WEEK-8-JQ-NOTES

▼ What is jQuery

jQuery is a fast, lightweight JavaScript library that simplifies DOM manipulation, event handling, animations, and AJAX—while hiding browser differences.

CDN (Content Delivery Network)

```
<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
```

Advantages:

- Faster loading (cached globally)
- Saves your server bandwidth
- Easy to use

Disadvantages:

- Needs internet
- If CDN blocked → jQuery fails

Local jQuery

```
<script src="js/jquery.min.js"></script>
```

Advantages:

- Works offline
- Full control
- No external dependency

Disadvantages:

- Slightly slower first load
- You manage updates

Document Ready Function (MOST IMPORTANT)

Why Needed?

HTML loads **top to bottom**.

JS may run **before DOM is ready**.

Problem Code

```
$("#btn").click(function () {
alert("Clicked");
});
```

If script runs before button loads → error

Solution: Document Ready

```
$(document).ready(function () {
 $("#btn").click(function () {
alert("Clicked");
 });
});
```

Short-hand (Most Used)

```
$(function () {
 $("#btn").click(function () {
alert("Clicked");
 });
});
```

▼ jQuery Selector Engine

jQuery selectors are used to **find HTML elements** and return them as a **jQuery object**.

Internally, jQuery uses a powerful engine called **Sizzle** to parse selectors and match elements.

```
$("selector")
```

- Input → CSS-like selector
- Output → jQuery object (collection of elements)

Basic Selectors

Tag Selector

```
$("p")
```

Selects **all** `<p>` **elements**

```
$("p").css("color","blue");
```

ID Selector

```
$("#title")
```

Selects element with `id="title"`

Class Selector

```
$(".box")
```

Selects **all elements** with class `box`

Attribute Selector

Basic Attribute Selector

```
$("[type='text']")
```

Selects:

```
<input type="text">
```

Common Attribute Variants

Selector	Meaning
[name]	Has attribute
[type='text']	Exact match
[href^='https']	Starts with
[href$='.pdf']	Ends with
[href*='google']	Contains

Example

```
$("input[type='password']").css("border","2px solid red");
```

Hierarchy (Relationship) Selectors

These depend on **HTML structure**.

Descendant Selector (Space)

```
$("div p")
```

Selects **all** `<p>` **inside** `<div>` (any depth)

```
<div>
<section>
<p>Hello</p><!-- selected -->
</section>
</div>
```

Child Selector (`>`)

```
$("div > p")
```

Selects **direct children only**

```
<div>
<p>Hello</p><!-- selected -->
<section>
<p>Hi</p><!-- NOT selected -->
</section>
</div>
```

Pseudo Selectors

Pseudo selectors filter elements **by position or state**.

Position-based

```
$("li:first")
$("li:last")
$("li:even")
$("li:odd")
```

Indexing starts from 0

Selector	Index
`:even`	0, 2, 4
`:odd`	1, 3, 5

Example

```
$("li:even").css("background","lightgray");
```

Visibility-based

```
$(":visible")
```

```
$(":hidden")
```

`:hidden` includes:

- `display: none`
- `type="hidden"`
- `width/height = 0`

Edge Case

```
<div style="visibility:hidden"></div>
```

Still counts as **visible** in jQuery

(because space exists)

Multiple Selectors (Comma `,`)

Select **multiple unrelated elements**.

```
$("p, h1, .box")
```

Selects:

- all `<p>`
- all `<h1>`
- all `.box`

Example

```
$("input, textarea").css("border","1px solid blue");
```

▼ DOM Manipulation

Getter vs Setter

If method has NO argument → Getter

If method has argument → Setter

`.html()` – Work with HTML Content

Getter

```
let content = $("#box").html();
```

Returns **HTML string** inside element

```
<div id="box"><b>Hello</b></div>
```

Setter

```
$("#box").html("<i>Welcome</i>");
```

Replaces inner HTML

`.text()` – Work with Plain Text

Getter

```
$("#box").text();
```

Returns **text only** (HTML ignored)

```
<p>Hello<b>World</b></p>
```

Setter

```
$("#box").text("<b>Hello</b>");
```

Output:

```
&lt;b&gt;Hello&lt;/b&gt;
```

`.val()` – For Form Inputs (VERY IMPORTANT)

