## What is the main.hbs file & why it is in the layouts folder?

* main.hbs is the main handlebars template file. It is typically located in the layouts folder because it serves as the layout template that other templates will be injected into. The other handlebars templates, such as ones for specific pages or components, are typically included within main.hbs using the handlebars {{{body}}} helper.

## How would you use the main.hbs file with a template?

### This is the main.hbs file:

**<!**DOCTYPE html**>**

**<**html**>**

**<**head**>**

**<**title**>**My Website**</**title**>**

**</**head**>**

**<**body**>**

**<**header**>**

**<**nav**>**

**<**a href**=**"/"**>**Home**</**a**>**

**<**a href**=**"/about"**>**About**</**a**>**

**<**a href**=**"/contact"**>**Contact**</**a**>**

**</**nav**>**

**</**header**>**

**{{{**body**}}}**

**<**footer**>**

Copyright © **{{**year**}}**

**</**footer**>**

**</**body**>**

**</**html**>**

In this example, {{{body}}} is where the other templates are included.

### For example, let's say we have a template called home.hbs that looks like this:

**<**div **class=**"home-page"**>**

**<**h1**>**Welcome to my website**!</**h1**>**

**<**p**>**Here you can find information about me **and** my projects**.</**p**>**

**</**div**>**

### When the server renders the home.hbs template, it will be injected into the {{{body}}} section of main.hbs, resulting in the following HTML being sent to the client:

**<!**DOCTYPE html**>**

**<**html**>**

**<**head**>**

**<**title**>**My Website**</**title**>**

**</**head**>**

**<**body**>**

**<**header**>**

**<**nav**>**

**<**a href**=**"/"**>**Home**</**a**>**

**<**a href**=**"/about"**>**About**</**a**>**

**<**a href**=**"/contact"**>**Contact**</**a**>**

**</**nav**>**

**</**header**>**

**<**div **class=**"home-page"**>**

**<**h1**>**Welcome to my website**!</**h1**>**

**<**p**>**Here you can find information about me **and** my projects**.</**p**>**

**</**div**>**

**<**footer**>**

Copyright © **{{**year**}}**

**</**footer**>**

**</**body**>**

**</**html**>**

## How would this work with multiple templates?

When you have multiple templates, you can use the same main.hbs layout template for all of them by including the {{{body}}} helper in the appropriate location, and then rendering the appropriate template in the route handler function of the server.js file.

### For example, let's say you have two additional templates called about.hbs and contact.hbs, in addition to home.hbs from the previous example. The about.hbs template might look like this:

**<**div **class=**"about-page"**>**

**<**h1**>**About Me**</**h1**>**

**<**p**>**I am a software developer **with** experience **in** various programming languages **and** frameworks**.</**p**>**

**</**div**>**

### And the contact.hbs template might look like this:

**<**div **class=**"contact-page"**>**

**<**h1**>**Contact Me**</**h1**>**

**<**p**>**You can reach me at **<**a href**=**"mailto:example@email.com"**>**example**@**email**.**com**</**a**>.</**p**>**

**</**div**>**

In the server.js file, you would have route handler functions for each of the routes, such as '/', '/about', and '/contact', that render the appropriate template using the res.render() function, passing in the name of the template as the first argument.

### For example, the handler function for the '/about' route might look like this:

app**.**get**(**'/about'**,** **(**req**,** res**)** **=>** **{**

res**.**render**(**'about'**);**

**});**

### When a client makes a request to the '/about' route, the server will render the about.hbs template and inject it into the {{{body}}} section of main.hbs, resulting in the following HTML being sent to the client:

**<!**DOCTYPE html**>**

**<**html**>**

**<**head**>**

**<**title**>**My Website**</**title**>**

**</**head**>**

**<**body**>**

**<**header**>**

**<**nav**>**

**<**a href**=**"/"**>**Home**</**a**>**

**<**a href**=**"/about"**>**About**</**a**>**

**<**a href**=**"/contact"**>**Contact**</**a**>**

**</**nav**>**

**</**header**>**

**<**div **class=**"about-page"**>**

**<**h1**>**About Me**</**h1**>**

**<**p**>**I am a software developer **with** experience **in** various programming languages **and** frameworks**.</**p**>**

**</**div**>**

**<**footer**>**

Copyright © **{{**year**}}**

**</**footer**>**

**</**body**>**

**</**html**>**

### Similarly, when a client makes a request to the '/contact' route, the server will render the contact.hbs template and inject it into the {{{body}}} section of main.hbs.

## Local Storage value

Local storage is a type of web storage that allows web applications to store data locally within the user's browser. It provides a key-value pair storage system, like cookies, but with much larger storage capacity and no expiration date. Data stored in local storage is only accessible to the web application that created it and is not accessible to other websites.

You can use local storage to store data that you want to persist between page reloads or browser sessions. For example, you could use local storage to store user preferences, form data, or session data.

In general, using Localstorage is a good practice in the workplace as it provides a way to store data within the browser, which can be useful in many scenarios, such as keeping track of user preferences or form data. However, it is important to keep in mind that Localstorage data can be accessed and modified by any script running on your website, so it may not be suitable for sensitive data.

## Local Storage & Session Storage

Session storage and local storage are both ways to store data in a user's browser. The key difference between them is how long the data is stored for.

Session storage is only stored for the duration of a user's browser session. This means that if a user closes their browser or navigates away from the website, the data stored in session storage will be deleted. It's useful for storing temporary data like user preferences or form data while they are still filling it out.

Local storage, on the other hand, stores data permanently, even after the user closes their browser or turns off their computer. This makes it useful for storing data that should persist between sessions, like user preferences or login information.

Both session storage and local storage are stored in the browser's memory and the data is accessible through JavaScript.

To store data in session storage, you can use the setItem() method. It take two arguments, the first is the key, and the second is the value. For example:

**sessionStorage.setItem("buttonValue", buttonValue);**

To retrieve data from session storage, you can use the getItem() method. It takes one argument, the key of the item you want to retrieve. For example:

**var buttonValue = sessionStorage.getItem("buttonValue");**

It's similar for **the local storage**, you can use the setItem and getItem method as well.

Both session storage and local storage have a limit of around 5-10 MB, depending on the browser.

## Alternatives to Session and Local (in case cookies r no allowed)

An alternative to using cookies would be to use the browser's URL to store data. This can be done by appending the data as a query string to the URL when redirecting to the next page. For example, instead of redirecting to "restaurant.html", you would redirect to "restaurant.html?buttonValue=1206". Then, on the next page, you can use JavaScript to read the query string from the URL and retrieve the data. This method is called URL encoding.

Another alternative method would be to use the HTML5 Web Storage API. This API allows you to store key-value pairs in the browser, similar to cookies. The data is stored on the client side and can be accessed by JavaScript. There are two types of storage: localStorage and sessionStorage. localStorage stores data with no expiration date, meaning it will persist even after the browser is closed, while sessionStorage stores data for a single session, it will be deleted when the browser is closed.

However, it's worth noting that the user can disable the web storage in their browser and some browser also have private mode that doesn't allow the storage of any data.

## REFERENCES:

1. <https://developers.google.com/maps/documentation/places/web-service/search-text>
2. <https://stackoverflow.com/questions/16562577/how-can-i-make-a-button-redirect-my-page-to-another-page>
3. <https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage>