Intro to Webpack

The Coding Bootcamp

Class Objectives

- Babel Exercise
- Create a basic Webpack configuration file.
- Bundle their JavaScript code into a single file.
- Add Webpack plugins to their Webpack configs.
- Convert web applications to PWA using Webpack
- Perform code splitting using webpac

## BABEL-IZE

# BABEL-IZE ||| 0-Mongo-Crud-Babel

### Make sure MONGO DB "notetaker" exists

### Install packages and Start App to make sure application works
npm i
npm start

### Install both babel-cli and babel-preset-env
npm i babel-cli babel-preset-env

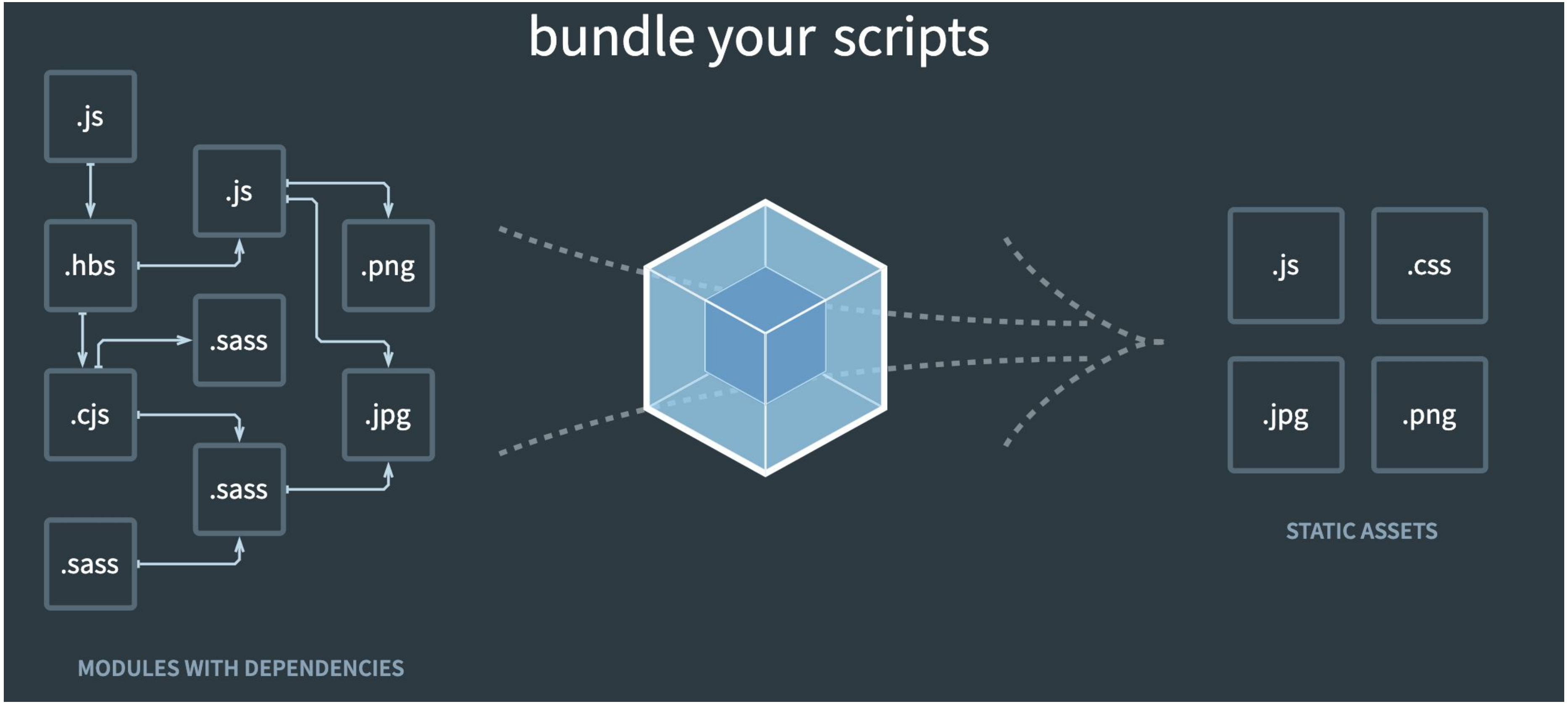
### Add script to package.json
"build": "babel public -d public/babel"

### Run Build Script
npm run build

### Update index.html
Change `<script src="app.js"></script>` to `<script src="babel/app.js"></script>`

### Start App
npm start

Webpack


bundle your scripts
.js
.js
.hbs
.png
.sass
.cjs
.jpg
.sass
.sass
MODULES WITH DEPENDENCIES
.js
.css
.jpg
.png
STATIC ASSETS

Intro Webpack

☝️ In terms of performance, what are the skills we have learned so far?

☝️ What if our application has dependencies? Do we minify those by-hand?

☝️ Today we're learning webpack. What do you think webpack does?

Intro Webpack

webpack.config.js

```
const config = {
  entry: "./src/app.js",
  output: {
    path: __dirname + "/dist",
    filename: "bundle.js"
  },
  mode: "development"
};

module.exports = config;
```

Here the application starts executing and webpack starts bundling

The target directory for all output files must be an absolute path and bundled filename.

