Major Project 1

Understanding the Web Design Process

Being a web-developer puts you in the shoes of a developer, as well as a designer. Earlier, you were working on assignments and projects where you were already told the task you'd need to do, and you already had an idea about where to start. But, if we ask you to design and build a website from scratch, you might not be able to do it. You would need to understand the web design process, to be able to create full-fledged websites. Here, we'll talk about that in-depth. There are certain steps in the web design process, which are explained below:

- **Setting goals and targets:** The first step should be to set your goals. You should know the reason and purpose of the development. After that, define the scope of the project. Clear goals will increase your focus and they'll stop you from having unreal expectations. Expecting an unreal user base can also turn out to be a huge letdown, in the end, so select your target audience accordingly.
- **Figuring out what to implement:** Once you're clear about what you want to do, you'll need to figure out how you're gonna make things work!
    - **Features**: Make sure you know what features you want to include for the users and how will you implement them.
    - **Screens**: Plan all the screens and pages of the website so that they fulfill all your goals: what you want to show to your users, how will the intended feature be placed on a page, how many pages should be there in total, etc. Looking at similar websites will help you a lot, but don't try to copy everything. Just try to learn and take ideas.
- **Design:** Here you'll plan, how your website will look. The design includes making wireframes and deciding breakpoints which will help you a lot.

    - **Wireframes**: They're like a blueprint of your website screens, like an outline, providing a detailed view of the page content. They'll guide you in defining the hierarchy of all the elements on the page. They're generally colorless and emphasize more on the structure. These are of two types:
        - **Low Fidelity** - They're like a rough sketch and give a rough idea about the page structure. They're like the first visual representation of the designer's idea. They generally show UI elements such as boxes, lines, and buttons, without much detail.
        - **High Fidelity** - They're comparatively more technical and fine, and provide a detailed idea about the website. They're like the final design of the website. It takes effort to create them and they generally show all the features and provide a view similar to actual UI elements.
    - **Breakpoints/approach**: Decide the right approach for your website; whether it be mobile-first or desktop first and set breakpoints accordingly.
- **Development**: When all the above steps are completed, then the final development phase begins, where you'll actually write code for all the earlier decided features keeping performance and responsiveness in mind.
- **Launch**: After the development and testing are done, you would launch the website.
- **Maintenance**: Like vehicles, websites and applications also require maintenance. This is the final stage of this process, where the site's content and performance are maintained according to the needs of the users, and any bugs or performance issues are fixed by implementing patches on a regular basis.

CSS Variables

We have variables in CSS like other programming languages. They're also referred to as CSS custom properties. You can create a variable, assign a value to it, and reuse it throughout the document.

Syntax:

- To declare a variable, you'll need to write **double hyphens (--)** followed by a variable name that further needs a **value** that can be any valid css value.
- To access/use that variable, you'll need to use the **var()** function.

Example:

```
body{
    --custom-color: orange;
}
```

Now, this variable can be accessed using the var() function:

```
body{
    --custom-color: orange;
}
span {
    background-color: var(--custom-color);
}
```

Advantages of using CSS Variables:

- It increases the code readability
- Instead of copy-pasting values over and over again, you can place them as a variable and then use them. Making changes will be much easier as you just have to change the value of the variable, which will affect all places where it has been used.

  For example: If you want multiple elements to have the same color, you can store the value of the color in a variable and use this variable in place of the color value. In case, you want to make any changes, just changing the color value of the variable will make changes everywhere, which will save a lot of time.

The var() function: It's used to access the value of a variable. It can take two parameters:

- **Name** (required): Name of the variable whose value you want to access
- **Value** (optional): Here, you can specify a **default** value, in case if you mistyped the variable name, the optional value will be used.

Look at the example below, where we create a custom variable(--customHeight) to set height. When we mistype the variable in the ***var() function,*** the default value (200px) will be used.

```
<head>
<style>
body {
    --customHeight: 100px;
}
div{
    margin: 100px;
    border: 1px solid white;
    background-color: seagreen;
    width: 100px;
    height: var(--customHeighst, 200px);
}
</style>
</head>

<body>
    <div>
    </div>
</body>
```

CSS Variables Scope

CSS variables can be global or local.

Global variables can be used anywhere on the entire page, whereas, the local variables can only be used inside the selectors where they're declared.

You can create global variables by declaring them inside the**: root selector** or in the HTML element, as all the elements come under the **HTML** element, CSS variables declared in it, or in the**: root** selector can be inherited directly by all the elements on the page.

Generally, a variable declared in an element will only be visible to its child elements.

For example:

```
<head>
    <style>
        body {
            --customColor: yellow;
        }
```

```
        p {
            color: var(--customColor);
        }
    </style>
</head>

<body>
    <p>Hi!</p>

</body>
```

Output:

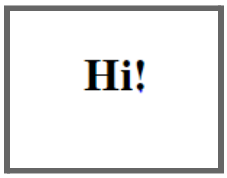
```
Hi!
```

As we can see, the variable **--customColor** is declared in the **<body>** and is inherited by its child element **<p>.** But the reverse of the above condition is invalid, as variables can only be inherited by the child elements and as **<body>** is not a child of the **<p>,** a variable declared inside **<p>** will not be visible to the **<body>.** So, we can say that the ability to inherit the variables depends upon a parent-child relationship.

As discussed earlier, you can set a default value in the var(), so that, if you mis-typed the variable name somehow, that default value will be used.

```
<head>
    <style>
        body {
            --customColor: yellow;
        }
        h1 {
            color: var(--customColorr, black);
        }
    </style>
</head>
<body>
    <h1>Hi!</h1>
</body>
```

Output:



As we can see in the above example, after mistyping the variable name, the default (black) value was used.

If we declare a variable using the **:root** selector, then it'll be visible to every element in the page.

```
<head>
    <style>
        :root {
            --customSize: 40px;
        }
        p {
            font-size: var(--customSize);
        }
    </style>
</head>

<body>
    <p>Hi!</p>
</body>
```

If any browser doesn't support CSS variables, make sure you set a default value in the var() function, as it'll be used in place of the variable.

You can read more about CSS variables from the link below:

- https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties
- https://codeburst.io/css-variables-explained-with-5-examples-84adaffaa5bd