Hands-on JavaScript Document Object Model (DOM)

Clarusway

> Purpose of the this hands-on training is to teach the students JavaScript DOM.

Learning Outcomes

> At the end of the this hands-on training, students will learn;

- How to change the content of HTML elements
- How to change the style (CSS) of HTML elements
- How to react to HTML DOM events
- How to add and remove HTML elements

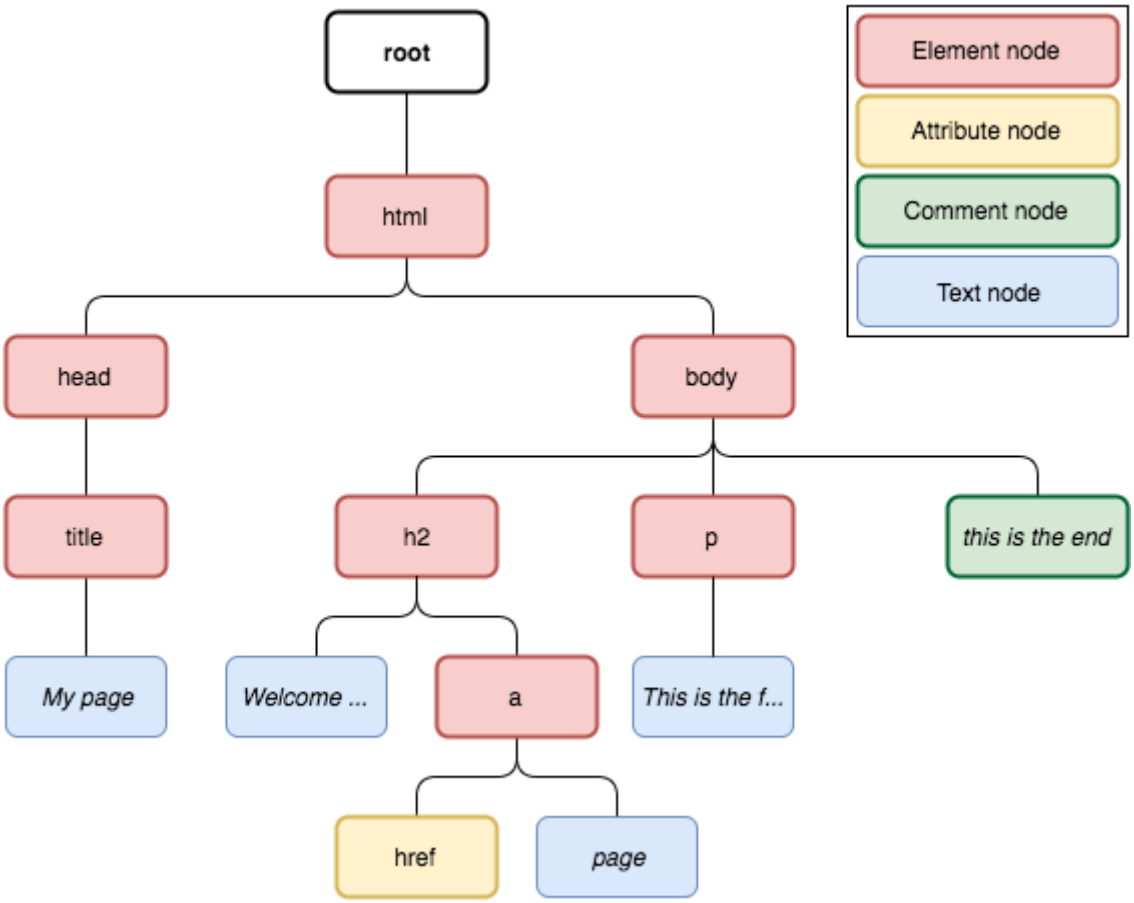
Outline

- Part 1 - What is the DOM?
- Part 2 - Selecting Elements
- Part 3 - JavaScript HTML DOM - Changing HTML
- Part 4 - JavaScript HTML DOM - Changing CSS
- Part 5 - Add Events Using the HTML DOM
- Part 6 - Event Listener
- Part 7 - Remove Event Listener (`removeEventListener()`)

Part 1 - What is the DOM?

