General Assembly

STUDY GUIDE

THE DOM

SEI Fundamentals Unit 11

Here are some notes on what's been covered in this chapter. Feel free to copy this and extend it to make your own cheat sheet.

The DOM

The browser pulls in HTML documents, parses them, and creates object models of the pages in its memory. This model is the **Document Object Model (DOM)**.

DOM Node

Each element in the HTML document is represented by a**DOM node**. These nodes can be accessed and changed using JavaScript.

When the model is updated, those changes are reflected on screen.

DOM Node

Accessing Elements

Before we can update a page, we need to find, or**select**, the element(s) that we want to update. In order to find an element, we need to search through the document. The syntax for the search looks something like this:

```
document.getElementById('main')
```

» Here are the methods that can be used to select an element or elements:

Method	Description
getElementById()	Selects an individual element within a document using a specific id
querySelector()	Uses CSS selector to select the first matching element within a document
getElementsByClassName()	Allows you to select all elements with a given class attribute
getElementsByTagName()	Locates all elements that match a given tag name
querySelectorAll()	Uses CSS selector to select one or more elements

Cache

If we'd like to work with that element multiple times, a variable should be used to store, o**rcache**, the results of our query.

```
const sidebar = document.getElementById('sidebar');
```

Traversing the DOM

The process of selecting another element based on its relationship to a previously selected element.

Property	Description
parentNode	Locates the parent element of an initial selection
previousSibling	Finds the previous sibling of a selected element
nextSibling	Finds the next sibling of a selected element
firstChild	Finds the first child of a selected element

» The syntax for using these properties looks like this:**javascript document.querySelector('li').parentNode**

NodeList

A **NodeList** is a list of node objects numbered similarly to arrays.

To locate the fourth item in this nodeList:

```
document.getElementsByTagName('li')[3];
```

Accessing and Updating Content

The **innerHTML** and **textContent** properties can be used to access or update content:

Property	Description
innerHTML	Get or set the HTML content of an element.
textContent	Get or set the text content of an element.

The syntax for getting content looks like this:

```
const firstListItem = document.querySelector('li').innerHTML;
// Remember, `querySelector()` selects the first element that matches the provided selector.
```

The syntax for updating content looks like this:

```
document.querySelector('li').innerHTML = 'Email <a href="mom@gmail.com">Mom</a>.';
```

Adding Content

To add new elements to the page, we'll need to use a three step process:

1. We will use the **createElement()** method to create a new element, which can then be added to the page. When this node is created, it will be *empty*. This element will be stored in a variable.
2. Next we will add content to the element using the **innerHTML** or **textContent** properties.
3. Now that our element has been created, we can add it as a child of an element using the **appendChild()** method. This will add an element as the last child of the parent element.

To add a sixth item to our list we can execute the following code:

```
// First up, let's create a new list item and store it in a variable.
const newListItem = document.createElement('li');

// Now let's update the text content of that list item.
newListItem.textContent = 'Jalapenos';

// And finally, let's add that list item as a child of the ul.
document.querySelector('ul').appendChild(newListItem);
```

Getting and Setting Attributes

Property	Description
className	Change the value of the class attribute for an element

```
document.getElementById('important').className = 'highlight';
```

Method	Description
setAttribute()	Sets an attribute of an element
removeAttribute()	Removes an attribute from an element

```
document.getElementsByTagName('a')[0].setAttribute('href', 'http://newurl.com');
document.getElementsByTagName('a')[0].removeAttribute('id');
```

Events

Actions taken by a user that can trigger updates in the DOM.

For example, when a user clicks on a website's menu icon, a sidebar menu should slide out from the side of the page. Or, if the user has typed an incorrect format into a form field, the field should become outlined in red.

Event Handler

We can set up **event handlers** in our scripts that will **listen**, or wait, for an event to occur and then trigger a function.

