadeTo() method.

Fading Example

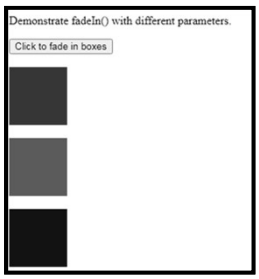
```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeIn("slow");
    $("#div3").fadeIn(3000);
  });
});
</script>
</head>
<body>

<p>Demonstrate fadeIn() with different parameters.</p>

<button>Click to fade in boxes</button><br><br>

<div id="div1" style="width:80px;height:80px;display:none;background-color:red;"></div><br>
<div id="div2" style="width:80px;height:80px;display:none;background-color:green;"></div><br>
<div id="div3" style="width:80px;height:80px;display:none;background-color:blue;"></div>

</body>
</html>
```



jQuery Effects - Sliding

- Examples
- jQuery slideDown()
  Demonstrates the jQuery slideDown() method.
- jQuery slideUp()
  Demonstrates the jQuery slideUp() method.
- jQuery slideToggle()
  Demonstrates the jQuery slideToggle() method.

Animate Example

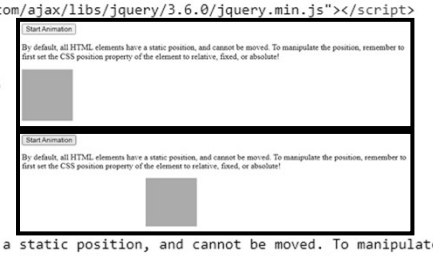
```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({left: '250px'});
  });
});
</script>
</head>
<body>

<button>Start Animation</button>

<p>By default, all HTML elements have a static position, and cannot be moved. To manipulate
the position, remember to first set the CSS position property of the element to relative,
fixed, or absolute!</p>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>

</body>
</html>
```



TASKS

- Make homepage slider with effects have following Elements:
- One Image
- Three Buttons (Fade In, Fade out, Animate the Width)
- Also Change BgColor by clicking on the above buttons (R,G,B)

Ajax

- AJAX = Asynchronous JavaScript and XML.
- In short; AJAX is about loading data in the background and display it on the webpage, without reloading the whole page.
- Examples of applications using AJAX: Gmail, Google Maps, Youtube, and Facebook tabs.

jQuery load() Method

- The jQuery load() method is a simple, but powerful AJAX method.
- The load() method loads data from a server and puts the returned data into the selected element.
- Syntax:
- $(selector).load(URL,data,callback);
- The required URL parameter specifies the URL you wish to load.
- The optional data parameter specifies a set of querystring key/value pairs to send along with the request.
- The optional callback parameter is the name of a function to be executed after the load() method is completed.
- The callback function can have different parameters:
- **responseTxt** - contains the resulting content if the call succeeds
- **statusTxt** - contains the status of the call
- **xhr** - contains the XMLHttpRequest object

Ajax example

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr){
      if(statusTxt == "success")
        alert("External content loaded successfully!");
      if(statusTxt == "error")
        alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
  });
});
</script>
</head>
<body>

<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>

<button>Get External Content</button>

</body>
</html>
```

Post/get ajax requests

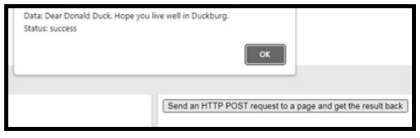
```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.post("demo_test_post.asp",
    {
      name: "Donald Duck",
      city: "Duckburg"
    },
    function(data,status){
      alert("Data: " + data + "\nStatus: " + status);
    });
  });
});
</script>
</head>
<body>

