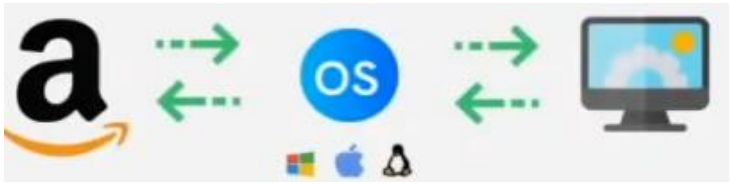
Software : it is a set of instructions to the hardware

OS : it is the main software that makes other software's work with the hardware.
It is an interface between hardware and software.



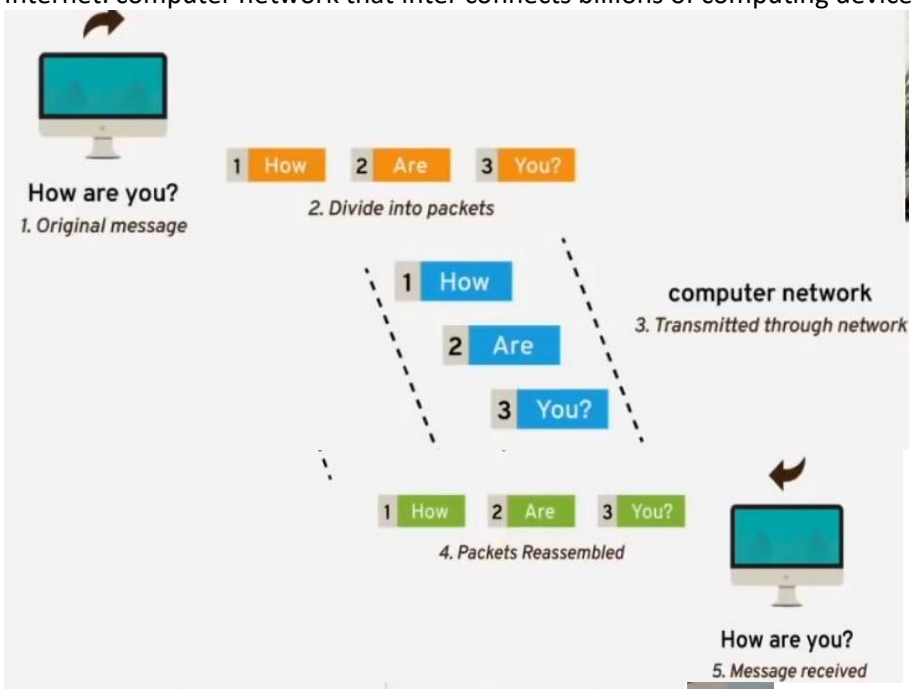
Different OS:
Computer OS

- Windows
- Ubuntu
- Mac os

Mobile os

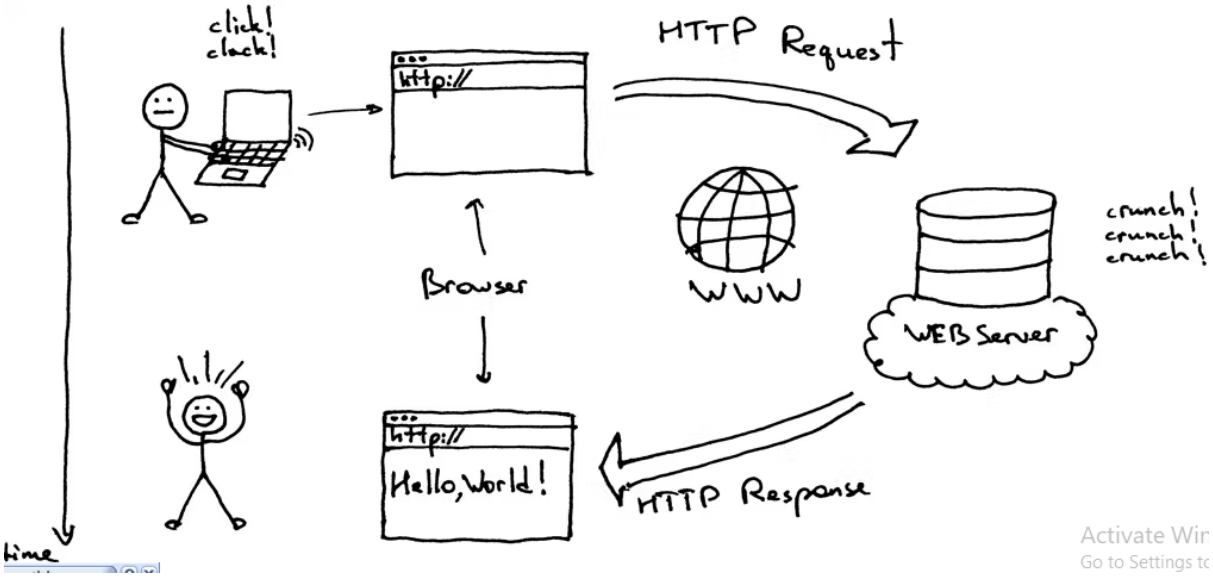
- Android 11
- Ios 13

Networks :
Internet: computer network that inter connects billions of computing device throughout the world



Cargo Scenario	Network Scenar
Cargo	Data
Trucks	Packets
Entire cargo can't be sent in single truck	Entire data can't be transferred in single packet
Junctions	Routers
Roads	Routes
Multiple paths	Multiple routes
Traffic	Traffic (congestion)

Client : one that requests
Servers: one that responds to requests
Exchange of information between client and server happens over the Network



Browsers: which has capable to read html code
Servers: which has capable to receive a request and send a response

Introduction to front end
Application development : when we say application it refers to the software developed for a specific purpose
Ex: browser, whats app, Mobile games etc..

