Javascript DOM

Get Elements

`document.getElementById('id')`

- `id` : an element ID in the DOM
- Retrieves the element from the DOM

`document.getElementByClassName('rice')`

- Retrieve every element with class of `rice`
- Returns a HTML collection (not an array!)

`document.getElementByTagName('li')`

- Retrieve every element with class of `nice`
- Returns a HTML collection (not an array!)

Converting HTML collections into arrays

- We can use `for` loops with HTML collections, but what if we want to do cool array stuff?

`Array.from(collection)`

- Converts an HTML collection into an array

ex.

```
Array.from(collection).forEach(item => {
  console.log(item);
});
```

Query Selectors

`document.querySelector('#id')`

- Selects the element with id `id`
- Can get quite specific (using CSS selectors)

```
document.querySelector('#book-list li:nth-child(2) .name');
```

- Grabs element of id `booklist` which is the 2nd `li` element of class `name`
- Only returns first element

`document.querySelectorAll()`

- Grabs every element according to the selector
- Returns a Node List (not an HTML collection, not an array!)
- We do not have to turn it into an array

Manipulating HTML

`elem.textContent`

- The text content inside the HTML element `elem`

`elem.innerHTML`

- Change the HTML

ex.

```
elem.innerHTML += '<h1>epic epic epic!!!</h1>'
```

Nodes

- Everything in the DOM is a node

Node Types

`elem.nodeType`

- A number representing the node type

`elem.nodeName`

- The node name

`elem.hasChildNodes()`

- Returns true/false

`elem.cloneNode(true)`

- Clones the element
- Argument `true` clones all the child node rather than just the top one

Forms

`document.forms`

- Returns an HTML collection of forms

```
document.forms[0]
document.forms['id']
```

- When clicking a button in a form, the form emits a `submit` event that also refreshes the page

```
const value = form.querySelector('input[type="text"]').value;
```

- Grab input from form

Traversing the DOM

`elem.parentNode`

- Grabs reference to the parent node
- Basically the same as below

`elem.parentElement`

- Grabs reference to the parent element
- Basically the same as above

You can chain these together to traverse upwards `.parentElement.parentElement.parentElement`

`elem.childNodes`

- A node list of child nodes
- Includes `text` elements which are line breaks

`elem.children`

- Only element children, no filler `text` nodes

`elem.nextSibling` / **`elem.previousSibling`**

- The next/previous sibling node (includes `text` line break)

`elem.nextElementSibling` / **`elem.previousElementSibling`**

- The next/previous element (no `text` line breaks)

ex.

```
elem.previousElementSibling.querySelector('p').innerHTML += '<h1>epic!!!!!</h1>'
```

- Select `<p>` only from previous element and add `epic!!!!!` as an `<h1>` to the HTML

Events

- Attach event listeners on elements to listen for events

```
elem.addEventListener('click', e => {
  console.log(e.target); // target of event
  console.log(e); // event itself
})
```

- `e.target` is the element

Prevent default behaviour

- ex. prevent default behaviour of `<a>` link

```
elem.addEventListener('click', e => e.preventDefault());
```

Bubbling

- Default: The event bubbles up from the target element
- The event bubbles up to the parent element, and if there is an event listener on the parent element the callback function would also fire as well
- Bubbling keeps happening until we hit the top

ex. Attaching event listeners to every button is expensive, we can use bubbling to our advantage

```
list.addEventListener('click', e => {
  if(e.target.className == 'delete'){
    //grab the ul
    const li = e.target.parentElement;
    list.removeChild(li);
  }
});
```

- ex. list is a `ul` element and we want to remove one of its `li` elements that contain `btn` with class `delete` as children

Add Elements

`document.createElement('div')`

- Creates a div element
- Floating around, not attached to anything

`elem.appendChild(node)`

- Appends `node` to the DOM

Styles and Classes

`elem.style.[css]`

- Set element CSS

`elem.className`

- Element class

`elem.classList`

- A list of the element's classes

`elem.classList.add` / `elem.classList.remove`

- Add/remove a class to `elem`
- Preferred over `elem.className += '...'`

Attributes

`elem.getAttribute('')`

- Gets element attribute

```
elem.getAttribute('class');
elem.getAttribute('src');
```

`elem.setAttribute('', '')`

- Sets element attribute
- First parameter is attribute, second parameter is what to set to

```
elem.getAttribute('class', 'cool');
elem.getAttribute('src', 'photo.png');
```

`elem.hasAttribute('')`

- Returns true or false

`elem.removeAttribute('')`

- Removes element attribute

`DOMContentLoaded`

```
document.addEventListener('DOMContentLoaded', () => {});
```

- Attaches everything in {} once the DOM is ready

Danny Wu