<button>Send an HTTP POST request to a page and get the result back</button>

</body>
</html>
```

Tip: Here is how the ASP file looks like ("demo_test_post.asp"):

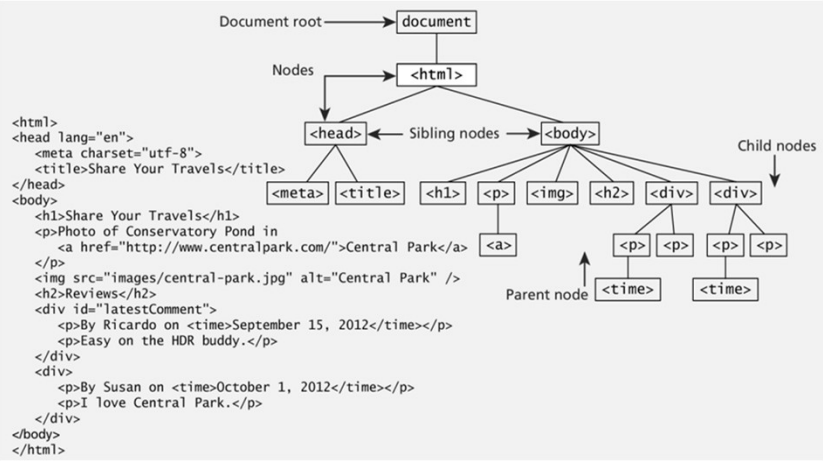
```
<%
dim fname,city
fname=Request.Form("name")
city=Request.Form("city")
Response.Write("Dear " & fname & ". ")
Response.Write("Hope you live well in " & city & ".")
%>
```



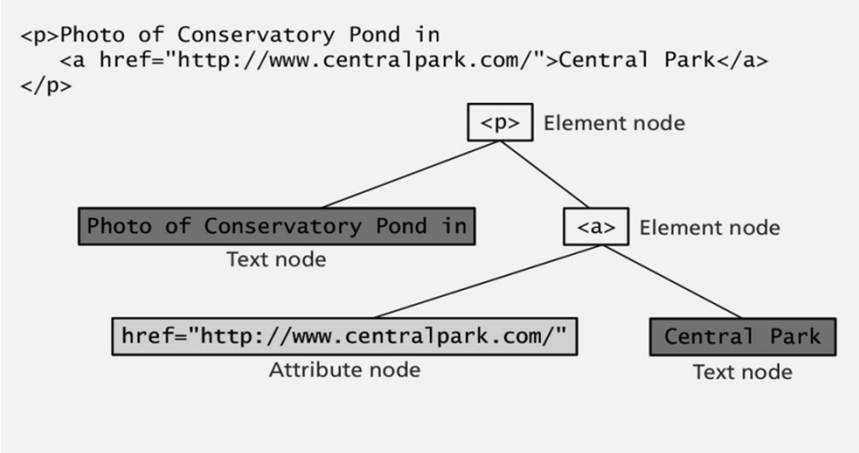
Document Object Model - DOM

- JavaScript is almost always used to interact with the HTML document in which it is contained.
- This is accomplished through a programming interface (API) called the **Document Object Model.**
- According to the W3C, the DOM is a:
- *Platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.*

DOM Tree



DOM Node



Adding Nodes

- **var newText = document.createTextNode(inText);**

We start by creating a new text node (called **newText**) using the **createTextNode()** method. This will contain whatever text was found in **textarea.**

- **var newGraf = document.createElement("p");**

We create a new element node using the **createElement()** method. The tag we are trying to create here is paragraph.

Adding Nodes

- **newGraf.appendChild(newText);**

In order to put the new text into the new paragraph, we have to call **appendChild().** That's a method of **newGraf,** Which, when passed **newText,** puts the text node into the paragraph.

- **evt.preventDefault();**

Calling the event not to run any default events.

Add Node

- **var docBody = document.getElementsByTagName("body")[0];**

In order to add a new node into the body of our document, we need to figure out where the body is. The **getElementsByTagName()** method gives us every **body tag** on our page.

- **docBody.appendChild(newGraf);**

Appending **newGraf** onto **docBody** (using **appendChild()** again) puts the user's new text onto the page.

Deleting Nodes

- **var allGrafs = document.getElementsByTagName("p");**

This line uses the **getElementsByTagName** method to collect all the paragraph tags in our page and store them in the **allGrafs** array.

- **if (allGrafs.length > 1)**

We check that **allGraf** length is greater than one. We don't want to delete
something that doesn't exists.

Deleting Nodes

- **var lastGraf = allGrafs[allGrafs.length-1];**

If there are paragraphs, get the last one on the page by subtracting one from the **length** and using that as our array index.

Subtracting one from the **length** gives us the last paragraph on the page.

Deleting Nodes