Chosen mode tells webpack to use its built-in optimizations accordingly.

Webpack script changes in JSON

```
{
    "name": "webpack-demo",
    "version": "1.0.0",
    "description": "",
    "main": "app.js",
    "scripts": {
        "webpack": "webpack --watch",
        "build": "npm run webpack",
        "test": "echo \"Error: no test specified\" && exit 1"
    },
    "keywords": [],
    "author": "",
    "license": "ISC",
    "devDependencies": {
        "webpack": "^4.31.0",
        "webpack-cli": "^3.3.2"
    }
}
```

Student Do: Budget Tracker (10 mins)

```
* Run the following command: `npm install webpack webpack-cli -D`

* Create a file called `webpack.config.js`.

* Using the entry point of `src/app.js`, make Webpack output a bundle file in a folder
called `dist/`.

* In `index.html`, change the JavaScript file src to be your new bundle file.

* Add the necessary scripts to `package.json`, then run Webpack with the `--watch` option.
```

~~* Update this application to accomplish the following:~~

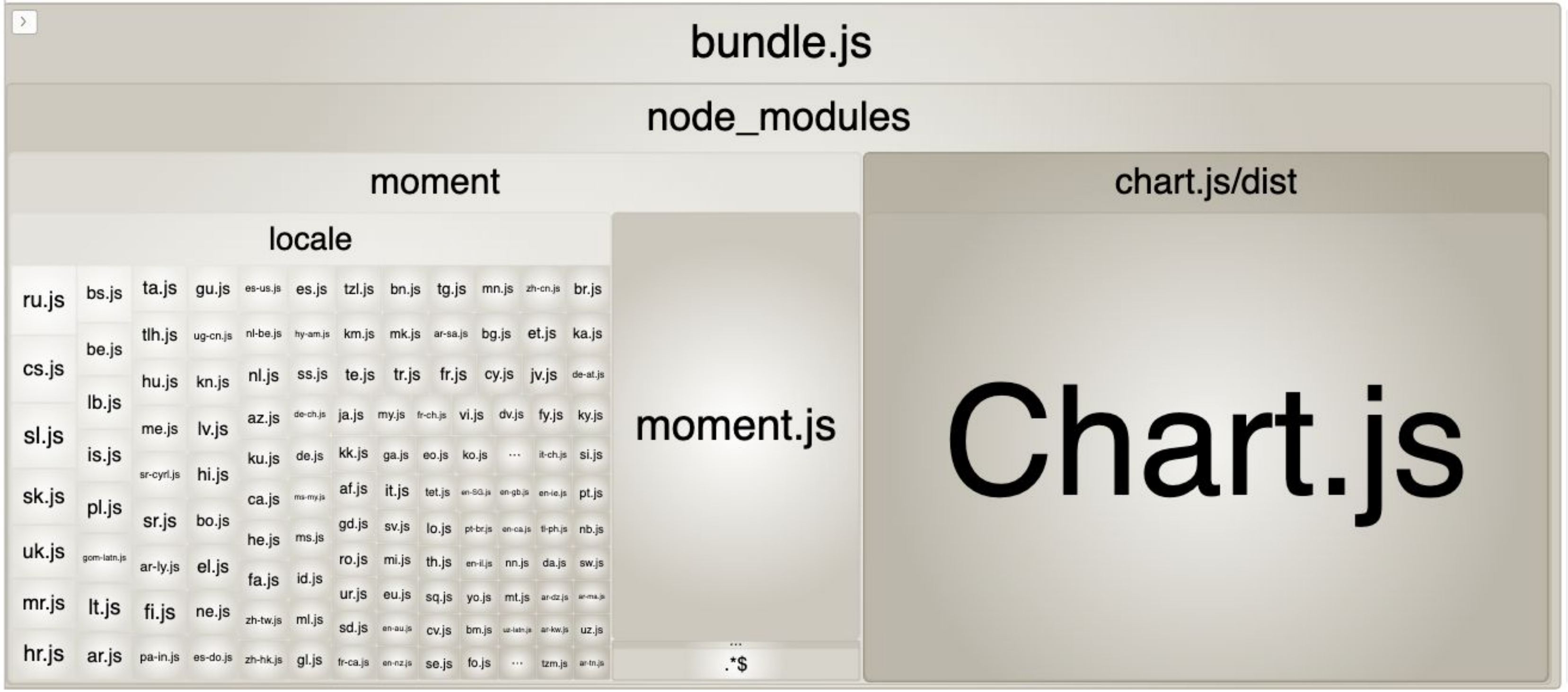
~~* When the user types in a value to the price field and clicks submit, the remaining balance should be updated.~~

~~* Using the `require` module and `module.exports`, move the code that calculates the new budget to a file named `calculations.js`.~~

~~* Update the `reset` function so that when clicked, it sets the current balance back to its original balance and clears the list of expenses.~~

Bundle Analyzer Plugin

This plugin helps us analyze the different impacts that libraries have on the bundle size of our application



SWPrecacheWebpackPlugin and WebpackPwaManifest

The first plugin, `SWPrecacheWebpackPlugin` allows us to cache our external project dependencies. It generates a service worker using an existing service worker and adds it to our build directory.

