Changing HTML attributes

* Dot notation provides easy access
* img.src gives us an image location
* Read and write properties
* Add attributes that don’t exist
* Be careful of reserved words

Query Selector

* document.querySelectorAll(selector).length;
* console.log(count("p")); // All <p> elements
* console.log(count(".animal")); // Class animal
* console.log(count("p .animal")); // Animal inside of <p>
* console.log(count("p > .animal")); // Direct child of <p>

Working with restricted attributes

* Dot notation not convenient
* Some names are restricted in JavaScript
* node.getAttribute(attributeName) gets value
* node.setAttribute(attributeName, value) sets value
* node.hasAttributeName(attributeName) boolean
* node.removeAttribute(attributeName) deletes attribute

Note

* Sometimes it’s harder to access and modified attribute by using dot notation then it’s more reliable and useful to use methods like getAttribute(), setAttribute(), hasAttribute(), removeAttribute() etc.

Detecting data attributes

* Users can type anything as an attribute
* Browsers ignore them, but it’s not valid HTML
* Create your own attributes using data
* data-coolness valid attribute
* node.dataset property lets us access them

Data Attribute

Definition and Usage

* The data-\* attributes is used to store custom data private to the page or application.
* The data-\* attributes gives us the ability to embed custom data attributes on all HTML elements.
* The stored (custom) data can be used in the page's JavaScript to create a more engaging user experience (without any Ajax calls or server-side database queries).
* The data-\* attributes consist of two parts:

1. The attribute name should not contain any uppercase letters, and must be at least one character long after the prefix "data-"
2. The attribute value can be any string

* **Note:** Custom attributes prefixed with "data-" will be completely ignored by the user agent.

Controlling Classes with classList

* Class properties can have more than one value
* Dot notation and even setAttribute method is not convenient if the element has more than one class
* HTML5 adds the datalist property to nodes
* This is new object that can be add all nodes in our dom
* jQuery like class modifications for example we add or remove toggle classes.
* Lousy IE browser.

Controlling classes with classList Syntax

* node.classList.add(class) adds a class without affecting any existing classes
* node.classList.remove(class) removes a class without affecting any existing classes
* node.classList.toggle(class) turns class on/off – first time it adds the class and in 2nd click it removes the class.
* node.classList.length – how many elements currently classes has.
* node.classList.contains – class name an element contains a specific class

Targeting the attributes property

* node.attributes returns a node list
* Accessed in a variety of ways
* By numeric index
* By named index
* Using dot notation

Using text content modifiers

* node.innerHTML changes text as HTML
* innerHTML is both used for read and write
* node.outerHTML changes element’s tags which includes tags with the node – supports by newer tag.
* insertHMTL returns just inside the targeted node without including the node whereas outerHTML returns everything including html tags.
* node.insertAdjacentHTML(insertionPoint, htmlText)

Element.insertAdjacentHTML()

* 'beforebegin' - Before the element itself.
* 'afterbegin' - Just inside the element, before its first child.
* 'beforeend' - Just inside the element, after its last child.
* 'afterend' - After the element itself.

Visualization of position names

<!-- **beforebegin** -->

**<p>**

<!-- **afterbegin** -->

foo

<!-- **beforeend** -->

**</p>**

<!-- **afterend** -->

Using text content modifiers

* node.innerText just the text of a node
* node.textcontent in firefox

if(node.innerText) {

myText = node.innerText;

} else {

myText = node.textContent;

}

* innerText or textContent is also read/write property