* **If you were to describe semantic HTML to the next cohort of students, what would you say?**

Semantic HTML is markup that conveys meaning, not appearance, to search engines to make everything easier to identify.

* **Name two big differences between display: block; and display: inline;.**

block starts on a new line and takes up the full width of the content.  
inline starts on the same line as previous content, in line with other content, and takes up only the space needed for the content.

* **What are the 4 areas of the box model?**

content, padding, border, margin

* **While using flexbox, what axis does the following property work on: align-items: center?**

cross-axis

* **Explain why git is valuable to a team of developers.**

Allows you to dole out tiny pieces of a large project to many developers who can all work towards the same goal on their own chunks, allows roll back if you make a mistake; version control.

* **What is the difference between an adaptive website and a fully responsive website?**

An adaptive website "adapts" to fit a pre-determined set of screen sizes/devices, and a responsive one changes to fit all devices.

* **Describe what it means to be mobile first vs desktop first.**

It means you develop/code the site with mobile in mind first and work your way outward in screen size.

* **What does font-size: 62.5% in the html tag do for us when using rem units?**

This setting makes it so that 1 rem = 10 px for font size, easier to calculate.

* **How would you describe preprocessing to someone new to CSS?**

Preprocessing is basically a bunch of functions and variables you can use to store CSS settings in different ways that make it easier to code CSS.

* **What is your favorite concept in preprocessing? What is the concept that gives you the most trouble?**

Favorite is (parametric) mixins; but I don't have a lot of trouble with preprocessing. What gives me the most trouble is knowing ahead of time what would be good to go in a mixin for a given site.

* **Describe the biggest difference between .forEach & .map.**

forEach iterates over an array item by item, and map calls a function on each array item, but returns another/additional array, unlike forEach.

* **What is the difference between a function and a method?**

Every function is an object. If a value is a function, it is a method. Methods have to be 'received' by something; functions do not.

* **What is closure?**

It is code identified elsewhere that we can use later; gives the ability to put functions together. If a variable isn't defined, a function looks outward for context.

* **Describe the four rules of the 'this' keyword.**
  1. Window/global binding -- this is the window/console object. 'use strict'; to prevent window binding.
  2. Implicit binding -- when a function is called by a dot, the object preceding the dot is the 'this'. 80 percent of 'this' is from this type of binding.
  3. New binding -- points to new object created & returned by constructor function
  4. Explicit binding -- whenever call, bind, or apply are used.
* **Why do we need super() in an extended class?**

Super ties the parent to the child.

* **What is the DOM?**

Document object model, the 'window' or container that holds all the page's elements

* **What is an event?**

An event is something happening on or to the page, like a mouse click, doubleclick, key up/down, pointer out of element/over element, things like this. There are tons of "events" that javascript can detect.

* **What is an event listener?**

Javascript command that 'listens' for an event to happen on the page to a given element and then runs a function when that event happens

* **Why would we convert a NodeList into an Array?**

A NodeList isn't a real array, so it won't have access to array methods such as slice or map.

* **What is a component?**

Reusable pieces of code to display info in a consistent repeatable way

* **What is React JS and what problems does it try and solve? Support your answer with concepts introduced in class and from your personal research on the web.**

ReactJS is a library used to build large applications. It's very good at assisting developers in manipulating the DOM element to create rich user experiences. We need a way to off-load the state/data that our apps use, and React helps us do that.

* **What does it mean to *think* in react?**

It makes you think about how to organize/build apps a little differently because it's very scalable and allows you to build huge apps. React's one-way data flow makes everything modular and fast. You can build apps top-down or bottom-up.

* **Describe state.**

Some data we want to display.

* **Describe props.**

Props are like function arguments in JS and attributes in HTML.

* **What are side effects, and how do you sync effects in a React component to state or prop changes?**

Side effects are anything that affects something outside the executed function's scope like fetching data from an API, a timer, or logging.

* **Explain benefit(s) using client-side routing?**

Answer: It's much more efficient than the traditional way, because a lot of data isn't being transmitted unnecessarily.

* **Why would you use class component over function components (removing hooks from the question)?**

Because some companies still use class components and don't want to switch their millions of dollars' worth of code over to all functional hooks, and also there's currently a lot more troubleshooting content out there for classes that isn't out there for hooks. Also, functional components are better when you don't need state, presentational components.

