Program1) Callback hell Situation & Solution: using callbacks, clean callbacks, promises & Generators.

Program2) Task List Program: Add Task, Remove particular Task, Clear All Tasks, Filter task, Add Task to Local Storage & Make data persistent using local storage

Programs 3) ES6 class Inheritance: create subclass-superclass relationship using extends keyword, super keyword to call parent class constructor and use parameters available in parent class.

Program 4) BookList Program: Add books, remove books, show alert messages, using ES5 Prototypes syntax and ES6 Classes  
  
Program 5) Fetch API: access file from remote server

Program 6) Create elements dynamically: Create 4 boxes dynamically, all boxes are hidden by default, four buttons are there, if we click on first button, first box needs to be shown and rest all buttons needs to be hidden, if we click on second button then second box needs to be shown and rest all buttons needs to be hidden and so on. Also four more buttons are there to assign background color to box, red-green-blue & yellow buttons, if we click on red button then background color will be red and if we click on green button then background color will be green. Also one cross button will appear on every box. On click of close button, box needs to be hidden.

Program 7): How can we get the user’s current location?

Program 1) callback hell solution:

* A)callback hell issue by multiple nested callbacks: through this callback hell occurs
* B)clean callbacks way,
* C)promises and
* D)generators

index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Callback Hell</title>

<style>

.error { color:red; font-size: 15px; }

</style>

</head>

<body>

<!-- include jquery -->

<script data-require="jquery@2.1.1" data-semver="2.1.1" src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>

<script data-require="traceur@\*" data-semver="0.0.0-20140302" src="https://traceur-compiler.googlecode.com/git/bin/traceur.js"></script>

<!-- bluebird.min.js is required for generators -->

<script src="//cdnjs.cloudflare.com/ajax/libs/bluebird/1.2.2/bluebird.min.js"></script>

<h1>Javascript Async fun!</h1>

<ul id="status"></ul> <!-- ul status end -->

<h4>profile</h4>

<pre id="profile-pre"></pre>

<h4>tweets</h4>

<pre id="tweets-pre"></pre>

<h4>Friend</h4>

<pre id="friend-pre"></pre>

<!-- async start here:

all below methods are executing asynchronously, Remember, if first request is unable to execute then next 2 requests will not be executed, similarly if 2nd request is unable to execute in that case, 3rd request will not be executed.. This is because they are rely upon each other

-->

<!-- 1) using callbacks -->

<!-- callback hell here: multiple callbacks putting inside each other and for each and every request, separate error message will be there, it looks very ugly. Your code becomes extremely hard to understand and hard to maintain. See below code: -->

<!--

<script>

var $status = $("#status");

// first ajax request..

$.ajax({ // $.ajax is used to make ajax request, request to a remote server

type: 'GET',

url:'profile.json', //rest api

success: function(profile) { // parameter holds the data fetched from server

$status.append('<li>got profile</li>'); // append li into ul status

$('#profile-pre').html(JSON.stringify(profile)) // added fetched data to profile-pre

// 2nd ajax request inside 1st request

$.ajax({

type: 'GET',

url:'tweets.json',

success:function(tweets) { // parameter holds the data fetched from server

$status.append('<li>get tweets</li>'),

$('#tweets-pre').html(JSON.stringify(tweets));

// 3rd ajax request inside 2nd request

$.ajax({

type:'GET',

url: 'friend.json',

success: function(friend) {

$status.append('<li>got mentioned friend</li>');

$('#friend-pre').html(JSON.stringify(friend));

},

error: function(xhr, status, error){

$status.append('<li class="error">error in third request:'+error.toString()+'</li>');

}

}) // 3rd ajax request end

}, // 2nd ajax success end

error: function(xhr, status, error) {

$status.append('<li class="error">error in second request:'+error.toString()+'</li>');

}

}) // 2nd ajax request end

}, // first ajax success end

error: function(xhr, status, error) {

$status.append('<li class="error">error in first request error:'+error.toString()+'</li>');

}

})

</script>

-->

<!-- 2) clean callbacks way: only one error message will be handled for all requests and on success we will be calling one function, it looks more clean than callback hell -->