- The Document Object Model (DOM) is a programming interface for HTML and XML documents.
- Every element in a document—the document as a whole, the head, tables within the document, table headers, text within the table cells—is part of the document object model for that document, so they can all be accessed and manipulated using the DOM and a scripting language like JavaScript.

```
<html>
  <head>
    <title>My page</title>
  </head>
  <body>
    <h2>Welcome to my <a href="#">page</a></h2>
    <p>This is the first paragraph.</p>
    <!-- this is the end -->
  </body>
</html>
```



The HTML DOM Tree of Objects

Part 2 - Selecting Elements

Object	Property
`getElementById()`	select an element by id.
`getElementsByName()`	select elements by name.
`getElementsByTagName()`	select elements by a tag name.
`getElementsByClassName()`	select elements by one or more class names.
`querySelector()`	select elements by CSS selectors.

JavaScript `getElementById()` Method

- The getElementById() allows you to select an element by its id.

Syntax :

```
let element = document.getElementById(id);
```

- If multiple elements share the same id, even though it is not valid, the getElementById() returns still only the first element it encounters.

JavaScript `getElementsByName()` Method

- Elements on an HTML document can have a name attribute. Unlike the id attribute, multiple element can share the same value of the name attribute.
- To get all elements with a specified name, you use the getElementsByName() method of the document object:

```
let elements = document.getElementsByName(name);
```

```

```

JavaScript `getElementsByTagName()` Method

- The getElementsByTagName() method accepts a tag name and returns a live HTMLCollection of elements with the matching tag name in the order which they appear in the document.

Syntax :

```
let elements = document.getElementsByTagName(tagName);
```

```

```

JavaScript `getElementsByClassName()` Method

The `getElementsByClassName()` method returns an object containing all the elements with the specified class names in the document as objects.Each element in the returned object can be accessed by its index. This method can be called upon any individual element to search for its descendant elements with the specified class names.

```
let elements = document.getElementsByClassName(classNames);
let elements = parentElement.getElementsByClassName(classNames);
```

```

```

JavaScript `querySelector()` and `querySelectorAll()` Methods

- The querySelector() is a method of the Element interface. The querySelector() allows you to find the first element, which is a descendant of the parent element on which it is invoked, that matches a CSS selector or a group of CSS selectors.

Syntax :

```
let element = parentNode.querySelector(selector);
```

- Besides the querySelector(), you can use the querySelectorAll() method to find all elements that match a CSS selector or a group of CSS selector:

```
let elementList = parentNode.querySelectorAll(selector);
```

The following example finds the first h1 element in the document:

```
let firstHeading = document.querySelector("h1");
```

To select an element based on the value of its Id, you use the Id selector syntax:

```
let idExm = document.querySelector("#nameOfId");
```

To find the element with a given class attribute, you use the class selector syntax:

```
let classExm = document.querySelector(".nameOfClass");
```

```

```

Part 3 - JavaScript HTML DOM - Changing HTML

- By using JavaScript, you can change any part of an HTML document.

```

```

❗ Never use `document.write()` after the document is loaded. It will overwrite the document.

- The Element property `innerHTML` gets or sets the HTML or XML markup contained within the element.

Syntax:

```
document.element.innerHTML = newHTML;
```

```

```

- If you need to change the value of an HTML attribute, use this syntax:

```
document.getElementById(id).attribute = newValue;
```

```

```

Part 4 - JavaScript HTML DOM - Changing CSS

- Just as you can use JavaScript to change the HTML in a web page, you can also use it to change CSS styles. The process is very similar.

Syntax :

```
document.element.style.nameOfProp = newStyle;
```

```

```

Part 5 - Add Events Using the HTML DOM

- You can add events to HTML using Javascript.

Syntax :

```
document.element.nameOfEvents = nameOfFunction;
```

```

```

Part 6 - Event Listener

- EventListener is an object that handles an event. It can be passed to `addEventListener()` instead of passing a Function. Any JavaScript object with a handleEvent method can be used as an EventListener.

```
document.element.addEventListener(event, listener);
```

```

```

- When **passing parameter** values, use an "anonymous function" that calls the specified function with the parameters:

```
document,
  element.addEventListener("event", function () {
    nameOfFunction(p1, p2);
  });
```

```

```

Part 7 - Remove Event Listener (`removeEventListener()`)

- The JavaScript `removeEventListener()` method is used to remove an event listener from an element that has been previously attached with the `addEventListener()` method.

Syntax :

```
document.element.removeEventListener("event", listener);
```

```

```