* **Name three lifecycle methods and their purposes.**

componentDidMount = do the stuff inside this 'function' after the component mounted

componentDidUpdate = do the stuff inside this function after the component updated

componentWillUnmount = clean-up in death/unmounting phase

* **What is the purpose of a custom hook?**

allow you to apply non-visual behavior and stateful logic throughout your components by reusing the same hook over and over again

* **Why is it important to test our apps?**

Gets bugs fixed faster, reduces regression risk, makes you consider/work out the edge cases, acts as documentation, acts as safety net when refactoring, makes code more trustworthy

* **What problem does the context API help solve?**

You can store data in a context object instead of prop drilling.

* **In your own words, describe actions, reducers and the store and their role in Redux. What does each piece do? Why is the store known as a 'single source of truth' in a redux application?**

Everything that changes within your app is represented by a single JS object called the store. The store contains state for our application. When changes are made to our application state, we never write to our store object but rather clone the state object, modify the clone, and replace the original state with the new copy, never mutating the original object. Reducers are the only place we can update our state. Actions tell our reducers “how” to update the state, and perhaps with what data it should be updated, but only a reducer can actually update the state.

* **What is the difference between Application state and Component state? When would be a good time to use one over the other?**

App state is global, component state is local. Use component state when you have component-specific variables.

* **Describe redux-thunk, what does it allow us to do? How does it change our action-creators?**

Redux Thunk is middleware that provides the ability to handle asynchronous operations inside our Action Creators, because reducers are normally synchronous.

* **What is your favorite state management system you've learned and this sprint? Please explain why!**

Redux, because I felt it was easier to understand than the context API.

* **Explain what a token is used for.**

Many services out in the wild require the client (our React app, for example) to provide proof that it’s authenticated with them. The server running these services can issue a JWT (JSON Web Token) as the authentication token, in exchange for correct login credentials.

* **What steps can you take in your web apps to keep your data secure?**

As we build our web apps, we will most likely have some “protected” routes - routes that we only want to render if the user has logged in and been authenticated by our server. The way this normally works is we make a login request, sending the server the user’s username and password. The server will check those credentials against what is in the database, and if it can authenticate the user, it will return a token. Once we have this token, we can add two layers of protection to our app. One with protected routes, the other by sending an authentication header with our API calls (as we learned in the above objective).

* **Describe how web servers work.**

The “world wide web” (which we’ll refer to as “the web”) is just a part of the internet - which is itself a network of interconnected computers. The web is just one way to share data over the internet. It consists of a body of information stored on web servers, ready to be shared across the world. The term “web server” can mean two things:

* a computer that stores the code for a website
* a program that runs on such a computer

The physical computer device that we call a web server is connected to the internet, and stores the code for different websites to be shared across the world at all times. When we load the code for our websites, or web apps, on a server like this, we would say that the server is “hosting” our website/app.

* **Which HTTP methods can be mapped to the CRUD acronym that we use when interfacing with APIs/Servers.**

Create, Read, Update, Delete

* **Mention two parts of Express that you learned about this week.**

Routing/router, Middleware, convenience helpers

* **Describe Middleware?**

array of functions that get executed in the order they are introduced into the server code

* **Describe a Resource?**

* everything is a **resource**.
* each resource is accessible via a **unique URI**.
* resources can have multiple **representations**.
* communication is done over a **stateless** protocol (HTTP).
* management of resources is done via **HTTP methods**.

* **What can the API return to help clients know if a request was successful?**

200 status code/201 status code

* **How can we partition our application into sub-applications?**

By dividing the code up into multiple files and 'requiring' them in the main server file.

* **Explain the difference between Relational Databases and SQL.**

SQL is the language used to access a relational database.

* **Why do tables need a primary key?**

To uniquely identify each record/row.

* **What is the name given to a table column that references the primary key on another table.**

Foreign key

* **What do we need in order to have a *many to many* relationship between two tables.**

An **intermediary table** that holds foreign keys that reference the primary key on the related tables.

* **What is the purpose of using *sessions*?**

The purpose is to persist data across requests.

* **What does bcrypt do to help us store passwords in a secure manner?**

* password hashing function.
* implements salting both manual and automatically.
* accumulative hashing rounds.

* **What does bcrypt do to slow down attackers?**

Having an algorithm that hashes the information multiple times (rounds) means an attacker needs to have the hash, know the algorithm used, and how many rounds were used to generate the hash in the first place. So it basically makes it a lot more difficult to get somebody's password.

* **What are the three parts of the JSON Web Token?**

Header, payload, signature