The syntax for setting up an event handler looks like this:

element.addEventListener('nameOfEvent', functionToRun);

Types of Events

There are many events that can trigger a function. Here are a few:

Event	Description
'click'	When a button (usually a mouse button) is pressed and released on a single element.
'keydown'	When the user first presses a key on the keyboard.
'keyup'	When the user releases a key on the keyboard.
'focus'	When an element has received focus.
'blur'	When an element loses focus.
'submit'	When the user submits a form.
'load'	When the page has finished loading.
'resize'	When the browser window has been resized.
'scroll'	When the user scrolls up or down on the page.

This

this

A term used in event handling functions to refer to the specific object with which the user interacted.

Accessing the DOM

1. Use the **querySelectorAll()** method to select all elements with the **current** class and then, using array syntax (our trusty square brackets), update the selection to only select the *second* element with the **current** class. After you've made your selection, add the following code onto the end of the selection: **.textContent = "The Violent Bear It Away"; javascript document.querySelectorAll('.current')[1].textContent = "The Violent Bear It Away";**
2. Use the **getElementById()** method to find the element with the ID **next**. After you've made your selection, add the following code onto the end of the selection: **.textContent = "Me Talk Pretty One Day"; javascript document.getElementById('next').textContent = "Me Talk Pretty One Day";**
3. Find the first **li** using the **querySelector()** method. After you've made your selection, add the following code onto the end of the selection: **.textContent = "Brothers Karamazov"; javascript document.querySelector('li').textContent = "Brothers Karamazov";**
4. Use the **getElementsByTagName()** method to select all **li** elements. Then, use array syntax to select the fourth **li**. After you've made your selection, add the following code onto the end of the selection: **.textContent = "JavaScript is Fun!"; javascript document. getElementsByTagName('li')[3].textContent = "JavaScript is Fun!";**

Manipulating the DOM

1. Create a **for** loop. For the **for** loop, **i** should have an initial value of **1** and the loop should run three times. Each time the loop runs, create a **div** element using the **createElement()** method and store it in a variable **boxElement**. **javascript for (let i = 1; i <= 3; i++) { const boxElement = document.createElement('div'); }**
2. Alright! Next let's add some styles for each **div**. In the CSS tab we've defined some styles for you to use. Each time the for loop runs, we want to add a class to the current **boxElement** using the **className** property. This will apply the styles in our CSS that are associated with that class.
   - The class name for the first box should be **box-1**.
   - The class name for the second box should be **box-2**.
   - The class name for the third box should be **box-3**. **javascript for (let i = 1; i <= 3; i++) { const boxElement = document.createElement('div'); boxElement.className = 'box-' + i; }**
3. Also within the **for** loop, use the **appendChild()** method to append each **boxElement** to the body. **javascript for (let i = 1; i <= 3; i++) { const boxElement = document.createElement('div'); boxElement.className = 'box-' + i; document.getElementsByTagName('body')[0].appendChild(boxElement); }**

Events

1. First create a function called **addItem**. For now it should be empty. ```javascript function addItem() {

};

2. Add a `click` event handler to the button. When the button is clicked, run the `addItem` function.
javascript
function addItem() {

};

document.getElementsByTagName('button')[0].addEventListener('click', addItem);

3. Alright, now within the `addItem` function we need to find out what the user entered into the `input` field. We'll be using a new property - the `value` property - to find out what the user entered into the `input`.
- Get the user's todo item from the `input` element and store it in a variable `newItemText`.
javascript
function addItem() {
const input = document.getElementsByTagName('input')[0];
const newItemText = input.value;
};

document.getElementsByTagName('button')[0].addEventListener('click', addItem);

4. Now we want to add the new list item to the `ul` with the id `todo-list`. Remember, in order to add a new element to the page, there are a few steps we will have to take:
- Use the `createElement()` method to create a list item element and store it in the variable `newItem`.
- Add content to this new list item using the `innerHTML` property.
- Use the `appendChild()` method to append this element to the element with the id `todo-list`.
javascript
function addItem() {
const input = document.getElementsByTagName('input')[0];
const newItemText = input.value;

const newItem = document.createElement('li');
newItem.innerHTML = newItemText;
document.getElementById('todo-list').appendChild(newItem);
};

document.getElementsByTagName('button')[0].addEventListener('click', addItem);
```
```