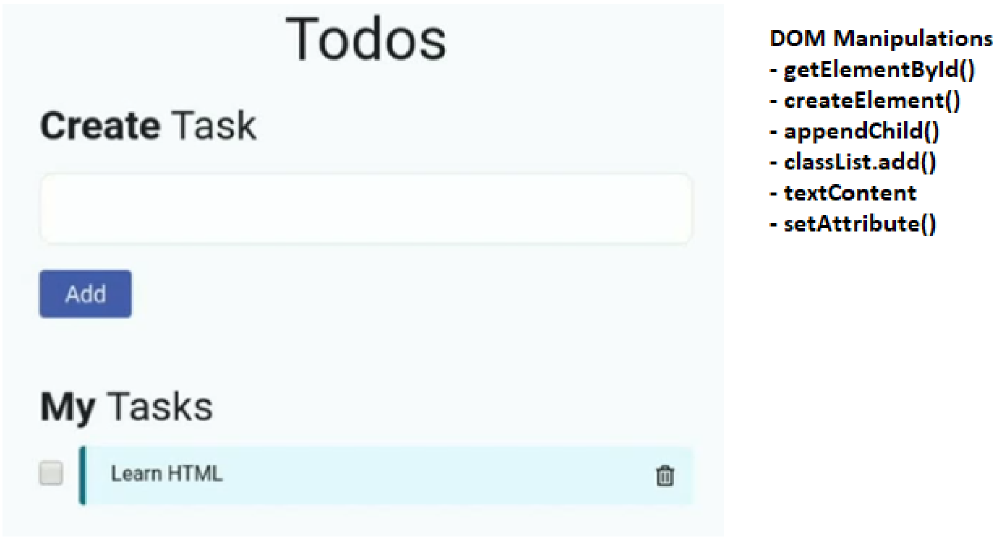
In [1]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js1.png",width=500, height=20)
```

Out[1]:



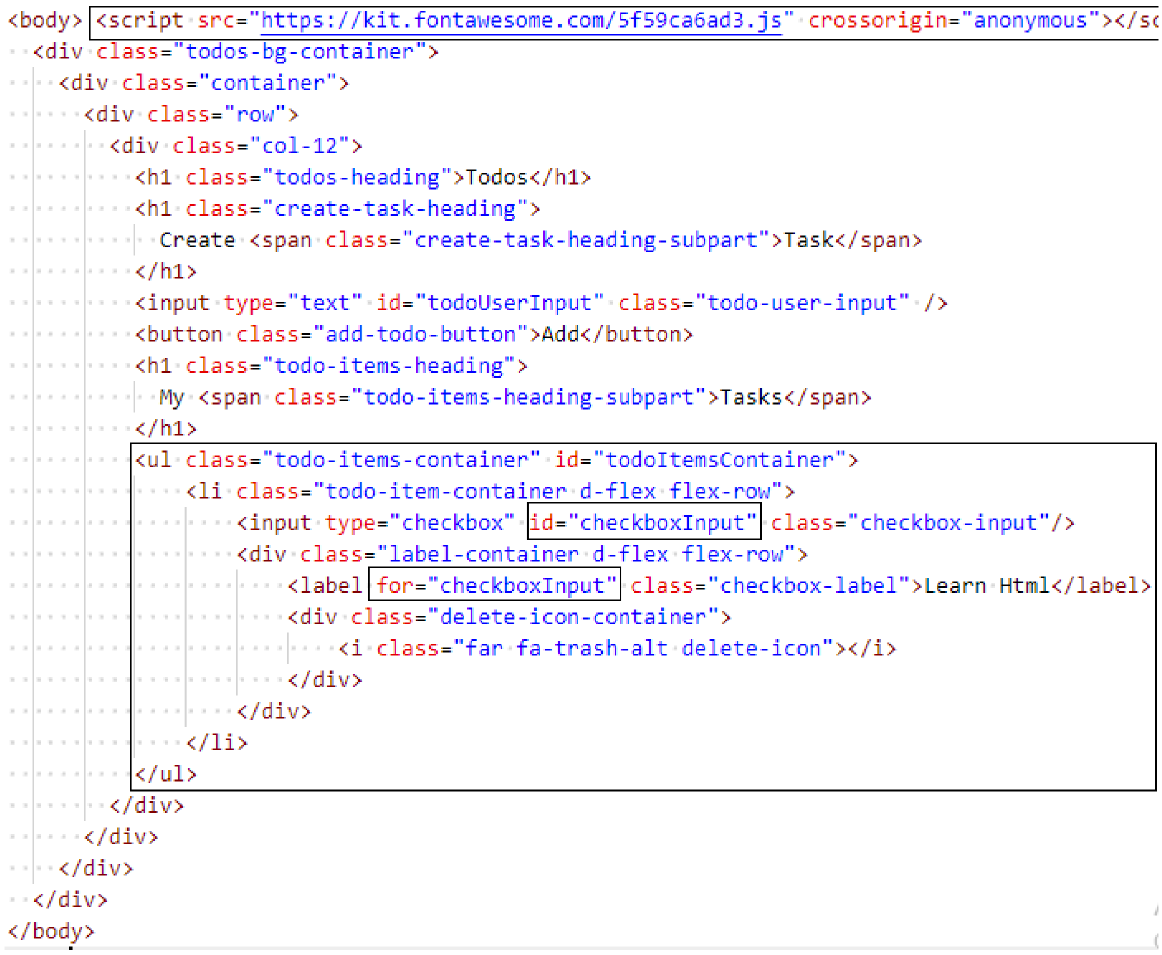
DOM Manipulations
- getElementById()
- createElement()
- appendChild()
- classList.add()
- textContent
- setAttribute()

Lets Create the Todo item statically

In [2]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js2.png")
```

Out[2]:

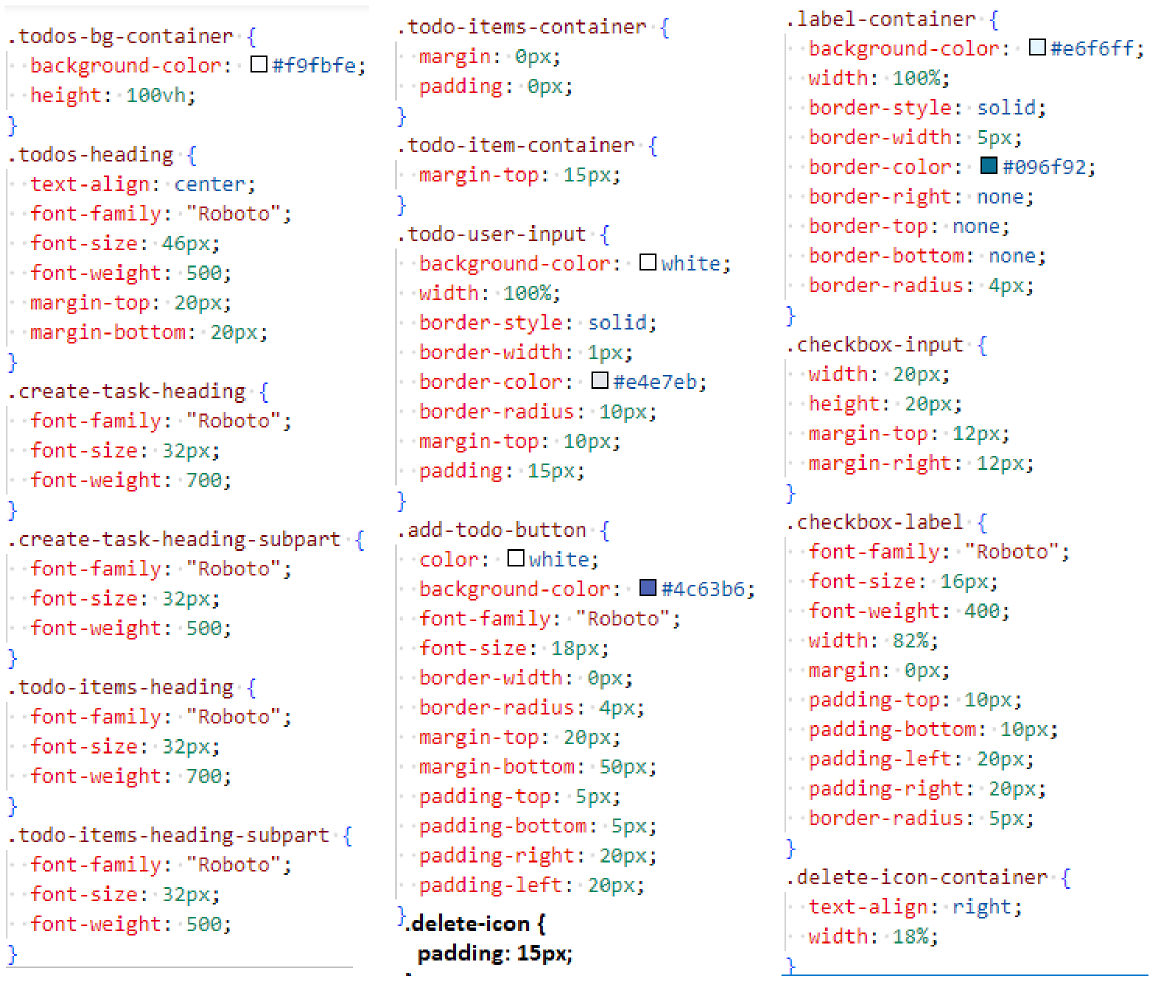


```
<body> <script src="https://kit.fontawesome.com/5f59ca6ad3.js" crossorigin="anonymous"></sc
  <div class="todos-bg-container">
    <div class="container">
      <div class="row">
        <div class="col-12">
          <h1 class="todos-heading">Todos</h1>
          <h1 class="create-task-heading">
            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-item-container d-flex flex-row">
                  <input type="checkbox" id="checkboxInput" class="checkbox-input"/>
                  <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>
</body>
```

In [3]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/js3.png")
```

Out[3]:

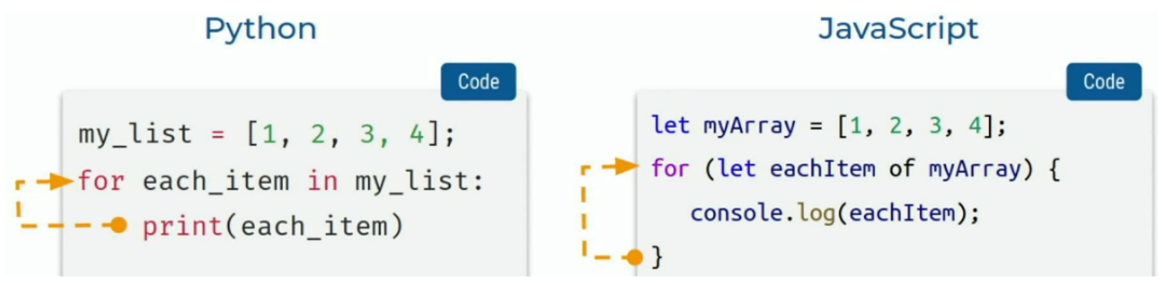


```css
.todos-bg-container {
  background-color: #f9fbfe;
  height: 100vh;
}
.todos-heading {
  text-align: center;
  font-family: "Roboto";
  font-size: 46px;
  font-weight: 500;
  margin-top: 20px;
  margin-bottom: 20px;
}
.create-task-heading {
  font-family: "Roboto";
  font-size: 32px;
  font-weight: 700;
}
.create-task-heading-subpart {
  font-family: "Roboto";
  font-size: 32px;
  font-weight: 500;
}
.todo-items-heading {
  font-family: "Roboto";
  font-size: 32px;
  font-weight: 700;
}
.todo-items-heading-subpart {
  font-family: "Roboto";
  font-size: 32px;
  font-weight: 500;
}
.todo-items-container {
  margin: 0px;
  padding: 0px;
}
.todo-item-container {
  margin-top: 15px;
}
.todo-user-input {
  background-color: white;
  width: 100%;
  border-style: solid;
  border-width: 1px;
  border-color: #e4e7eb;
  border-radius: 10px;
  margin-top: 10px;
  padding: 15px;
}
.add-todo-button {
  color: white;
  background-color: #4c63b6;
  font-family: "Roboto";
  font-size: 18px;
  border-width: 0px;
  border-radius: 4px;
  margin-top: 20px;
  margin-bottom: 50px;
  padding-top: 5px;
  padding-bottom: 5px;
  padding-right: 20px;
  padding-left: 20px;
}.delete-icon {
  padding: 15px;
.label-container {
  background-color: #e6f6ff;
  width: 100%;
  border-style: solid;
  border-width: 5px;
  border-color: #096f92;
  border-right: none;
  border-top: none;
  border-bottom: none;
  border-radius: 4px;
}
.checkbox-input {
  width: 20px;
  height: 20px;
  margin-top: 12px;
  margin-right: 12px;
}
.checkbox-label {
  font-family: "Roboto";
  font-size: 16px;
  font-weight: 400;
  width: 82%;
  margin: 0px;
  padding-top: 10px;
  padding-bottom: 10px;
  padding-left: 20px;
  padding-right: 20px;
  border-radius: 5px;
}
.delete-icon-container {
  text-align: right;
  width: 18%;
}
```

In [4]:

```python
from IPython.display import Image
Image("E:/code/frontend/img/js4.png")
```

Out[4]:



