Working with data attributes

Until we got CSS variables (and now Custom Properties) there was one way of setting custom attributes in an HTML element or directly in Javascript (note the difference with CSS variables) that can be later used from Javascript to change the value of the page: **custom data attributes**.

According to the HTML standard data attributes specification

> Custom data attributes are intended to store custom data, state, annotations, and similar, private to the page or application, for which there are no more appropriate attributes or elements.
>
> These attributes are not intended for use by software that is not known to the administrators of the site that uses the attributes. For generic extensions that are to be used by multiple independent tools, either this specification should be extended to provide the feature explicitly, or a technology like microdata should be used (with a standardized vocabulary).

Data attributes are not the same as CSS custom attributes (Houdini) and CSS variables. The first is defined in HTML and can be used in either CSS or Javascript.

CSS variables and custom attributes are defined in Javascript but, primarily, used in CSS to add or modify behavior.

In this post we'll explore some uses for data attributes and how can we use them both in CSS and Javascript.

Sample Use Cases

We use data attributes in situations where a class or ID attribute would not be appropriate because it would add information that is not related to styling the content associated with it.

Each attribute that we want to add to an element means another CSS class to add. More classes make it harder to parse the attribute using Javascript.

If your class starts with a number you'll have to do more work in CSS to parse it (CSS selectors can only start with letters, dashes or underscores). According to the latest CSS 2 specification

> In CSS, identifiers (including element names, classes, and IDs in selectors) can contain only the characters [a-zA-Z0-9] and ISO 10646 characters U+0080 and higher, plus the hyphen (-) and the underscore (_); they cannot start with a digit, two hyphens, or a hyphen followed by a digit. Identifiers can also contain escaped characters and any ISO 10646 character as a numeric code

For example, let's say that we're marking up an album of our favorite artists. We want to provide the title and duration for each song... The easiest way to do it would be to mark the data as an ordered list and tell the users what the time stands for or let them figure it out.

The markup for the initial list looks like this:

```
<ol>
  <li>Force Ten    4:31</li>
  <li>Time Stand Still    5:09</li>
  <li>Open Secrets    5:38</li>
  <li>Second Nature    4:36</li>
  <li>Prime Mover    5:19</li>
  <li>Lock and Key    5:09</li>
  <li>Mission    5:16</li>
  <li>Turn the Page    4:55</li>
  <li>Tai Shan    4:17</li>
  <li>High Water    5:33</li>
</ol>
```

We can hide the song duration without removing it by adding it as a data attribute. I choose to do it with data attributes because there are no other valid attributes in HTML that CSS will accept as valid.

So the second iteration of the list looks like this:

```
<ol>
  <li data-duration="4:31">Force Ten</li>
  <li data-duration="5:09">Time Stand Still</li>
  <li data-duration="5:38">Open Secrets</li>
  <li data-duration="4:36">Second Nature</li>
  <li data-duration="5:19">Prime Mover</li>
  <li data-duration="5:09">Lock and Key</li>
  <li data-duration="5:16">Mission</li>
  <li data-duration="4:55">Turn the Page</li>
  <li data-duration="4:17">Tai Shan</li>
  <li data-duration="5:33">High Water</li>
</ol>
```

Using data attributes from CSS

Using CSS we'll add the duration of each song using the `after` pseudo attribute. This will add the string `Duration:` and the value of the `data-duration` attribute at the end of each item.

```
li:after {
  content: ' Duration: ' attr(data-duration);
  font-weight: 700;
}
```

Example using CSS is in this Codepen

Using data attributes from Javascript

Javascript is a little more complex but gives us more chances to modify the data that we work with.

We first capture all the `li` elements and, in a separate step, we convert the collection of items into an array using destructuring. We do this because *the HTML Collection of li items is not an array* and we can't run array methods in it.

Once we have the array we use .forEach to execute a function for each member of the array. In this function we change the inner HTML content of the item to a string literal containing the existing text, a string, and the value of the item's duration.

```
const items = document.getElementsByTagName('li');

const itemsArray = [...items];

itemsArray.forEach(function(item) {
  item.innerHTML = `${item.innerText}.
  Duration: ${item.dataset.duration}`;
})
```

Example available in this Codepen

Other ideas

These are fairly contrived examples using multiple items and modifying the items at the same time.

Other uses may include animation instructions for specific items, grouping of different items into categories and give them different colors or other attributes.

The possibilities are endless.

Links and Resources

- How You Can Use HTML5 Custom Data Attributes and Why
- data attributes specification