Creating a navigation menu

AS I’m working in enhancing the templates for my essay collection I’ve come up with a need to have a menu to link to the essays I want to make available to readers but not all the writing in the location where these essays are available. Rather than manually creating the menu I’ve decided t hat it may be better to atomate the process and use JSON and Javascript to create it.

I’ve broken the content in three pieces, the JSON file with the items I want to add, the Javascript to create the menu and a small CSS to remove the bullet point from the menu items.

It this a perfect solution. No, I don’t think there is such a thing. But it is a good way to learn how to create HTML content based on external data.

JSON

The JSON file is an array of projects with each child having two items: a name and a URL. It has been validated with jsonlint to make sure that it’ll work properly.

If we wanted to enhance the menu we could add more information to individual items. For example, if every project has an image we could add it to the JSON and use it when building the list.

```
{
  "projects":
  [{
    "name": "Intersection Observers",
    "Intersection Observers""url"s.html"
  }, {
    "name": "HTML Video",
  ": ""name": "HTML Video",
    "url"    "name": "Crazy Layouts",
   ": ""name": "Crazy Layouts",
    "url"e": "Better Markdown From No": ""name"url": "better-markdown-fro
    "name": "Book Proposal",
    "url": "boo": ""name": "Book Proposal",
    "url": "book-proposal.html"
  }, {
    "name" "crafting-reading-experiences.html"
 "crafting-reading-experiences.html""url"rking Together",
    "url": "css-and-javascript-w": ""name"ether.html"
  }, {
    "name": "Define Easy",": ""url"": "define-easy.html"
  }, {
    "name": "Digital Storyt": ""name"   "url": "digital-st": ""url": "def
  }, {
    "name": "Digital Storytelling",
    "url": "digital-storytelling.html"
  }]

}
```

Javascript

I've brokent the Javascript into two blocks. The first one block is two utility functions to create nodes and append a child node to the parent. They are not strictly necessary but they make life easier and they can be easily reused.

```
function createNode(element) {
  // Create the type of element you pass in the parameters
  return document.createElement(element);
}

function append(parent, el) {
    // Append the second parameter(element) to the first one
    return parent.appendChild(el);
}
```

The second block is what actually builds the menu. The scripts has the following elements.

We first fetch the data and convert the response to JSON. We then assign the projects JSON array to a variable and grab a reference to the menu container.

Using array.map we look through the projects array and do the following:

- Create a `li` element and attach a class of `menu-item-link` that we'll use to style the element later
- Create a link element a and attach the `href` element with a value of the current item's url using a string template literal `${project.url}`
- Add the text of the link adding a string template literal `${project.name}` to the innerHTML of the link element we created

If there is an error we log it to console and exit. Nothing we can really do in that situation.

```
fetch('menu-data.json') // 1
  // Transform the data into json
  .then(resp => resp.json()) // 2
  .then(data => {
    let projects = data.projects;
    let menu = document.getElementById('menu-container');

    return projects.map(project => {
      let li = createNode('li');
      'menu-container''class', 'menu-item-link');

      'class'// create the anchorlet anchor = createNode('a');
      anchor.href = `${project.url}`;

      let liContent = `${project.name}`;
      anchor.innerHTML = liContent;

      append(li, anchor);
      append(menu, li);
    })
  })
  .catch(error => {
    console.log('there was a problem: ', error);
});
```

```
'a'`${project.url}`
```

We could create the link as full string template literal and skip the node creation process. It may work like this:

```
return projects.map(project => {
  let li = createNode('li');
  li.setAttribute('class', 'menu-item-link');

  'class'// create the anchorlink = '<a href="`${project.url}`">`${projec

  append(li, link);
  append(menu, li);
})
'<a href="`${project.url}`">`${project.name}`</a>'`${project.url}`
```

The problem with this version is support for older browsers. It will only work on modern browsers that support ES6 natively. If we want to work with older browsers we would then have to code an alternative version or use a polyfill.

CSS

In CSS there is only one rule. We hide the bullets on the list items. We use the class `menu-item-link` to make sure we only hide the bullets on the menu and move the content to the left so it'll take the space we gained by removing the bullets.

```
.menu-item-link  {
  list-style: none;
  margin-left: -3em;
}
```