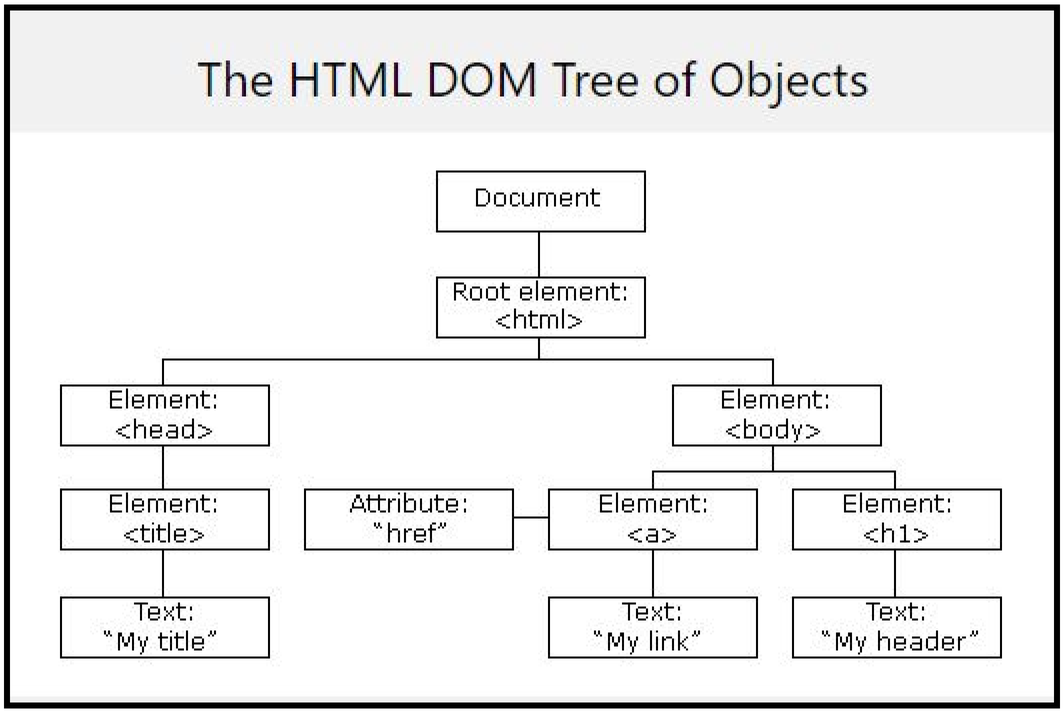
DOM:

- With the HTML DOM (Document Object Model), JavaScript can access and change all the elements of an HTML document.
- When a web page is loaded, the browser creates a Document Object Model of the page.
- The HTML DOM document object is the owner of all other objects in your web page. The document object represents your web page.
- If you want to access any element in an HTML page, you always start with accessing the document object.
- The HTML DOM model is constructed as a tree of Objects:



- With the object model, JavaScript gets all the power it needs to create dynamic HTML:
    - JavaScript can change all the HTML elements in the page.
    - JavaScript can change all the HTML attributes in the page.
    - JavaScript can change all the CSS styles in the page.
    - JavaScript can remove existing HTML elements and attributes.
    - JavaScript can add new HTML elements and attributes.
    - JavaScript can react to all existing HTML events in the page.
    - JavaScript can create new HTML events in the page.
- The HTML DOM is a standard object model and programming interface for HTML. It defines:
    - The HTML elements as objects.
    - The properties of all HTML elements.
    - The methods to access all HTML elements.
    - The events for all HTML elements.

  Ie. The HTML DOM is a standard for how to get, change, add, or delete HTML elements.
- The HTML DOM can be accessed with JavaScript and with other programming languages.
  E.g. Consider the document.getElementById(). The "document" part references the DOM.

