前端網絡開發人員課程
(二) 進階網絡程式設計

4. JS DOM IV: Attributes and Styling

Presented by Krystal Institute

Learning Objective

- Understand attributes, and its relationship with properties
- Know how to manipulate element CSS with javascript and DOM

Content

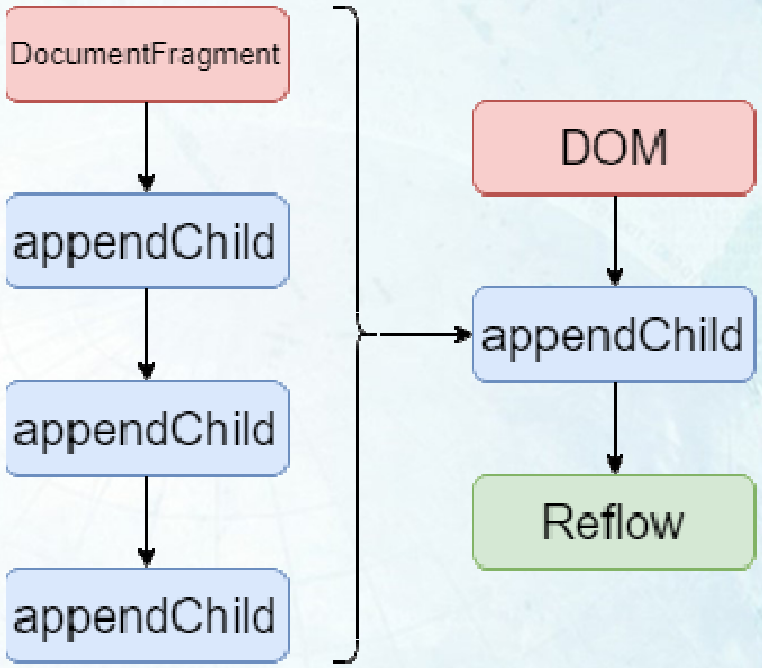
4.1 Revise on the previous lesson

Document Fragment

- Document is a lightweight version of the document separate from DOM
- Changing elements in the document fragment will not cause any performance issues

Reflow

- Document Fragment reduce the number of reflows, increasing the performance of the document, while not affecting the original DOM Tree before appending it in



Inserting Nodes

- insertBefore inserts a new Node before the specified child node of a parent node
- It can be used to create another helper function insertAfter as it is not yet supported in most web broswers

```
<body>
    <ul id="list">
        <!-- insertBefore places home here -->
        <li>Services</li>
        <li>About</li>
        <li>Contacts</li>
    </ul>
<script>
    let home = document.createElement("li");
    home.textContent = "Home";
    let list = document.querySelector("#list")
    list.insertBefore(home, list.firstElementChild)
</script>
</body>
```

Inserting Nodes

- Append inserts a set of nodes after the last child of the specified parent node
- appendChild accepts one node while append accepts one or more Nodes or DOMStrings

```
<body>
    <ul id="list">
        <li>Home</li>
    </ul>
<script>
    let namelist = ["Products", "Services", "About"];
    let list = document.querySelector("#list");
    for (let i = 0; i < namelist.length; i++) {
        let li = document.createElement("li");
        li.textContent = namelist[i];
        list.append(li);
    };
</script>
</body>
```

Inserting Nodes

- Prepend inserts a set of nodes before the first child of the specified parent node
- Every prepend puts element at the very top

```
<body>
    <p id="para"></p>
<script>
    let p = document.querySelector("#para")
    p.prepend("Mary", "James", "Albert")
</script>
</body>
```

insertAdjecentHTML

- insertAdjacentHTML inserts text adjacent to the specified element
- Beforebegin inserts before the element
- Afterbegin inserts before the first child of the element
- Beforeend inserts after the last child of the element
- Afterend inserts after the element

```
    <h2>JS Tutorial</h2>
<!--  beforebegin  -->
    <ul id="list">
<!--    afterbegin    -->
        <li>JS</li>
        <li>DOM</li>
        <li>ES6</li>
<!--    beforeend    -->
    </ul>
<!--  afterend  -->
    <h2>JS Tutorial</h2>
```

Node Manipulation

- replaceChild uses a new Node to replace the old Node
- cloneNode is a method that allows you to clone an element, returns the cloned element, and is called from the target element
- Deep argument is a Boolean that tells the function if the cloned node would keep all of its descendants
- removeChild function removes a child node from a parent node

4.2 Attributes

Attributes and Properties

- When a web browser loads an HTML page
- It generates a DOM object based on the created nodes
- Attributes of HTML elements will be converted to DOM object properties automatically

Attributes and Properties

- For instance, when a input element on the right is used

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

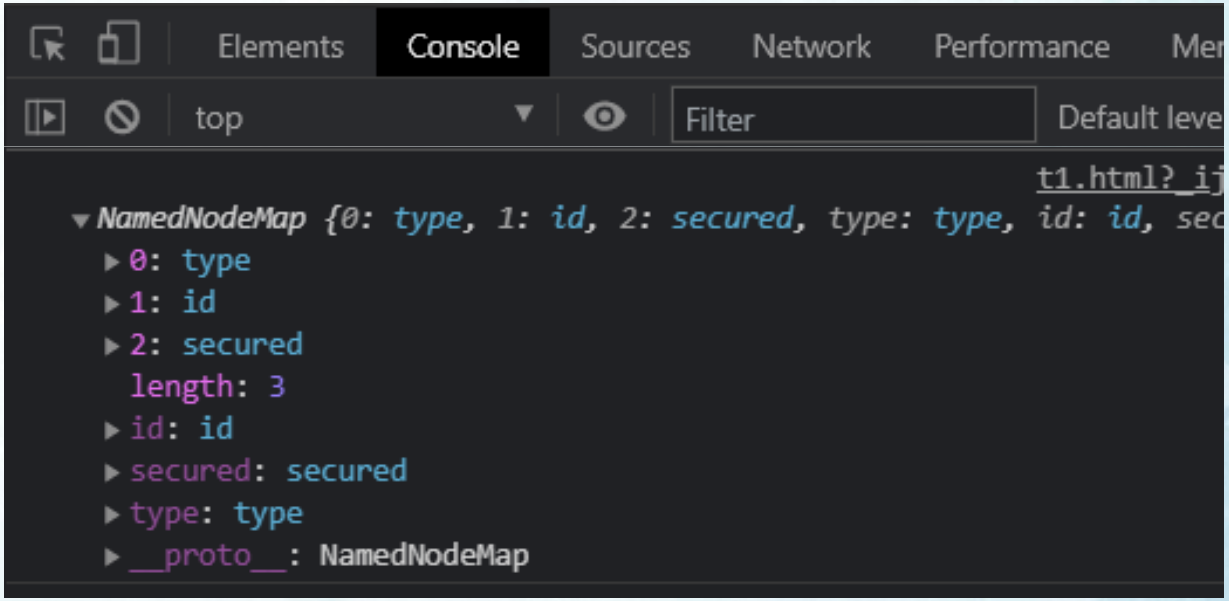
- It has 2 attributes: type and id
- The generated HTMLInputElement will have 2 properties: input.type and input.id

The web browser will convert attributes of the input element to the properties of the Dom Object

Attributes and Properties

- element.attributes returns a live collection of attributes available
- The live collection is not an array, array functions cannot be used