- **var docBody = document.getElementsByTagName("body")[0];**
- **docBody.removeChild(lastGraf);**

In order to modify the document we need to get the content of the **body.** Once we have got that, its simply a matter of calling the **docBody.removeChild()** method and passing it **lastGraf.**

Which tells JavaScript which paragraph we want to delete. Our page should immediately show one less paragraph

Deleting Specific Nodes

- **nodeChgArea = document.getElementById("modifiable");**

As our paragraph has multiple paragraphs, it could be confusing to keep track of which can cant be deleted. Now, we setup an entirely new area: a div with the **id** of **modifiable.** Here, we set the global variable **nodeChgArea** to that element node.

Deleting Specific Nodes

- **var grafChoice = document.getElementById("grafCount").selectedIndex;**
- **var allGrafs = nodeChgArea.getElementsByTagName("p");**
- **var oldGraf = allGrafs.item(grafChoice);**

We read the number from the **grafCount** field and store it in **GrafChoice.** The **allGrafs** variable is then set to be all the paragraph within the **nodeChangingArea,** and the paragraph to be deleted is then stored in **oldGraf.**

Deleting Specific Node

- **nodeChgArea.removeChild(oldGraf);**

When it run well see paragraphs disappear from the middle of our page.

- **var grafChoice = document.getElementById("grafCount").selectedIndex;**
- **var inText = document.getElementById("textArea").value;**

We need two things to know in order to insert a paragraph.

grafChoice : Where users want to insert.

intext: What text they want to insert.

Insert Nodes

- **var newText = document.createTextNode(inText);**
- **var newGraf = document.createElement("p");**

Creating a new paragraph and filling it with the users input.

- **var allGrafs = nodeChgArea.getElementsByTagName("p");**
- **var oldGraf = allGrafs.item(grafChoice);**

Once, We get all the **P** tags in our region. And then store the target paragraph in **oldGraf.**

Insert Node

- **nodeChgArea.insertBefore(newGraf,oldGraf);**

To insert the new paragraph , we call **insertBefore()** with two parameters: the new node and the existing node, that we want the new node to be inserted before.

To replace the node we will use all the code of **inserting a node,** except for

the last line. Which will be replace by the line below

- **nodeChgArea.replaceChild(newGraf,oldGraf);**

It takes two parameters: the paragraph we want to swap in and the paragraph

we want to swap out.

Folder Name: bootstrap

File Name: bootstrapExample.html