Used with:

- `<input>`
- `<textarea>`
- `<select>`

Getter

```
let name = $("#username").val();
```

Setter

```
$("#username").val("Nisharg");
```

Sets input value

`.attr()` – Get / Set HTML Attributes

Getter

```
$("#link").attr("href");
```

Reads attribute value

Setter

```
$("#link").attr("href","https://google.com");
```

Updates HTML attribute

Set Multiple Attributes

```
$("#img").attr({
src:"a.jpg",
alt:"Photo"
});
```

`.prop()` – Get / Set DOM Properties (IMPORTANT)

Used for:

- `checked`
- `selected`
- `disabled`
- `readonly`

Example

```
$("#chk").prop("checked",true);
```

Reflects current state

Getter

```
$("#chk").prop("checked");// true / false
```

`.removeAttr()` – Remove HTML Attribute

```
$("#input").removeAttr("disabled");
```

Removes attribute from markup

`.css()` – Apply Inline Styles

Single Property

```
$("#box").css("color","red");
```

Applies inline style

Not reusable

Multiple Properties (Object Form)

```
$("#box").css({
color:"white",
backgroundColor:"black",
padding:"10px"
});
```

`.addClass()` – Apply CSS Class

HTML

```
<div id="box"></div>
```

CSS

```
.active {
background: green;
color: white;
}
```

jQuery

```
$("#box").addClass("active");
```

Add Multiple Classes

```
$("#box").addClass("active highlight");
```

`.removeClass()` – Remove CSS Class

```
$("#box").removeClass("active");
```

`.toggleClass()` – Add / Remove Automatically

Basic Toggle

```
$("#box").toggleClass("active");
```

- If class exists → remove
- If not → add

Conditional Toggle (Advanced)

```
$("#box").toggleClass("active", isLoggedIn);
```

`true` → add

`false` → remove

.hasClass() – Check Class Presence

```
if ($("#box").hasClass("active")) {
console.log("Box is active");
}
```

Returns `true` / `false`

Common in logic conditions

▼ Event Handling

1. Mouse Events

Used for mouse interactions.

Event	Description
click()	Fires when element is clicked
dblclick()	Fires on double click
mouseenter()	Mouse enters element (no bubbling)
mouseleave()	Mouse leaves element (no bubbling)
mouseover()	Mouse enters element (bubbles)
mouseout()	Mouse leaves element (bubbles)
mousemove()	Mouse moves over element
mousedown()	Mouse button pressed

Event	Description
mouseup()	Mouse button released
contextmenu()	Right-click event

Example

```
$("#btn").click(function () {
alert("Button clicked");
});

$("#btn").click(); // triggers event
```

2. Keyboard Events

Triggered by keyboard actions.

Event	Description
keydown()	Key is pressed
keyup()	Key is released
keypress()	Key is pressed & released (deprecated)

Example

```
$("#input").keydown(function () {
console.log("Key pressed");
});
```

3. Form Events

Used while interacting with form elements.

Event	Description
submit()	Form is submitted
change()	Value changes (input, select)
focus()	Element gains focus
blur()	Element loses focus
focusin()	Focus (bubbles)

Event	Description
`focusout()`	Blur (bubbles)
`select()`	Text is selected
`input()`	Fires on every input change

Example

```
$("input").focus(function () {
  $(this).css("background","lightyellow");
});
```

4. Document & Window Events

Used for page load, resize, scroll, etc.

