CS 240
Computer Systems

#25: Towards Frontends and Finale!

Nov. 30, 2021 · Wade Fagen-Ulmschneider

Frontend Technology

In app development, the frontend component of an application is often the only element that the user will see. Even the very best system will fail if the interface to that system is poor.

In general, a frontend developer either works directly with the technology they're developing or works with a library that provides an abstraction over a native frontend components:

- [Application Native Frontends]:

- [Frontend Abstraction Libraries]:

Additionally, there are very few native front end technologies:

- [HTML]:

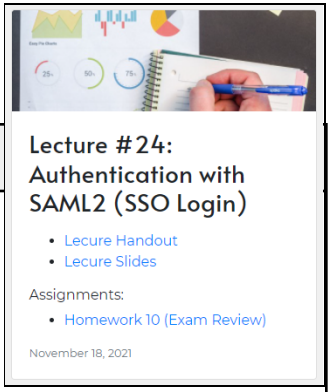
  + CSS:

- [Native Desktop]:

- [Native Phone]:

HTML Development Overview

HTML is a tree-based structured markup language that provides a structure for the layout of content:



CS 240 Website HTML

```
 1  <body>
...  [...]
 3  <main role="main" class="container">
 4  <div class="row">
...  [...]
 6  <div class="col-md-4" id="lec24">
 7    <div class="card mb-4 box-shadow">
 8      <img class="card-img-top"
    src="[...]lecture1.jpg" alt="Authentication with SAML2 (SSO
    Login)">
 9        <div class="card-body">
10          <h3>Lecture #24: Authentication with SAML2 (SSO
    Login)</h3>
11          <ul>
12            <li><a href="[...]cs240fa21_24-handout.pdf">Lecture
    Handout</a></li>
13            <li><a href="[...]cs240fa21_24-slides.pdf">Lecture
    Slides</a></li>
14          </ul>
15          <h6>Assignments:</h6>
16          <ul>
17            <li><a href="/cs240/fa2021/homeworks/hw10/">Homework
    10 (Exam Review)</a></li>
18          </ul>
19        </div>
20        <div class="d-flex justify-content-between
    align-items-center">
21          <small class="text-muted">November 18, 2021</small>
22        </div>
23      </div>
24    </div>
25  </div>
...  [...]
27  </div>
28  </main>
...  [...]
30  </body>
```

Internally, this HTML is maintained inside of a data structure in your web browser called the _____.

...every time we change the website, we update this data structure!

Interacting with the DOM Directly

The most bare-bones way of creating dynamic content on web pages is to modify the DOM directly. Usually, this is done via two steps:

1. Get a reference to the element you want to manipulate using a query selector: **`document.querySelector(selectors)`**

   The selectors parameter is:
   - The name of an HTML tag (**`"main"`**).
   - The name of an **id** associated with a tag, where the class name is modified with a **`#`** (**`"#lec24"`**).
   - The name of a **class** associated with a tag, where the class name is modified with a period (**`".card-body"`**).
   - Other, even more specific queries,
   - ...or any combination of any of the above!

2. Manipulate the element using the methods provided by the browser (specified by the "DOM Standard").

   Common options:
   - Change the HTML: **`e.innerHTML = "new HTML"`**
   - Add a CSS class: **`e.classList.add("alert")`**
   - ...or any number of other changes.

Using a Library to Change the DOM

Instead of interacting with the DOM directly, there are many libraries that provide a light-weight wrapper around the DOM. These libraries were particularly helpful in the earlier days of the Internet when all browsers did not have the same DOM API.

- The most popular library is **jQuery**: this library exports a single variable to the global score: **`$`** (yup, $ is a valid JS variable name).
  - The $ acts as a function that will take a query selector as the argument: **`$("#lec24")`**.
  - The $ also acts as s function to call user-supplied functions when the page is finished loading:
    `$( function() { print("Done!") } )`
  - There are hundreds of utility functions included with jQuery, many of which have been moved directly into the DOM specification.

Using a Framework to Manipulate the DOM

As your application grows, you may want to use an object-based design to manipulate the presentation of your page. Frameworks provide a level of abstraction away from directly accessing the DOM and creates frameworks that internally manipulate the DOM.

React Example: "A Stateful Component"

```
 1 class Timer extends React.Component {
 2   constructor(props) {
 3     super(props);
 4     this.state = { seconds: 0 };
 5   }
 6
 7   tick() {
 8     this.setState(state => ({
 9       seconds: state.seconds + 1
10     }));
11   }
12
13   componentDidMount() {
14     this.interval = setInterval(() => this.tick(), 1000);
15   }
16
17   componentWillUnmount() {
18     clearInterval(this.interval);
19   }
20
21   render() {
22     return (
23       <div>Seconds: {this.state.seconds}</div>
24     );
25   }
26 }
27
28 ReactDOM.render(
29   <Timer />,
30   document.getElementById('timer-example')
31 );
```

Fetching Data from a Middleware

Fetching Data using the JavaScript fetch API

```
1 fetch("http://cs240-adm.cs.illinois.edu:8000/time", {
2   method: "PUT",
3   body: JSON.stringify(data),
4   ...many other options...
5 }).then( response => console.log(response) )
```