<!--

<script>

var $status = $('#status');

$.ajax({

type:'GET',

url:'profile.json',

success: getTweets, // call getTweets function on success

error:handleError /\* only one error function, its better than callback hell where we need to initilize 3 error messages for every request respectively \*/

});

function getTweets(profile){

$status.append('<li>got profile</li>');

$("#profile-pre").html(JSON.stringify(profile));

$.ajax({

type:'GET',

url:'tweets.json',

success:getMentionedUser, // call getMentionedUser function on success

error:handleError // only one error function calling for every ajax request

});

} // function getTweets end

function getMentionedUser(tweets) {

$status.append('<li>got tweets</li>');

$("#tweets-pre").html(JSON.stringify(tweets));

$.ajax({

type:"GET",

url:'friend.json',

success: function(friend) {

$status.append('<li>got friend</li>');

$("#friend-pre").html(JSON.stringify(friend));

},

error: handleError // only one error function calling for every ajax request

});

}

function handleError(xhr, status, error) { // only one error function handling for all ajax requests

$status.append("<li>error occured:" + error.toString() + "</li>")

}

</script>

-->

<!-- 3) using promises: Best way -->

<!--

<script>

var $status = $("#status");

$.get('profile.json') /\*The $.get() method requests data from the server with an HTTP GET request \*/

.then(function(profile) { /\* first then start, and it will return remote json file to next 'then' \*/

$status.append('<li>got profile</li>');

$("#profile-pre").html(JSON.stringify(profile));

return $.get('tweets.json');

}) /\* first then closed \*/

.then(function(tweets){ /\* second then start, and it will return remote json file to next 'then' \*/

$status.append('<li>got tweets</li>');

$("#tweets-pre").html(JSON.stringify(tweets));

return $.get('friend.json');

}) // second then closed

.then(function(friend) {

$status.append("<li>got friend</li>");

$("#friend-pre").html(JSON.stringify(friend));

}, handleError); /\* third then closed alongwith error handling message for all ajax requests \*/

function handleError(xhr, status, error) {

$status.append("<li>error </li>" + error.toString() + "</li>")

}

/\* $.get and $.post method requests data from the server \*/

</script>

-->

<!-- 4) using ES6 generators -->

<script>

var $status = $('#status');

Promise.coroutine(function\* () {

var profile = yield $.get('profile.json'); /\* this request will execute first then second and third request will be executed, if it fails then next requests will not be executed \*/

$status.append('<li>got profile</li>');

$('#profile-pre').html(JSON.stringify(profile));

var tweets = yield $.get('tweets.json'); /\* this request will execute secondly \*/

$status.append('<li>got tweets</li>');

$('#tweets-pre').html(JSON.stringify(tweets));

var friend = yield $.get('friend.json');

$status.append('<li>got friend</li>');

$('#friend-pre').html(JSON.stringify(friend));

})().catch(function(errs) {

//handle errors on any events

})

</script>

</body>

</html>

--------------------------index.html end---------------------  
-------------------------------------------------------------------

profile.json  
{

"id": 12302151,

"username": "supafly2014",

"authLevel": "user",

"prefs": {

"homepageTweetsToShow": 2,

"showSidebar": false,

"fontSize": 12

}

}

--------------------------profile.json end--------------------

tweets.json  
[

{

"id": 1,

"tweet": "OMG, worst day ever, my BF @BobbyBoo dumped me",

"usersMentioned": [

{

"id": 10,

"username": "BobbyBoo"

}

]

},

{

"id": 2,

"tweet": "OMG, best day ever, my BF came back to me"

},

{

"id": 3,

"tweet": "OMG, worst day ever, just don't ask"

},

{

"id": 4,

"tweet": "@BobbyBoo OMG...just OMG!",

"usersMentioned": [

{

"id": 10,

"username": "BobbyBoo"

}

]

}

]

--------------------------tweets.json end--------------------

friend.json  
  
{

"id": 10,

"username": "BobbyBoo"

