Implement webpack bundle for checking Login Credentials

Ex: No: 11 **Use Web Pack for Application**

Webpack is an open-source JavaScript module bundler. Webpack allows you to split your JavaScript into separate modules in development (better for maintenance) while letting you compile those modules into a single bundle in production (better for performance).

Steps for bundling App using WebPack:

1) Install webpack. In this reagard, we need to create folder called webpack-example

```
cd webpack-example
```

```
npm init -y
```

```
npm install webpack webpack-cli --save-dev
```

Once you nstalled, your package.json will be

```
{
  "name": "webpack-example",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "webpack": "^5.27.2",
    "webpack-cli": "^4.5.0"
  }
}
```

2) Setting up a project to bundle

In the same location as my `package.json` file, I'm going to add the following:

- A folder called `src`
- An `index.html` file inside `src`
- An `index.js` file inside `src`
- A `dist` folder

3) In this example, we uses two JavaScript utility libraries:

Flicking – A JavaScript carousel

Panzoom – A pan/zoom framework

```
npm install panzoom --save
```

```
npm install @egjs/flicking -save
```

Now my `package.json` has the following appended below the `devDependencies` section:

```
"dependencies": {

  "@egjs/flicking": "^3.8.1",

  "panzoom": "^9.4.1"

}
```

4) To demonstrate that these utilities are successfully bundled, I’m going to add the following to my `index.js` file in the `src` folder:

```
import panzoom from 'panzoom';

import flicking from '@egjs/flicking';



console.log(panzoom);

console.log(flicking);
```

```
npx webpack
```

5) Configuring webpack to generate HTML

   ```
   npm install html-webpack-plugin --save-dev
   ```

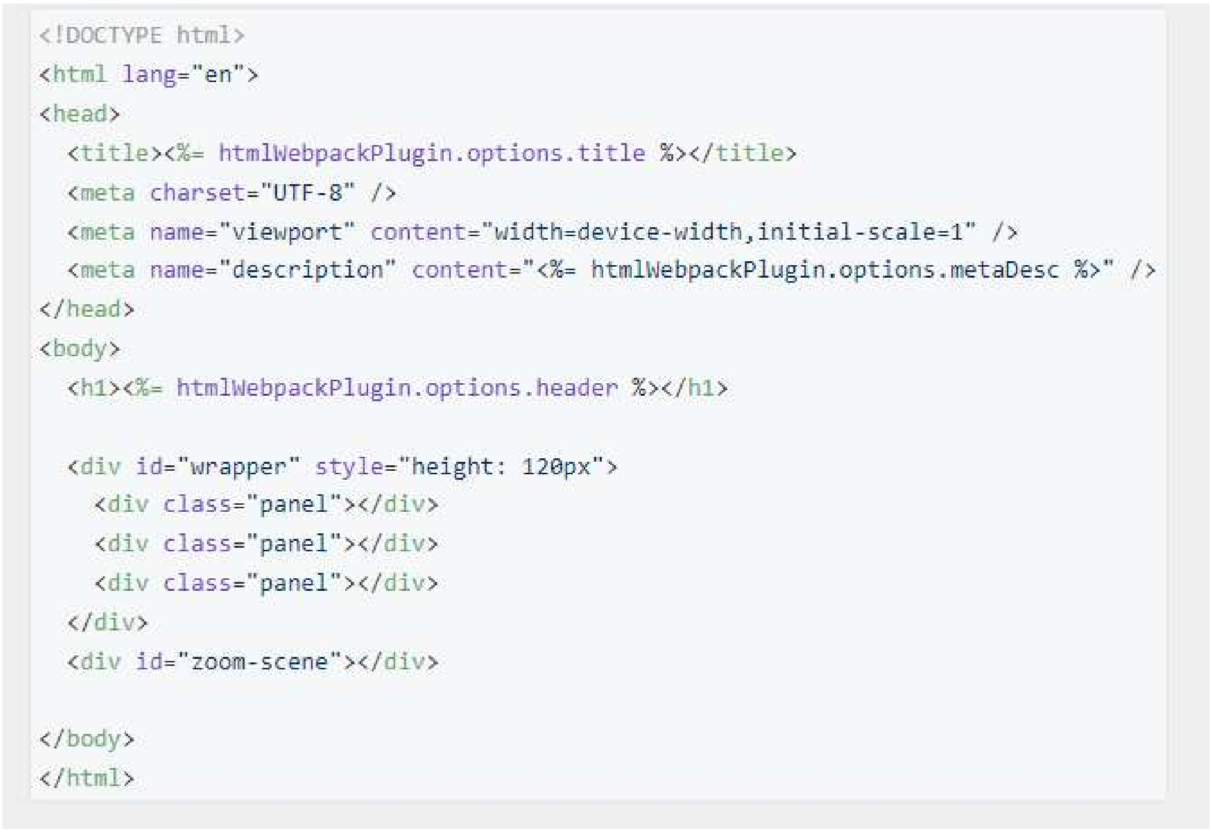
My devDependencies in `package.json` will reflect the change:

```
"devDependencies": {

  "html-webpack-plugin": "^5.3.1",

  "webpack": "^5.27.2",

  "webpack-cli": "^4.5.0"

},
```