The second plugin, `WebpackPwaManifest` generates a manifest.json file to be included in our build directory. While most of the properties are the same as a regular manifest.json, this plugin also automatically resizes all of our icons to the appropriate sizes and allows for the use of ES6 features and JavaScript comments.

```
plugins: [
  new SWPrecacheWebpackPlugin({
    cacheId: "my-domain-cache-id",
    dontCacheBustUrlsMatching: /\.\w{8}\./,
    filename: "service-worker.js",
    minify: true,
    staticFileGlobsIgnorePatterns: [/\.map$/,
/manifest\.json$/]
  }),
  new WebpackPwaManifest({
    name: "Images App",
    short_name: "Images App",
    description: "An application for images",
    background_color: "#01579b",
    theme_color: "#ffffff",
    "theme-color": "#ffffff",
    start_url: "/",
    icons: [
      {
        src:
path.resolve("public/assets/images/icons/icon192x192.png
"),
        sizes: [96, 128, 192, 256, 384, 512],
        destination: path.join("assets", "icons")
      }
    ]
  })
  ],
```

Student Do: Gallery App with Webpack (10 mins)

In this activity we will adjust our Gallery app so that Webpack minifies and bundles our code.

## Instructions

* Before you begin, make sure mongod is running and to install all of the necessary dependencies with “npm install” then “node seeders/seed.js”, .

* Run the following command: `npm install sw-precache-webpack-plugin webpack-pwa-manifest -D`.

* In a separate tab in your terminal, start a mongodb server with `mongod`.

* Run `npm start` to make sure that the application works as expected.

* Additional Instructions in Readme

Pure vs Impure Functions

Pure vs Impure Functions. The term **pure function**, means that the **function's** return value is completely determined by its inputs and that executing the **function** produces no side effects. A **function** that is not **pure** is called '**impure**'.

A pure function always returns the same result for the same argument values and doesn’t have any *externally-visible side-effects*.

ES6 in all browsers

☝️ What tool do you think we will need to use to allow us to use ES6 in all browsers?

Babel! And the babel-loader in webpack config

```
const config = {
  entry: "./src/app.js",
  output: {
    path:   dirname + "/dist",
    filename: "bundle.js"
  },
  mode: "development",
  module: {
    rules: [{
      test: /\.m?js$/,
      exclude: /(node_modules)/,
      use: {
        loader: "babel-loader",
        options: {
          presets: ["@babel/preset-env"]
        }
      }
    }]
  }
};
module.exports = config;
```

Student Do: Gallery Pure Functions (15 mins)

```
In this activity we will adjust our Gallery app so that Webpack minifies and bundles our
code. 22-Stu-Chunking

* Run `npm install`.

* Using the ES6 import/export syntax, separate functions out into separate JavaScript files
to make your application more modular.

* While there are many ways that you can separate your JavaScript files, it is recommended
that you create somethings similar to the following file structure:

  * `app.js` `cardCreation.js` `domMethods.js` `hover.js` `rating.js` Handles the creation

* Adjust the files in the `FILES TO CACHE` array within `public/service-worker.js` so that
the Webpack bundle is cached instead.

  * Run `npm start` and make sure that the application still works as expected.
```

Lazy Loading and Bundles

☝️ What is lazy loading?

☝️ If there is JavaScript specific to a part of the page a user is using, when do you think it should be downloaded?

☝️ Could deferring the downloading save us time on page load?

Gallery Lazy Loading JavaScript

* Run `npm install`.

* In `webpack.config.js`, add entry points for JavaScript files for the three pages, home, detail, and favorites.

* Update `service-worker.js` file so that it caches the new bundles.

* Make sure to update each html file so that it also uses the appropriate bundle.

* Note that the gallery application has been upgraded with the ability to save your favorite images to IndexedDb.

* Once again, there are many ways that you can separate your JavaScript files. It is recommended that you create somethings similar to the following file structure to avoid chunking unused code:

  * `api.js` Loads images from the api.

  * `cardCreation.js` Responsible for all functions related to the creation of cards.

- MORE IN READ ME

Mini-Project

```markdown
# PWA Mini-Project

In this activity we will take an existing news aggregator application and transform it into a PWA that can be installed on a user's device. We will also utilize webpack's minify and chunking features to help reduce the total size of the application.

## Instructions

* Open the [Unsolved](Unsolved) folder and install dependencies by running `npm install` at the project root.

* Start the app by running `npm start` from the project root.

* Once the app starts open your browser to [localhost:3000](http://localhost:3000).

* Open [index.js](Unsolved/assets/js/index.js).

* There are 3 main sections in this application:

  * A section that allows you to manage a list of topics.

  * A section that displays different articles of a given topic. This page will also allow you to save articles to your favorites.

  * A favorites page to view a list of the user's favorite articles. This page also allows the user to remove articles from their favorites.
```

Questions?