10. jQuery

10.1 What is jQuery? Why do we need it and what can we do with it?

- **Definition**:
    - jQuery is just a JavaScript library that you include on your web page. As jQuery is built on top of JavaScript, it provides all the functionalities of JavaScript. We can say from above points jQuery is enhanced version of JavaScript.
    - As a library, it provides many built-in functions using which you can accomplish various tasks easily and quickly.
    - jQuery is different from vanilla JavaScript because is basically a simpler way of doing things in a more compatible way
    - jQuery library contains features like HTML/DOM manipulation, CSS manipulation, CSS manipulation and effects and animations
- **What can you do with jQuery?** jQuery is JavaScript and does not add anything to the JavaScript language itself. Whatever you do with vanilla JavaScript, you can do it with jQuery too. But jQuery does the following jobs in a very easy way for developers:
    - **DOM manipulation**:
        - **Selecting**: jQuery provides simpler and CSS-like way of selecting HTML elements
        - **Updating**: Multiple jQuery methods provide different ways to update, animate and loop through elements in a much simpler way
    - **Handling events:** Allows you to bind events with elements without the need to write a fall back for older versions
    - **Effects and animations**: jQuery also allows us to add animated effects on our web page which takes a lot of pain and lines of code with JavaScript.
- **Why is jQuery chosen?**
    - **Simplicity**: The main reason we need jQuery is because it makes JavaScript easier when we use it on website, in fact, jQuery's motto is "write less, do more". How?
        - jQuery takes a lot of common tasks that require many lines of vanilla JavaScript code to accomplish and wraps them into methods that you can call with a single line of code.

        - DOM manipulation in jQuery is simplified: The jQuery made it easy to select DOM elements, negotiate them and modify them using a cross-browser open-source selector engine called Sizzle
        - Loop through elements
        - Simplified animation and fading of elements
        - Better event handling
        - **Cross browser compatibility**: Unlike vanilla JavaScript , jQuery provides a consistent result throughout major browsers
    - **Simpler selection and updating method**: If you use vanilla JavaScript, some methods that you use for selection do not work on older browsers
    - **jQuery is Easy to learn:** jQuery is very simple as it is built on a shorter code than JavaScript. Also, the developers don't need any design talent or learn web designing as uses of jQuery provides an abundant set of built-in plugins.

10.2 Adding jQuery library to our web page

- There are two ways to include jQuery into your HTML
    - You can download the latest version, save it locally in your computer and include it in your HTML's <header> using a simple <script > tag. Link to offcicial jQuery website:
        - https://jquery.com/download/
    - Including jQuery from a CDN and adding it right before the closing </body> in <script> tag. Note: Your jQuery CDN needs to be placed before your custom JavaScript file
- It is always recommended to use the latest version available. Currently, 3.6.0 is the latest (Jan 2022)
        - https://code.jquery.com/jquery-3.6.0.min.js

10.3 Selecting elements with jQuery (id, class, element selectors)

- **jQuery selectors**: They select HTML elements based on their name, id, classes, types, attributes, values of attributes and much more.
    - jQuery selectors mostly use the existing CSS Selectors and, in some instances, custom selectors.

- **jQuery() vs $()**: All jQuery selectors use a function called jQuery(). However, there is a shorthand representation of the jQuery() method, just using the dollar sign at the start and add parentheses like this: $().
    - Therefore, jQuery() is the same as $()
    - **Note 1**: The jQuery function/ jQuery() always returns a jQuery object (that is based on an array). You will see this in example when jQuery selectors are explained
    - **Note 2**: Please note that jQuery() returns an object even if there are no elements that matches the selector. If the jQuery object contains no elements, it will simply do nothing.