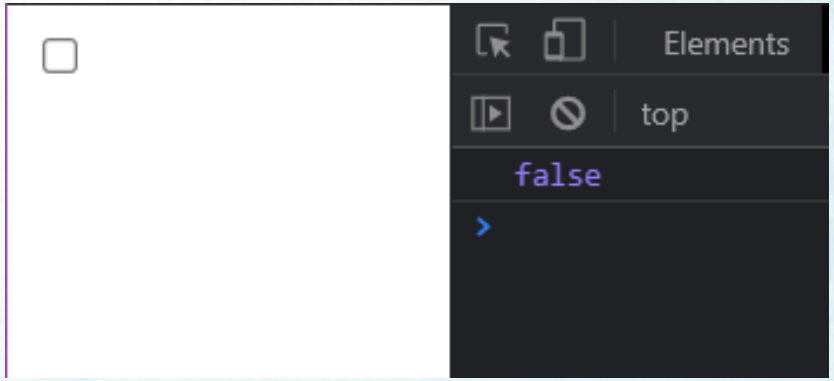
```
console.log(input.attributes);
```



Attributes and Properties

- Element attribute values is always a string
- When they are converted to DOM property, they can be typed as strings, integers, etc.

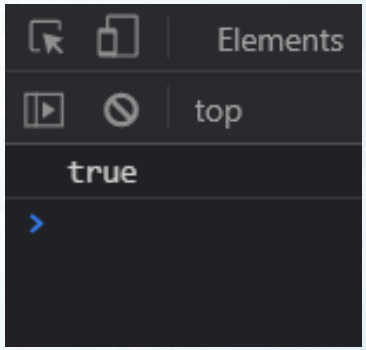
```
<body>
    <input type="checkbox" id="cb">
<script>
    let input = document.querySelector('#cb');
    console.log(input.checked);
</script>
</body>
```



Data-* attributes

- Custom attributes that starts with data- are reserved for developer use
- Using dataset property on the element returns a list of all data-* attributes

```
<body>
    <input type="checkbox" id="cb" data-secured="true">
<script>
    let input = document.querySelector('#cb');
    console.log(input.dataset.secured)
</script>
</body>
```



- Useful for more dynamic websites

Attributes and Properties Exercise

- Create a website with a <ul>, consisiting of at least 3
- The <li> elements should contain a animal name, the species name in a data attribute
- The <li> should display the animal's species on click in a element

Attributes and Properties Example

- 3 animals are shown in a list

- Goldfish
- Eagle
- Husky

- When one of the animal is clicked, a line will be displayed, showing the species of the animal

- Goldfish
- Eagle
- Husky

Husky is a dog

Attributes and Properties Solution

- Setting up the list of items and the display
- Note that onclick can be used on <li> and data-species is used

```
<body>
    <ul id="list">
        <li onclick="showspecies(this)" data-species="fish">Goldfish</li>
        <li onclick="showspecies(this)" data-species="bird">Eagle</li>
        <li onclick="showspecies(this)" data-species="dog">Husky</li>
    </ul>
    <div id="display"></div>
```

Attributes and Properties Solution

- In the function, the display and the species name is assigned to a variable
- The animal name and species is displayed inside the

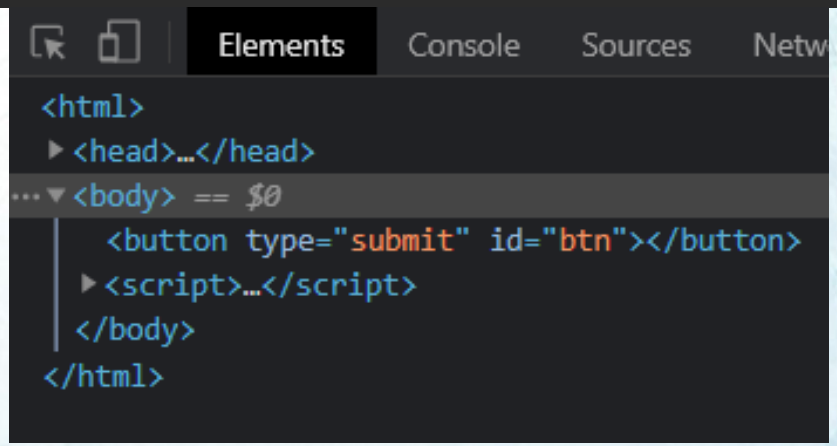
```
<script>
    function showspecies(animal) {
        let display = document.querySelector("#display");
        let species = animal.dataset.species;
        display.textContent = animal.textContent + " is a " + species;
    }
</script>
```