```python
my_list = [1, 2, 3, 4];
for each_item in my_list:
  print(each_item)
```

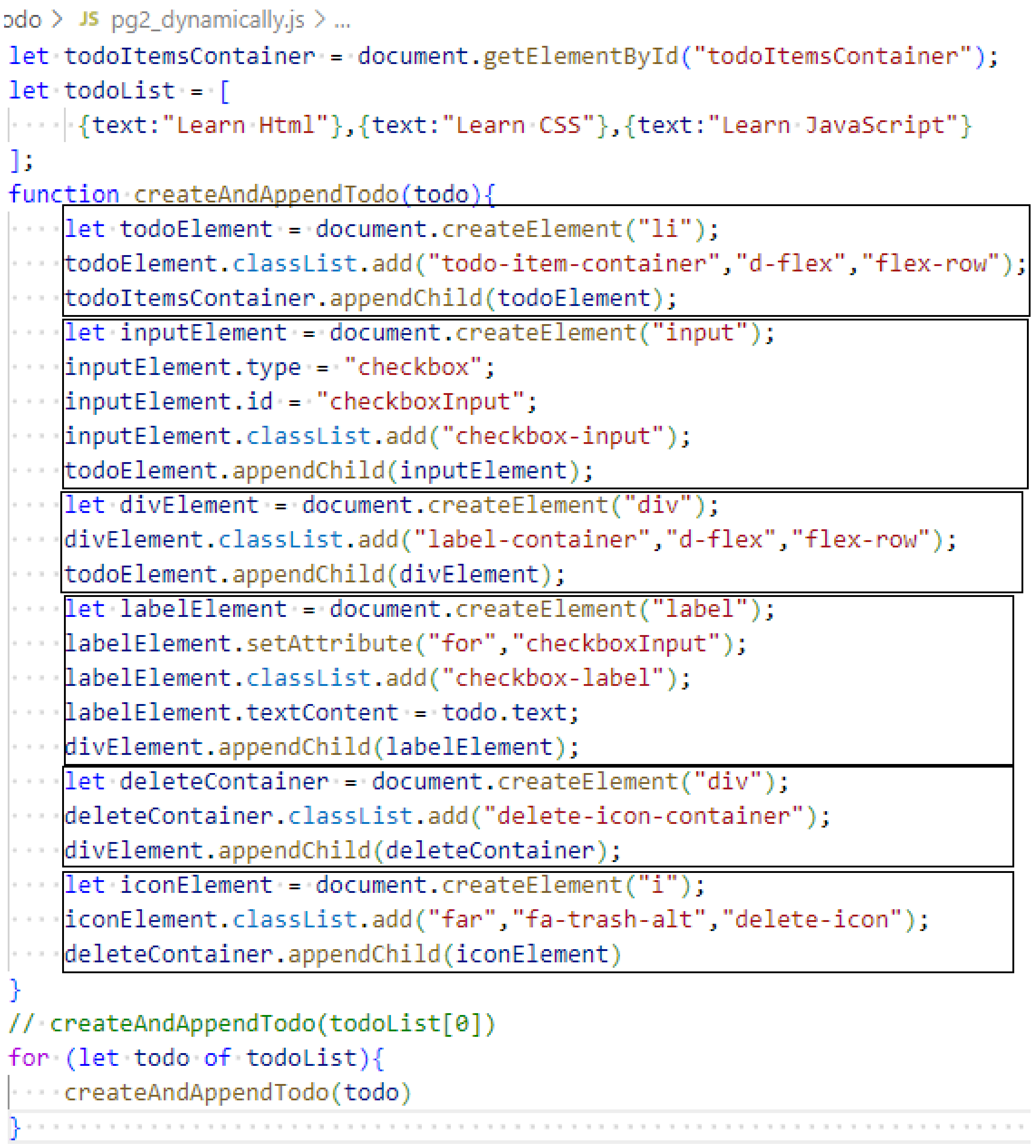
```javascript
let myArray = [1, 2, 3, 4];
for (let eachItem of myArray) {
  console.log(eachItem);
}
```

Lets Create the Todo item dynamically

In [5]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js5.png")
```

Out[5]:



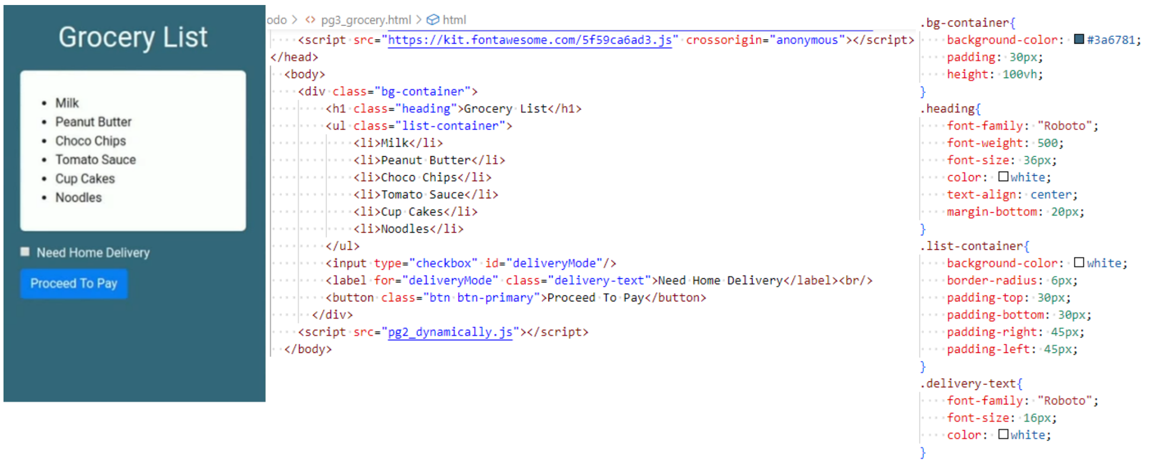
```
odo > JS pg2_dynamically.js > ...
let todoItemsContainer = document.getElementById("todoItemsContainer");
let todoList = [
    {text:"Learn Html"},{text:"Learn CSS"},{text:"Learn JavaScript"}
];
function createAndAppendTodo(todo){
    let todoElement = document.createElement("li");
    todoElement.classList.add("todo-item-container","d-flex","flex-row");
    todoItemsContainer.appendChild(todoElement);
    let inputElement = document.createElement("input");
    inputElement.type = "checkbox";
    inputElement.id = "checkboxInput";
    inputElement.classList.add("checkbox-input");
    todoElement.appendChild(inputElement);
    let divElement = document.createElement("div");
    divElement.classList.add("label-container","d-flex","flex-row");
    todoElement.appendChild(divElement);
    let labelElement = document.createElement("label");
    labelElement.setAttribute("for","checkboxInput");
    labelElement.classList.add("checkbox-label");
    labelElement.textContent = todo.text;
    divElement.appendChild(labelElement);
    let deleteContainer = document.createElement("div");
    deleteContainer.classList.add("delete-icon-container");
    divElement.appendChild(deleteContainer);
    let iconElement = document.createElement("i");
    iconElement.classList.add("far","fa-trash-alt","delete-icon");
    deleteContainer.appendChild(iconElement)
}
// createAndAppendTodo(todoList[0])
for (let todo of todoList){
    createAndAppendTodo(todo)
}
```

Approach to develop a Layout Statically

In [6]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js6.png")
```

Out[6]:



```
    <script src="https://kit.fontawesome.com/5f59ca6ad3.js" crossorigin="anonymous"></script>
</head>
  <body>
    <div class="bg-container">
        <h1 class="heading">Grocery List</h1>
        <ul class="list-container">
            <li>Milk</li>
            <li>Peanut Butter</li>
            <li>Choco Chips</li>
            <li>Tomato Sauce</li>
            <li>Cup Cakes</li>
            <li>Noodles</li>
        </ul>
        <input type="checkbox" id="deliveryMode"/>
        <label for="deliveryMode" class="delivery-text">Need Home Delivery</label><br/>
        <button class="btn btn-primary">Proceed To Pay</button>
      </div>
    <script src="pg2_dynamically.js"></script>
  </body>
```