- **jQuery #id Selector**: jQuery #id selector uses the id attribute of an HTML tag to find the specific element. Let's use the following HTML to explain id, class and element selectors in jQuery
    - **Example**: Let's select an element that has id name called "myParagraph"

```
<div class="myDiv">jQuery</div>
   <div>
    <p id="myParagraph">Greetings from p</p>
    <span>Greetings from span</span>
    <div class="myDiv">Greetings from div</div>
 </div>
var logParagraph = $("#myParagraph");
console.log(logParagraph);// logs our paragraph
```

- **jQuery .class Selector**: jQuery .class selector finds elements with a specific class.
    - **Example**:Using the above HTML, Let's select elements that have "myDiv" as class name

```
var logDiv = $(".myDiv");
console.log(logDiv);// logs the two divs with
"myDiv" class name
```

- **jQuery element selector**: jQury element selector selects elements based on the element name.
    - **Example**: Using the above HTML, let's select our span element

```
console.log($("span"));// returns our <span> element
```

10.4 Selecting elements with jQuery (filters)

- **jQuery (filters)**: jQuery provides several filter methods to narrow down the search for elements in a DOM tree.
    - **first()**: The first() method filters the set of matched elements and returns the first element of the specified elements. **Example**: The script below will select all s and filters the first <li> only and change its background color to pink

```
$("li").first().css("background-color", "pink");
```

    - **last()**: This method returns the last element of the specified elements. **Example**: The script below will select all s and filters the last only and change its background color to pink

```
$("li").last().css("background-color", "pink");
```

    - **even() and odd()** : These method filter the set of matched/selected elements and returns those with an even/odd index number. **Example**: Let's select all s and make background color of those s with even index green and make background color of those s with odd index purple

```
$("li").even().css("background-color", "green");
$("li").odd().css("background-color", "purple");
```

    - **nth-child() or nth-children()**: This method selects all elements that are the nth child/children, of their parent. **Note**: The element type is not considered here, if it is the nth child, this method returns that element. **Example**: Let's select a <ul> that has s as children and change the background color of the 2nd child <li> only

```
$("ul li:nth-child(2)").css("background-color", "lightgreen");
```

- **Content filters**
  - **has(selector)**: This method filters the set of matched elements and returns only those elements that has the specified descendant element. **Note**: To select elements that have multiple elements inside of them, use comma. **Example**: Let's select a that has <p> in it, and change this 's background color

```
$("div").has("p").css("background-color", "pink");
```

  - **:contains(text)**: This selector selects all elements containing the specified string. **Note**: The string can be contained directly in the element as text, or in a child element. **Example**: Now, let's select a <ul> that has "First list item" as text directly in it or in its children.

```
$("ul:contains(First list item)").css("background-
color", "pink");
```

  - **empty()**: The empty() method removes all child nodes and content from the selected elements.
- **Visibility**
  - **hidden()**: This method hides the selected elements just like CSS property display: none. **Example**: let's hide our <ul> that has id name of firstUL

```
$(" ul[id=firstUL] ").hide();
```

  - **visible()**: The method selects every element that is currently visible. If an element is already invisible (using ways like display: none, type="hidden"), the visible() method will not apply to it. **Example**: Let's select any <ul> that is visible and change its background color

```
$("ul:visible").css("background-color",
"lightgreen");
```

- **show()**: The method shows the hidden, selected elements. **Note**: The show() shows hidden with jQuery methods and display: none in CSS, it will not show if element is hidden using CSS property display: none. **Example**: Now, let's show a <ul> that ha is hidden, if there is any

```
$(" ul").show();
```

10.5 Updating or altering values: content and elements

- jQuery object has many methods that you can use on the element that you select. These are the common methods that you will use. Methods to get or change content of elements, attributes or nodes
- **Updating content**:
  - **html()**: This method sets/returns the content of selected elements (including HTML markup). **Example**: Let's print in the console the text, including our <li> tag, of our <li> that has a specific class name

```
console.log($("li[class=otherLi]").html());
```

  - **text()**: The method sets/returns the text content of selected elements. **Example**: Let's print in the console the text of our <li> that has a specific class name

```
console.log($("li[class=firstLi]").text());
```

  - **remove() and empty()**: The remove() method removes the selected element and all its child elements. The empty() method removes only the child elements of the selected element, meaning, the selected element itself will not be removed, but its child/children. This method sets/returns the value of form fields. **Example**: Let's remove our and its child <p>s first and then go to emptying our 's children. **Notice**: We styled our to show that it won't be affected by empty() method.

```
 <div style="border:1px solid; height:100px">I
am a parent div
```

```
<p>I am first</p>
<p>I am second</p>
<p>I am third</p>
</div>
$("div").remove();// removes our parent <div>
with child <p>
$("div").empty();// removes only the child <p>
```

- **Updating elements**:
  - **before()**: The before() method inserts specified content in front of (before) the selected elements. **Example**: Let's put our <p> with id="firstPar" right before our

```
<div id="divId" style="border:1px solid;
height:160px">I am a parent
<p id="firstPar">I am first</p>
<p>I am second</p>
<p>I am third</p>
</div>
$("#divId").before($("#firstPar"));
```

  - **after()**: The after() method inserts specified content in behind the selected elements. **Example**: Let's put our <p> with id="firstPar" right after our

```
 <div id="divId" style="border:1px solid;
height:160px"> I am a parent

<p id="firstPar">I am first</p>

<p>I am second</p>
<p>I am third</p>
</div>
$("#divId").after($("#firstPar"));
```

- **prepend()**: The prepend() method inserts specified content inside the selected element at the beginning of this selected element. In short, it puts the selected element at the first index. **Example**: Let's put our <p> that has "firstPar" id at the beginning of our

```
<div id="divId" style="border:1px solid
height:160px">
<p id="firstPar">I am first</p>
<p>I am second</p>
<p>I am third</p>
 </div>
$("#divId").prepend($("#firstPar"));
```

- **append()**: The append() method inserts specified content inside the selected element at the end of this selected element. In short, it puts the selected element inside an element at the last index. **Example**: Let's put our <p> that has "firstPar" id at the end of our

```
<div id="divId" style="border:1px solid
height:160px">
<p id="firstPar">I am first</p>
<p>I am second</p>
<p>I am third</p>
 </div>
$("#divId").append($("#firstPar"));
```

10.6 Altering values: attributes, form value, looping through elements

- **Attributes**:
  - **addClass()**: This method adds one or more property (CSS class) to each selected element. This method does not remove existing class attributes, it only adds one or more class names to the class attribute. **Example**: Let's add two CSS classes to our paragraphs using addClass() and change their background and text color.

```
.classOne {background-color: blueviolet;}
.classTwo {color: yellow;}
$("p").addClass("classOne classTwo");
```

- **removeClass()**: The removeClass() method removes one or more class names from the selected elements. **Example**: Let's remove one of the above CSS classes (the "classOne") from our paragraphs using removeClass(). This removes only the backgroundcolor from our paragraphs

```
$("p").removeClass("classOne");
```

- **css()**: The css() method in JQuery is used to change the style property of the selected element. **Example**: Let's change the background color of our <p>tags using the css() method

```
$("p").css("background-color", "green");
```

- **Form value**:
  - **val()**: This method sets or returns the value of form fields. The value of an input is for example, the value of the value attribute in an <input>. **Example**: Let's print in the console what we provided under the value attribute for our <input>

```
console.log($("input").val());
```

  - **$.isNumeric()**: This method checks whether a value is numeric/a number and returns true if only the argument passed to it a number or a numeric string. **Example**: Let's use the $.isNumeric method to see if our <input>'s value attribute has a numeric or non-numeric value. Since the value is "happy" string, the following script should return false

```
<form id="myForm" method="POST">
<label for="firstName">First Name</label>
<input id="firstInput" value="happy">
</form>
```

```
var inputValue = $("input#firstInput").val();
console.log($.isNumeric(inputValue));//
prints false
```

- **Finding elements (jQuery traversing)**: We have seen previously that traversing the DOM means moving through DOM/HTML elements to find/select them based on their relation to other elements. jQuery has its own traversing methods to select HTML elements based on ancestor, child, sibling and other related elements.
- **jQuery traversing**: **Methods to up the DOM**
    - **parent()**: returns the direct parent element of the selected element
        - **Example**: Let's style our using its child <ul> and applying

```
the parent() method
<div>
<p>I am div's child</p>
   <ul>
       <li>UL is my dad. Div is my granddad</li>
    </ul>
</div>
$(document).ready(function () {
$("ul").parent().css({ border: "2px solid red" });
});
```

    - **parents()**: returns all ancestor elements of the selected element
        - **Example**: Let's style all ancestor elements of our

```
<div>
    <ul>
    <li>ul and div are my ancestors. </li>
    </ul>
</div>
$("li").parents().css({ border: "2px solid red" });
```

- **Methods to traverse down the DOM**
  - **children()**: returns all direct children of the selected element.
    - **Example**: let's style all direct children of div. Here, grand children like the <li> will not be affected

```
<div>
    <p>div is my dad</p>
    <span>div is my dad too</span>
<ul>
    <li>I am div's descendant</li>
    <li>I also div's descendant</li>
</ul>
</div>
$(document).ready(function () {
   $("div").children().css({border: "2px solid" });
});
```

  - **find()**: returns descendant elements of the selected element
    - **Example**: Let's find only the <li> descendants of our and style them

```
<div>
<p>div is my dad</p>
<span>div is my dad too</span>
<ul>
<li>I am div's descendant</li>
<li>I also div's descendant</li>
</ul>
</div>
$("div").find("li").css({ border: "2px solid red" });
```

- **Methods to travers sideways in the DOM**
  - **siblings()**: returns all sibling elements of the selected element.

- **Example**: Let's style all sibling elements of our

```
<div>
 <p>div is my dad</p>
 <span>div is my dad too</span>
 <ul>
    <li>I am div's descendant</li>
    <li>I also div's descendant</li>
</ul>
  </div>
$("span").siblings().css({ border: "4px solid red" });
```

- **next()**: returns the next sibling element of the selected element.
  - **Example**: Let's use the above HTML and style 's next sibling, which is the

```
$("span").next().css({ border: "4px solid red" });
```

- **nextAll()**: returns all next sibling elements of the selected element.
  - **Example**: Let's use our <p> and style all of its next siblings

```
$("p").nextAll().css({ border: "4px solid red" });
```

- **Looping through elements**: Earlier we said that jQuery returns an object (array-like collection) when we create a new element or select an existing element using jQuery. Using jQuery to loop through elements is simplified way better than vanilla JavaScript. jQuery does the looping behind the scene. Let's compare the code will use if we want to change the background color of all <li> elements using vanilla JavaScript and jQuery.
  - **Assume you have the following HTML**

```
<ul>
    <li>I am 1st li</li>
    <li>I am 2nd li</li>
    <li>I am 2nd li</li>
```

```
    <li>I am 2nd li</li>
</ul>
```

- **Example of looping through elements using vanilla JavaScript**

```
var element1 =
document.getElementsByTagName("li");
for (let i = 0; i < element1.length; i++) {
var allLi = element1[i];
allLi.style.backgroundColor = "pink";
}
```

- **Example of looping through elements using jQuery**

```
$("li").css("background-color", "pink")
```

10.7 Handling events

- jQuery again makes binding and handling events easier
- We learned previously that events are actions taken by a user for which we write JavaScript code to respond to. We use JavaScript code, called event handlers, to respond to these events. We also learned that event methods trigger or attach a function to an event handler for the selected elements.
- To handle events in jQuery, just like any other method in jQuery, you start by selecting the element you want to respond to. We said that selection of elements in jQuery creates a jQuery object. It is on these objects that we apply the different jQuery event methods. jQuery has various methods to attach a handler function to an event for selected elements.
- **jQuery's on() method**: Below, we will discuss the most used jQuery event method, the on() method.
    - The on() method is used to handle various kinds of events on selected elements
    - **Syntax**: $(selector).on(event, childSelector, data, function, map)
    - **Note**: Event handlers attached using the on() method will work for both current and FUTURE elements (on new element created by JavaScript dynamically).

