LEMONCODE

HTML

Custom Properties

Las Custom Properties, también conocidas como Variables CSS, son una característica de CSS que te permite asignar valores a nombres específicos, los cuales luego pueden ser reutilizados a lo largo de tu hoja de estilos.

Ventajas de las Custom Properties

Las Custom Properties ofrecen varias ventajas:

1. Mantenimiento: Permite mantener un código más limpio y fácil de mantener.
2. Flexibilidad: Proporciona flexibilidad al poder reutilizar valores en diferentes partes del código.
3. Consistencia: Ayuda a mantener una paleta de colores y estilos coherentes en todo el proyecto.
4. Acceso y manipulación en JavaScript: A diferencia de las variables de pre-procesadores como Sass o Less, las Custom Properties están disponibles en tiempo de ejecución y pueden ser leídas y modificadas a través de JavaScript, lo que permite una mayor interacción dinámica en los proyectos.

Ejemplo de uso de Custom Properties

En el ejemplo, tenemos un formulario con un input y dos botones. Como vemos en el CSS, el botón tiene asignado un `background-color: blueviolet` y un `color: white` para el texto. Además vamos a añadir al botón la pseudo-clase `:hover` con un `background-color: blue` para ver que al posicionar el cursor encima del botón este cambia su color.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta chartset="UTF-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <link rel="stylesheet" href="style.css" />
        <title>Custom Properties</title>
    </head>
    <body>
        <form action="">
            <input type="checkbox" />
            <button>Cancelar</button>
            <button>Enviar</button>
        </form>
    </body>
</html>
```

```css
/* style.css */

body {
    font-family: sans-serif;
    font-size: 16px;
}

button {
    background-color: blueviolet;
    color: white;
    border-radius: 10px;
    border: none;
    padding: 16px;
}

button:hover {
    background-color: blue;
}
```

Ahora veremos como definir y utilizar las Custom Properties.

En el siguiente ejemplo, se definen tres Custom Properties en el selector `:root`, que hace referencia al elemento HTML. Estas variables están disponibles globalmente en la hoja de estilos.

Luego, se utiliza la función `var()` para acceder al valor de las Custom Properties en diferentes propiedades CSS. Por ejemplo, en el selector `button`, se utiliza `var(--primary)` para asignar el color primario al fondo del botón y `var(--secondary)` para el color del texto. También se utiliza `var(--accent)` en la pseudo-clase `:hover` para cambiar el color de fondo al interactuar con el botón.

```css
/* style.css */

:root {
    --primary: #845ec2;
    --secondary: #dbe7ed;
    --accent: #39829b;
}

body {
    font-family: sans-serif;
    font-size: 16px;
}

button {
    background-color: var(--primary);
    color: var(--secondary);
    border-radius: 10px;
    border: none;
    padding: 16px;
}
```

```
button:hover {
    background-color: var(--accent);
}
```

Las Custom Properties también se pueden utilizar para estilizar formularios de manera coherente. Por ejemplo:

```
/* style.css */

:root {
    --primary: #845ec2;
    --secondary: #dbe7ed;
    --accent: #39829b;
}

body {
    font-family: sans-serif;
    font-size: 16px;
}

button {
    background-color: var(--primary);
    color: var(--secondary);
    border-radius: 10px;
    border: none;
    padding: 16px;
}

button:hover {
    background-color: var(--accent);
}

input {
    accent-color: var(--primary);
}
```

En este caso, se utiliza la propiedad `accent-color` para definir un color de acento para todos los elementos `input` del formulario. Esto ayuda a integrar los elementos del formulario con la paleta de colores definida por las Custom Properties.

Una ventaja adicional de las Custom Properties es que se pueden acceder y manipular a través de JavaScript. Esto permite una mayor interacción y dinamismo en los proyectos.

Recuerda que al utilizar Custom Properties, puedes mejorar el mantenimiento, la consistencia y la flexibilidad de tus estilos CSS, así como aprovechar la manipulación de las variables a través de JavaScript para crear una experiencia más dinámica en tus proyectos.