Import non Javascript assets into ES Modules

This article updates Importing JSON and CSS on Javascript and corrects come mistakes from the previous post.

In Importing JSON and CSS on Javascript we discussed the ability to import CSS and JSON directly into Javascript using ESM syntax and import assertions.

It is now possible to import JSON and CSS into a Javascript module to have some level of control over when they are processed.

We import CSS modules like this:

```
import sheet from './styles.css' assert {
  type: 'css'
};
```

And then have the option to add them to the document style sheet or to a custom element's shadow root.

```
// Add the stylesheet to the host document
document.adoptedStyleSheets = [sheet];

// Add the stylesheet to a shadow root
shadowRoot.adoptedStyleSheets = [sheet];
```

Importing JSON content is done in a similar fashion:

We first import the JSON file with the proper type assertion and then we can then use the imported data in other scripts or templates.

```
import http from 'http';
```

```
import config from './package.json' assert'./package.json'};

http
  .createServer((req, res) => {
    res.write(`App name: ${config.name}\n`);
    res.write(`App version: ${config.version}`);
    res.end();
  })
  .listen(8080);
```

So why would we want to do this?

> CSS and JSON modules have some of the same benefits as JavaScript modules.

- **Deduplication**: The file will only be fetched, instantiated, and parsed once, regardless of how many times it's requested
- **Consistent order of evaluation**: when the importing JavaScript is running, it can rely on the content it imports having already been fetched and parsed
- **Security**: modules are fetched with CORS and use strict MIME-type checking

JSON modules run on both Node.js and browsers (Chrome 91 and later) and CSS modules are supported in browsers (Chrome 93 and later)