```
.bg-container{
    background-color: #3a6781;
    padding: 30px;
    height: 100vh;
}
.heading{
    font-family: "Roboto";
    font-weight: 500;
    font-size: 36px;
    color: white;
    text-align: center;
    margin-bottom: 20px;
}
.list-container{
    background-color: white;
    border-radius: 6px;
    padding-top: 30px;
    padding-bottom: 30px;
    padding-right: 45px;
    padding-left: 45px;
}
.delivery-text{
    font-family: "Roboto";
    font-size: 16px;
    color: white;
}
```

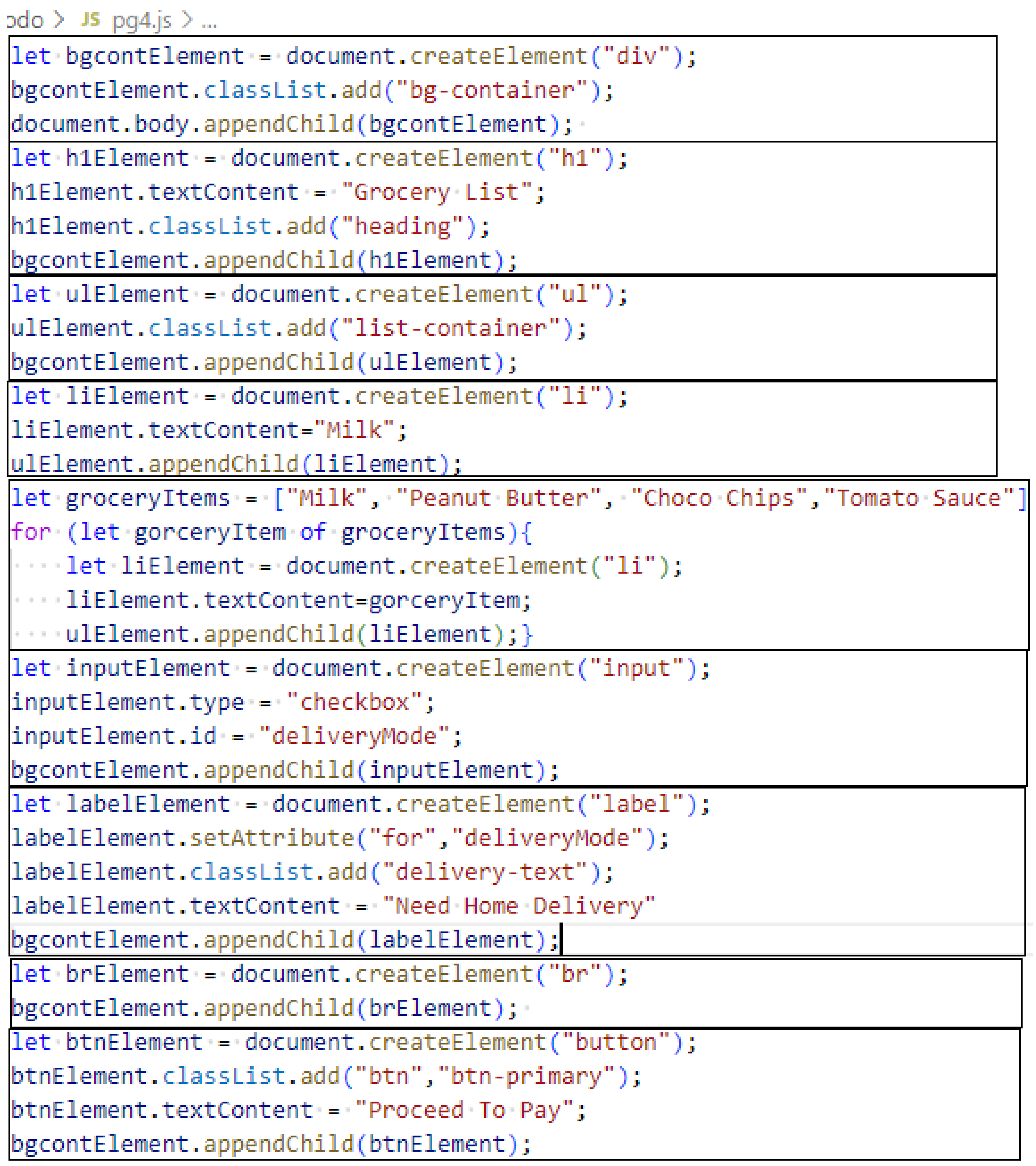
Approach to develop a Layout Statically

- html page with only body element
- css file with all styles

In [7]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js7.png")
```

Out[7]:



odo > JS pg4.js > ...

```
let bgcontElement = document.createElement("div");
bgcontElement.classList.add("bg-container");
document.body.appendChild(bgcontElement);
let h1Element = document.createElement("h1");
h1Element.textContent = "Grocery List";
h1Element.classList.add("heading");
bgcontElement.appendChild(h1Element);
let ulElement = document.createElement("ul");
ulElement.classList.add("list-container");
bgcontElement.appendChild(ulElement);
let liElement = document.createElement("li");
liElement.textContent="Milk";
ulElement.appendChild(liElement);
let groceryItems = ["Milk", "Peanut Butter", "Choco Chips","Tomato Sauce"]
for (let gorceryItem of groceryItems){
    let liElement = document.createElement("li");
    liElement.textContent=gorceryItem;
    ulElement.appendChild(liElement);}
let inputElement = document.createElement("input");
inputElement.type = "checkbox";
inputElement.id = "deliveryMode";
bgcontElement.appendChild(inputElement);
let labelElement = document.createElement("label");
labelElement.setAttribute("for","deliveryMode");
labelElement.classList.add("delivery-text");
labelElement.textContent = "Need Home Delivery"
bgcontElement.appendChild(labelElement);
let brElement = document.createElement("br");
bgcontElement.appendChild(brElement);
let btnElement = document.createElement("button");
btnElement.classList.add("btn","btn-primary");
btnElement.textContent = "Proceed To Pay";
bgcontElement.appendChild(btnElement);
```

1. HTML Input Element

1.1 Placeholder

- Placeholder is the text that appears in the HTML input element when no value is set. We can specify it using the HTML attribute placeholder.

2. JavaScript Built-in Functions

2.1 alert()

- The alert() function displays an alert box with a specified message and an OK button.

3. DOM Properties

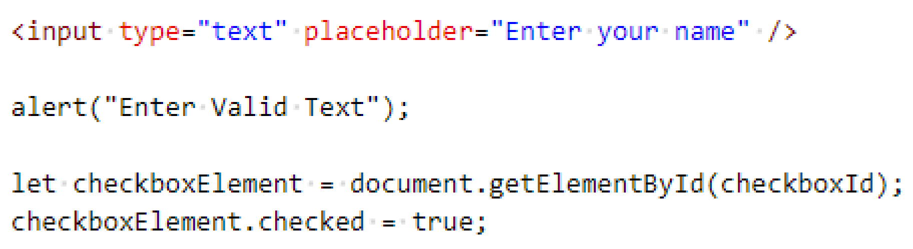
3.1 Checked

The checked property sets or returns the checked status of an HTML checkbox input element as a boolean value.