Attribute functions

- There are 4 attribute functions:
- setAttribute allows for changing or setting the value of a attribute
- If specified attribute doesn't exist in the element, a new one will be added instead

```
element.setAttribute(name, value);
```

```
<body>
    <button type="button" id="btn" ></button>
<script>
    let btn = document.getElementById("btn");
    btn.setAttribute("type", 'submit')
</script>
</body>
```



Attribute functions

- The name argument specifies the name of the attribute and will be automatically converted to lowercase

- The value argument specifies the value of the attribute and will be automatically converted to a string

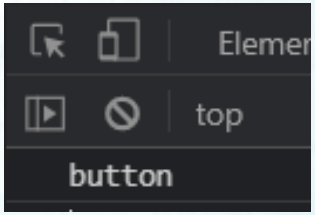
Attribute functions

- getAttribute returns the value of the attribute in the specified element

```
let value = element.getAttribute(name);
```

- If there is no specified attribute in the element, it returns null instead

```
    <button type="button" id="btn" ></button>
<script>
    let btn = document.getElementById("btn");
    console.log(btn.getAttribute("type"))
</script>
```



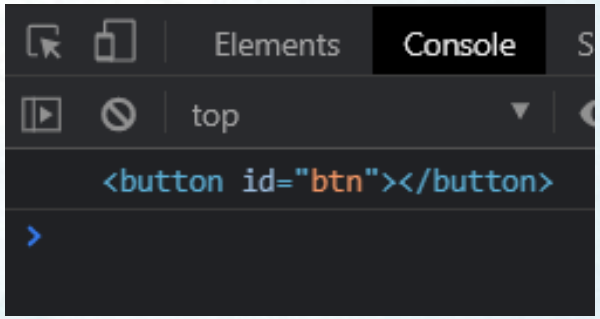
Attribute functions

- removeAttribute removes an attribute from the specified element

```
element.removeAttribute(name);
```

- Setting a Boolean with setAttribute will not work as it converts into a string
- removeAttribute should be used instead

```
    <button type="button" id="btn" ></button>
<script>
    let btn = document.getElementById("btn");
    btn.removeAttribute("type")
    console.log(btn)
</script>
```



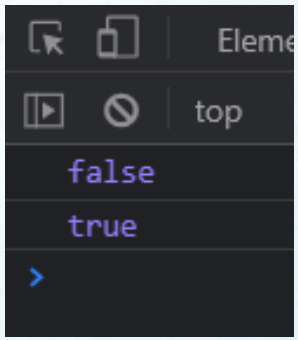
Attribute functions

- hasAttribute checks if the specified element has a specified attribute or not

```
let result = element.hasAttribute(name);
```

- It returns a Boolean value: true if the attribute exist, and false otherwise

```
    <button type="button" id="btn" ></button>
<script>
    let btn = document.getElementById("btn");
    console.log(btn.hasAttribute("name"));
    console.log(btn.hasAttribute("type"));
</script>
```



4.3 Styling

Style Property

- Recap: There are 3 methods of adding styles with CSS
- Inline styles – using the style attribute in elements
- Embedded styles - using style element in the document
- External styles – using style files externally

```
<html>
<head>
    <title>JSTutorial</title>
    <style>
        div {background-color: green}
    </style>
    <!--  This uses embedded style -->
    <link rel="stylesheet" href="css/index.css">
    <!--  This uses external style -->
</head>
<body>
    <div style="color: red">This uses inline style</div>
<script>
</script>
</body>
</html>
```

DOM Styling

- To style an element via JavaScript, use the style property of the element

```
element.style
```

- You can use the style property to change the css of the specified element

```
<body>
    <div style="color: red">This is red</div>
<script>
    let div = document.getElementsByTagName("div")[0];
    div.style.color = "yellow";
</script>
</body>
```

DOM Styling

- Multiple styles can be set at once using the cssText property or the setAttribute method