Html: hyper text markup language(the content we see in the application is written in html)
- Hyper text: text containing links
- Markup : the remarks that we add highlight

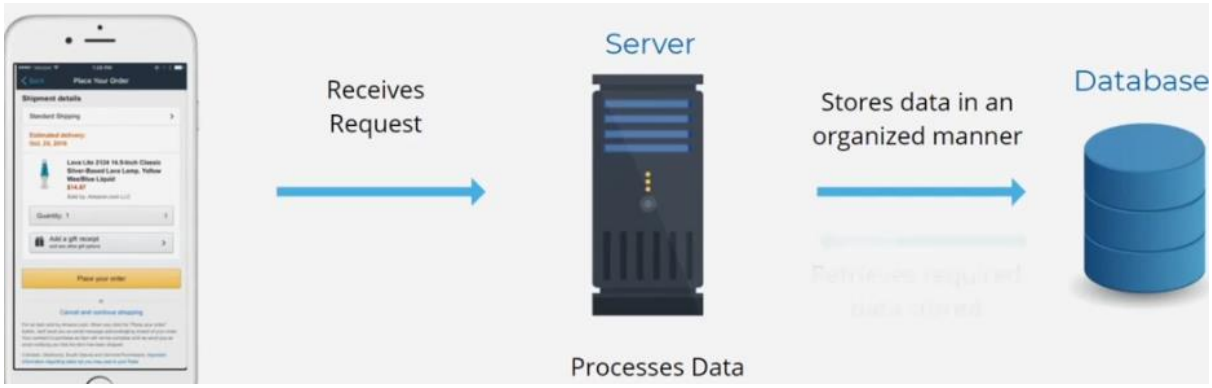
CSS: cascading Style sheets(to achieve style like font, color, size, shapes)
Javascript: can change contents of website dynamically without reloading the entire site.

Responsive web design: website view is same for all screen sizes
Cross platform technologies: developing an app for android and ios is time consuming. To overcome this React Js and flutter

Introduction to Back end
Data: any information that has to be stored
Ex: text, images, videos, audio
Space occupied by data is measured in the units of bits and bytes

1 Byte	8 Bits
1 Kilobyte	1024 bytes
1 Megabyte	1024 Kilobytes
1 Gigabyte	1024 Megabytes
1 Terabyte	1024 Gigabytes

Back end:



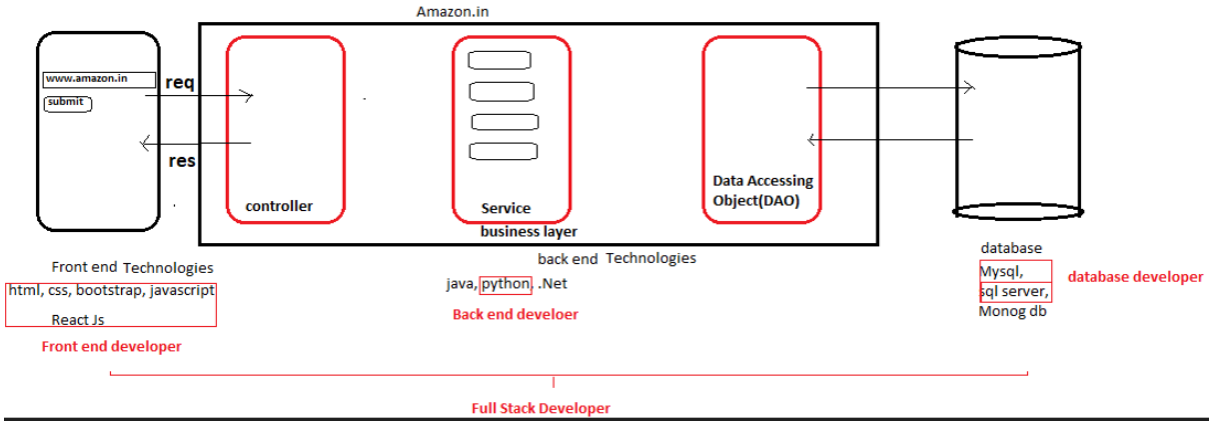
API: application programming interface
Communication between to different applications

Cloud: servers that accessed over the internet and the software's run on those servers
Cloud computing: on demand availability of server system, of any size, over the internet at the click of a button from your computer

Scaling: decreasing and increasing the servers
Horizontal scaling
Vertical scaling

Cloud service providers takes care of

- Security
- Maintenance
- Purchasing hardware
- Power and internet connection
- Replacing the faculty hardware

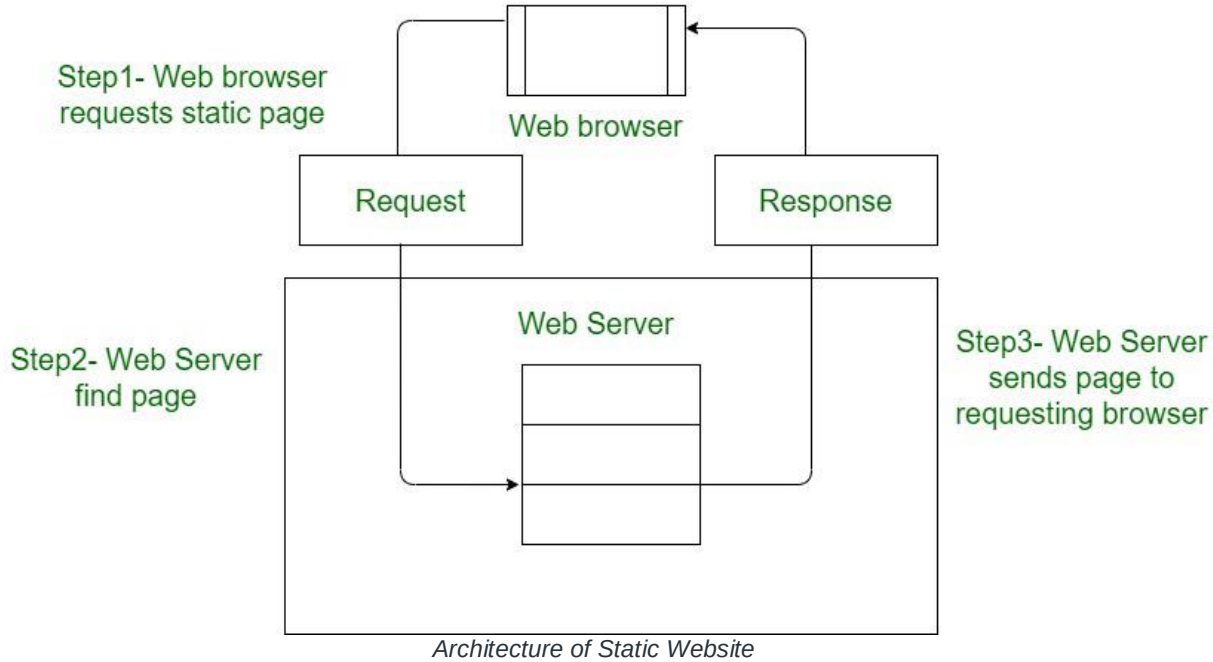


What is Website?
Website is the collection of web pages, different multimedia content such as text, images, and videos which can be accessed by the URL which you can see in the address bar of the browser. For example: https://www.geeksforgeeks.org