In [8]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js18.png")
```

Out[8]:



```
<input type="text" placeholder="Enter your name" />

alert("Enter Valid Text");

let checkboxElement = document.getElementById(checkboxId);
checkboxElement.checked = true;
```

Enhancements

1. Fixing checkbox issue

- we have to specify a Unique ID to each checkbox
- provide the same ID to the labels for attribute

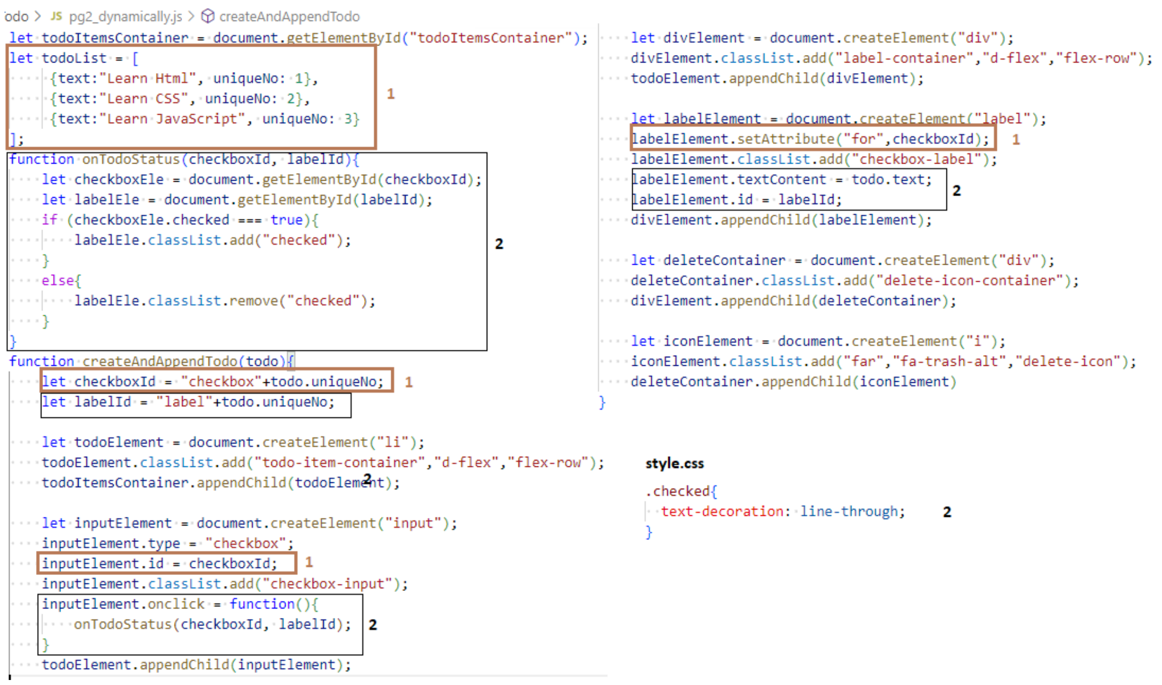
2. Striking through the label when selected

- adding required CSS to strike the text
- specifying ID to each Label Element
- Adding Event Listeners to Checkboxes
- Accesing the checkbox Elements

In [9]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js20.png")
```

Out[9]:



```
let todoItemsContainer = document.getElementById("todoItemsContainer");
let todoList = [
    {text:"Learn Html", uniqueNo: 1},
    {text:"Learn CSS", uniqueNo: 2},
    {text:"Learn JavaScript", uniqueNo: 3}
];
function onTodoStatus(checkboxId, labelId){
    let checkboxEle = document.getElementById(checkboxId);
    let labelEle = document.getElementById(labelId);
    if (checkboxEle.checked === true){
        labelEle.classList.add("checked");
    }
    else{
        labelEle.classList.remove("checked");
    }
}
function createAndAppendTodo(todo){
    let checkboxId = "checkbox"+todo.uniqueNo;
    let labelId = "label"+todo.uniqueNo;

    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 = checkboxId;
    inputElement.classList.add("checkbox-input");
    inputElement.onclick = function(){
        onTodoStatus(checkboxId, labelId);
    }
    todoElement.appendChild(inputElement);
    let divElement = document.createElement("div");
    divElement.classList.add("label-container","d-flex","flex-row");
    todoElement.appendChild(divElement);

    let labelElement = document.createElement("label");
    labelElement.setAttribute("for",checkboxId);
    labelElement.classList.add("checkbox-label");
    labelElement.textContent = todo.text;
    labelElement.id = labelId;
    divElement.appendChild(labelElement);

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

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

```
.checked{
  text-decoration: line-through;
}
```

4. DOM Manipulations

4.1 The removeChild() Method

- The removeChild() method removes an HTML child element of the specified HTML parent element from the DOM and returns the removed HTML child element.

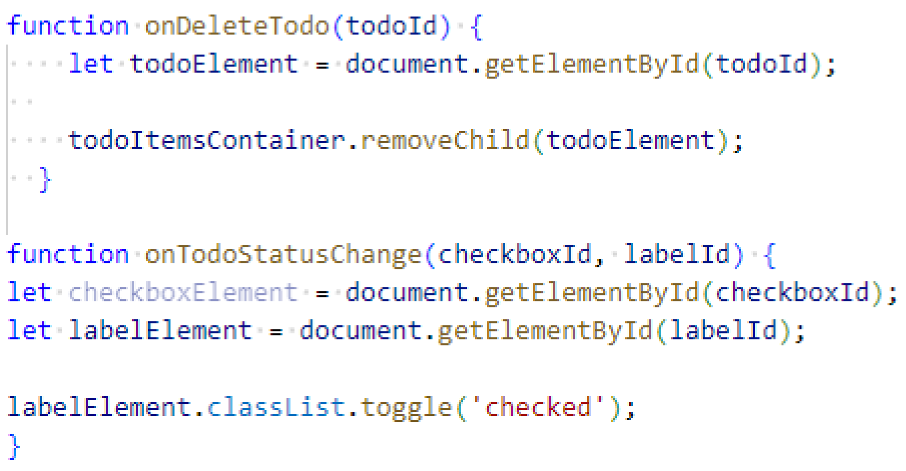
4.2 The classList.toggle() Method

- The classList.toggle() method is used to toggle between adding and removing a class name from an HTML element.

In [10]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js19.png")
```

Out[10]:



```
function onDeleteTodo(todoId) {
    let todoElement = document.getElementById(todoId);
  
    todoItemsContainer.removeChild(todoElement);
  }

function onTodoStatusChange(checkboxId, labelId) {
let checkboxElement = document.getElementById(checkboxId);
let labelElement = document.getElementById(labelId);

labelElement.classList.toggle('checked');
}
```