}

--------------------------friend.json end--------------------

Program 2) Task List Program: Add Task, Remove particular Task, Clear All Tasks, Filter task, Add Task to Local Storage & Make data persistent using local storage  
  
index.html

|  |  |
| --- | --- |
|  | <!DOCTYPE html> |
|  | <html lang="en"> |
|  | <head> |
|  | <meta charset="UTF-8"> |
|  | <meta name="viewport" content="width=device-width, initial-scale=1.0"> |
|  | <meta http-equiv="X-UA-Compatible" content="ie=edge"> |
|  | <link rel="stylesheet" href="<https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/css/materialize.min.css>"> |
|  | <link href="<https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css>" rel="stylesheet" integrity="sha384-wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0sibVcOQVnN" crossorigin="anonymous"> |
|  | <title>Task List</title> |
|  | </head> |
|  | <body> |
|  | <div class="container"> |
|  | <div class="row"> |
|  | <div class="col s12"> |
|  | <div id="main" class="card"> |
|  | <div class="card-content"> |
|  | <span class="card-title">Task List</span> |
|  | <div class="row"> |
|  | <form id="task-form"> |
|  | <div class="input-field col s12"> |
|  | <input type="text" name="task" id="task"> |
|  | <label for="task">New Task</label> |
|  | </div> |
|  | <input type="submit" value="Add Task" class="btn"> |
|  | </form> |
|  | </div> |
|  | </div> |
|  | <div class="card-action"> |
|  | <h5 id="task-title">Tasks</h5> |
|  | <div class="input-field col s12"> |
|  | <input type="text" name="filter" id="filter"> |
|  | <label for="filter">Filter Tasks</label> |
|  | </div> |
|  | <ul class="collection"></ul> |
|  | <a href="[#](file:///F:\Amol%20Mali\1.Advanced%20JS%20Videos\00000Udemy%20-%20Brad%20Traversy%20-%20JavaScript%20Beginner%20to%20very%20Advanced\Working%20Files\4-3-project-files\4_3_project_files\tasklist\index.html)" class="clear-tasks btn black">Clear Tasks</a> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | </div> |
|  |  |
|  | <script |
|  | src="<https://code.jquery.com/jquery-3.2.1.js>" |
|  | integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE=" |
|  | crossorigin="anonymous"></script> |
|  | <script src="<https://cdnjs.cloudflare.com/ajax/libs/materialize/0.100.2/js/materialize.min.js>"></script> |
|  | <script src="[app.js](file:///F:\Amol%20Mali\1.Advanced%20JS%20Videos\00000Udemy%20-%20Brad%20Traversy%20-%20JavaScript%20Beginner%20to%20very%20Advanced\Working%20Files\4-3-project-files\4_3_project_files\tasklist\app.js)"></script> |
|  | </body> |
|  | </html>  **// app.js** |

// Define UI Vars

const form = document.querySelector('#task-form');

const taskList = document.querySelector('.collection');

const clearBtn = document.querySelector('.clear-tasks');

const filter = document.querySelector('#filter');

const taskInput = document.querySelector('#task');

// Load all event listeners

loadEventListeners();

// Load all event listeners

function loadEventListeners() {

// DOM Load event

document.addEventListener('DOMContentLoaded', getTasks);

// Add task event

form.addEventListener('submit', addTask);

// Remove task event

taskList.addEventListener('click', removeTask);

// Clear task event

clearBtn.addEventListener('click', clearTasks);

// Filter tasks event

filter.addEventListener('keyup', filterTasks);

}

// Get Tasks from LS

function getTasks() {

let tasks;

if(localStorage.getItem('tasks') === null){

tasks = [];

} else {

tasks = JSON.parse(localStorage.getItem('tasks'));

}

tasks.forEach(function(task){

// Create li element

const li = document.createElement('li');

// Add class

li.className = 'collection-item';

// Create text node and append to li

li.appendChild(document.createTextNode(task));

// Create new link element

const link = document.createElement('a');

// Add class

link.className = 'delete-item secondary-content';

// Add icon html

link.innerHTML = '<i class="fa fa-remove"></i>';

// Append the link to li

li.appendChild(link);

// Append li to ul

taskList.appendChild(li);