Event	Description
`ready()`	DOM is fully loaded
`load()`	Page including images loaded
`resize()`	Window resized
`scroll()`	Page is scrolled
`unload()`	Page is unloaded

Example

```
$(document).ready(function () {
console.log("DOM Ready");
});
```

5. Clipboard Events

Used for copy-paste actions.

Event	Description
`copy()`	Content copied
`cut()`	Content cut
`paste()`	Content pasted

6. Animation Events

Related to animations.

Event	Description
animate()	Create custom animation
stop()	Stop animation
delay()	Delay animation
finish()	Complete animation immediately

7. Event Binding / Handling Methods

Ways to attach or remove events.

Method	Description
on()	Attach event(s)
one()	Attach event (runs once)
off()	Remove event
trigger()	Fire event manually
triggerHandler()	Fire without bubbling

Example

```
$("#btn").on("click mouseenter",function () {
console.log("Event fired");
});
```

Define Custom Event

```
$("#form").on("process",function () {
validate();
saveData();
});
```

Trigger Custom Event

```
$("#btn").click(function () {
  $("#form").trigger("process");
});
```

Passing Data with Custom Events

```
$("#box").on("highlight",function (e, color) {
  $(this).css("background", color);
});

$("#btn").click(function () {
  $("#box").trigger("highlight", ["yellow"]);
});
```

▼ DOM Tree Navigation

- **Up** → `parent, parents, closest`
- **Down** → `children, find`
- **Sideways** → `siblings, next, prev`
- **Iterate** → `each`

`.parent()` vs `.parents()` (UPWARD TRAVERSAL)

`.parent()`

```
$(".item").parent();
```

`Returns` **`immediate parent only`**

`.parents()`

```
$(".item").parents();
```

`Returns` **`all ancestors up to`** `<html>`

`.children()` (DOWN – DIRECT ONLY)

```
$("#list").children();
```

Returns **direct children only**

Filter Children

```
$("#list").children(".active");
```

`.siblings()` (SIDEWAYS)

```
$(".active").siblings();
```

Returns **all siblings except itself**

Filter Siblings

```
$(".active").siblings(".item");
```

`.next()` & `.prev()` (SIDEWAYS STEP-BY-STEP)

`.next()`

```
$(".item").next();
```

Next sibling only

`.prev()`

```
$(".item").prev();
```

Previous sibling only

`.find()` (DOWN – ANY DEPTH)

```
$("#box").find("p");
```

Finds **all descendants**

.closest()

Starts from **current element**, goes **up**, stops at **first match**

Example

```
<div class="card">
<button class="btn">Click</button>
</div>
```

```
$(".btn").closest(".card");
```

Returns `.card`

.each()

Syntax

```
$(".item").each(function (index, element) {
console.log(index, element);
});
```

▼ jQuery Validation

jQuery Validation Plugin is a JavaScript library that:

- Validates **HTML forms on the client side**
- Prevents submitting **invalid data**
- Shows **error messages automatically**
- Saves server load by catching errors early

```
<script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/jquery-validation@1.19.5/dist/j
```

```
query.validate.min.js"></script>
```

Basic Form Example

HTML

```
<form id="myForm">
<label>Username</label>
<input type="text"name="username">

<label>Email</label>
<input type="email"name="email">

<button type="submit">Submit</button>
</form>
```

Basic Validation Rules

JavaScript

```
$("#myForm").validate({
rules: {
username: {
required:true,
minlength:3
   },
email: {
required:true,
email:true
   }
 },
messages: {
username: {
required:"Username is required",
minlength:"Minimum 3 characters"
   },
email: {
```

```
required:"Email is required",
email:"Enter valid email"
   }
 }
});
```

Most Common Built-in Rules

Rule	Meaning
required: true	Field cannot be empty
minlength: 3	Minimum length
maxlength: 10	Maximum length
email: true	Valid email format
number: true	Only numbers
digits: true	Only digits
url: true	Valid URL
equalTo: "#id"	Match another field
range: [1,100]	Value range

Password + Confirm Password Example

HTML

```
<input type="password" id="pass" name="password">
<input type="password" name="confirm">
```

JS

```
$("#myForm").validate({
rules: {
password: {
required:true,
minlength:6
```

```
    },
confirm: {
equalTo:"#pass"
    }
  },
messages: {
confirm: {
equalTo:"Passwords do not match"
    }
  }
});
```

Styling Errors

```
.error {
color: red;
font-size:14px;
}
```

jQuery Validator **automatically adds** `.error` class.