- In the DOM, all HTML elements are defined as objects.
- A **property** is a value that you can get or set (like changing the content of an HTML element).
- A **method** is an action you can do (like add or deleting an HTML element).
- E.g. Consider the code snippet below:

```
<html>
<body>

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

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

  This code snippet changes the content (the innerHTML) of the <p> element with id="demo".
  In the example above, getElementById is a method, while innerHTML is a property.
- Below are some examples of how you can use the document object to access and manipulate HTML:

1. **Finding HTML Elements:**

Method	**Description**
document.getElementById(id)	Find an element by element id
document.getElementsByTagName(name)	Find elements by tag name
document.getElementsByClassName(name)	Find elements by class name

2. **Changing HTML Elements:**

Property	**Description**
element.innerHTML = new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.property = new style	Change the style of an HTML element
Method	**Description**
element.setAttribute(attribute, value)	Change the attribute value of an HTML element

3. **Adding and Deleting Elements:**

Method	**Description**
document.createElement(element)	Create an HTML element
document.removeChild(element)	Remove an HTML element
document.appendChild(element)	Add an HTML element
document.replaceChild(new, old)	Replace an HTML element
document.write(text)	Write into the HTML output stream

4. **Adding Events Handlers:**

Method	**Description**
document.getElementById(id).onclick = function(){code}	Adding event handler code to an onclick event

JQuery:

- JQuery is a lightweight, "write less, do more", JavaScript library.
- The purpose of JQuery is to make it much easier to use JavaScript on your website.
- JQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
- JQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

Adding JQuery to Your Web Pages:

- There are several ways to start using JQuery on your web site. You can:
    - Download the JQuery library from JQuery.com.
    - Include JQuery from a CDN, Content Delivery Network, like Google.
- You can download JQuery from jQuery.com. The JQuery library is a single JavaScript file, and you reference it with the HTML <script> tag.
  Note: The <script> tag should be inside the <head> section.
  E.g.
  ```
  <head>
  <script src="jquery-3.4.1.min.js"></script>
  </head>
  ```
- You should place the downloaded file in the same directory as the pages where you wish to use it.
- If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network), which is an external link.
- Both Google and Microsoft host jQuery.
- To use jQuery from Google or Microsoft, use one of the following:
- Google:
  ```
  <head>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">
  </script>
  ```

```
</head>
```

- Microsoft:

```
<head>
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.4.1.min.js"></script>
</head>
```

JQuery Syntax:

- With jQuery you select (query) HTML elements and perform "actions" on them.
- The jQuery syntax is tailor-made for selecting HTML elements and performing some action on the element(s).
- Basic syntax: **$(selector).action()**
    - A $ sign to define/access jQuery.
    - A (selector) to "query" HTML elements.
    - A jQuery action() to be performed on the element(s).
- **Note:** jQuery uses CSS syntax to select elements.
- E.g.
  $(this).hide(): hides the current element.
  $("p").hide(): hides all <p> elements.
  $(".test").hide(): hides all elements with class="test".
  $("#test").hide(): hides the element with id="test".

jQuery Selectors:

- jQuery selectors allow you to select and manipulate HTML element(s).
- jQuery selectors are used to select HTML elements based on their name, id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS selectors.
- All selectors in jQuery start with the dollar sign and parentheses: $().
- **The element Selector:**
- The jQuery element selector selects elements based on the element name.
- E.g. You can select all <p> elements on a page like this: **$("p")**
- E.g. When a user clicks on a button, all <p> elements will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
        $("p").hide();
  });
});
```

- **The #id Selector:**
- The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.
- An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element.
- To find an element with a specific id, write a hash character, followed by the id of the HTML element. E.g. **$("#test")**
- E.g. When a user clicks on a button, the element with id="test" will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
        $("#test").hide();
  });
});
```

- **The .class Selector:**
- The jQuery .class selector finds elements with a specific class.

- To find elements with a specific class, write a period character, followed by the name of the class. E.g. **$(".test")**
- E.g. When a user clicks on a button, the elements with class="test" will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
        $(".test").hide();
  });
});
```

- **More Examples of jQuery Selectors:**

Syntax	**Description**
$("*")	Selects all elements
$(this)	Selects the current HTML element
$("p.intro")	Selects all <p> elements with class="intro"
$("p:first")	Selects the first <p> element
$("ul li:first")	Selects the first <li> element of the first <ul>
$("ul li:first-child")	Selects the first <li> element of every <ul>
$("[href]")	Selects all elements with an href attribute
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"
$(":button")	Selects all <button> elements and <input> elements of type="button"
$("tr:even")	Selects all even <tr> elements
$("tr:odd")	Selects all odd <tr> elements

JQuery Event:

- jQuery is tailor-made to respond to events in an HTML page.
- All the different visitors' actions that a web page can respond to are called **events**.
- An **event** represents the precise moment when something happens.
- Examples of events include:
    - Moving a mouse over an element.
    - Selecting a radio button.
    - Clicking on an element.