});

}

// Add Task

function addTask(e) {

if(taskInput.value === '') {

alert('Add a task');

}

// Create li element

const li = document.createElement('li');

// Add class

li.className = 'collection-item';

// Create text node and append to li

li.appendChild(document.createTextNode(taskInput.value));

// Create new link element

const link = document.createElement('a');

// Add class

link.className = 'delete-item secondary-content';

// Add icon html

link.innerHTML = '<i class="fa fa-remove"></i>';

// Append the link to li

li.appendChild(link);

// Append li to ul

taskList.appendChild(li);

// Store in LS

storeTaskInLocalStorage(taskInput.value);

// Clear input

taskInput.value = '';

e.preventDefault();

}

// Store Task

function storeTaskInLocalStorage(task) {

let tasks;

if(localStorage.getItem('tasks') === null){

tasks = [];

} else {

tasks = JSON.parse(localStorage.getItem('tasks'));

}

tasks.push(task);

localStorage.setItem('tasks', JSON.stringify(tasks));

}

// Remove Task

function removeTask(e) {

if(e.target.parentElement.classList.contains('delete-item')) {

if(confirm('Are You Sure?')) {

e.target.parentElement.parentElement.remove();

// Remove from LS

removeTaskFromLocalStorage(e.target.parentElement.parentElement);

}

}

}

// Remove from LS

function removeTaskFromLocalStorage(taskItem) {

let tasks;

if(localStorage.getItem('tasks') === null){

tasks = [];

} else {

tasks = JSON.parse(localStorage.getItem('tasks'));

}

tasks.forEach(function(task, index){

if(taskItem.textContent === task){

tasks.splice(index, 1);

}

});

localStorage.setItem('tasks', JSON.stringify(tasks));

}

// Clear Tasks

function clearTasks() {

// taskList.innerHTML = '';

// Faster

while(taskList.firstChild) {

taskList.removeChild(taskList.firstChild);

}

// https://jsperf.com/innerhtml-vs-removechild

// Clear from LS

clearTasksFromLocalStorage();

}

// Clear Tasks from LS

function clearTasksFromLocalStorage() {

localStorage.clear();

}

// Filter Tasks

function filterTasks(e) {

const text = e.target.value.toLowerCase();

document.querySelectorAll('.collection-item').forEach(function(task){

const item = task.firstChild.textContent;

if(item.toLowerCase().indexOf(text) != -1){

task.style.display = 'block';

} else {

task.style.display = 'none';

}

});

}

----------------------------------------------------------------------------------------------------------------------------------------------------------------

Program 3) ES6 class Inheritance: create subclass-superclass relationship using extends keyword, super keyword to call parent class constructor and use parameters available in parent class.  
  
Ans:  
  
class Person {

constructor(firstName, lastName) {

this.firstName = firstName;

this.lastName = lastName;

}

greeting() {

return `Hello there ${this.firstName} ${this.lastName}`;

}

}

class Customer extends Person {

constructor(firstName, lastName, phone, membership) {

super(firstName, lastName);

this.phone = phone;

this.membership = membership;

}

static getMembershipCost() {

return 500;

}

}

const john = new Customer('John', 'Doe', '555-555-5555', 'Standard');

console.log(john.greeting());

console.log(Customer.getMembershipCost());

--------------------------------------------------------------------------------------------------------------------------------------------------------------

Program 4) BookList Program using: Add books, remove books, show alert messages, using ES5 Prototypes syntax and ES6 Classes  
  