```
<!DOCTYPE html>

<html lang="en">
<head>
<title>Bootstrap 5 Example</title>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<!-- Bootstrap CSS -->
<link crossorigin="anonymous"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
rel="stylesheet"/>
<!-- Bootstrap JS Bundle with Popper -->
<script crossorigin="anonymous" integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"></sc
ript>
<!-- Bootstrap Offline -->
<!-- <link  rel="stylesheet" type="text/css" href="/bootstrap-5.0.2-
dist/css/bootstrap.min.css"> -->
<!-- <script type="text/javascript" href="/bootstrap-5.0.2-
dist/js/bootstrap.bundle.min.js"></script> -->
</head>
<body>
<nav class="navbar navbar-expand-lg bg-body-tertiary">
<div class="container-fluid">
<a class="navbar-brand" href="#">Navbar</a>
<button aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation" class="navbar-toggler" data-bs-target="#navbarSupportedContent" data-bs-
toggle="collapse" type="button">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarSupportedContent">
<ul class="navbar-nav me-auto mb-2 mb-lg-0">
<li class="nav-item">
<a aria-current="page" class="nav-link active" href="#">Home</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Link</a>
```

```
</li>
<li class="nav-item dropdown">
<a aria-expanded="false" class="nav-link dropdown-toggle" data-bs-toggle="dropdown"
href="#" role="button">
          Dropdown
        </a>
<ul class="dropdown-menu">
<li><a class="dropdown-item" href="#">Action</a></li>
<li><a class="dropdown-item" href="#">Another action</a></li>
<li><hr class="dropdown-divider"/></li>
<li><a class="dropdown-item" href="#">Something else here</a></li>
</ul>
</li>
<li class="nav-item">
<a class="nav-link disabled">Disabled</a>
</li>
</ul>
<form class="d-flex" role="search">
<input aria-label="Search" class="form-control me-2" placeholder="Search"
type="search"/>
<button class="btn btn-outline-success" type="submit">Search</button>
</form>
</div>
</div>
</nav>
<div class="container-fluid p-5 bg-primary text-white text-center">
<h1>My First Bootstrap Page</h1>
<p>This is a paragraph.</p>
</div>
<div class="container mt-5">
<div class="row">
<div class="col-sm-4">
<h3>Column 1</h3>
<p>The term random refers to any collection of data or information with no determined
order, or is chosen in a way that is unknown beforehand. </p>
<p>For example, 5, 8, 2, 9, and 0 are single-digit numbers listed in random order. Random
data may be re-ordered, or sorted, by date, name, time, age, etc., in which case its order is no
longer random.</p>
</div>
<div class="col-sm-4">
<h3>Column 2</h3>
<p>Data can be randomly selected, or random numbers can be generated using a random
seed.</p>
```

```
<p>Computer games, web pages, programs, and encryption are a few examples of things
that need random values to operate.</p>
</div>
<div class="col-sm-4">
<h3>Column 3</h3>
<p>Data can be randomly selected, or random numbers can be generated using a random
seed.</p>
<p>Computer games, web pages, programs, and encryption are a few examples of things
that need random values to operate.</p>
</div>
</div>
</div>
</body>
</html>
```

File Name: BootstrapHelloWorld.html

```
<!DOCTYPE html>

<html>
<head>
<title>Basic HTML File</title>
<!-- Bootstrap CSS -->
<link crossorigin="anonymous"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
rel="stylesheet"/>
<!-- Bootstrap JS Bundle with Popper -->
<script crossorigin="anonymous" integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"></sc
ript>
</head>
<body>
<h1>Hello World....!!!</h1>
</body>
</html>
```

File Name: cssposition.html

```
<!DOCTYPE html>

<html>
<head>
<title>Basic HTML File</title>
<style type="text/css">
        .box
        {
                display: inline-block;
                border: 3px solid red;
                width: 200px;
                height: 200px;
                margin: 5px;
        }
        #rose
        {
                position: sticky;
      left: 30px;
      top: 0px;

        }
    /* #jasmine
    {
      position: sticky;
                        top:10px
    } */
    /*#daisy
    {
      position: fixed;
      bottom: 0px;
      right: 0px;
    }
    #sunflower
    {
      position: fixed;
      bottom: 0px;
    }*/
        .parent
        {
                border: 3px solid green;
                background: #ee46de40;
                height: 3000px;
```

```
        }
  </style>
<body>
<div class="parent">
<div class="box" id="sunflower">
<img height="200px" src="/Images/sunflower.jpg" width="200px"/>
</div>
<div class="box" id="jasmine">
<img height="200px" src="/Images/jasmine.jpg" width="200px"/>
</div>
<div class="box" id="rose">
<img height="200px" src="/Images/rose.jpg" width="200px"/>
</div>
<div class="box" id="daisy">
<img height="200px" src="/Images/daisy.jpg" width="200px"/>
</div>
</div>
</body>
</head></html>
```

Folder Name: jquery

File Name: addingJQuery.html

```
<!DOCTYPE html>

<html>
<head>
<title>Welcome to jQuery!</title>
<script src="http://code.jquery.com/jquery-2.1.0.js"></script>
<script src="addingjQueryScript.js" type="text/javascript"></script>
</head>
<body>
<h1 id="welcome"></h1>
</body>
</html>
```

File Name: addingjQueryScript.js

```
$(document).ready(function () {
        alert("welcome to jQuery!");
});
```

```
$(document).ready(function () {
	$("#welcome").append("welcome to jQuery!");
});
```

File Name: interactingWithjquery.html

