Detecting user’s color preferences

There are multiple ways to handle theming an app or a site.

The first one depends on whether the user has enabled light or dark mode at the operating system level.

We can use prefers-color-scheme` media queries like any other media queries, loading them in a stylesheet:

```
@media (prefers-color-scheme: dark) {
  /*
    all styles for dark
    color scheme go here
  */
}

@media (prefers-color-scheme: light) {
  /*
    all styles for light
    color scheme go here
  */
}
```

A more interesting approach is to use the media query as the value of the `media` attribute of the link element. This will only load stylesheets that match the query, potential saving download times by reducing the size of the stylesheet we load.

```
<link
  rel="stylesheet"
  media="(prefers-color-scheme:light)"
  href="https://example.com/themes/light.css"
/>
<link
  rel="stylesheet"
  media="(prefers-color-scheme:dark)"
  href="https://example.com/themes/dark.css"
  onload="document.documentElement.classList.add('theme-dark');"
/>
```

The final way to detect if a user has enabled a light or dark theme color preference in the operating system is using Javascript.

Using the matchMedia method of the window object we can query if the document matches the media query.

The matches property will perform an immediate test and return true if the document matches the query.

With that in place we check if the dark property returns true. If it does, we add the `theme-dark` class to the HTML document; otherwise we add the `theme-light` class.

This assumes that the theme-related classes are defined in CSS.

```
let dark = window.matchMedia("(prefers-color-scheme: dark)").matches;

if (dark) {
  document.documentElement.classList.add("theme-dark");
} else {
  document.documentElement.classList.add("theme-light");
}
```

The problem with this approach is that it’s static. If we change the color preference, the browser will not update until we reload the page. This may be what we want; we may not want the changes happen and potentially reload the page.

The first remedy is to attach a change listener to the media query and take action when it triggers.

We create a constant to test if tme media query matches (if OS is using a dark color theme).

If it does, then we remove the light theme class and add the dark theme class.

If it does not then we remove the dark theme class and add the light theme.

```
const darkModeMediaQuery = window.matchMedia('(prefers-color-scheme: dark
darkModeMediaQuery.addListener((e) => {
  const darkModeOn = e.matches;

  if (darkModeOn) {
    document.documentElement.classList.remove("theme-light");
    document.documentElement.classList.add("theme-dark");
    console.log('Dark')
  } else {
    document.documentElement.classList.remove("theme-dark");
    document.documentElement.classList.add("theme-light");
    console.log('Light')
  }
});
```