Hello, World DOMination: Console.log, Element.innerHTML, and the Date Object

In this section, we'll learn about how to use `console.log`to print element values. We'll also use `Element.innerHTML`to fill in the HTML of a DOM element. Finally, we'll learn about the Javascript Date object and how to use it to construct a clock that keeps the current time.

Console Logging Element Values

Along with the other developer tools, the console is a valuable tool Javascript developers use to debug and check that scripts are running correctly. In this exercise, we'll practice logging to the console.

Create an HTML file that contains the following:

HTML

```html
<!DOCTYPE html>
<html>
  <head> </head>
  <body>
    <ul id="your-best-friend">
      <li>Has your back</li>
      <li>Gives you support</li>
      <li>Actively listens to you</li>
      <li>Lends a helping hand</li>
      <li>Cheers you up when you're down</li>
      <li>Celebrates important moments with you</li>
    </ul>
  </body>
</html>
```

In the above code, we see an id with which we can reference the `ul`element. Recall that we previously used `document.querySelectorAll()`to store multiple elements with the same class name in a single variable, as a NodeList. However, in the above example, we see only one id for the parent element. We can reference the parent element via its id to get access to the content of its children.

Javascript

```javascript
window.addEventListener("DOMContentLoaded", event => {
  const parent = document.getElementById("your-best-friend");
  const childNodes = parent.childNodes;
  for (let value of childNodes.values()) {
    console.log(value);
  }
});
```

In your browser, use the developer tools to open the console and see that the values of each `li`have been printed out.

Using Element.innerHTML

Thus far, we have referenced DOM elements via their id or class name and appended new elements to existing DOM elements. Additionally, we can use the inner HTML property to get or set the HTML or XML markup contained within an element.

In an HTML file, create a `ul`element with the id "your-worst-enemy" that has no children. We'll add some JavaScript to construct a string that contains six `li`tags each containing a random number and set the inner HTML property of`ul#your-worst-enemy`to that string.

HTML

```html
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript" src="example.js"></script>
  </head>
  <body>
    <ul id="your-worst-enemy"></ul>
  </body>
</html>
```

Javascript

```javascript
// generate a random number for each list item
const getRandomInt = max => {
  return Math.floor(Math.random() * Math.floor(max));
};

// listen for DOM ready event
window.addEventListener("DOMContentLoaded", event => {
  // push 6 LI elements into an array and join
  const liArr = [];
  for (let i = 0; i < 6; i++) {
    liArr.push("<li>" + getRandomInt(10) + "</li>");
  }
  const liString = liArr.join(" ");

  // insert string into the DOM using innerHTML
  const listElement = document.getElementById("your-worst-enemy");
  listElement.innerHTML = liString;
});
```

Save your changes, and refresh your browser page. You should see six new list items on the page, each containing a random number.

Inserting a Date Object into the DOM

We've learned a lot about accessing and manipulating the DOM! Let's use what we've learned so far to add extra functionality involving the Javascript Date object.

Our objective is to update the title of the document to the current time at a reasonable interval such that it looks like a real clock.

We know we'll be starting with an HTML document that contains an empty title element. We've learned a couple of different ways to fill the content of an element so far. We could create a new element and append it to the title element, or we could use `innerHTML`to set the HTML of the title element. Since we don't need to create a new element nor do we care whether it appears last, we can use the latter method.

Let's give our title an id for easy reference.

HTML

```html
<title id="title"></title>
```

In our Javascript file, we'll use the Date constructor to instantiate a newDate object.

```javascript
const date = new Date();
```

Javascript

```javascript
window.addEventListener("DOMContentLoaded", event => {
  const title = document.getElementById("title");
  const time = () => {
    const date = new Date();
    const seconds = date.getSeconds();
    const minutes = date.getMinutes();
    const hours = date.getHours();

    title.innerHTML = hours + ":" + minutes + ":" + seconds;
  };
  setInterval(time, 1000);
});
```

Save your changes and refresh your browser. Observe the clock we inserted dynamically keeping the current time in your document title!

What We Learned:

- What the DOM is and how we can access it
- How to access DOM elements using `document.getElementById()` and `document.querySelectorAll()`
- How to create new elements with `document.createElement()` and `document.createTextNode`, and append them to existing DOM elements with `Element.appendChild()`
- The difference between `window.onload` and `DOMContentLoaded`
- How to access children nodes with `NodeList.childNodes`
- Updating DOM elements with `Element.innerHTML`
- The Javascript Date object