How to access Websites?
When we type a certain URL in a browser search bar, the browser requests the page from the Web server and the Web server returns the required web page and its content to the browser. Now, it differs how the server returns the information required in the case of static and dynamic websites.
Types of Website:

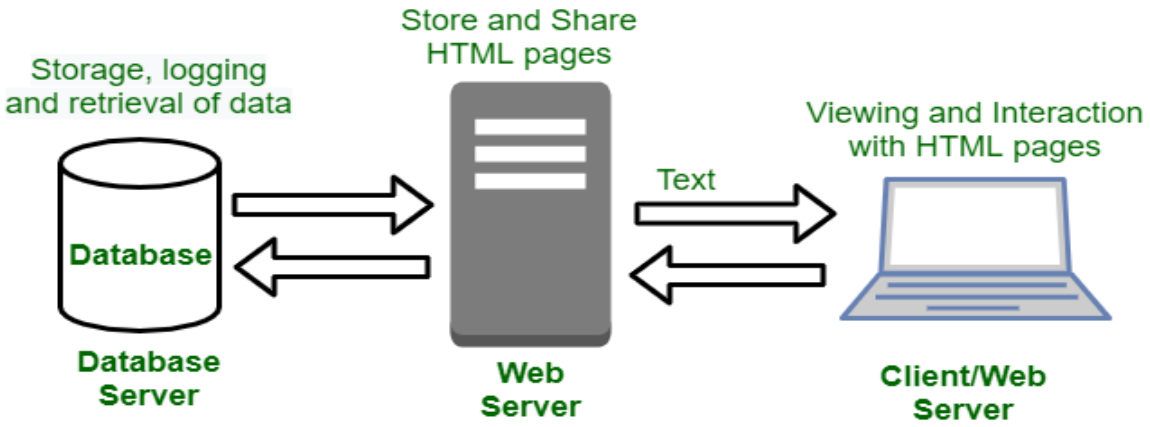
- Static Website
- Dynamic Website

Static Website: In Static Websites, Web pages are returned by the server which are prebuilt source code files built using simple languages such as HTML, CSS, or JavaScript. There is no processing of content on the server (according to the user) in Static Websites. Web pages are returned by the server with no change therefore, static Websites are fast. There is no interaction with databases. Also, they are less costly as the host does not need to support server-side processing with different languages.



Architecture of Static Website

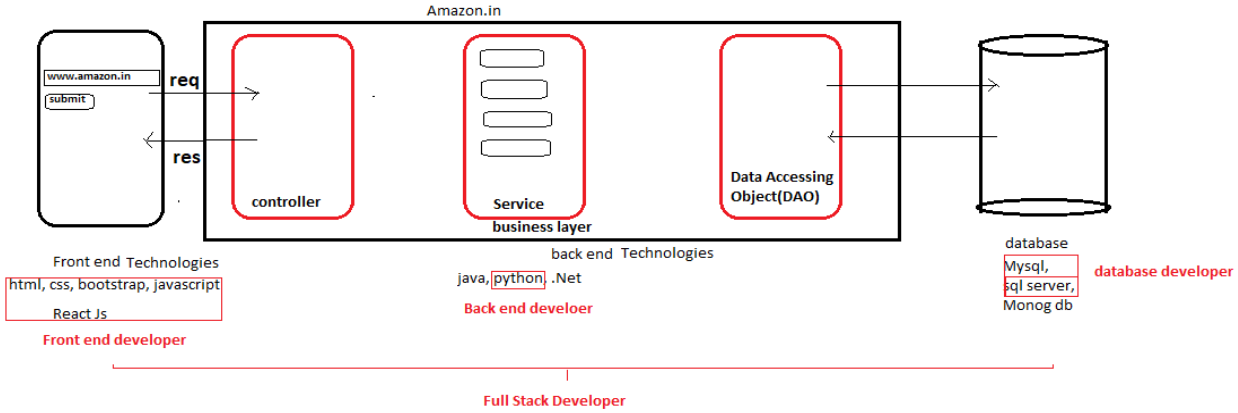
Note: Static does not mean that it will not respond to user actions, These Websites are called static because these cannot be manipulated on the server or interact with databases (which is the case in Dynamic Websites).
Dynamic Website: In Dynamic Websites, Web pages are returned by the server which are processed during runtime means they are not prebuilt web pages but they are built during runtime according to the user's demand with the help of server-side scripting languages such as PHP, Node.js, ASP.NET and many more supported by the server. So, they are slower than static websites but updates and interaction with databases are possible.
Dynamic Websites are used over Static Websites as updates can be done very easily as compared to static websites (Where altering in every page is required) but in Dynamic Websites, it is possible to do a common change once and it will reflect in all the web pages.



Architecture of Dynamic Website

Difference Between Static and Dynamic Websites:

Static Website	Dynamic Website
Content of Web pages can not be change at runtime.	Content of Web pages can be changed.
No interaction with database possible.	Interaction with database is possible
It is faster to load as compared to dynamic website.	It is slower than static website.
Cheaper Development costs.	More Development costs.
No feature of Content Management.	Feature of Content Management System.
HTML, CSS, Javascript is used for developing the website.	Server side languages such as PHP, Node.js are used.
Same content is delivered everytime the page is loaded.	Content may change everytime the page is loaded.



1. Introduction to Dynamic Web Applications

- Dynamic web applications: introduction
- Introduction to java script: why java script? writing first java script program
- Java Script variables: Declaring Variables

Dynamic web applications: personalized content
Ex: Instagram , Youtube, Live score updates, latest stats about covid-19 cases, live traffic updates
Similarly Gmail sign in (user interactivity)- form data validations

Building Dynamic web applications mainly involves:

- Client server communications
- Manipulating HTML and CSS
- Writing Application Logic

Ways to achieve:

- Using Java Script
- Using Web Assembly

Why Java script?
- It is the most popular programming language which is adopted and Understood by major browsers
- 69.7% developers use Java Script
- big companies like google, Netflix etc use java Script

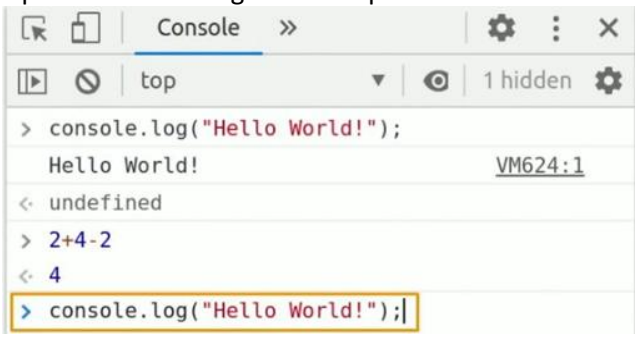
Html provides the basic structure for web pages.
CSS is used for styling
JS is used primarily by web browsers to create a dynamic and interactive experience for users.