Custom Validation

Sometimes built-in rules are **not enough**.

Example: Username must start with a letter

Step 1: Create Custom Rule

```
$.validator.addMethod("startsWithLetter",function (value) {
return /^[A-Za-z]/.test(value);
},"Must start with a letter");
```

Step 2: Use It

```
$("#myForm").validate({
rules: {
username: {
required:true,
startsWithLetter:true
   }
 }
});
```

Custom Validation with Parameters

Example: Minimum Age Check

```
$.validator.addMethod("minAge",function (value, element, age) {
return value >= age;
},"Age is too low");
```

```
rules: {
age: {
required:true,
minAge:18
 }
}
```

Prevent Form Submit Until Valid

```
$("#myForm").validate({
submitHandler:function (form) {
alert("Form is valid!");
   form.submit();
 }
});
```

▼ jQuery Utility Functions

- Called as **static methods** on `$`

- Syntax:

```
$.methodName(...)
```

$.each() – Iterate Anything

Used to loop through:

- Arrays
- Objects
- jQuery collections

Array Example

```
let arr = ["JS","jQuery","React"];

$.each(arr,function (index, value) {
console.log(index, value);
});
```

Object Example

```
let user = {name:"Nisharg",age:21 };

$.each(user,function (key, value) {
console.log(key, value);
});
```

$.map() – Transform Data

Example

```
let nums = [1,2,3];

let squares = $.map(nums,function (n) {
return n * n;
```

```
});

console.log(squares);// [1, 4, 9]
```

$.trim() – Remove Extra Spaces

```
let name ="   Nisharg   ";
console.log($.trim(name));// "Nisharg"
```

$.extend() – Merge Objects

Used for:

- Config merging
- Default options
- Plugin development

Shallow Copy

```
let defaults = {theme:"light",show:true };
let options  = {show:false };

let settings = $.extend(defaults, options);
```

Result:

```
{theme:"light",show:false }
```

Deep Copy

```
$.extend(true, {}, obj1, obj2);
```

Parameter	Meaning
true	deep copy

Parameter	Meaning
{}	target
others	sources

$.type() – Accurate Type Detection

```
$.type([]);// "array"
$.type({});// "object"
$.type(null);// "null"
$.type(function(){});// "function"
```

$.inArray() – Find Item Index

```
let fruits = ["apple","banana","mango"];

$.inArray("banana", fruits);// 1
$.inArray("grape", fruits);// -1
```

$.isArray() – Check for Array

```
$.isArray([1,2,3]);// true
$.isArray({});// false
```

$.isFunction() – Check for Function

```
$.isFunction(function(){});// true
$.isFunction(123);// false
```

$.grep() – Filter an Array

`$.grep()` **filters elements** of an array based on a condition.

```
let numbers = [1,2,3,4,5];

let even = $.grep(numbers,function (n) {
return n %2 ===0;
});

console.log(even);// [2, 4]
```

invert Filter

```
let odd = $.grep(numbers,function (n) {
return n %2 ===0;
},true);

console.log(odd);// [1, 3, 5]
```

`invert = true` → reverse selection

$.merge() — Combine Two Arrays

Syntax

```
$.merge(array1, array2);
```

Basic Example

```
let a = [1,2];
let b = [3,4];

$.merge(a, b);

console.log(a);// [1, 2, 3, 4]
```