**[Thrid-Party APIs – Day 1]**

**DOM**: Document Object Model interface enables developers to interact with the application’s HTML and CSS using JavaScript.

**DOM Web API**: Create HTML elements / Changing HTML content and attributes / Binding listeners to execute a function upon a certain event.

<Web API example>

Const newElement = document.createElement(“span”);

newElement.textContent = “Hello World”;

const heading = document.querySelector(“h1”);

heading.setAttribute(‘style’, “font-family:10px”);

heading.append(newElement);

heading.addEventListner(‘click’, function() {

console.log(“Hello from Heading”);

})

API = DOM + JS

(HTML or XML Page)

DOM is provided by the browser Web API / not built it in the JavaScript

**<The Languages of the Web>**

1. HTML: used to write content.
2. CSS: used to format content.
3. JavaScript: used to create dynamic web applications that take in user inputs, change what’s displayed to users, animate elements, and much more.

**<JavaScript and APIs>**

Browser-based APIs we’ve used include the following:

1. The DOM to interact with our HTML and CSS.
2. The console to help us debug and print messages.
3. Timer functions with setTimeout and setInterval.

**<Third-Party APIs>**

Q. Are there APIs we can use that are not provided by the browser?

=> Yes, a lot. => **Third-Party APIs**

1. Reduce JavaScript code for DOM manipulation.

2. Set up a styled and laid-out webpage quickly with premade CSS.

Carry out tedious tasks like converting time or currency values.

e.g. ‘**day.js**’

Third-Party API example: **jQuery**

**Bootstrap**: **getbootstrap.com**

We need to be more **Productive** / Save time.

**cdn: Content Deliver Network**

**cdn.jsdelivr.net**

<link href=[https://**cdn.jsdelivr.net**/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css](https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css) …>

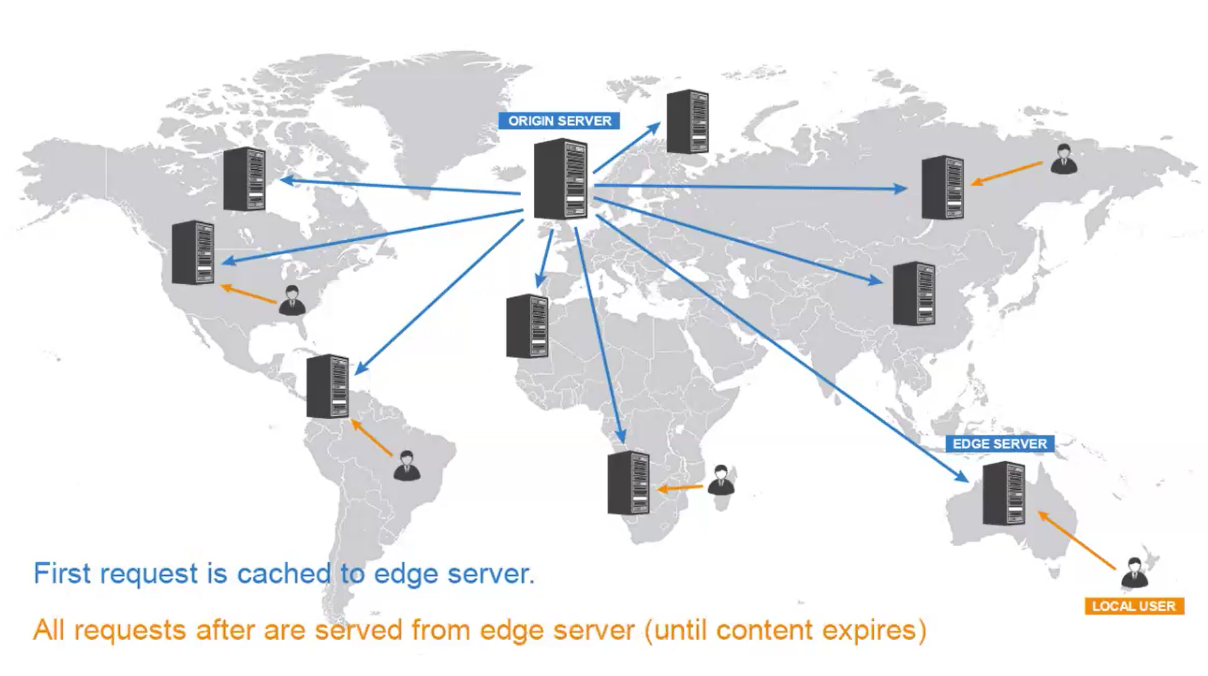
<script src=”[https://**cdn.jsdelivr.net**/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js](https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js)” …>