Java script can be used in both client side and Server side.

- We can add animations
- We can add live dashboards
- We can add browser based games

```
console.log("Hello world!")
```

Running Java Script code using Browser Console:
Open browser → right click inspect → console → write the java script code → enter



Java Script Variables:
Variables are like containers for storing values
We can create a variable using **let** keyword → declaring a variable: let message
We can put data into a variable using assignment operator(=) → let message = 'pawan';
Printing a variable → console.log(message);

```
console.log("Hello world!");

let m;
console.log(m);

let message = 'pawan kumar puppala'
console.log(message);

console.log("hellooooo");

console.log(123);
```

```
Hello world!
undefined
pawan kumar puppala
hellooooo
123
```

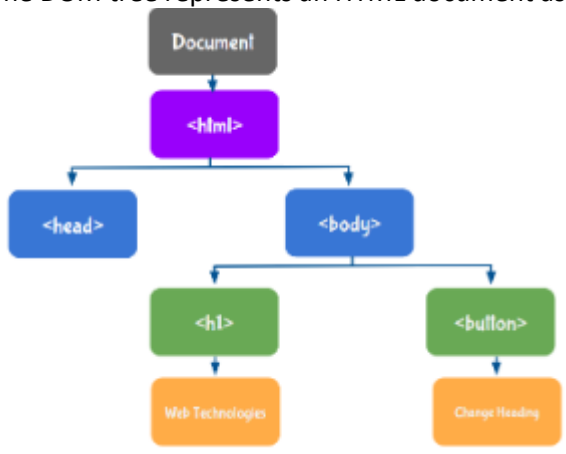
2. DOM and Event Fundamentals
Document Object Model: Manipulating HTML and CSS, HTML DOM Tree
DOM Methods: getElementById()
Events: onclick Event

How do we manipulate HTML and CSS Using JS?

Document Object Model:
The DOM is the structured representation of the content in the document created by the browser.
HTML DOM: Tree structured representation of the HTML document created by the browser.
It allows JavaScript to manipulate, structure, and style your website.

Document Object
It is the entry point of the DOM. For accessing any HTML Element, you should always start with accessing the document object first.

HTML DOM:
The DOM tree represents an HTML document as nodes. Each node is referred to as an Object.




```
<!DOCTYPE html>
<html>
<head>
    <script src="example.js"></script>
</head>
<body>
    <h1>web technologies</h1>
    <button>change heading</button>
</body>
</html>
```

- ➔ Select the heading element
- ➔ Manipulate the text content of the heading element
- ➔ Manipulate the styles of the heading element.

```
<!DOCTYPE html>
<html>
<head>
    <script src="example.js"></script>
</head>
<body>
    <h1 id="headingElement">Web Technologies</h1>
    <button>change heading</button>
</body>
</html>
console.log(document.getElementById("headingElement"));
```

Output:

```
      <h1 id="headingElement">Web Technologies</h1>
```

```
document.getElementById("headingElement").textContent="4.0 Technologies";
document.getElementById("headingElement").style.color = "blue";
```

getElementById
The getElementById() method helps to select the HTML Element with a specific ID.

Properties:
textContent: To manipulate the text within the HTML Element, we use textContent Property.

Style:
The style property is used to get or set a specific style of an HTML Element using different CSS properties.
Use Camel Case naming convention (starting letter of each word should be in the upper case except for the first word) for naming the Style Object Properties.
For example: color , fontFamily , backgroundColor , etc.

Events
Events are the actions by which the user or browser interacts with the HTML Elements. Actions can be anything like clicking a button, pressing keyboard keys, scrolling the page, etc.
onclick Event
The onclick event occurs when the user clicks on an HTML Element. We will give the name of the function as a value for the HTML onclick attribute.

How to manipulate HTML and CSS Based on user actions? (when we click on button)

ex1_heading.html > ...

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="ex3style.css" />
</head>
<body>
    <h1 id="headingElement">Web Technologies</h1>
    <button onclick="manipulateStyles()">change heading</button>
    <script src="ex1_heading.js"></script>
</body>
</html>
```

```
function manipulateStyles(){
    //console.log("Hi Pawan puppala");
    document.getElementById("headingElement").textContent="4.0 Technologies";
    document.getElementById("headingElement").style.color = "blue";
}
```

Steps:

- Make Switches listen to User Actions
- Make bulb Go On and Off
- Make cat Visible and Invisible
- Update Switch Status
- Change the background color of the switch

```
<body>
  <div class="dark-background text-center">
    <div>
      <img
        src="https://d1tgh8fmlzexmh.cloudfront.net/ccbp-dynamic-webapps/bulb-go-on-img.png"
        class="bulb-image"
        id="bulbImage"/>
    </div>
    <div>
      <img
        src="https://d1tgh8fmlzexmh.cloudfront.net/ccbp-dynamic-webapps/cat-img.png"
        class="cat-image"
        id="catImage"/>
    </div>
    <div class="d-flex flex-row justify-content-center pt-5">
      <div class="switch-board">
        <h1 class="switch-status" id="switchStatus">Switched On</h1>
        <button class="off-switch" id="offSwitch" onclick="switchOff()">OFF</button>
        <button class="on-switch" id="onSwitch" onclick="switchOn()">ON</button>
      </div>
    </div>
  </div>
</body>
```

JS ex2_catandlight.js > ...

```
function switchOff() {
    document.getElementById("bulbImage").src =
      "https://d1tgh8fmlzexmh.cloudfront.net/ccbp-dynamic-webapps/bulb-go-off-img.png";
    document.getElementById("catImage").src =
      "https://d1tgh8fmlzexmh.cloudfront.net/ccbp-dynamic-webapps/cat-eyes-img.png";
    document.getElementById("switchStatus").textContent = "Switched Off"
    document.getElementById("onSwitch").style.backgroundColor = "#22c55e";
    document.getElementById("offSwitch").style.backgroundColor = "#cbd2d9";
  }
```

```
function switchOn() {
    document.getElementById("bulbImage").src =
        "https://d1tgh8fmlzexmh.cloudfront.net/ccbp-dynamic-webapps/bulb-go-on-img.png";
    document.getElementById("catImage").src =
        "https://d1tgh8fmlzexmh.cloudfront.net/ccbp-dynamic-webapps/cat-img.png";
    document.getElementById("switchStatus").textContent = "Switched On";
    document.getElementById("offSwitch").style.backgroundColor = "#e12d39";
    document.getElementById("onSwitch").style.backgroundColor = "#cbd2d9";
}
```

3. Primitive Types & Conditionals

- Primitive types: undefined string Boolean number
- Converting String into number: parseInt()
- Conditionals: if… else Statements

What kind of other values can we assign to the variables?
Primitive Types:

- Number: all numbers are of Number type(int, float, real)