```
<!DOCTYPE html>

<html>
<head>
<title>Welcome to jQuery!</title>
<link href="interactingWithjQueryCascading.css" rel="stylesheet" type="text/css"/>
<script src="http://code.jquery.com/jquery-2.1.0.js"></script>
<script src="interactingWithjQueryScript.js" type="text/javascript"></script>
</head>
<body>
<h1 id="colorMe">Pick a color</h1>
<p>
<a id="red">Red</a>
<a id="green">Green</a>
<a id="blue">Blue</a>
</p>
</body>
</html>
```

File Name: interactingWithjQueryCascading.css

```
a {
	display: block;
	float: left;
	padding: 10px;
	margin: 10px;
	font-weight: bold;
	color: white;
	background-color: gray;
}

a:hover {
	color: black;
```

```
	background-color: silver;
}
```

File Name: interactingWithjQueryScript.js

```
$(document).ready(function () {

	$("a").hover(function(){
		$(this).css({
			"color": $(this).attr("id"),
			"background-color": "silver"
		});
	});

	$("a").mouseout(function(){
		$(this).css({
			"color": "white",
			"background-color": "silver"
		});
	});

	$("a").click(function(evt){
		$("#colorMe").css({
			"color": $(this).attr("id")
		});
		evt.preventDefault();
	});
});
```

Folder Name: nodes

File Name: insertAndReplaceNodes.html

```
<!DOCTYPE html>

<html>
<head>
<title>Deleting Selected Nodes</title>
<script src="insertAndReplaceNodesScript.js"></script>
</head>
<body>
```

```
<form id="theForm">
<p>
<textarea cols="30" id="textArea" rows="5">
                                                </textarea>
</p>
<p>
<label>
<input name="nodeAction" type="radio"/>Add Node
                                                </label>
<label>
<input name="nodeAction" type="radio"/>Delete Node
                                                </label>
<label>
<input name="nodeAction" type="radio"/>Insert Before Node
                                                </label>
<label>
<input name="nodeAction" type="radio"/>Replace Node
                                                </label>
</p>
                                        Paragraph #: <select id="grafCount"></select>
<input type="submit" value="Submit"/>
</form>
<div id="modifiable"> </div>
</body>
</html>
```

File Name: insertAndReplaceNodesScript.js

```
window.addEventListener("load",initAll,false);
var nodeChgArea;

function initAll()
{
        document.getElementById("theForm").addEventListener("submit", nodeChanger,
false);
        nodeChgArea = document.getElementById("modifiable");
}

function addNode()
{
        var inText = document.getElementById("textArea").value;
        var newText = document.createTextNode(inText);
```

```
	var newGraf = document.createElement("p");
	newGraf.appendChild(newText);
	nodeChgArea.appendChild(newGraf);
}

function delNode()
{
	var grafChoice = document.getElementById("grafCount").selectedIndex;
	var allGrafs = nodeChgArea.getElementsByTagName("p");
	var oldGraf = allGrafs.item(grafChoice);
	nodeChgArea.removeChild(oldGraf);
}

function insertNode()
{
	var grafChoice = document.getElementById("grafCount").selectedIndex; // where
user wants to enter
	var inText = document.getElementById("textArea").value; // what user wants to
enter
	var newText = document.createTextNode(inText);
	var newGraf = document.createElement("p");
	newGraf.appendChild(newText);
	var allGrafs = nodeChgArea.getElementsByTagName("p"); // array of p
	var oldGraf = allGrafs.item(grafChoice);
	nodeChgArea.insertBefore(newGraf,oldGraf);
}

function replaceNode()
{
	var grafChoice = document.getElementById("grafCount").selectedIndex;
	var inText = document.getElementById("textArea").value;
	var newText = document.createTextNode(inText);
	var newGraf = document.createElement("p");
	newGraf.appendChild(newText);
	var allGrafs = nodeChgArea.getElementsByTagName("p");
	var oldGraf = allGrafs.item(grafChoice);
	nodeChgArea.replaceChild(newGraf,oldGraf);
}

function nodeChanger(evt)
{
	var actionType = -1;
	var pGrafCT = nodeChgArea.getElementsByTagName("p").length;
```