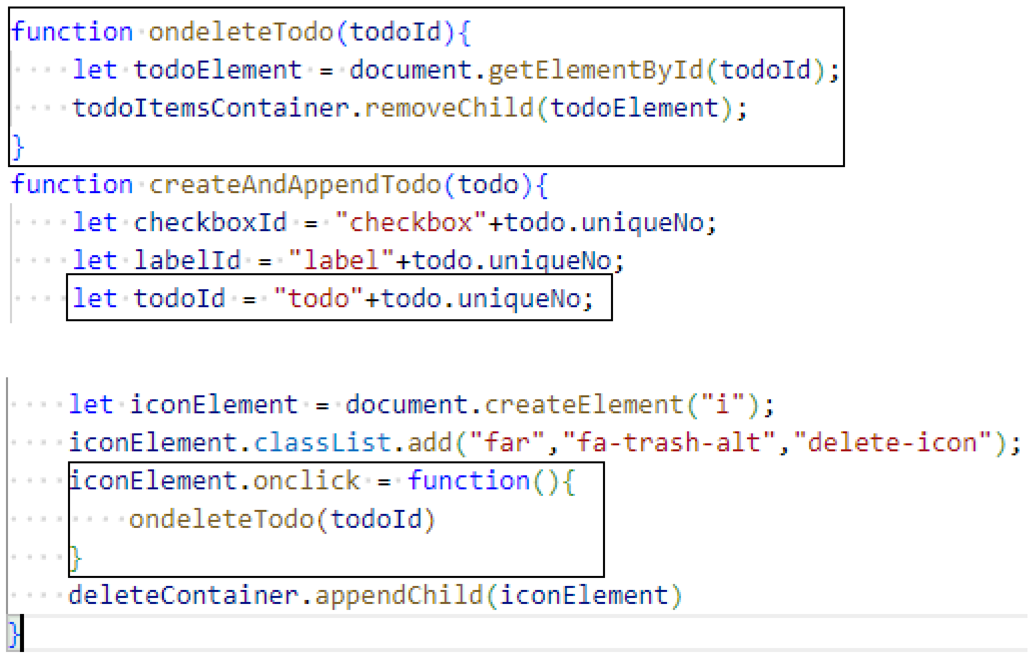
Deleting ToDo item

- Specifying ID to each Todo item
- Add Event Listeners to Delete icon
- Delete Todo item from the Todo item container

In [11]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js21.png")
```

Out[11]:



```
function ondeleteTodo(todoId){
    let todoElement = document.getElementById(todoId);
    todoItemsContainer.removeChild(todoElement);
}
function createAndAppendTodo(todo){
    let checkboxId = "checkbox"+todo.uniqueNo;
    let labelId = "label"+todo.uniqueNo;
    let todoId = "todo"+todo.uniqueNo;


    let iconElement = document.createElement("i");
    iconElement.classList.add("far","fa-trash-alt","delete-icon");
    iconElement.onclick = function(){
        ondeleteTodo(todoId)
    }
    deleteContainer.appendChild(iconElement)
}
```

Adding ToDo item

- Add Event Listner to the Add button
- Access user input value
- create new todo item

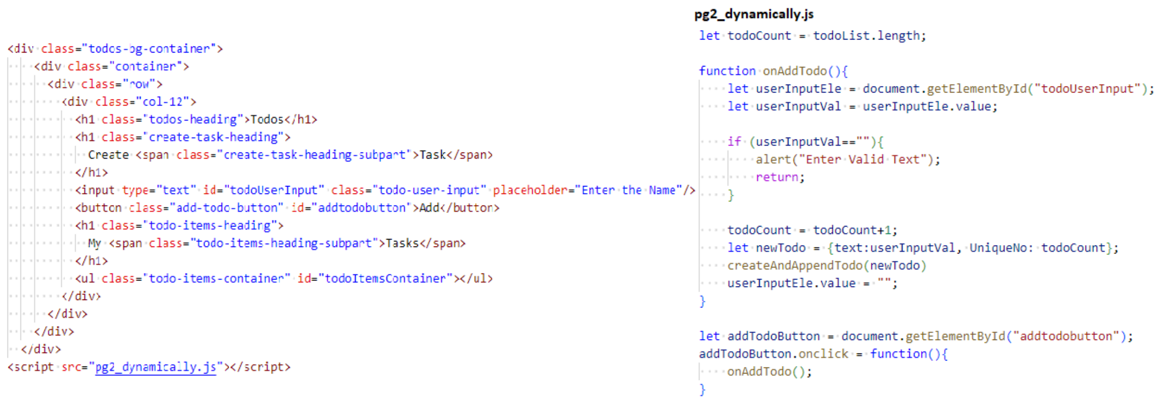
showing warning message

placeholder text

In [12]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js22.png")
```

Out[12]:



```
<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" placeholder="Enter the Name"/>
                <button class="add-todo-button" id="addtodobutton">Add</button>
                <h1 class="todo-items-heading">
                    My <span class="todo-items-heading-subpart">Tasks</span>
                </h1>
                <ul class="todo-items-container" id="todoItemsContainer"></ul>
            </div>
        </div>
    </div>
</div>
<script src="pg2_dynamically.js"></script>
```

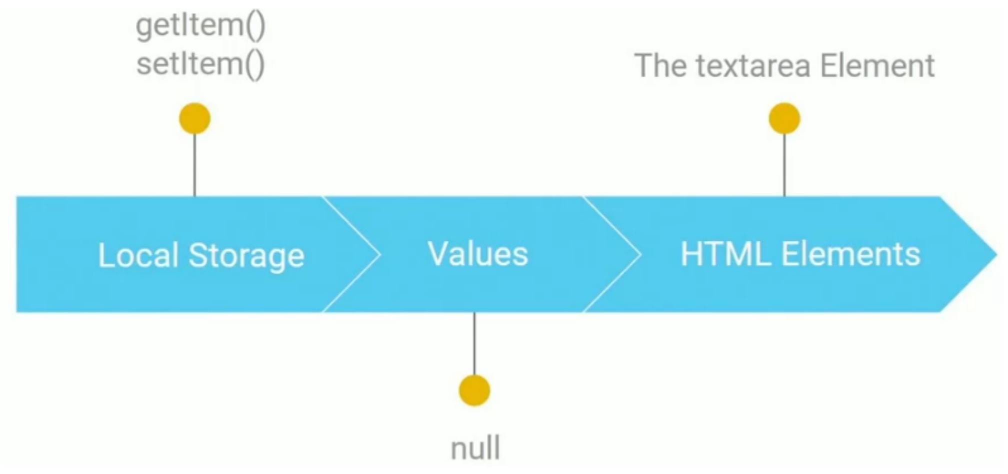
pg2_dynamically.js

```
let todoCount = todoList.length;

function onAddTodo(){
    let userInputEle = document.getElementById("todoUserInput");
    let userInputVal = userInputEle.value;

    if (userInputVal==""){
        alert("Enter Valid Text");
        return;
    }

    todoCount = todoCount+1;
    let newTodo = {text:userInputVal, UniqueNo: todoCount};
    createAndAppendTodo(newTodo)
    userInputEle.value = "";
}

let addTodoButton = document.getElementById("addtodobutton");
addTodoButton.onclick = function(){
    onAddTodo();
}
```