- The on() method in jQuery accepts an object containing multiple events and handlers. Meaning, it does the following things:
- **Bind any event triggered on the selected elements to an event handler**
    - **Example**: When any <p> is clicked, it will print "'<p> was clicked'

```
$( "p" ).on( "click", function() {
    console.log( "<p> was clicked" );
});
```

- **Bind multiple events to one event handler**
    - **Example**: Let's trigger the same handler function whenever the mouse hovers over or leaves our

```
$( "div" ).on( "mouseenter mouseleave",
function() {
console.log( "mouse hovered over or left a div" );
});
```

- **Bind multiple events and multiple handlers to the selected elements**
    - **Example**: Let's assign multiple event handler functions for multiple events to the selected div

```
$("div").on({
    click: function () {
    console.log("clicking div");
    },
    mouseenter: function () {
    $("div").css("background-color", "yellow");
    },
    mouseleave: function () {
    console.log("mouse leaving div");
    },
    });
```

- **Other most popular jQuery event methods:**
  - **UI/DOM element events**: focus, blur, change
    - **blur()**: This method triggers the blur event or attaches a function when blur event occurs. Blur event occurs when an element loses focus. **Example**: Let's alert a text and also change the input's background color to red when the <input> loses focus because user moved away the pointer from it.

```
$("input").blur(function () {
    $("input").css("background-color", "red");
    alert("This text box has lost its focus");
    });
```

  - **Keyboard events**:
    - **keypress():** This method attaches a function to run when a keypress event occurs. **Note**: keypress event is not fired for all keys (Ex. ALT, CTRL, SHIFT, ESC). **Example**:

```
$("input").keypress(function () {
    $("label").css("background-color", "pink");
    });
```

  - **Mouse events**: click, dbclick, mouseenter, mouseleave
    - **click()**: The click() method attaches an event handler function to an HTML element when the user clicks on the HTML element. **Example**: Let's make a hide when <p> is clicked

```
$("p").click(function(){
  $("div").hide();
});
```

  - **Form events**: submit, change
    - **submit()**: This method is used to submit an HTML form i.e. <form> or to attach a function to run when a submit event occurs. **Note**: This event can

only be used on <form> elements. **Example**: Let's change the color of our <input> when we submit our form

```
$("form").on("submit",
backgroundChanger);
 function backgroundChanger() {
$("#firstInput").css("background-
color", "pink");
}
```

- **Browser/window /document events**: ready(), load(), resize(), unload(), error(), scroll()
    - **$(document).ready()**: This method allows us to execute a function when the document is fully loaded. It is good practice to wait for the document (HTML and CSS) to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section. **Example**: Below, let's show an alert once the document is fully loaded

```
$(document).ready(function(){
    alert("Document is now ready!");
 });
```

- Please refer this link to find the list of all jQuery event methods:
    - https://www.w3schools.com/jquery/jquery_ref_events.asp

10.8 Effects and animations in jQuery

- One of the advantages of jQuery is, it lets us easily enhance our web pages by adding some transition and movement effects
    - **Some of the popular jQuery effects**: hide(), show(), toggle()
        - **toggle()**: The toggle() method is used to check the visibility of selected elements to toggle between hide() and show() for the selected elements. **Note**: We can pass speed in milliseconds for the toggle effects. In the following example paragraph element will take 2000 milliseconds time to

get hidden and displayed completely. **Example**: Let's hide and show our <p> with "thirdPar" id name when we click our <p> that has an id of "firstPar". It will take 1000 milliseconds for the <p> to hide or show upon click

```
<p id="firstPar">I am first</p>
<p id="thirdPar">I am second</p>
$("p#firstPar").on("click", function () {
$("p#thirdPar").toggle(1000);
});
```

Some of jQuery's sliding effects: slideUp(), slideDown(), slideToggle()

- **slideUp()**: It is a method that slides-up (gradually hides) the selected elements. **Example**: Let's slide our <p> up with id of "thirdPar" when a button is clicked

```
$("#myButton").click(function () {
    $("#thirdPar").slideUp(3000);
    });
```

- **slideDown()**: This method slides (gradually hides) an element down. Example: Let's slide our <p> up with id of "thirdPar" when a button is clicked

```
$("#myButton").click(function () {
    $("#thirdPar").slideUp(3000);
    });
```

- **slideToggle()**: This method toggles between the slideDown and slideUp methods. Meaning, if the element(s) has been slid down already, slideToggle() will slide it/them down and vice versa. We can basically avoid the use of slideUp() and slideDown() and just use slideToggle().