CssText
setAttribute

```
<body>
    <div style="color: red">CssText</div>
    <div style="color: yellow">setAttribute</div>
<script>
    let div1 = document.getElementsByTagName("div")[0];
    let div2 = document.getElementsByTagName("div")[1];
    div1.style.cssText = "color:black;background-color:white";
    div2.setAttribute("style", "color:white;background-color:black")
</script>
</body>
```

DOM Styling

- Using the style property only returns inline styles on that element
- It doesn't return styles from an external stylesheet or an embedded style tag

```
fontKerning: ""
fontOpticalSizing: ""
fontSize: ""
fontStretch: ""
fontStyle: ""
fontVariant: ""
fontVariantCaps: ""
fontVariantEastAsian: ""
fontVariantLigatures: ""
```

```
<head>
    <title>JSTutorial</title>
    <style>
        div {
            font-size: 100px;
        }
    </style>
</head>
<body>
    <div style="color: red">CssText</div>
<script>
    let div = document.getElementsByTagName("div")[0];
    console.log(div.style)
</script>
</body>
```

Computed Style

- Computed style is the actual style property after every CSS has been applied to the element
- That includes those styles using embedded and external styling

Computed Style

- getComputedStyle is a method of the window object, and returns the computed style of an element

```
let style = window.getComputedStyle(element, "pseudoElement");
```

- Takes in 2 arguments: the specified element, as well as it's pseudo element that is null by default

Computed Style Activity

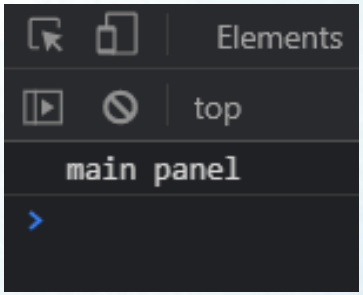
- Activity: display the value of the color property of an element from an embedded style tag
- Create a with font-size:72px and text inside
- Display it in another the value of the fontSize property

```
<head>
    <title>JSTutorial</title>
    <style>
        div {
            font-size: 72px;
        }
    </style>
</head>
<body>
    <div>This is a red text.</div>
    <div id="display"></div>
<script>
    let div = document.getElementsByTagName("div")[0];
    let display = document.querySelector("#display")
    display.textContent = getComputedStyle(div).fontSize
</script>
</body>
```

Class Name

- className is a property that returns a list of css classes of the element
- The returned value is a string, with space separating each class

```
<body>
    <img class="image main" src="#">
    <div class="main panel"></div>
    <div class="panel"></div>
<script>
    let mainpanel = document.querySelector("div")
    console.log(mainpanel.className)
</script>
</body>
```



Class Name

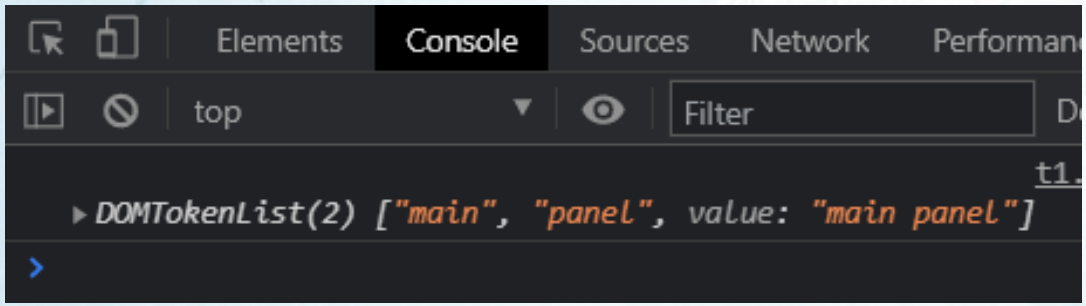
- Changing and adding Classes is similar to other element properties
- Use = to overwrite the element classes
- Use += to add classes (make sure to add a space for a new class)

```
    <img class="image main" src="#">
    <div class="main panel"></div>
    <div class="panel"></div>
<script>
    let mainpanel = document.querySelector("div");
    mainpanel.className = "main";
    let img = document.querySelector("img");
    img.className += " panel";
</script>
```