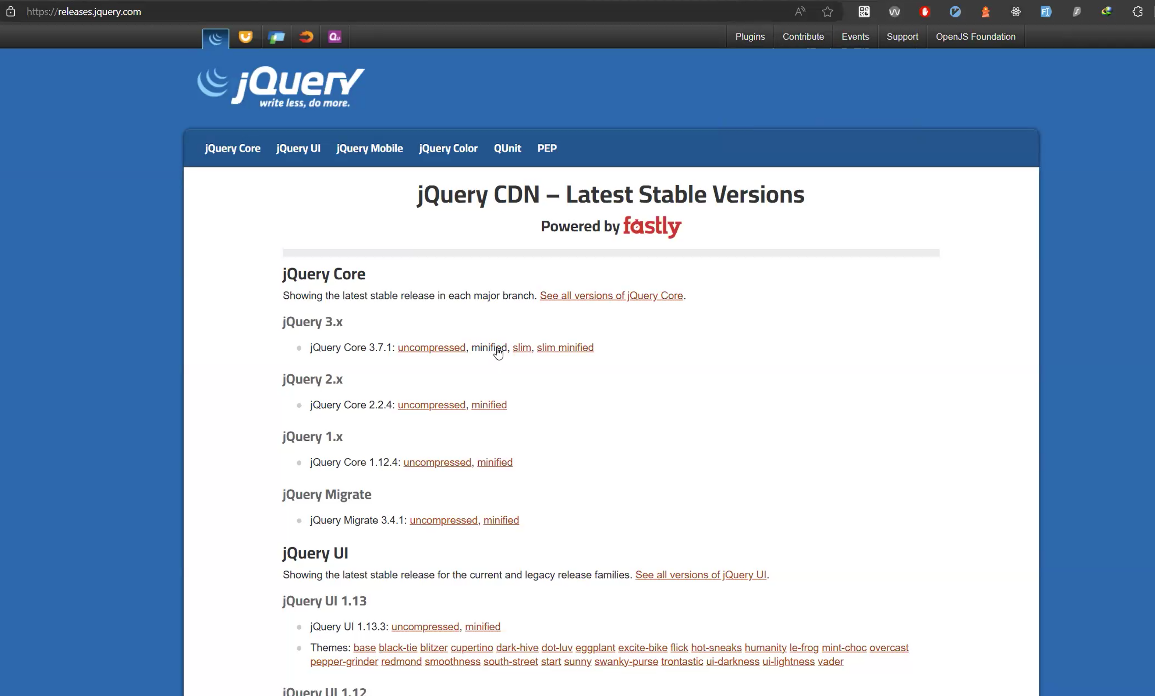
<How to learn Third-Party APIs>

1. Read the official documentation and practice with the provided examples.
2. Reverse-engineer finished code to see how something was accomplished.
3. Build something from scratch.
4. Debug a broken app using Chrome DevTools.
5. And most importantly, ask questions!

**<How to integrate jQuery into your project?>**

**Cdn: Content Delivery Network**

* Cdn working -> copy files in other servers



Usually, we use ‘**minified’** or ‘slim’ version

Copy the script and put it in the HTML file.

**jQuery cdn**

**[First Exercise: 01- jQuery-Elements]**

In **HTML** file:

<body …>

<div …>

\*\*\* All 3rd party APIs goes/defined/included first

**<script src=**[**https://code.jquery.com/jquery-3.5.1min.js**](https://code.jquery.com/jquery-3.5.1min.js)**></script>**

// min: mini version of jquery

\*\*\* Then, your own custom javascript file is included

<script type=”text/javascript” src=”./assets/js/script.js”></script>

</body>

In .**js** file:

// const rootEL = document.getElementById(“root”)

Const rootEl = $(‘#root’);

// $(“selector”) => $(“#your\_element\_id”)

// const titleEL = document.createElement(“h1”);

Const titleEl = $(‘<h1>’);

// ($”h1”) => selector, selecting element h1

// $(.class\_goes\_here”)