```
let a = 900
let b = 9.2
console.log(typeof(a))  number
console.log(typeof(b))  number
```

- Boolean : true and false

```
let isApproved = false;
console.log(typeof(isApproved)); boolean
console.log(typeof(true));       boolean
```

- String: collections of characters

```
console.log(typeof('pawan')); string
console.log(typeof("pawan")); string
console.log(typeof(`pawan`)); string
```

- Undefined : if a value is not assigned to the variable

```
let age;
console.log(age); undefined
```

Reference

Ex3counter.html

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="ex3style.css" />
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.j
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.
  </head>
  <body>
    <div class="bg-container text-center d-flex flex-column justify-content-center">
      <h1 class="counter-heading">Counter</h1>
      <p id="counterValue" class="counter-value">0</p>
      <div>
        <button onClick="onDecrement()" class="button">DECREASE</button>
        <button onClick="onReset()" class="button">RESET</button>
        <button onClick="onIncrement()" class="button">INCREASE</button>
      </div>
    </div>
  </body>
</html>
```

Ex3style.css

```
@import url("https://fonts.googleapis.

.bg-container {
  background-color: #f1f5f8;
  height: 100vh;
}
.counter-heading {
  font-family: "Roboto";
  font-size: 50px;
  font-weight: 800;
}
.counter-value {
  font-size: 75px;
  font-weight: 900;
}
.button {
  background-color: #f1f5f8;
  border-width: 2px;
  border-color: black;
  border-radius: 7px;
  margin: 8px;
  padding: 5px;
}
```

Ex3.js

```
let counterElement = document.getElementById("counterValue");

function onIncrement() {
  let previousCounterValue = counterElement.textContent;
  let updatedCounterValue = parseInt(previousCounterValue) + 1;
  if (updatedCounterValue > 0) {
    counterElement.style.color = "green";
  }
  else if (updatedCounterValue < 0) {
    counterElement.style.color = "red";
  }
  else {
    counterElement.style.color = "black";
  }
  counterElement.textContent = updatedCounterValue;
}
function onDecrement() {
  let previousCounterValue = counterElement.textContent;
  let updatedCounterValue = parseInt(previousCounterValue) - 1;
  if (updatedCounterValue > 0) {
    counterElement.style.color = "green";
  }
  else if (updatedCounterValue < 0) {
    counterElement.style.color = "red";
  }
  else {
    counterElement.style.color = "black";
  }
  counterElement.textContent = updatedCounterValue;
}
function onReset() {
  let counterValue = 0;
  counterElement.textContent = counterValue;
  counterElement.style.color = "black";
}
```

Input Element and Math Functions

- Math Functions: Math.ceil(), Math.random()
- Input element: Text Input, password Input
- Comparison Operators: Loose Equal to(==), Strict Equal to(===)

How to generate the random number in java script?
Math.random() : returns a random number from 0 to 1

```
console.log(Math.random()) 0.05428668905811462
console.log(Math.random()) 0.562598656650753
console.log(Math.random()) 0.3197352645104976
```

```
console.log(Math.random()*100) 96.18901881384545
console.log(Math.random()*100) 98.75588433562761
console.log(Math.random()*100) 49.14240199781603
```

Math.ceil() : always rounds a number up to the next largest integer.

```
console.log(Math.ceil(0.054)) 1
console.log(Math.ceil(0.562)) 1
console.log(Math.ceil(0.319)) 1
console.log(Math.ceil(96.189)) 97
console.log(Math.ceil(98.755)) 99
console.log(Math.ceil(49.142)) 50
```

How to take input from the user?
The HTML input element creates interactive controls to accept the data from the user.
There are different types of inputs.

- Text
- Password
- Radio
- Date
- Checkbox

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="ex4style.css" />
  </head>
  <body>
    <p>Enter your Name</p>
    <input type="text" id="inputElement" />
    <p>Enter your Password</p>
    <input type="password" />
    <div>
      <button onclick="signIn()">Sign In</button>
    </div>
    <p id="signInText"></p>
  </body>
</html>
```

```
let inputElement = document.getElementById("inputElement");
let signInTextElement = document.getElementById("signInText");
function signIn() {
  let inputValue = inputElement.value;
  let verifyText = "Hi " + inputValue + ", verifying your account...";
  signInTextElement.textContent = verifyText;
}
```

Loose equal to (==): Loose equality compares two values for equality but doesn't compare types of values.
Strict equal to (===): Strict equality compares two values for equality including types of values.

```
console.log(2=='2')
console.log(2==='2')
```

```
<body>
  <div class="container-fluid bg-container">
    <div class="row bg-white">
      <div class="col-12 col-md-6 m-auto bg-white pt-5 pb-5">
        <img
          class="guess-game-img"
          src="https://d1tgh8fmlzexmh.cloudfront.net/ccbp-dynamic-webapps/guess-game-img.png"/>
        <p class="text-center game-description">
          Find out the right number between 1 to 100
        </p>
      </div>
    </div>
    <div class="row">
      <div class="col-12 guess-bg-container text-center">
        <h1 class="guess-heading">
          Guess The Number
          <img
            class="guess-number-image"
            src="https://d1tgh8fmlzexmh.cloudfront.net/ccbp-dynamic-webapps/guess-number-img.png"/>
        </h1>
        <input type="text" class="user-input" id="userInput" />
        <div>
          <button class="btn btn-info check-guess" onclick="checkGuess()">
            Check
          </button>
        </div>
        <p class="game-result" id="gameResult"></p>
      </div>
    </div>
  </div>
</body>
```



```
let gameResult = document.getElementById("gameResult");
let userInput = document.getElementById("userInput");
let randomNumber = Math.ceil(Math.random() * 100);
function checkGuess() {
  let guessedNumber = parseInt(userInput.value);
  if (guessedNumber > randomNumber) {
    gameResult.textContent = "Too High! Try Again.";
    gameResult.style.backgroundColor = "#1e217c";
  }
  else if (guessedNumber < randomNumber) {
    gameResult.textContent = "Too Low! Try Again.";
    gameResult.style.backgroundColor = "#1e217c";
  }
  else if (guessedNumber === randomNumber) {
    gameResult.textContent = "Congratulations! You got it right.";
    gameResult.style.backgroundColor = "green";
  }
  else {
    gameResult.textContent = "Please provide a valid input.";
    gameResult.style.backgroundColor = "#1e217c";
  }
}
```

Arrays and more DOM Manipulations:

- Arrays: methods Operations
- Functions: Declaration Expression
- DOM Manipulations: classList.remove(), add(), appendChild(), createElement()

1. Data Structures

Data Structures allow us to store and organize data efficiently. This makes us access and performs operations on the data smoothly.

In JavaScript, we have built-in Data Structures like,

- Arrays : An Array holds an ordered sequence of items.
- Objects
- Maps
- Sets