Class List

- classList is a read-only property of an element
- Returns a live collection of CSS classes

```
<body>
    <div class="main panel"></div>
    <div class="panel"></div>
<script>
    let mainpanel = document.querySelector("div");
    let classes = mainpanel.classList
    console.log(classes)
</script>
</body>
```



Class List

- As class list is a collection of classes, some methods can help with manipulating the classes
- The add() method adds one or more classes. Separated by comma

```
<body>
    <div class="main panel"></div>
    <div class="panel"></div>
<script>
    let mainpanel = document.querySelector("div");
    let classes = mainpanel.classList;
    classes.add("info", "index");
    console.log(classes);
</script>
</body>
```

Class List

- remove() can be used to remove any existing classes inside an element

```
<body>
    <div class="main panel"></div>
    <div class="panel"></div>
<script>
    let mainpanel = document.querySelector("div");
    let classes = mainpanel.classList;
    classes.remove("panel");
</script>
</body>
```

- replace(OldClass, NewClass) replaces the old existing class with a new one

```
<script>
    let mainpanel = document.querySelector("div");
    let classes = mainpanel.classList;
    classes.replace("panel", "warning");
</script>
```

Class List

- contains(Class) checks if the element contains the target class
- Returns true if the class is inside the element classList, false if not

```
<body>
    <div class="main panel"></div>
    <div class="panel"></div>
<script>
    let mainpanel = document.querySelector("div");
    let classes = mainpanel.classList;
    console.log(classes.contains("main"));
</script>
</body>
```

Class List

- toggle(Class) adds the class into the element if it doesn't exist, and removes it if it does

```
<body>
    <div class="main panel"></div>
    <div class="panel"></div>
<script>
    let mainpanel = document.querySelector("div");
    let classes = mainpanel.classList;
    classes.toggle("panel");
    console.log(classes);
</script>
</body>
```

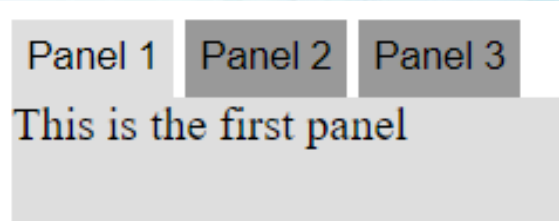
Exercise

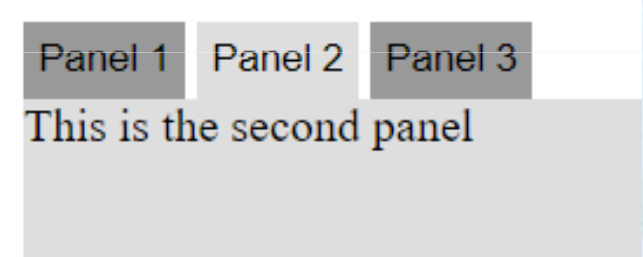
- Create a website with 3 large panels, and their respective tabs and a dark theme button
- Clicking on the tab will switch to that respective panel
- Hint: use buttons as tabs, use display: none to hide the other panels
- Finish this exercise by the end of this lesson

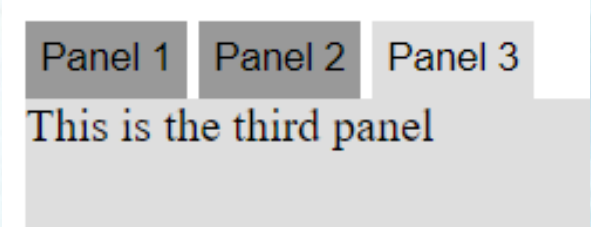
Exercise Example

- A panel with 3 tabs should be seen on load, with panel 1 having different color than the other panel tabs

- Upon clicking on the tabs, the respective panel will appear







References

- Use these if you need more explanations!
- https://www.javascripttutorial.net/es6/
- https://javascript.info/

- Use this if you need more specific answers!
- https://stackoverflow.com/