index.html

|  |  |
| --- | --- |
|  | <!DOCTYPE html> |
|  | <html lang="en"> |
|  | <head> |
|  | <meta charset="UTF-8"> |
|  | <meta name="viewport" content="width=device-width, initial-scale=1.0"> |
|  | <meta http-equiv="X-UA-Compatible" content="ie=edge"> |
|  | <link rel="stylesheet" href="<https://cdnjs.cloudflare.com/ajax/libs/skeleton/2.0.4/skeleton.css>" /> |
|  | <style> |
|  | .success, .error { |
|  | color: white; |
|  | padding: 5px; |
|  | margin: 5px 0 15px 0; |
|  | } |
|  |  |
|  | .success { |
|  | background: green; |
|  | } |
|  |  |
|  | .error { |
|  | background: red; |
|  | } |
|  | </style> |
|  | <title>Book List</title> |
|  | </head> |
|  | <body> |
|  | <div class="container"> |
|  | <h1>Add Book</h1> |
|  | <form id="book-form"> |
|  | <div> |
|  | <label for="title">Title</label> |
|  | <input type="text" id="title" class="u-full-width"> |
|  | </div> |
|  | <div> |
|  | <label for="author">Author</label> |
|  | <input type="text" id="author" class="u-full-width"> |
|  | </div> |
|  | <div> |
|  | <label for="isbn">ISBN#</label> |
|  | <input type="text" id="isbn" class="u-full-width"> |
|  | </div> |
|  | <div> |
|  | <input type="submit" value="Submit" class="u-full-width"> |
|  | </div> |
|  | </form> |
|  | <table class="u-full-width"> |
|  | <thead> |
|  | <tr> |
|  | <th>Title</th> |
|  | <th>Author</th> |
|  | <th>ISBN</th> |
|  | <th></th> |
|  | </tr> |
|  | </thead> |
|  | <tbody id="book-list"></tbody> |
|  | </table> |
|  | </div>  <!-- <script src="es5.js"></script> --> |
|  |  |
|  | <script src="[es6.js](file:///F:\Amol%20Mali\1.Advanced%20JS%20Videos\00000Udemy%20-%20Brad%20Traversy%20-%20JavaScript%20Beginner%20to%20very%20Advanced\Working%20Files\6-5-project-files\6_5_project_files\booklist\appes6.js)"></script> |
|  | </body> |
|  | </html>  [es5.js](file:///F:\Amol%20Mali\1.Advanced%20JS%20Videos\00000Udemy%20-%20Brad%20Traversy%20-%20JavaScript%20Beginner%20to%20very%20Advanced\Working%20Files\6-5-project-files\6_5_project_files\booklist\appes6.js): ES5 Prototypes Syntax way:  // Book Constructor  function Book(title, author, isbn) {  this.title = title;  this.author = author;  this.isbn = isbn;  }  // UI Constructor  function UI() {}  // Add Book To List  UI.prototype.addBookToList = function(book){  const list = document.getElementById('book-list');  // Create tr element  const row = document.createElement('tr');  // Insert cols  row.innerHTML = `  <td>${book.title}</td>  <td>${book.author}</td>  <td>${book.isbn}</td>  <td><a href="#" class="delete">X<a></td>  `;  list.appendChild(row);  }  // Show Alert  UI.prototype.showAlert = function(message, className) {  // Create div  const div = document.createElement('div');  // Add classes  div.className = `alert ${className}`;  // Add text  div.appendChild(document.createTextNode(message));  // Get parent  const container = document.querySelector('.container');  // Get form  const form = document.querySelector('#book-form');  // Insert alert  container.insertBefore(div, form);  // Timeout after 3 sec  setTimeout(function(){  document.querySelector('.alert').remove();  }, 3000);  }  // Delete Book  UI.prototype.deleteBook = function(target){  if(target.className === 'delete') {  target.parentElement.parentElement.remove();  }  }  // Clear Fields  UI.prototype.clearFields = function() {  document.getElementById('title').value = '';  document.getElementById('author').value = '';  document.getElementById('isbn').value = '';  }  // Event Listener for add book  document.getElementById('book-form').addEventListener('submit', function(e){  // Get form values  const title = document.getElementById('title').value,  author = document.getElementById('author').value,  isbn = document.getElementById('isbn').value  // Instantiate book  const book = new Book(title, author, isbn);  // Instantiate UI  const ui = new UI();  // Validate  if(title === '' || author === '' || isbn === '') {  // Error alert  ui.showAlert('Please fill in all fields', 'error');  } else {  // Add book to list  ui.addBookToList(book);  // Show success  ui.showAlert('Book Added!', 'success');    // Clear fields  ui.clearFields();  }  e.preventDefault();  });  // Event Listener for delete  document.getElementById('book-list').addEventListener('click', function(e){  // Instantiate UI  const ui = new UI();  // Delete book  ui.deleteBook(e.target);  // Show message  ui.showAlert('Book Removed!', 'success');  e.preventDefault();  });  [es6.js](file:///F:\Amol%20Mali\1.Advanced%20JS%20Videos\00000Udemy%20-%20Brad%20Traversy%20-%20JavaScript%20Beginner%20to%20very%20Advanced\Working%20Files\6-5-project-files\6_5_project_files\booklist\appes6.js): ES6 