```
var radioButtonSet = document.getElementById("theForm").nodeAction;

for (var i = 0; i < radioButtonSet.length; i++)
{
	if (radioButtonSet[i].checked)
	{
		actionType = i;
	}
}

switch(actionType)
{
	case 0:
		addNode();
		break;
	case 1:
		if (pGrafCT > 0)
		{
			delNode();
			break;
		}
	case 2:
		if (pGrafCT > 0)
		{
			insertNode();
			break;
		}
	case 3:
		if (pGrafCT > 0)
		{
			replaceNode();
			break;
		}
	default:
		alert("No valid action was chosen");
}

document.getElementById("grafCount").options.length = 0;

for (i = 0; i < nodeChgArea.getElementsByTagName("p").length; i++)
{
	document.getElementById("grafCount").options[i] = new Option(i+1);
}
```

```
        evt.preventDefault();
}
```

File Name: SectionA-quiz1.html

```
<!DOCTYPE html>

<html lang="en">
<head>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1.0" name="viewport"/>
<title>Quiz-01 Solution</title>
</head>
<script>

  // Question-01
  submit_form = () => {
    var name = document.getElementById("name").value
    var email = document.getElementById("email").value
    var male = document.getElementById("male")
    var female = document.getElementById("female")
    var color = document.getElementById("color").value

    if ( name === "" || email === "" || (!male.checked && !female.checked) || color === "") {
      alert("Please fill all the fields")
      return
    }
  }

  // Question - 02
  calc_sum = () => {
    var num1 = document.getElementById("num1").value
    var num2 = document.getElementById("num2").value
    var res = document.getElementById("res")

    if (num1 === "" || num2 === "") {
      alert("Please fill all the fields")
      return
    }

    res.value = parseInt(num1) + parseInt(num2)
```

```
  }

</script>
<body>
<!-- Question - 01 -->
<form>
<label>Name</label>
<input id="name" placeholder="Enter Name" type="text"/>
<label>Email</label>
<input id="email" placeholder="Enter Email" type="email">
<label>Gender</label>
<input id="male" name="gender" type="radio"/>
<label>Male</label>
<input id="female" name="gender" type="radio"/>
<label>Female</label>
<label>Faviroute Color</label>
<select id="color">
<option value="red">Red</option>
<option value="blue">Blue</option>
<option value="green">Green</option>
</select>
<button onclick="submit_form()" type="submit">Submit</button>
</input></form>
<!-- Question - 02 -->
<input id="num1" onchange="calc_sum()" placeholder="Enter Number 01" type="text"/>
<input id="num2" onchange="calc_sum()" placeholder="Enter Number 02" type="text"/>
<input id="res" placeholder="Result" type="text"/>
</body>
</html>
```

Folder Name: html

File Name: index.html

```
<!DOCTYPE html>

<html lang="en">
<head>
<title>My first Web Page</title>
<style>
        img {
           width: 100px;
           border-radius: 50px;
```

```
            float: left;
            margin-right: 10px;
          }
          p {
            font-weight: bold;
          }
      </style>
</head>
<body>
<!-- absolute URL -->
<img alt="Image of sunflower" src="/Images/sunflower.jpg"/>
<!-- relative URL -->
<img alt="Image of sunflower" src="../../Desktop/html/Images/sunflower.jpg"/>
<!-- Hyperlink -->
<a href="#Section-One"> Lorem ipsum</a>
<a href="https://google.com" target="_blank"> Google</a>
<a href="mailto:xyz"> Email</a>
<p>Hi, Im a sunflower!</p>
<p id="Section-One">Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam
reprehenderit earum nihil neque eius magnam labore sunt non,
        maiores quo aut assumenda, modi et impedit fugit numquam temporibus commodi!
Incidunt.Lorem ipsum dolor sit amet consectetur adipisicing elit.
        Nam reprehenderit earum nihil neque eius magnam labore sunt non,
        maiores quo aut assumenda,
        modi et impedit fugit numquam temporibus commodi! Incidunt.</p>
</body>
</html>
```

File Name: index.js

```
// var -> Function
// let -> Block

function sayHello(){
          for (var i = 0; i<5; i++){
                    console.log(i);
          }
          console.log(i);
}
sayHello();
```