```
//Creating an Array
let myArray = [5, "six", 2, 8.2];
console.log(myArray);

//Accessing an Array Item
console.log(myArray[0]);
console.log(myArray[1]);

//  Modifying an Array Item
myArray[1] = 6;
console.log(myArray);

//Finding Array Length
let lengthOfArray = myArray.length;
console.log(lengthOfArray);

//Array Methods
myArray.push(true);
console.log(myArray);

let lastItem = myArray.pop();
console.log(myArray);

console.log(lastItem);
```

```
[ 5, 'six', 2, 8.2 ]
5
six
[ 5, 6, 2, 8.2 ]
4
[ 5, 6, 2, 8.2, true ]
[ 5, 6, 2, 8.2 ]
true
```

More DOM manipulations:

Creating an HTML Element using Java Script?

```
let h1Element = document.createElement("h1");
h1Element.textContent = "Web Technologies";
console.log(h1Element);
```

o/p : <h1>Web Technologies</h1>

Appending to Document Body Object:

```
document.body.appendChild(h1Element);
```

Appending to Existing Container Element:

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <div id="myContainer">

    </div>
    <script src="ex1.js"></script>
  </body>
</html>
```

ex1.js

```
let containerElement = document.getElementById("myContainer");
containerElement.appendChild(h1Element);
```

Adding Event Listeners Dynamically

```
let btnElement = document.createElement("button");
btnElement.textContent = "Change Heading";
document.getElementById("myContainer").appendChild(btnElement);
```

Web Technologies

Change Heading

Functions: there is another function syntax for creating a function which is called Function Expression.

```
let showMessage = function(){
    console.log("Hello pawan")
}

showMessage();
```

Hello pawan

Function Declaration

```
function showMessage() {
  console.log('Hello');
}
showMessage();
```

Function Expression

```
let showMessage = function() {
  console.log('Hello');
};
showMessage();
```

```
btnElement.onclick = function(){
    console.log("click event triggered")
};
```

Providing Class Names Dynamically

```
btnElement.onclick = function(){
  h1Element.textContent = "4.0 Technologies";
  h1Element.classList.add("heading");
  console.log(h1Element);
};
```

```
.heading {
    color: blue;
    font-family: "Caveat";
    font-size: 40px;
    text-decoration: underline;
}
```

```
btnElement.onclick = function(){
  h1Element.textContent = "4.0 Technologies";
  h1Element.classList.add("heading");
  h1Element.style.color = "blue";
  h1Element.style.font = "40px";
  h1Element.style.fontFamily = "Caveat"
  h1Element.style.textDecoration = "underline"
  };
```

Removing Class Names Dynamically

```
let removeStylesBtnElement = document.createElement("button");
removeStylesBtnElement.textContent = "Remove Styles";

document.getElementById("myContainer").appendChild(removeStylesBtnElement);
removeStylesBtnElement.onclick = function(){
h1Element.classList.remove("heading");
};
```

Js file

```
let h1Element = document.createElement("h1");
h1Element.textContent = "Web Technologies";
document.body.appendChild(h1Element);

let containerElement = document.getElementById("myContainer");
containerElement.appendChild(h1Element);

let btnElement = document.createElement("button");
btnElement.textContent = "Change Heading";
document.getElementById("myContainer").appendChild(btnElement);

btnElement.onclick = function(){
  h1Element.textContent = "4.0 Technologies";
  h1Element.classList.add("heading");
  // h1Element.style.color = "blue";
  // h1Element.style.font = "40px";
  // h1Element.style.fontFamily = "Caveat"
  // h1Element.style.textDecoration = "underline"
  };

let removeStylesBtnElement = document.createElement("button");
removeStylesBtnElement.textContent = "Remove Styles";

document.getElementById("myContainer").appendChild(removeStylesBtnElement);
removeStylesBtnElement.onclick = function(){
h1Element.classList.remove("heading");
};
```

Html file

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="ex1.css"></link>
  </head>
  <body>
    <div id="myContainer">

    </div>
    <script src="ex1.js"></script>
  </body>
</html>
```

Objects:

- **Objects: Creating an object**
- **Object properties: Dot Notation, Bracket Notation, object destructuring**
- **Property value: function array object**

Object

An Object is a collection of properties.

A property is an association between a name (or key) and a value.

For example, a person has a name, age, city, etc. These are the properties of the person

Key	Value
firstName	Rahul
lastName	Attuluri
age	28
city	Delhi

Creating an object: We can add properties into {} as key: value pairs.

```
let person = {
    firstName: "Pawan",
    lastName: "puppala",
    age: 28,
};

console.log(person);     { firstName: 'Pawan', lastName: 'puppala', age: 28 }
```

Identifiers

A valid Identifier should follow the below rules:

- It can contain alphanumeric characters, _ and $
- It cannot start with a number.

Valid Identifiers: invalid identifiers:

```
firstName;
$firstName;
_firstName;
firstName12;
```

```
1 12firstName;
2 firstName 12;
```

```
let person = {
    firstName: "Rahul",
    lastName: "Attuluri",
    age: 28,
    "1": "value1",
    "my choice": "value2",
    };

console.log(person);
```

```
{
  '1': 'value1',
  firstName: 'Rahul',
  lastName: 'Attuluri',
  age: 28,
  'my choice': 'value2'
}
```

Accessing the object properties

```
let person = {
    firstName: "Rahul",
    lastName: "Attuluri",
    age: 28,
    "1": "value1",
    "my choice": "value2",
    };

console.log(person);
console.log(person.firstName);
```

Dot notation:

Note: use dot notation when the key is valid identifier

Bra;cket notation:

```
console.log(person['firstName']);
console.log(person['my choice']);
```

Accessing non existent properties:

```
console.log(person['gender']);
```

o/p:

```
undefined
```

```
let fname = "firstName"
console.log(person.fname);
console.log(person['fname']);
```

```
undefined
undefined
```

Object destructuring:

To unpack properties from Objects, we use Object Destructuring.

The variable name should match with the key of an object.