Example: Let's just toggle between sliding up and sliding down of our <p> when button is clicked.

```
<button>click for slide toggle</button>
 <p>Hello there!</p>
$("button").click(function () {
$("p").slideToggle(3000);
});
```

- **Some of jQuery's fading Effects**: fadeIn(), fadeOut(), fadeToggle()

```
<button id="myButton"> click for FadeOut </button>

<button id="myOhterButton">click for
FadeIn</button>
<p id="firstPar">I am first</p>
<p id="secondPar">I am second</p>
<p id="thirdPar">I am third</p>
```

fadeOut(): This method gradually changes the opacity, for selected elements, from visible to hidden (fading effect). **Example**: Let's use the HTML above and fade out our <p>s with different speed

```
$("#myButton).click(function () {
    $("#firstPar").fadeOut();
    $("#secondPar").fadeOut("slow");
    $("#thirdPar").fadeOut(3000);
    });
```

- **FadeIn()**: This method gradually changes the opacity, for selected elements, from hidden to visible. **Example**: Using the above HTML, let's gradualy change the opacity of our third <p> when #myOhterButton is clicked.

```
$("#myOhterButton").click(function () {
    $("#thirdPar").fadeIn(1000);
    });
```

- **Custom**: delay(), stop(), animate()
    - **delay()**: This method sets a timer to delay the execution of the next item in the queue. The method is for instance used to make a delay between the queued jQuery effects. **Example**: Let's delay the sliding up and down of our <p> by 3000 milliseconds when our <button> is clicked.

```
<button>click for slide toggle</button>
<p>Hello there!</p>
$("button").click(function () {
$("p").delay(3000).slideToggle();
});
```

    - **animate()**: This method makes the CSS property value change gradually to create an animation effect. Note: Only CSS properties with numerical values (such as padding: 15px) can be animated, unless we want the animation to be between "hide" and "show" or "toggle". **Example**: Let's add animation on our <p> by changing the already existing CSS width, height and background color when the <button> is clicked.

```
<button>click for animation</button>
 <p style="width:50px;height:50px
;background-color:red;">Hello there!</p>
$("button").click(function () {
$("p").animate({ height: 300 }, "3000");
$("p").animate({ width: 300 }, "3000");
$("p").css("background-color", "blue");
});
```

- **stop()**: This method is used to stop animations or effects before it is finished. **Example**: Assuming there was an animation (slideUp on our <p>), let's stop that animation using the stop() method. While the <p> is sliding up, click on the <button> for stop slide up to stop <p> from continuing to slide up.

```
<button id="button1">click for slide up</button>
<button id="button2">click to slop slide up</button>
<p style="background-color:red">Hello there!</p>
$("#button1").click(function () {
$("p").slideUp(3000);
});
$("#button2").click(function () {
$("p").stop();// stops the above slideUp effect if
clicked
});
```

10.9 Conclusion

- jQuery is just a JavaScript library that you include on your web page. As jQuery is built on top of JavaScript, it provides all the functionalities of JavaScript. It is important to know that jQuery is an enhanced version of JavaScript.
- jQuery makes DOM manipulation and event handling much simpler as opposed to vanilla JavaScript. It also comes in handy when we want to include animation and effects on our website/application. Moreover, jQuery takes a lot of common tasks that require many lines of vanilla JavaScript code to accomplish and wraps them into methods that you can call with a single line of code.
- Unlike vanilla JavaScript, jQuery provides a consistent result throughout major browsers and is therefore cheered for its cross-browser compatibility.