Testing - Spikes

Advanced CSS

How can we use advanced CSS features to create complex custom components?

Questions to consider

1. What are “combinator” selectors? Can you provide examples where they’re useful?
    - “Combinator” selectors select “relatives” of an element.
    - E.g. The " " (space) combinator selects nodes that are descendants of the first element.
    - div span { } will match all elements that are inside a element.

Other combinator selectors:

- Child combinator > selects nodes that are direct children of the first element.
- General sibling combinator ~ The second element follows the first (though not necessarily immediately), and both share the same parent.
- Adjacent sibling combinator + Matches the second element only if it immediately follows the first element.
- Column combinator || selects nodes which belong to a column.

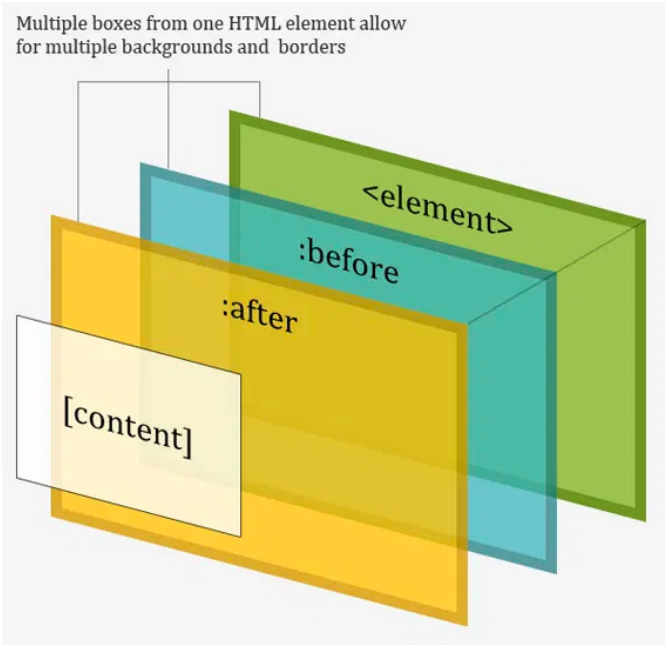
2. What are pseudo-elements? Can you provide examples where they’re useful?

    Pseudo-elements are keywords added to a selector that lets you style a specific part of the selected element(s). This might be useful if you were looking to style the font of the first line of a paragraph.

```
/* The first line of every <p> element. */
p::first-line {
  color: blue;
  text-transform: uppercase;
}
```

(mdn web docs)

Pseudo-elements can be used to create multiple background canvases when absolutely positioned relative to their parent element.



(css-tricks.com)

3. How might you create custom-styled checkboxes using both of the above?

   In the same way that using the : pseudo with a:visited would match all the <a> elements that have been visited by the user, you might use the : pseudo with the <*checkbox*> element.

   E.g.,
   input[type=radio] : visited {
   colour: blue;
   }

Advanced DOM

How can we use advanced DOM features to make rendering complex UIs easier.

Questions to consider

1. What is a NodeList?

   In JavaScript, a NodeList is an object that represents a collection of DOM elements, usually returned by methods such as querySelectorAll() or getElementsByTagName().

- How is it different from an array?
    i. An array is a collection of elements of the same type that is stored in a single block of memory (node lists use dynamic memory allocation which means that they can be added or removed without the need for contiguous memory.)
- What's the difference between "live" and "static" NodeLists?
    i. A live NodeList is a dynamic list of elements that updates automatically when the DOM is modified. For example, adding or removing elements will be automatically reflected in the live NodeList.
    ii. A static NodeList does not update automatically according to the DOM and instead shows a snapshot of a list of elements present in the DOM at the time the NodeList was created.

2. What is the <template> element?

   The <template> element holds HTML and stores it for subsequent use when it is later rendered using JavaScript (as opposed to when the page is initially loaded)

    - How can we use this to render dynamic UI?
        i. xyz

Checking our code

What are all the different ways to make sure our code is correct?

Questions to consider

1. What is Prettier? How might it help us write better code?
    a. Prettier is a vscode formatting extension that applies uniform spacing and quotation marks to the page.
2. What is static analysis? How can a linter help us avoid bugs?
3. What are the pros and cons of unit, integration and end-to-end tests?

Unit tests

Pros:

    i. They are fast and easy to run.

ii. They test individual units of code in isolation, making it easier to locate and fix bugs.
iii. They help ensure that changes to one unit of code do not break other units.

Cons:

iv. They only test individual units of code in isolation and cannot test the interactions between different units of code.
v. They do not test the functionality of the system as a whole.
vi. Integration tests:

Integration tests

Pros:

- They test the interactions between different units of code.
- They help ensure that changes to one unit of code do not break the interactions with other units.
- They can catch errors that only occur when multiple units of code are working together.

Cons:

- They are slower and more complex than unit tests.
- They can be difficult to set up and maintain.

End-to-end tests

Pros:

- They test the functionality of the system as a whole, including interactions between different components.
- They help ensure that the system is working as expected from the user's perspective.
- They can catch errors that only occur when all the components are working together.

Cons:

- They are slower and more complex than both unit tests and integration tests.
- They can be more difficult to set up and maintain, especially if the system being tested is complex.

Testing methodologies

How do different testing methodologies try to improve the way we write tests?

Questions to consider

1. What is Test-Driven Development (TDD)? Can it help us write better code?
    a. TDD is a method of programming where tests are written *before* the code itself.
    b. Some developers claim it to be life changing, while others see it as unnecessarily time consuming.
2. What is Behavior-Driven Development (BDD)? How do we translate user requirements into automated tests?
    a. BDD is a branch of TDD that uses user stories as the basis for software tests.

E.g. A test for a transfer in a cryptocurrency wallet might look like this:

```
Story: Transfers change balances

As a wallet user
In order to send money
I want wallet balances to update

Given that I have $40 in my balance
And my friend has $10 is their balance
When I transfer $20 to my friend
Then I should have $20 in my balance
And my friend should have $30 in their balance.
```

(medium.com)

3. What is test coverage? Can this tell us about the *quality* of our tests?
    a. Test coverage is used to find untested code. High coverage is possible with low quality testing and it is therefore not to be used as a measure of quality