// $(“<h1>”) => creating new element

// $() => jquery method

// titleEl.textContent = “Hello friends”;

titleEl.text(‘Hello friends’); => text mothod() <= setter

// cost elementText = $(elementSelector).text(); <= getter

// titleEl.className = ‘fancy’; => setting a class name

// titleEl.setAttributes(‘class’, ‘fancy’); => setting an attribute

titleEl.attr(‘class’, ‘fancy’);

//titleEl.classList.add(‘p-5’) -> (‘p-5’ is for padding)

titleEl.addClass(‘p-5’);

// titleEl.style.border = “rgb(122,242,242) 5px solid”;

titleEl.css(‘border’, ‘rgb(122,242,242) 5px solid’);

// rootEl.appendChild(titleEl);

rootEl.append(titleEl);

// titleEl.append(“Welcome to jQuery”);

rootEl.append(‘<h2>With jQuery we can:</h2>’);

const abilities = [

‘Select’,

‘Create’,

‘Style’,

‘Animate’,

‘Traverse’,

‘Event Listen’,

‘much more’,

];

For (const ability of abilities) {

// Create new ‘<li>’ for each ability and its text content

Const abilityEl = $(‘<li>’);

// abilityEl.textContent = ability;

AbilityEl.text(ability);

// ‘my-3’ cass adds margin on top and bottom

AbilityEl.addClass(‘my-3’); // my: margin in the y-axis // py: padding …

// mx: margin in the x-axis

// Alternatively

// const abilityEl = $(`<li class=’my-3’>${ability}</li>`);

// Add this new `<li>~ to the `<div>` root element.

rootEl.append(abilityEl);

}

// select all `<li>` elements on the page

// document.querySeletorAll(‘li’);

Console.log($(‘li’));

// Alternatively use the `$.each` method

// $.each(abilities, function(I, ability) {

// rootEl.append(`<li>${index+1} - ${ability}</li>`);

// })

$.each([“Yo”, “Bye”], function(index, siglevalue) {

rootEl.append(`<h${index + 1}>${singleValue}</h${index + 1}>`

});

Const liElement = $(“<li>”) // create element

$(“li:first”) // selecting all list items -> select the first list item

Const allLiElements = $(“li”) // selecting all list items

Const firstLiElement = $(“li:first”)

$.each(allLiElements, function(index, singleElement) {

If (index === 0) {

Console.log(‘this msg will only be printed for first element’)

}

})

// DOM jquery

// select, change attribute, class, style, append

**[Third Exercise: 03 Click-Events]**

// adding Event

// liElement.addEventListener(‘click’, function(){ }

In HTML file:

<button id=”alert-button” class=”btn btn-danger”>Alert</button>

<button id=”theme-button” class=”btn btn-info”>Toggle Theme</button>

<button id=”lottery-button” class=”btn btn-primary”>Play Lottery</button>

<span id=”lottery-number”></span>

<button id=”refresh-button” class=btn btn-refresh>

In .js file:

Const alertButtonEl = $(‘#alert-button’);

Const themeButtonEl = $(‘#theme-button’);

Const lotteryButtonEl = $(‘#lottery-button’);

Const refreshButtonEl = $(‘#refresh-button’);

Let isDark = true;

alertButtonEl.on(‘click’, function() {

alert(‘Hello World’);

});

themeButtonEl.on(‘click’, function() {

if (isDark) {

$(‘body’).css({ ‘background-color’: '#d9e9e8', color: '#1a1a1a' });

// $(element).css(‘property’, ‘value’)

/\* $(element).css(

{

‘property-1’: ‘value-1’,

‘property2’: ‘value-2’,

}

)

isDark = !isDark;

}

});

lotteryButtonEl.on(‘click’, functon() {

const random = Math.floor(Math.random() \* 100000000) + 10000000;

console.log(random);

lotteryNumberEl.text(random);

});

refreshButtonEl.on(‘click’, function() {

location.reload();

});

Math.round(100.60) => 101

Math.floor(100.60) => 100

Math.ceil(100.20) => 101

**[Fifth Exercise: 05 Form-Events]**

<input type=”text” id=”first-name”/>

<textarea></textarea>

<select>

<option value=”1”>first option</option>

</select>

Const firstNameFieldValue = document.querySelector(‘#firstName’).value

Const firstNameFieldValue = $(‘#firstName’).val(); // getter => get the value

Document.querySelector(‘#firstName’).value = “10”