Classes Syntax:  class Book {  constructor(title, author, isbn) {  this.title = title;  this.author = author;  this.isbn = isbn;  }  }  class UI {  addBookToList(book) {  const list = document.getElementById('book-list');  // Create tr element  const row = document.createElement('tr');  // Insert cols  row.innerHTML = `  <td>${book.title}</td>  <td>${book.author}</td>  <td>${book.isbn}</td>  <td><a href="#" class="delete">X<a></td>  `;    list.appendChild(row);  }  showAlert(message, className) {  // Create div  const div = document.createElement('div');  // Add classes  div.className = `alert ${className}`;  // Add text  div.appendChild(document.createTextNode(message));  // Get parent  const container = document.querySelector('.container');  // Get form  const form = document.querySelector('#book-form');  // Insert alert  container.insertBefore(div, form);  // Timeout after 3 sec  setTimeout(function(){  document.querySelector('.alert').remove();  }, 3000);  }  deleteBook(target) {  if(target.className === 'delete') {  target.parentElement.parentElement.remove();  }  }  clearFields() {  document.getElementById('title').value = '';  document.getElementById('author').value = '';  document.getElementById('isbn').value = '';  }  }  // Event Listener for add book  document.getElementById('book-form').addEventListener('submit', function(e){  // Get form values  const title = document.getElementById('title').value,  author = document.getElementById('author').value,  isbn = document.getElementById('isbn').value  // Instantiate book  const book = new Book(title, author, isbn);  // Instantiate UI  const ui = new UI();  console.log(ui);  // Validate  if(title === '' || author === '' || isbn === '') {  // Error alert  ui.showAlert('Please fill in all fields', 'error');  } else {  // Add book to list  ui.addBookToList(book);  // Show success  ui.showAlert('Book Added!', 'success');    // Clear fields  ui.clearFields();  }  e.preventDefault();  });  // Event Listener for delete  document.getElementById('book-list').addEventListener('click', function(e){  // Instantiate UI  const ui = new UI();  // Delete book  ui.deleteBook(e.target);  // Show message  ui.showAlert('Book Removed!', 'success');  e.preventDefault();  });  --------------------------------------------------------------------------------------------------------------------------------------------------  Program 5) Fetch API: access file from remote server  index.html   |  |  | | --- | --- | |  | <!DOCTYPE html> | |  | <html lang="en"> | |  | <head> | |  | <meta charset="UTF-8"> | |  | <meta name="viewport" content="width=device-width, initial-scale=1.0"> | |  | <meta http-equiv="X-UA-Compatible" content="ie=edge"> | |  | <link rel="stylesheet" href="<https://cdnjs.cloudflare.com/ajax/libs/skeleton/2.0.4/skeleton.css>" /> | |  | <title>Ajax Sandbox</title> | |  | </head> | |  | <body> | |  | <div class="container"> | |  | <button id="button1">Get Customer</button> | |  | <button id="button2">Get Customers</button> | |  | <br><br> | |  | <h1>Customer</h1> | |  | <div id="customer"></div> | |  | <h1>Customers</h1> | |  | <div id="customers"></div> | |  | </div> | |  |  | |  | <script src="[app.js](http://localhost/fetch%20api/app.js)"></script> | |  | </body> | |  | </html>  app.js |   document.getElementById('button1').addEventListener('click', loadCustomer);  document.getElementById('button2').addEventListener('click', loadCustomers);  // Load Single Customer  function loadCustomer(e) {  const xhr = new XMLHttpRequest();  xhr.open('GET', 'customer.json', true);  xhr.onload = function(){  if(this.status === 200) {  // console.log(this.responseText);  const customer = JSON.parse(this.responseText);  const output = `  <ul>  <li>ID: ${customer.id}</li>  <li>Name: ${customer.name}</li>  <li>Company: ${customer.company}</li>  <li>Phone: ${customer.phone}</li>  </ul>  `;  document.getElementById('customer').innerHTML = output;  }  }  xhr.send();  }  // Load Customers  function loadCustomers(e) {  const xhr = new XMLHttpRequest();  xhr.open('GET', 'customers.json', true);  xhr.onload = function(){  if(this.status === 200) {  // console.log(this.responseText);  const customers = JSON.parse(this.responseText);  let output = '';  customers.forEach(function(customer){  output += `  <ul>  <li>ID: ${customer.id}</li>  <li>Name: ${customer.name}</li>  <li>Company: ${customer.company}</li>  <li>Phone: ${customer.phone}</li>  </ul>  `;  });  document.getElementById('customers').innerHTML = output;  }  }  xhr.send();  } |

customer.json

{

"id": 1,

"name": "John Doe",

"company": "123 Designs",

"phone": "444-555-6666"

}

customers.json

[

{

"id": 1,

"name": "John Doe",

"company": "123 Designs",

"phone": "444-555-6666"

},

{

"id": 2,

"name": "Steve Smith",

"company": "Hello Productions",

"phone": "333-222-2222"

},

{

"id": 3,

"name": "Tara Williams",

"company": "Traversy Media",

"phone": "111-222-3333"

}

]

------------------------------------------------------------------------------------------------------------------------------------------

Program 6) Create elements dynamically: Create 4 boxes dynamically, all boxes are hidden by default, four buttons are there, if we click on first button, first box needs to be shown and rest all buttons needs to be hidden, if we click on second button then second box needs to be shown and rest all buttons needs to be hidden and so on. Also four more buttons are there to assign background color to box, red-green-blue & yellow buttons, if we click on red button then background color will be red and if we click on green button then background color will be green. Also one cross button will appear on every box. On click of close button, box needs to be hidden.

Ans:

<!DOCTYPE html>

<html>

<head>

<title>Dynamically create DOM elements</title>

<meta charset="utf-8" />

<style>

\*{

box-sizing: border-box;

padding: 0;

margin: 0;

}

section#userInput{

display: block;

height: 40px;

width: 100%;

}

#userInput .display\_boxes > div.box{

min-width: 50px;

padding: 12px;

background-color: antiquewhite;

display: inline-block;

cursor: pointer;

}

#choose\_color .color

{

min-width: 40px;

background: aliceblue;

display: inline-block;

float: left;

padding: 8px;

border: 1px solid cornflowerblue;

cursor: pointer;

}

#choose\_color .color + .color{

margin-left: 15px;

}

section{

display: inline-block;

margin-top: 20px;

width: 48%;

float: left;

}

.container{

width: 92%;

margin: auto;

}

.mainDiv{

height: 200px;

float: left;

padding: 15px;

background: #bcbca1;

overflow: hidden;

}

.header\_part{

height: 30px;

border-bottom: 1px solid #795548;

float: left;

width: 100%;

}

.close\_{

width: 20px;

height: 20px;

float: right;

text-align: center;

color: beige;

cursor: pointer;

background: black;

font-size: 24px;

margin-top: -5px;

border-radius: 4px;

}

.mainContent{

background: aqua;

width: 100%;

height: 80%;

float: left;

padding: 10px;

margin-top: 8px;

}

.disn{

display: none;

}

.block{

display: block;

}

</style>

</head>

<body>

<section id="userInput">

<div class="container">

<div id="display\_boxes" class="display\_boxes" style="float:left; width: 45%;">

<div class="box">Show First</div>

<div class="box">Show Second</div>

<div class="box">Show Third</div>

<div class="box">Show Fourth</div>

</div>

<div id="choose\_color" style="float:right; width: 45%;">

<div class="color">Yellow</div>

<div class="color">antiquewhite</div>

<div class="color">Pink</div>

<div class="color">Coral</div>

<div class="color">red</div>

</div>

</div>

</section>

<script>

for(var i = 0; i <=3; i++ ){

var body\_ele = document.querySelector("body");

var \_section = body\_ele.appendChild(document.createElement("section"));