In [13]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js23.png",width=500, height=20)
```

Out[13]:



What happens when we reload the Todos Application?

How to Persist Todo items even on reload?

1. Execution Context

- The environment in which JavaScript Code runs is called Execution Context.
- Execution context contains all the variables, objects, and functions.
- Execution Context is destroyed and recreated whenever we reload an Application.

2. Storage Mechanisms

2.1 Client-Side Data Storage

- Client-Side Data Storage is storing the data on the client (user's machine).
    - Local Storage
    - Session Storage
    - Cookies
    - IndexedDB and many more.

2.2 Server-Side Data Storage

- Server-Side Data Storage is storing the data on the server.

3. Local Storage

- It allows web applications to store data locally within the user's browser.
- It is a Storage Object. Data can be stored in the form of key-value pairs.
- Please note that both key and value must be strings. If their type is other than a string, they get converted to strings automatically.
- To access and work with Local Storage we have below methods:
    - setItem()
    - getItem()
    - clear()
    - removeItem()

3.1 The setItem() Method

- The setItem() method can be used for storing data in the Local Storage.
- Syntax: localStorage.setItem("Key", "Value");

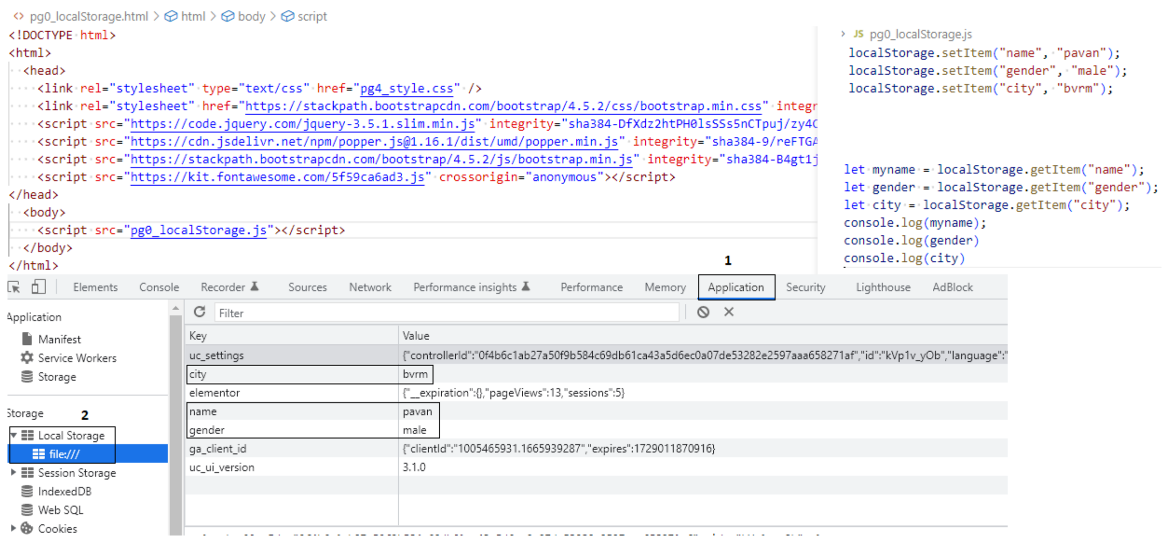
3.2 The getItem() Method

- The getItem() method can be used for getting data from the Local Storage.
- Syntax: localStorage.getItem("Key");

In [14]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js24.png")
```

Out[14]:



pg0_localStorage.html > html > body > script

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="pg4_style.css" />
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integr
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGA
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha384-B4gt1j
    <script src="https://kit.fontawesome.com/5f59ca6ad3.js" crossorigin="anonymous"></script>
</head>
  <body>
    <script src="pg0_localStorage.js"></script>
  </body>
</html>
```

pg0_localStorage.js

```
localStorage.setItem("name", "pavan");
localStorage.setItem("gender", "male");
localStorage.setItem("city", "bvrm");


let myname = localStorage.getItem("name");
let gender = localStorage.getItem("gender");
let city = localStorage.getItem("city");
console.log(myname);
console.log(gender)
console.log(city)
```

Key	Value
uc_settings	{"controllerId":"0f4b6c1ab27a50f9b584c69db61ca43a5d6ec0a07de53282e2597aaa658271af","id":"kVp1v_yOb","language":"
city	bvrm
elementor	{"__expiration":{},"pageViews":13,"sessions":5}
name	pavan
gender	male
ga_client_id	{"clientId":"1005465931.1665939287","expires":1729011870916}
uc_ui_version	3.1.0

4. Values

4.1 null

- We use null in a situation where we intentionally want a variable but don't need a value to it.
- let occupation = localStorage.getItem("occupation");
- console.log(occupation);

5. HTML Elements

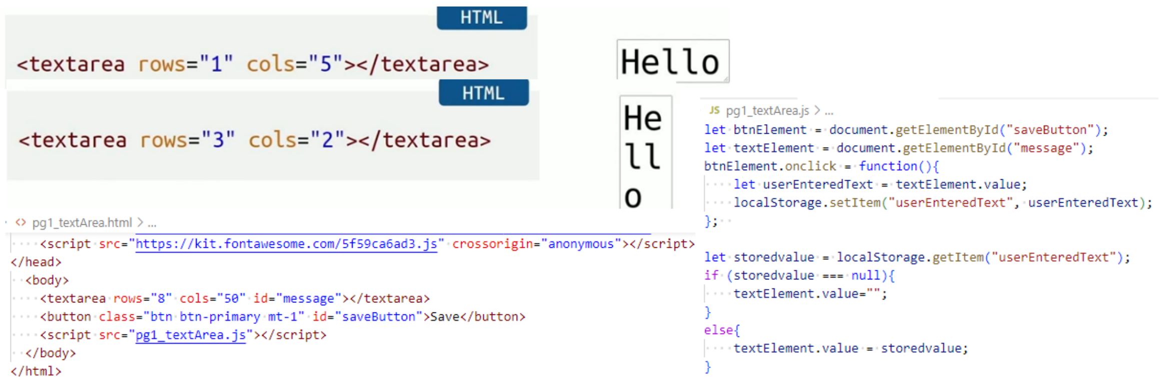
5.1 The textarea Element

- The HTML textarea element can be used to write the multiline text as an input.
  - textarea rows="8" cols="55"></textarea
  - The HTML rows attribute specifies the number of lines.
  - The HTML cols attribute specifies the number of characters per each line.

In [15]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js25.png")
```

Out[15]:



```html
<textarea rows="1" cols="5"></textarea>
```

```html
<textarea rows="3" cols="2"></textarea>
```

pg1_textArea.html > ...

```html
    <script src="https://kit.fontawesome.com/5f59ca6ad3.js" crossorigin="anonymous"></script>
</head>
  <body>
    <textarea rows="8" cols="50" id="message"></textarea>
    <button class="btn btn-primary mt-1" id="saveButton">Save</button>
    <script src="pg1_textArea.js"></script>
  </body>
</html>
```

pg1_textArea.js > ...

```javascript
let btnElement = document.getElementById("saveButton");
let textElement = document.getElementById("message");
btnElement.onclick = function(){
    let userEnteredText = textElement.value;
    localStorage.setItem("userEnteredText", userEnteredText);
};

let storedvalue = localStorage.getItem("userEnteredText");
if (storedvalue === null){
    textElement.value="";
}
else{
    textElement.value = storedvalue;
}
```

1. JavaScript Object Notation (JSON)

- JSON is a data representation format used for:
  - Storing data (Client/Server)
  - Exchanging data between Client and Server

1.1 Supported Types

```
- Number
- String
- Boolean
- Array
- Object
- Null
```

1.2 JS Object vs JSON Object

- In JSON, all keys in an object must be enclosed with double-quotes. While in JS, this is not necessary.