$(‘#firstName’).val(‘10’); // setter => set the value

In HTML FILE:

<form id=”pizza-form”>

            <div class="row">

              <div class="col-12 col-sm-6">

                <label for="first-name">First Name</label>

                <input id="first-name" type="text" class="form-input" **name="first-name"** placeholder="Enter first name" />

              </div>

              <div class="col-12 col-sm-6">

                <label for="last-name">Last Name</label>

                <input id="last-name" type="text" class="form-input" **name="last-name"** placeholder="Enter last name" />

              </div>

            </div>

**Const formEl = $(‘#pizza-form’);**

**Const firstNameEl = $(‘input[name=”first-name”]’);**

* **Input that has ‘name’ = ‘first-name’**

**Const lastNameEl = $(‘input[name=’last-name”]);**

Function handleFormSubmit(event) {

// Prevent the default behavior

Event.preventDefault();

**Console.log(‘First Name:’, firstNameEl.val()); // printing into the console**

**Console.log(‘Last Name:’, lastNameEl.val());**

**$(‘input[type=”text”]’).val(‘’);**

**formEl.on(‘submit’, handleFormSubmit)**

**formEl.addEventListener(‘submit’, handleFormSubmit)**

**[Check Boxes]**

**Const checkedEl = $(‘input: checked’); // grab all the elements checked**

**Const selected = [];**

// Loop through checed options to store in array

**$.each(chedkedEl, function() {**

**Selected.push($(this).val());**

**// ‘this’ refers to the current element in the loop**

**// $(this).val() => current element’s value**

**// whatever that value, we’re pushing into the array**

**// selected = [‘peperoni’, ‘onion’]**

});

Console.log(‘Toppings: ‘, **selected.join(‘, ‘)**);

// **[‘peperoni’, ‘onion’] => peperoni, onion**

**// clear input fields**

**$(‘input[type=”text”]’).val(‘’);**

**$(‘input[type=”email”]’).val(‘’);**

**$(‘input[type=”checkbox”]’).prop(‘checked’, false); // changing property, not value**

<**input** id=”email” … **disabled**> => won’t accept any input.

< **input** type = “**checkbox**” id=”cb1” value=”pepperoni” **checked** />

// this check box will always be selected

How to do the same thing with jquery?

* **$(“input[name=’email’]”).prop(‘disabled’, true)**

// => <input name=”email” disabled/>

Same with check boxes:

* **$(“input[type=’checkbox]”).prop(‘checked, false)**

// run through all checkbox, and tick them off

**[Sixth Exercise: 06Form-Events]**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8" />

  <title>jQuery Shopping List</title>

  <link rel="stylesheet" type="text/css" href="./assets/css/jass.css" />

  <link rel="stylesheet" type="text/css" href="./assets/css/style.css" />

</head>

<body class="bg-dark text-light">

  <main class="my-5 mx-auto">

    <h1>Shopping List</h1>

    <hr />

    <h2>Add an item to the list.</h2>

    <form id="shopping-form">

      <input type="text" class="form-input w-100" id="shopping-input" name="shopping-input" placeholder="Enter item name" />

      <button class="btn btn-info">Add Item</button>

    </form>

    <ul id="shopping-list"></ul>

  </main>

  <!-- Added link to the jQuery library -->

  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>

  <!-- Your JavaScript file -->

  <script type="text/javascript" src="assets/js/script.js">

  </script>

</body>

</html>

const shoppingFormEl = $('#shopping-form');

const shoppingListEl = $('#shopping-list');

// create function to handle form submission

function handleFormSubmit(event) {

  event.preventDefault();

  // select form element by its `name` attribute and get its value

  const shoppingItem = $('input[name="shopping-input"]').val();

  // if there's nothing in the form entered, don't print to the page

  if (!shoppingItem) {

    console.log('No shopping item filled out in form!');

    return;

  }

  // print to the page

  shoppingListEl.append(`<li>${shoppingItem}</li>`);

  // clear the form input element

  $('input[name="shopping-input"]').val('');

}

// Create a submit event listener on the form element

shoppingFormEl.on('submit', handleFormSubmit);

Procedures:

1. Call the handle function
2. Grab the element into input field
3. Receive the value
4. Append it to the list
5. Set its value to blank