\_section.className = "disn"

var container\_ele = \_section.appendChild(document.createElement("div"));

container\_ele.className = "container";

var \_div = container\_ele.appendChild(document.createElement("div"));

\_div.setAttribute("class","mainDiv");

var div\_header = \_div.appendChild(document.createElement("div"));

div\_header.setAttribute("class","header\_part");

var right\_header = div\_header.appendChild(document.createElement("div"));

right\_header.setAttribute("class","right\_header");

var close\_btn = right\_header.appendChild(document.createElement("div"));

close\_btn.setAttribute("class","close\_");

close\_btn.innerHTML = "&times";

close\_btn.style.fontWeight = "700";

// main Content start here

var \_mainContent = \_div.appendChild(document.createElement("div"));

\_mainContent.className = "mainContent";

\_mainContent.appendChild(document.createTextNode("Lorem Ipsum dolor sit amet, lorem ipsum dolor sit amet, Lorem Ipsum dolor sit amet, lorem ipsum dolor sit amet, Lorem Ipsum dolor sit amet, lorem ipsum dolor sit amet. Lorem Ipsum dolor sit amet, lorem ipsum dolor sit amet, Lorem Ipsum dolor sit amet, lorem ipsum dolor sit amet, Lorem Ipsum dolor sit amet, lorem ipsum dolor sit amet."));

}

// get Index of clicked element..

var list = document.getElementById("display\_boxes"),

items = list.getElementsByClassName("box");

// display required section..

var box\_list = document.getElementsByClassName("disn");

list.onclick = function(e) {

var evt = e || window.event, // get Event first

src = evt.target || evt.srcElement; // Get the element that triggered event

var myIndex = findIndex(src);

// alert(myIndex);

};

function findIndex( elem ) {

var i, len = items.length;

for(i=0; i<len; i++) {

if (items[i]===elem) {

// return i;

box\_list[i].className += " block";

// assign background-color as per userInput - Get Background color first

var chooseColor = document.getElementById("choose\_color"),

color\_items = chooseColor.getElementsByClassName("color");

chooseColor.onclick = function(e) {

var evt = e || window.event, // get Event first

src = evt.target || evt.srcElement; // get exact element which trigger the event

var getColor = src.innerHTML; // get color Name

// console.log(getColor);

// Set Background color here

var \_mainDiv = document.querySelector(".block .mainDiv");

\_mainDiv.style.backgroundColor = getColor;

};

/\* hide box on close button \*/

var blockEle = document.querySelector('.block');

var closeBtn = document.querySelector('.block .close\_');

closeBtn.addEventListener('click', function(){

blockEle.classList.remove('block');

});

}

else box\_list[i].className = "disn";

}

}

</script>

</body>

</html>

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Que. 7: How can we get the user’s current location?

## The HTML Geolocation API is used to locate a user's position. Locate the User's Position

The HTML Geolocation API is used to get the geographical position of a user.

Since this can compromise privacy, the position is not available unless the user approves it.

Ex:

<!DOCTYPE html>

<html>

<body>

<p>Click the button to get your coordinates.</p>

<button onclick="getLocation()">Try It</button>

<p id="demo"></p>

<script>

var x = document.getElementById("demo");

function getLocation() {

if (navigator.geolocation) {

navigator.geolocation.getCurrentPosition(showPosition);

} else {

x.innerHTML = "Geolocation is not supported by this browser.";

}

}

function showPosition(position) {

x.innerHTML = "Latitude: " + position.coords.latitude +

"<br>Longitude: " + position.coords.longitude;

}

</script>

</body>

</html>  
  
  
  
Output:   
Click the button to get your coordinates.

Try It // this is button

Latitude: 18.4992061 // this can be vary as user’s position  
Longitude: 73.94565690000002 // this can be vary as user’s position