- Here are some common DOM events:

Mouse Events	**Keyboard Events**	**Form Events**	**Document/Window Events**
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

- Some commonly used JQuery methods include:
    - **click():**
    - The click() method attaches an event handler function to an HTML element.
    - The function is executed when the user clicks on the HTML element.
    - E.g. When a click event fires on a <p> element; hide the current <p> element:

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

    - **dblclick():**
    - The dblclick() method attaches an event handler function to an HTML element.
    - The function is executed when the user double-clicks on the HTML element.
    - E.g.

```
$("p").dblclick(function(){
  $(this).hide();
});
```

    - **mouseenter():**
    - The mouseenter() method attaches an event handler function to an HTML element.
    - The function is executed when the mouse pointer enters the HTML element.
    - E.g.

```
$("#p1").mouseenter(function(){
  alert("You entered p1!");
});
```

    - **mouseleave():**
    - The mouseleave() method attaches an event handler function to an HTML element.
    - The function is executed when the mouse pointer leaves the HTML element.
    - E.g.

```
$("#p1").mouseleave(function(){
  alert("Bye! You now leave p1!");
});
```

    - **mousedown():**
    - The mousedown() method attaches an event handler function to an HTML element.
    - The function is executed, when the left, middle or right mouse button is pressed down, while the mouse is over the HTML element.

- E.g.

```
$("#p1").mousedown(function(){
  alert("Mouse down over p1!");
});
```

- **mouseup():**
- The mouseup() method attaches an event handler function to an HTML element.
- The function is executed, when the left, middle or right mouse button is released, while the mouse is over the HTML element.
- E.g.

```
$("#p1").mouseup(function(){
  alert("Mouse up over p1!");
});
```

- **hover():**
- The hover() method takes two functions and is a combination of the mouseenter() and mouseleave() methods.
  Note: hover() is not an actual event. If you use on() with this, nothing happens.
- The first function is executed when the mouse enters the HTML element, and the second function is executed when the mouse leaves the HTML element.
- E.g.

```
$("#p1").hover(function(){
  alert("You entered p1!");
},
function(){
  alert("Bye! You now leave p1!");
});
```

- **focus():**
- The focus() method attaches an event handler function to an HTML form field.
- The function is executed when the form field gets focus.
- E.g.

```
$("input").focus(function(){
  $(this).css("background-color", "#cccccc");
});
```

- **blur():**
- The blur() method attaches an event handler function to an HTML form field.
- The function is executed when the form field loses focus.
- E.g.

```
$("input").blur(function(){
  $(this).css("background-color", "#ffffff");
});
```

- **on():**
- The on() method attaches one or more event handlers for the selected elements.
- E.g. Attach a click event to a <p> element.

```
$("p").on("click", function(){
  $(this).hide();
});
```

- E.g. Attach multiple event handlers to a <p> element.

```
$("p").on({
  mouseenter: function(){
```

```
  $(this).css("background-color", "lightgray");
},
mouseleave: function(){
        $(this).css("background-color", "lightblue");
},
click: function(){
        $(this).css("background-color", "yellow");
}
});
```

- **Document Ready:**
- The document ready event is to prevent any jQuery code from running before the document is finished loading. It is good practice to wait for the document 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.
- Here are some examples of actions that can fail if methods are run before the document is fully loaded:
    - Trying to hide an element that is not created yet.
    - Trying to get the size of an image that is not loaded yet.
- Syntax:

```
$(document).ready(function(){
  // jQuery methods go here...
});
```