```
// ***********************************
// const -> Block
// cannot reassign

const x = 5;
x = 1;

// ***********************************
const person = {
name: "Saif",
walk: function(){},
talk() {}
};

person.talk();
person.name = "";

const targetMember = "name";
person[targetMember] = "Ali";

//************************************

const person = {
	name: "Saif",
	walk() {
		console.log(this);
	}
};

//ref to object
person.walk();

//standalone object
const walkOne = person.walk;
walkOne();

// BIND
const walk = person.walk.bind(person);
walk();

//************************************

// Arrow function
```

```
const squareOne = function(number){
	return number * number;
}

const squareTwo = number => number * number;

console.log(squareOne(5));
console.log(squareTwo(5));

const jobs = [
	{id:1, isActive:true},
	{id:2, isActive:true},
	{id:3, isActive:false}
];

const activeJobsOne = jobs.filter(function(job) {return job.isActive;});
const activeJobsTwo = jobs.filter(job => job.isActive);
console.log(activeJobsOne);
console.log(activeJobsTwo);

// Arrow functions dont rebind with 'this' keyword
const person = {
	talk () {
		console.log("This: ", this);
		var self = this;
		setTimeout(function() {
			console.log("This: ", this);
			console.log("Self: ", self);
		}, 1000)
		setTimeout(()=> {
				console.log("this", this)
		}, 1000);
	}
};

person.talk();

// Object Restructuring:
// const address = {
//	street: '',
//	city: '',
//	country: ''
// };
```

```
//  street = address.street;
//  city = address.city;
//  country = address.country;

// const {street, city, country} = address;
// const { street: st } = address;

// spread operator
const first = [1,2,3];
const second = [4,5,6];

var combined = first.concat(second);
combined = [...first, 'a', ...second, 'saif'];
console.log(combined);
var clone = [...first];
console.log(first);
console.log(clone);

// spread operator - in objects
const One = {Name: "Saif"};
const Two = {Job: "Lecturer"};

const Merge = {...One, ...Two, Location: "NUCES-FAST, Lahore Campus"};
console.log(Merge);

// Classes

class Human {
	constructor(name){
		this.name = name;
	}
	walk(){
		console.log("Human walk");
	}
}

const human = new Human();
human.walk();

// Inheritance

class Human {
```

```
        constructor(name){
                this.name = name;
        }
        walk(){
                console.log("Human walk");
        }
}

class Teacher extends Human {
        constructor(name, degree){
                super(name)
                this.degree = degree;
        }
        teach(){
                console.log("teach");
        }
}

const teacher = new Teacher('Saif', 'MSc');

// Modules
import  {Teacher} from './teacher'
const teacherOne = new Teacher('Saif', 'MSc');
// Default -> import ... from '';
// Named -> import {...} from '';

// example React, {Component} from 'react';
```

File Name: indexThree.html

```
<!DOCTYPE html>

<html lang="en">
<head>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1.0" name="viewport"/>
<title>Document</title>
<style>
      img {
         width: 500px;
         height: 500px;
         object-fit: cover;
```

```
    }
  </style>
</head>
<body>
<img alt="picture of flower" src="/Images/sunflower.jpg"/>
</body>
</html>
```

File Name: IndexTwo.html

```
<!DOCTYPE html>

<html lang="en">
<head>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1.0" name="viewport"/>
<!-- For SEO -->
<meta keywords="CSS, HTML"/>
<!-- The content we will to see on google -->
<meta content="..."/>
<title>Document</title>
</head>
<body>
<!-- Entities -->
<h1>HeadingOne &lt;Example&gt; ©</h1>
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam reprehenderit earum nihil neque eius magnam labore sunt non,
    maiores quo aut assumenda, modi et impedit fugit numquam temporibus commodi! Incidunt.Lorem ipsum dolor sit amet consectetur adipisicing elit.
    Nam reprehenderit earum nihil neque eius magnam labore sunt non,
    maiores quo aut assumenda, numquam temporibus numquam temporibus numquam temporibus
    modi et impedit fugit numquam temporibus commodi! Incidunt.</p>
</body>
</html>
```

