Constructable style sheets

One of the outstanding issues about web components is CSS reuse. Right now we either have to use `::part` to create an external stylesheet to match the specified parts of the document or encapsulate styles inside each custom element. `::part` works well if all custom elements use the `::part` attribute and if all the names are unique accross the document, otherwise we may get unexpected results.

We've always been able to create style sheets in Javascript but it's not always a straight forward process. It involves creating a `style` element, using `innerHTML` to insert a string containing the CSS we want to use and then appending the new `style` element to the body of the page

```
const sheet = document.createElement('style');

sheet.innerHTML = "h1 {color: blue;} body {font-family: Verdana}";

document.body.appendChild(sheet);
```

This may work fine in some situations but for the most part it doesn't lend itself well to create reusable code.

Composable style sheets give you an alternative way to create and use stylesheets programmatically and use them in multiple places whether in individual pages or in the shadow DOM of custom elements.

The first step is to create the stylesheet and add the rules we want to use on it.

1. We first create a new style sheet using the style sheet constructor
2. The `replace` method provides a promise-based API
3. `replaceSync`, as the name indicates, provides a synchronous API

```
const sheet = new CSSStyleSheet(); // 1

sheet.replace('h1 { color: blue }') 'h1 { color: blue }'// 2('success'))
  .catch(console.log('There was a problem'));
```

```
sheet.replaceSync('body {font-family: verd'success'/ 3
// 3
```

The second component of composable style sheets allows us to add the stylesheet we just created to an existing document using the adoptedStyeSheets property available in shadow roots and documents.

One important thing to point out is that we're working with a frozen array so methods like `push()` would cause an error. If we want to make sure we keep existing style sheets when adding a new one, we can use concatenate to create an array that contains both the old and new style sheets... this causes no problems if there are no `adoptedStyleSheets` in the document, the array will be empty.

This is an example of how to do it for the document style sheets.

```
document.adoptedStyleSheets = [
  ...document.adoptedStyleSheets,
  sheet];
```

And this is an example of how to do it with a shadow root, inside or outside a custom element.

```
const node = document.createElement('div');
const shadow = node.attachShadow({ mode: 'open' });
shadow.adoptedStyleSheets = [
  ...shadow.adoptedStyleSheets,
  sheet
];
```

This provides for a way to style elements on the page programmatically without having to change the document using `::part` to pierce the style boundary but it is Javascript, so it may not fully appeal to everyone... CSS has a different steep learning curve that seems to be easier for most people.

I'm torn about which system to use as they both have their appeal and their preferred use cases. Will have to do more testing