```
let per = {
    firstName: "pawan",
    lastName: "puppala",
    age: 28,
};

let { gender, age } = per;
console.log(gender);        undefined
console.log(age);           28
```

Modifying objects

Dot Notation

```
let person = {
  name: "Rahul",
  age: 28
};
person.name = "Abhi";
console.log(person.name);
```

Bracket Notation

```
let person = {
  name: "Rahul",
  age: 28
};
person['name'] = "Abhi";
console.log(person['name']);
```

Adding Object Property

Dot Notation

```
let person = {
  name: "Rahul",
  age: 28
};
person.gender = "Male";
console.log(person);
```

Bracket Notation

```
let person = {
  name: "Rahul",
  age: 28
};
person['gender'] = "Male";
console.log(person);
```

Output

```
Object {name: "Rahul", age: 28, gender: "Male"}
```

Property Value

The Value of Object Property can be

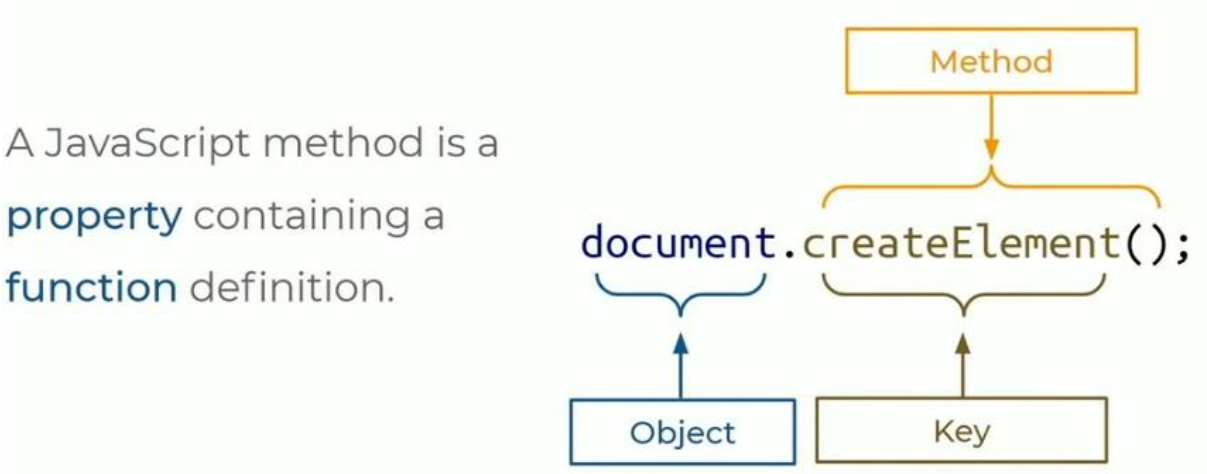
- Function
- Array
- Object

Function as a value

```
let profile = {
  firstName: "pawan",
  lastName: "puppala",
  age: 28,
  run: function () {
    console.log("Start Running.");
  },
};

profile.run();                    Start Running.
```

Methods:



Array as a Value

```
let profile = {
  firstName: "Pawan",
  lastName: "Puppala",
  age: 28,
  habits: ["Playing Chess", "Singing"],
};

console.log(profile.habits);
console.log(profile.habits[0]);
```

```
[ 'Playing Chess', 'Singing' ]
Playing Chess
```

Object as a Value

```
let profile = {
  firstName: "Pawan",
  lastName: "Puppala",
  age: 28,
  habits: ["Playing Chess", "Singing"],
  car: {
    name: "Audi",
    model: "A6",
    color: "White",
  },
};

console.log(profile.car.name);
console.log(profile.car["model"]);
```

```
Audi
A6
```

Single line comments

```
// Change Heading
document.getElementById("heading").textContent = "Web Technologies";

// Change Paragraph
document.getElementById("paragraph").textContent = "HTML, CSS & JS";
```

Multi line comments

Multi-line comments start with /* and end with */

```
/* Multiple Lines
of text or code */
```

Note: Comments are also used to prevent from execution.

W.A. JS P to log an array with the given values ['Orange', 25, 100, true, 33.58]

```
let myArray = ['Orange',25,100,true,33.58];
console.log(myArray);
```

Access the values of an array.
Update values in an array.
Find the length of an array.
Add a value to the end of the array.
Delete the last value of the array.

Call a function

```
function greet() {
  return "Hello! Have a nice day";
}
console.log(greet());
```

Return the sum of two integers passed as arguments to the function

```
let integer11 = 1;
let integer22 = 2;
function getSumOfTwoIntegers(integer1, integer2) {
  let sum = integer1 + integer2;
  return sum;
}

let sumOfTwoIntegers = getSumOfTwoIntegers(integer11, integer22);
console.log(sumOfTwoIntegers);
```

Create a function

```
function getNationalBird() {
  let bird = "Peacock";
  return bird;
}
let nationalBird = getNationalBird();
console.log(nationalBird);
```

Passing an argument to the function

```
process.stdin.resume();
process.stdin.setEncoding("utf-8");
let inputString = "";
let currentLine = 0;
process.stdin.on("data", (inputStdin) => {
    inputString += inputStdin;
});

process.stdin.on("end", (_) => {
    inputString = inputString
        .trim()
        .split("\n")
        .map((str) => str.trim());

    main();
});

function readLine() {
    return inputString[currentLine++];
}

function main() {
    let personName = readLine();

    function greetWithName(personName) {
        let greetings = "Hi " + personName;
        return greetings;
    }

    let result = greetWithName(personName);
    console.log(result);
}
```

Create a function with parameters

```
function main() {
    let firstInteger = parseInt(readLine());
    let secondInteger = parseInt(readLine());

    function getAverageOfTwoNumbers(integer1,integer2) {
            let average = (integer1 + integer2)/2;
            return average;
    }

    let averageOfTwoNumbers = getAverageOfTwoNumbers(firstInteger,secondInteger);
    console.log(averageOfTwoNumbers);
```

Create an expression

```
function main() {
  let minutes = parseInt(readLine());

  let convertMinutesToSeconds = function() {
      let seconds = minutes*60;
      return seconds;
  }

  let result = convertMinutesToSeconds(minutes);
  console.log(result);
}
```

To do's application Introduction:

- Html input element : checkbox
- Html elements : Label
- Dom Manipulations : html For setAttribute

How to add a check box statically using HTML?

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <input type="checkbox" id="mycheckbox"/>
    <label for="mycheckbox">Graduated</label>
  </body>
</html>
```

How to add a check box dynamically using HTML?
Ex1.html

```html
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>

    <script src="ex1.js"></script>
  </body>
</html>
```

Ex1.js

```
let inputElement = document.createElement('input');
inputElement.type = 'checkbox';
inputElement.id = 'myCheckbox';
document.body.appendChild(inputElement);

let labelElement = document.createElement('label');
labelElement.htmlFor = "myCheckbox";
labelElement.textContent = "Graduated";
document.body.appendChild(labelElement)
```

We can use
setAttribute() method to set any HTML attribute name and its corresponding value. If the attribute already exists, the value is updated. Otherwise, a new attribute is added with the specified name and value.
Syntax: Element.setAttribute(name, value);

```
let labelElement = document.createElement('label');
//labelElement.htmlFor = "myCheckbox";
labelElement.setAttribute('for','myCheckbox');
labelElement.textContent = "Graduated";
document.body.appendChild(labelElement)
```

Steps:

- Create a single Todo Item
- Create Multiple Todo items
- Take User Input and Create Todos Dynamically
- Add Delete Todo item Functionality

Lets create To do item Statically

```
<div class="todos-bg-container">
    <div class="container">
        <div class="row">
            <div class="col-12">
                <h1 class="todos-heading">Todos</h1>
                <h1 class="create-task-heading">
                    Create <span class="create-task-heading-subpart">Task</span>
                </h1>
                <input type="text" id="todoUserInput" class="todo-user-input" />
                <button class="add-todo-button">Add</button>
                <h1 class="todo-items-heading">
                    My <span class="todo-items-heading-subpart">Tasks</span>
                </h1>
                <ul class="todo-items-container" id="todoItemsContainer">

                    <li class="todo-items-container d-flex flex-row">
                        <input type="checkbox" class="checkbox-input" id="checkboxInput"/>
                        <div class="label-container d-flex flex-row">
                            <label for="checkboxInput" class="checkbox-label">Learn HTML</label>
                            <div class="delete-icon-container">
                                <i class="far fa-trash-alt delete-icon"></i>
                            </div>
                        </div>
                    </li>

                </ul>
            </div>
        </div>
    </div>
</div>
```

To do's Application:

- Loops: for.. of loop
- To do Application: adding multiple To do's Dynamically

Application Flow

Creating to do element

HTML

```
<ul class="todo-items-container" id="todoItemsContainer">
  <li class="todo-item-container d-flex flex-row">
  </li>
</ul>
```

JS

```
let todoElement = document.createElement("li");
todoElement.classList.add("todo-item-container", "d-flex", "flex-row");
todoItemsContainer.appendChild(todoElement);
console.log(todoItemsContainer);
```

Creating a checkbox

HTML

```
<li class="todo-item-container d-flex flex-row">
  <input type="checkbox" id="checkboxInput" class="checkbox-input" />
</li>
```

JS

```
let inputElement = document.createElement("input");
inputElement.type = "checkbox";
inputElement.id = "checkboxInput";
inputElement.classList.add("checkbox-input");
todoElement.appendChild(inputElement);
```

Creating a label container

HTML

```
...
<input type="checkbox" id="checkboxInput" class="checkbox-input" />
<div class="d-flex flex-row label-container">
</div>
...
```

JS

```
let labelContainer = document.createElement("div");
labelContainer.classList.add("label-container", "d-flex", "flex-row");
todoElement.appendChild(labelContainer);
```

Creating label element dynamically

HTML

```
<div class="d-flex flex-row label-container">
  <label for="checkboxInput" class="checkbox-label">
    Learn HTML
  </label>
</div>
```

JS

```
let labelElement = document.createElement("label");
labelElement.setAttribute("for", "checkboxInput");
labelElement.classList.add("checkbox-label");
labelElement.textContent = "Learn HTML";

labelContainer.appendChild(labelElement);
```

Creating a delete icon container

HTML

```
<div class="d-flex flex-row label-container">
  ...
  <div class="delete-icon-container">
    <i class="far fa-trash-alt delete-icon"></i>
  </div>
</div>
```

JS

```
let deleteIconContainer = document.createElement("div");
deleteIconContainer.classList.add("delete-icon-container");

labelContainer.appendChild(deleteIconContainer);
```

Adding icon

HTML

```
<div class="d-flex flex-row label-container">
  ...
  <div class="delete-icon-container">
    <i class="far fa-trash-alt delete-icon"></i>
  </div>
</div>
```

JS

```
let deleteIcon = document.createElement("i");
deleteIcon.classList.add("far", "fa-trash-alt", "delete-icon");

deleteIconContainer.appendChild(deleteIcon);
```

Creating single to do Object

```
let todoItemsContainer = document.getElementById("todoItemsContainer")

let todo1 = {
  text: "Learn HTML"
}
...
```

Creating multiple To do objects

```
let todo1 = {
  text: "Learn HTML"
}
let todo2 = {
  text: "Learn CSS"
}
let todo3 = {
  text: "Learn JavaScript"
}
```

Creating list of To do Objects

```
...
let todoList = [
  {
    text: "Learn HTML"
  },
  {
    text: "Learn CSS"
  },
  {
    text: "Learn JavaScript"
  }
];
...
```

```
let todoItemsContainer = document.getElementById("todoItemsContainer");
let todoList = [
  {
    text: "Learn HTML"
  },
  {
    text: "Learn CSS"
  },
  {
    text: "Learn JavaScript"
  }
];
```

```
function createAndAppendTodo(todo) {
    let todoElement = document.createElement("li");
    todoElement.classList.add("todo-item-container", "d-flex", "flex-row");
    todoItemsContainer.appendChild(todoElement);

    let inputElement = document.createElement("input");
    inputElement.type = "checkbox";
    inputElement.id = "checkboxInput";
    inputElement.classList.add("checkbox-input");
    todoElement.appendChild(inputElement);

    let labelContainer = document.createElement("div");
    labelContainer.classList.add("label-container", "d-flex", "flex-row");
    todoElement.appendChild(labelContainer);

    let labelElement = document.createElement("label");
    labelElement.setAttribute("for", "checkboxInput");
    labelElement.classList.add("checkbox-label");
    labelElement.textContent = todo.text;
    labelContainer.appendChild(labelElement);

    let deleteIconContainer = document.createElement("div");
    deleteIconContainer.classList.add("delete-icon-container");
    labelContainer.appendChild(deleteIconContainer);

    let deleteIcon = document.createElement("i");
    deleteIcon.classList.add("far", "fa-trash-alt", "delete-icon");
    deleteIconContainer.appendChild(deleteIcon);
  }

for (let todo of todoList) {
createAndAppendTodo(todo);
}
```

Approach to develop a layout Dynamically

- Adding the background container
-