HTML Classes: Separation of Concerns

In this lesson we teach you some design patterns to use when handling numerous HTML elements and also multiple JS and CSS files.

Recall the `button <button class="btn-large js-hello">Greet</button>`. You can see two classes in there, `btn-large` and `js-hello`. The first class is used solely for styling, and the second class is used solely for referencing in JavaScript.

No-one forces you to write code this way, but it pays off to separate classes for styling and functionality. When multiple people work on the same codebase, this separation pays off. This way, a person responsible for styling can add, delete, or rename any styling classes without affecting functionality. JavaScript developers can do the same thing with the `js-` prefixed classes.

> `js-` prefixed classes should be used for functionality, while regular classes should be used for styling.

Why aren't we using HTML IDs?

You may know from your studies or previous work that we can also reference DOM nodes using their ID attributes:

```
<div id="intro">This is an introduction</div>
```

If you have the above node in the DOM, you can reference it using

```
document.getElementById('intro')
```

The problem with ID attributes is that they have to be unique for the whole document. If you violate this rule, you get an HTML error.

In big websites and applications, most developers never know the context in which their markup will appear. Chances are that if you use an ID attribute

name, someone else will do the same elsewhere.

> As two DOM nodes cannot have the same ID, using ID attributes is not advised in most cases.

Once a business owner asked me to try out his affiliate plugin, because it was not working for him. Sure enough, I filled in the form, but once I pressed register, nothing happened. I checked the developer tools of the browser, and it turned out that there were some duplicated ID attributes in the markup.

I asked him how many times he included the affiliate plugin. He said, once for desktop computers, and once for mobile. He proudly told me that he made the mobile version hidden when someone checks it in a desktop.

Unfortunately, he only managed to hide it using a CSS style (`display: none`). The markup was in his document with the exact same ID attributes as the other version.

```
<div id="affiliate-plugin" class="desktop">
    <!-- Content goes here -->
</div>
<div id="affiliate-plugin" class="mobile  hidden">
    <!-- Content goes here -->
</div>
```

The above markup is erroneous due to the duplicated ID attributes. My entrepreneur friend, lacking web development knowledge, lost hours on not knowing the fundamentals. You can now save these hours next time you encounter a similar situation.

What do we do with multiple JavaScript and CSS files?

In large projects, placing all the JavaScript code in one file is not advised for development. Therefore, we often have hundreds if not thousands of JavaScript and CSS files.

Placing all these files in the HTML markup is not advised either, because enumerating hundreds of files in the markup is neither convenient, nor efficient. You may also encounter JavaScript errors if you include the JavaScript files in the wrong order.

> For large projects, use Webpack and npm (Node Package Manager) to bundle your files into one file that you can include in your markup.

In a large project, I highly recommend that you learn and use SASS. SASS not only enriches your CSS syntax, but it helps you structure it better.