Lesson Wednesday

Intermediate JavaScript (/intermediate-javascript) / Test-Driven Development and Environments with JavaScript (/intermediate-javascript/test-driven-development-and-environments-with-javascript) / Managing Images with webpack

Text

The primary focus of this section should be test-driven development, including writing more complex business logic along with tests. However, many students also want to incorporate images into their projects. While this should be a lower priority than using test-driven development (and adding images isn't required for this section's project), you are welcome to experiment with adding images if you have completed other key aspects of your project.

Why webpack for Images?

Now that we are using webpack as a module bundler, all of our source code is being combined in a single *dist* folder. For that reason, we need to use webpack to manage images and correctly add them to our *dist* directory.

Configuring webpack for Images

We'll need to add two packages to manage images with webpack. First, we'll install the **file-loader** package:

```
$ npm install file-loader --save-dev
```

Next, we'll install the **html-loader** package:

```
$ npm install html-loader@1.3.2 --save-dev
```

Finally, we need to configure webpack to use these new tools. We'll create two new entries in the `rules` array within the `module` object of `webpack.config.js`:

webpack.config.js

```
...

{
  test: /\.(gif|png|avif|jpe?g)$/,
  use: [
    {
      loader: 'file-loader',
      options: {
        name: '[name].[ext]',
        outputPath: 'assets/images/'
      }
    }
  ]
},
{
  test:/\.html$/,
  use: [
    'html-loader'
  ]
},

...
```

Let's walk through this new code:

- The first `test` section states which file types this loader will be applied to. We list `/\.(gif|png|avif|jpe?g)$/` to instruct webpack to handle `.gif`, `.png`, `.jpg`, `.avif`, and `.jpeg` extensions. If there is another file extension that we want the file loader to handle, we simply need to add it to the list in the webpack configuration.
- `use` specifies `file-loader` as the webpack loader responsible for handling these file types. Because this dependency has more settings than other loaders, we add an `options` object with additional settings in key-value pairs:
  - `name` tells webpack what to name the image file it places in *dist*. By stating `[name].[ext]`, we tell it to simply use the

file's existing name and extension.
    - `outputPath` tells it where in *dist* it should place this image.
- The second `test` section begins a new rule for `.html` files.
- The value corresponding to the `use` key in this rule states `.html` file types should have our new `html-loader` applied to them. As stated in this loader's GitHub Documentation (https://github.com/webpack-contrib/html-loader), this dependency invokes webpack to load the corresponding image resource for any `<img>` tags it spots in our HTML.

Saving Images in a Project

We can now begin adding images to our site. There's no strict, universal rule for where to place image files. However, **it's common practice to house resources such as images, fonts, and icons in an *assets/* directory that contains corresponding subdirectories for each type of resource**, such as

- `assets/images/`
- `assets/fonts/`

We've configured webpack to use file-loader to output images to `dist/assets/images/`, and we'll go ahead and use this same naming for locating our images in our source code: `src/assets/images/`. Go ahead and create this directory and subdirectory now.

Next, we'll add an image. For this example we'll download this free stock image of a puppy (https://unsplash.com/photos/-Go4DH2pZbc) from Unsplash.com (https://unsplash.com) and save it in a file named `stock-puppy-photo.jpg` in our `src/assets/images` directory.

Rendering webpack-Bundled Images

Now let's add our image to our HTML so we can see it in the browser. We'll add the following `<img>` tag to *index.html*:

src/index.html

```
<html lang="en-US">
<head>
</head>
<body>
  <div>Hello world!</div>
  <img src="./assets/images/stock-puppy-photo.jpg" alt="suc
h a pup!">
</body>
</html>
```

We link to the location of our new image and add an `alt` property.

Let's see our new image in action. Close the project's development server if it's open, and run `$ npm run start`.

Previous (/intermediate-javascript/test-driven-development-and-environments-with-javascript/haiku-creator-rpg-sudoku-solver-two-day-project-part-1)
Next (/intermediate-javascript/test-driven-development-and-environments-with-javascript/es6-array-and-object-destructuring)
Lesson 42 of 49
Last updated more than 3 months ago.

disable dark mode

Epicodus (http://www.epicodus.com)