6) I'm going to create a `webpack.config.js` file in my project's root folder. Inside this file, I'm going to add the following to utilize this newly installed plugin:

```
const HtmlWebpackPlugin = require('html-webpack-plugin');

const path = require('path');



module.exports = {

  plugins: [

    new HtmlWebpackPlugin()

  ]

};
```

7) make the following changes to src/index.html:



```
<!DOCTYPE html>
<html lang="en">
<head>
  <title><%= htmlWebpackPlugin.options.title %></title>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <meta name="description" content="<%= htmlWebpackPlugin.options.metaDesc %>" />
</head>
<body>
  <h1><%= htmlWebpackPlugin.options.header %></h1>

  <div id="wrapper" style="height: 120px">
    <div class="panel"></div>
    <div class="panel"></div>
    <div class="panel"></div>
  </div>
  <div id="zoom-scene"></div>

</body>
</html>
```

8) Next I'm going to add some options to `HtmlWebpackPlugin()` in my `webpack.config.js` file. The `plugins: []` section of my `webpack.config.js` file now looks like this:

```
plugins: [

  new HtmlWebpackPlugin({

    hash: true,

    title: 'Webpack Example App',

    header: 'Webpack Example Title',

    metaDesc: 'Webpack Example Description',

    template: './src/index.html',

    filename: 'index.html',

    inject: 'body'

  })

]
```

9) Enabling development mode in webpack

10) To enable development mode I'll add a line to my `webpack.config.js` file so the complete file will now look like this:

```
const path = require('path');

module.exports = {
  plugins: [
    new HtmlWebpackPlugin({
      hash: true,
      title: 'Webpack Example App',
      header: 'Webpack Example Title',
      metaDesc: 'Webpack Example Description',
      template: './src/index.html',
      filename: 'index.html',
      inject: 'body'
    })
  ],
  mode: 'development'
};
```

11) Cleaning up the dist folder

12) I'm going to add another line to my `webpack.config.js` file, which will now look like this (again, note the additional comma for proper syntax):

```
const path = require('path');

module.exports = {
  plugins: [
    new HtmlWebpackPlugin({
      hash: true,
      title: 'Webpack Example App',
      header: 'Webpack Example Title',
      metaDesc: 'Webpack Example Description',
      template: './src/index.html',
      filename: 'index.html',
      inject: 'body'
    })
  ],
  mode: 'development',
  output: {
    clean: true
  }
};
```

13) Running webpack with an npm script

14) In my `package.json` file, there's a "scripts" section. I'm going to add a line to that so it looks like this (again, note the extra comma):

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "build": "webpack"
},
```

15) Now I can run webpack using the following command in my project's root directory:

```
npm run build
```

16) Ff Alternatively, if I want to use scripts to differentiate between development and production builds, I can do the following:

```
"scripts": {

  "test": "echo \"Error: no test specified\" && exit 1",

  "dev": "webpack --mode development",

  "build": "webpack --mode production"

},
```

17) Now I can run either `npm run dev` or `npm run build`, depending on what I want to do with my project.

18) Installing and running a server with hot reload

19) To install a server as a developer dependency, I'm going to run the following command in my project's root directory:

```
20) npm install webpack-dev-server --save-dev
```

21) Once that's installed, I'm going to add a few lines to my `webpack.config.js` file:

```
const HtmlWebpackPlugin = require('html-webpack-plugin');
const path = require('path');

module.exports = {
  plugins: [
    new HtmlWebpackPlugin({
      hash: true,
      title: 'Webpack Example App',
      header: 'Webpack Example Title',
      metaDesc: 'Webpack Example Description',
      template: './src/index.html',
      filename: 'index.html',
      inject: 'body'
    })
  ],
  mode: 'development',
  output: {
    clean: true
  },
  devServer: {
    contentBase: './dist',
    open: true
  }
};
```

22) One final thing I need to do is add the server as part of my build script in `package.json`:

```
"scripts": {

  "test": "echo \"Error: no test specified\" && exit 1",

  "dev": "webpack serve --mode development",

  "build": "webpack --mode production"

},
```

23) `npm run dev`

24) Final example of a webpack implementation

25) With all the above in place, the `npm run dev` command will produce my build each time it's executed. I can then use the following command to build my project for production:

```
26) npm run build
```

27) This executes the `build` script in production mode (as outlined in my `package.json`). In my case, this produces a minified version of the following inside `index.html` in the `dist` folder:

```html
<!doctype html>
<html lang="en">
  <head>
    <title>Webpack Example App</title>
    <meta charset="UTF-8"/>
    <meta name="viewport" content="width=device-width,initial-
scale=1"/>
    <meta name="description" content="Webpack Example
Description"/>
  </head>

  <body>
    <h1>Webpack Example Title</h1>
    <div id="wrapper" style="height: 120px">
      <div class="panel"></div>
      <div class="panel"></div>
      <div class="panel"></div>
    </div>
    <div id="zoom-scene"></div>
    <script defer="defer"
src="main.js?097d8b8eda8ecc97a023"></script>
  </body>
</html>
```