```
// 1. Global Variables and Constants
var globalVariable = 'Global'; // Declare global variables using 'var'
let anotherGlobalVariable = 'Global'; // Declare global variables using 'let'
const GLOBAL_CONSTANT = 'Global Constant'; // Declare global constants using 'const'
// 2. Functions
function myFunction(parameter1, parameter2) {
  // Function code here
  var localVariable = 'Local'; // Declare local variables within functions
  return result; // Return a value if necessary
}
// 3. Event Listeners
document.addEventListener('DOMContentLoaded', function () {
  // Code to run when the DOM is fully loaded
});
element.addEventListener('click', function () {
  // Code to run when the element is clicked
});
// 4. Conditional Statements
if (condition) {
  // Code to run if the condition is true
} else if (anotherCondition) {
  // Code to run if another condition is true
} else {
  // Code to run if none of the conditions are true
}
// 5. Loops
for (let i = 0; i < length; i++) {
  // Code to repeat in a 'for' loop
}
```

```
while (condition) {
  // Code to repeat in a 'while' loop
}
// 6. Object and Array Declaration
let myObject = {
  key1: 'value1',
  key2: 'value2'
};
let myArray = [element1, element2, element3];
// 7. DOM Manipulation
let element = document.getElementById('elementId');
element.innerHTML = 'New content'; // Modify HTML content
element.style.color = 'red'; // Change CSS styles
// 8. Error Handling (try-catch)
try {
  // Code that might cause an error
} catch (error) {
  // Code to handle the error
}
// 9. Call Functions
let result = myFunction(argument1, argument2); // Call a function
```

HTML (Hypertext Markup Language):

HTML is the standard markup language for creating web pages. It consists of elements enclosed in tags, and it provides the structure and content for a web page.

1. Basic HTML Structure:

```
<!DOCTYPE html>
<html>
<head>
```

```
    <title>Page Title</title>
</head>
<body>
    <h1>Heading 1</h1>
    <p>This is a paragraph.</p>
</body>
</html>
```

2. Common HTML Elements:

- `<div>`: A container for grouping elements.
- `<p>`: Paragraph.
- `<h1>`, `<h2>`, ... `<h6>`: Headings.
- `<a href="url">Link Text</a>`: Hyperlink.
- `<img src="image.jpg" alt="Description">`: Image.
- `<ul>` and `<ol>`: Unordered and ordered lists.
- `<li>`: List item.
- `<table>`, `<tr>`, `<td>`: Table elements.

3. Forms and Input Types:

- `<form>`: Container for user input.
- `<input type="text">`: Single-line text input.
- `<input type="password">`: Password input.
- `<input type="checkbox">`: Checkbox.
- `<input type="radio">`: Radio button.
- `<input type="submit">`: Submit button.
- `<select>` and `<option>`: Dropdown menus.
- `<textarea>`: Multiline text input.

CSS (Cascading Style Sheets):

CSS is used for styling and layout of web pages. It allows you to control the appearance of HTML elements.

1. Basic CSS Syntax:

```
selector {
    property: value;
}
```

2. Selectors:

- Type Selector: `p { }`
- Class Selector: `.classname { }`
- ID Selector: `#idname { }`

3. Common CSS Properties:

- `color`: Text color.
- `font-family`: Font type.
- `font-size`: Font size.
- `margin` and `padding`: Spacing around elements.
- `border`: Border around elements.
- `background-color`: Background color.

4. CSS Box Model:

- Content, Padding, Border, Margin.

5. CSS Layout:

- `display`: Block, inline, inline-block.
- `position`: Relative, absolute, fixed.

JavaScript:

JavaScript is a scripting language used for enhancing interactivity and functionality on web pages.

1.DOM Manipulation:

- Access and manipulate HTML elements using JavaScript.
- `document.getElementById()`, `document.querySelector()`, etc.

2. Events:

- Respond to user actions like clicks, inputs, etc.
- `addEventListener()`.

These are some of the fundamental concepts and elements in HTML, CSS, and JavaScript. As you continue learning, you'll explore more advanced topics and techniques in web development.