Javascript

- let profile = { name: "pavan", age: 29, designation: "Web Developer" };

JSON:

- let profile = { "name": "pavan", "age": 29, "designation": "Web Developer" };

1.3 JSON Methods

1.3.1 JSON.stringify()

- It converts the given value into JSON string.
- Syntax: JSON.stringify(value)
- JSON.stringify(profile)

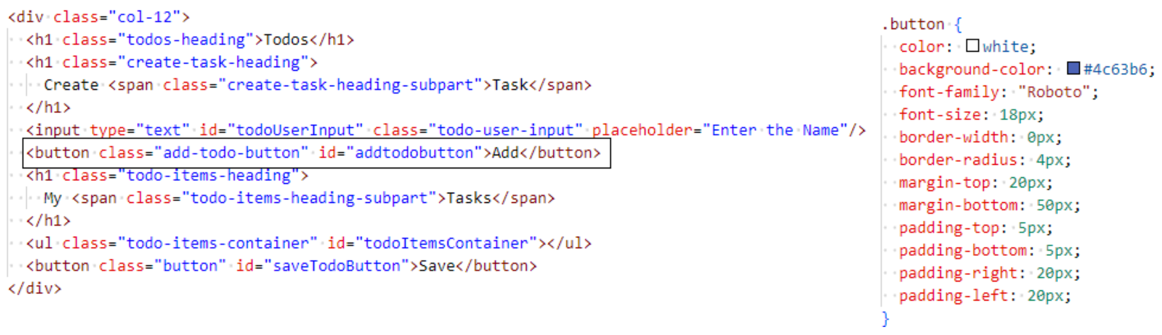
1.3.2 JSON.parse()

- It parses a JSON string and returns a JS object.
- Syntax: JSON.parse(value)
- JSON.parse(profile)

In [16]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js26.png")
```

Out[16]:



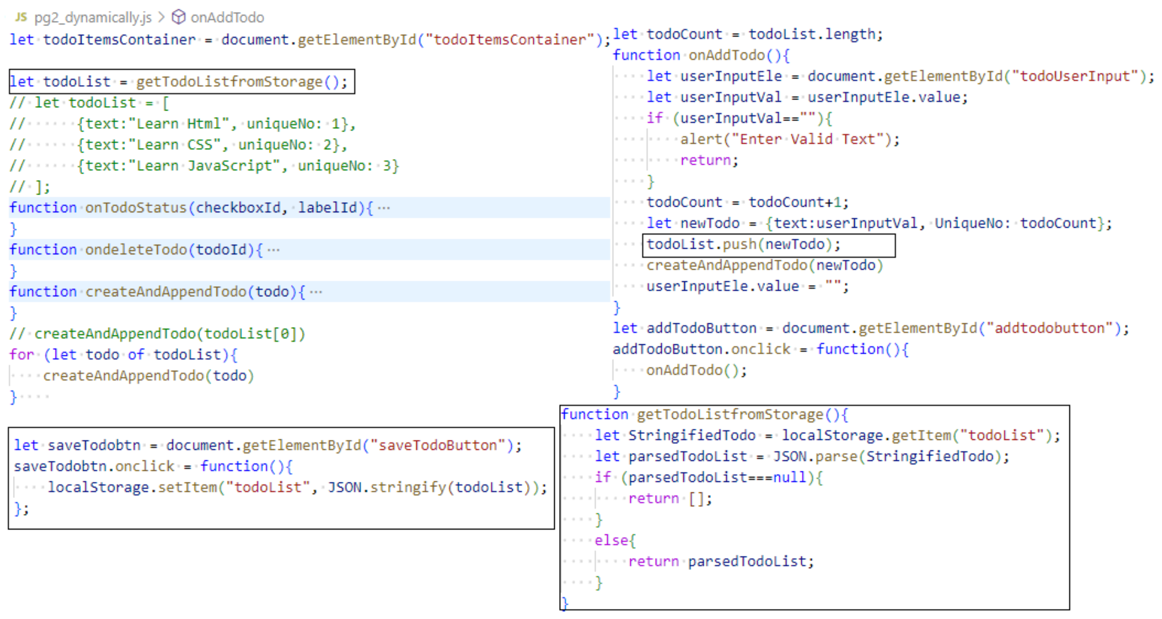
```
<div class="col-12">
  <h1 class="todos-heading">Todos</h1>
  <h1 class="create-task-heading">
    Create <span class="create-task-heading-subpart">Task</span>
  </h1>
  <input type="text" id="todoUserInput" class="todo-user-input" placeholder="Enter the Name"/>
  <button class="add-todo-button"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

```
.button {
  color: white;
  background-color: #4c63b6;
  font-family: "Roboto";
  font-size: 18px;
  border-width: 0px;
  border-radius: 4px;
  margin-top: 20px;
  margin-bottom: 50px;
  padding-top: 5px;
  padding-bottom: 5px;
  padding-right: 20px;
  padding-left: 20px;
}
```

In [17]:

```
from IPython.display import Image
Image("E:/code/frontend/img/js27.png")
```

Out[17]:



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

let todoList = getTodoListfromStorage();
// let todoList = [
//      {text:"Learn Html", uniqueNo: 1},
//      {text:"Learn CSS", uniqueNo: 2},
//      {text:"Learn JavaScript", uniqueNo: 3}
// ];
function onTodoStatus(checkboxId, labelId){ ···
}
function ondeleteTodo(todoId){ ···
}
function createAndAppendTodo(todo){ ···
}
// createAndAppendTodo(todoList[0])
for (let todo of todoList){
    createAndAppendTodo(todo)
}

let saveTodobtn = document.getElementById("saveTodoButton");
saveTodobtn.onclick = function(){
    localStorage.setItem("todoList", JSON.stringify(todoList));
};
```

```
let todoCount = todoList.length;
function onAddTodo(){
    let userInputEle = document.getElementById("todoUserInput");
    let userInputVal = userInputEle.value;
    if (userInputVal==""){
        alert("Enter Valid Text");
        return;
    }
    todoCount = todoCount+1;
    let newTodo = {text:userInputVal, UniqueNo: todoCount};
    todoList.push(newTodo);
    createAndAppendTodo(newTodo)
    userInputEle.value = "";
}
let addTodoButton = document.getElementById("addtodobutton");
addTodoButton.onclick = function(){
    onAddTodo();
}
function getTodoListfromStorage(){
    let StringifiedTodo = localStorage.getItem("todoList");
    let parsedTodoList = JSON.parse(StringifiedTodo);
    if (parsedTodoList===null){
        return [];
    }
    else{
        return parsedTodoList;
    }
}
```