**[2nd Day of Module-5: Third-Party APIs]**

* **Bootstrap:**
* **Getbootstrap.com**
* **Color.adobe.com**
* **Paletton.com**
* **Fonts.google.com**
* **Discord // tool for communication**

**[3rd Day of Module-5: Third-Party APIs]**

* **Bootstrap:**

Write less css and achieve more

* **Bootstrap Components:**

Pre-built code for different UI Parts/component copy/paste to save time

* **Bootstrap layout / grid system:**
* Mobile-first flexbox grid system, pre-built pattern+classes to achieve responsive design based on 12-col spaces
* **<div class=”container”>**

**<div class=”row”>**

**<div class=”col-12 col-md-6 col-lg-3”>**

//

</div>

**<div class=”col-12 col-md-6 col-lg-3”>**

//

</div>

<Microsoft Bing>

**Bootstrap grid system**

Carousel : Breakpoints

* **Bootstrap Utilities / pre-built classes for more granual control than creating your own classes for margin/ padding/ background ect**
* **P-5 // paddin: 5**
* **Mb-4 // margin-bottom: 4**
* **Bg-dark // background: dark**
* **Text-light // text-color: light**
* **Extending bootstrap style to give more personal / custom visual experience on project**
* **Google fonts / to use more fitting text style for your projects**
* **Jqueryui.com**
* **Jqueryui – libararies using jquery**

**Jqueryui -> dialog -> button to create new Task on the fly \*\*\*\*\*\*\*\*\*\***

**‘Modal Form’ example // bottom -> view source**

* **Include following files in ‘index.html’ file:**
* **<link rel=”stylesheet” href=**[**https://code.jquery.com/ui/1.13.3/themes/base/jquery-ui.css**](https://code.jquery.com/ui/1.13.3/themes/base/jquery-ui.css)**>**
* **<script scr=”**[**https://code.jquery.com/jquery-3.7.1.js”**](https://code.jquery.com/jquery-3.7.1.js”)**></script>**
* **<script scr=”**[**https://code.jquery.com/ui/1.13.3/jquery-ui.js”**](https://code.jquery.com/ui/1.13.3/jquery-ui.js”)**></script>**

**<script>**

**$(function() {**

**$(“#dialog”) dialog();**

**});**

**</script>**

1. Add CSS from Bootstrap
2. Then Add jquery-ui CSS

* **CSS (<head>)**
  + **Include any other 3rd party files CSS / JS**
  + **Include jquery ui CSS file:**

**<link rel=”stylesheet” href=**[**https://code.jquery.com/ui/1.13.3/themes/base/jquery-ui.css**](https://code.jquery.com/ui/1.13.3/themes/base/jquery-ui.css)**>**

* **JS section (</body>):**
  + **Include jquery js file:**

**<script scr=”**[**https://code.jquery.com/jquery-3.7.1.js”**](https://code.jquery.com/jquery-3.7.1.js”)**></script>**

* + **Include jquery ui js file:**

**<script scr=”**[**https://code.jquery.com/ui/1.13.3/jquery-ui.js”**](https://code.jquery.com/ui/1.13.3/jquery-ui.js”)**></script>**

* **Bing: jquery ui cdn**

jQuery UI 1.13 – minified version

In script.js file:

Function () { // when entire page is loaded

})

// older version of jquery

$(document).ready(function() { // target the document, and when the doc is ready

$(‘#dialog’).dialog();

// $(‘#dialog’). -> target element dialog // and attach a method

}

$(‘#dialog’).dialog(); // $(‘#dialog’). -> target element dialog // and attach a method

// newer version of jquery version:

$(function(){ // $() – jQuery method // when document is ready, function()

}

* **Jqueryui.com/datepicker**
* **View source**
* **Jqueryui.com/Interactions**
* **Draggable**
* **Droppable**

**[script.js file]**

**Const dateInputEl = $(‘#datepicker’);**

**$(dateInputEl).datepicker();**

**Datepicker -> ‘Display month & year menus’ option**

‘Ctrl + Shift + T’ => Open the previously closed tab

‘Ctrl + T’ => Open the New Tab

* **DayJS // 3rd Party API**
* **Handling Dates**
* **Day js // Microsoft Bing**

Browser:

<script src=”path/to/dayjs/dayjs.min.js”></script>

<script>

Dayjs().format()

<script>

CDN resource

<script src=<https://cdn.jsdelivr.net/npm/dayjs@1/dayjs.min.js>”></script>