**BÁO CÁO 9: TÌM HIỂU WEBPACK VÀ JEST**

1. **Webpack**
2. **Webpack là gì?**

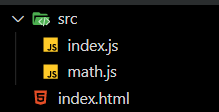
* Webpack là công cụ giúp compile các module Javascript. Nó hay được gọi là “module bundler”.

1. **Ưu điểm**

* Dành cho các dự án lớn, dễ dàng phát triển, quản lý và customize
* Tăng tốc độ cho project
* Phân chia các module và chỉ load khi thật sự cần
* Đóng gói tất cả các file nguồn thành 1 file duy nhất. Nhờ vào loader mà nó có thể biên dịch các loại file khác nhau.
* Biến các tài nguyên tĩnh (image, css,..) thành module.
* Biến đổi các mã nguồn: js, less, sass -> js, css or es6 -> es5 (nhằm hỗ trợ các trình duyệt)
* Đóng gói các module => sẽ giải duyết được các đường dẫn trong react
* JSX -> JS ( Browser sẽ không hiểu JSX nên webpack sẽ giúp chuyển đổi)
* Áp dụng Css preprocessor để xây dựng stylesheet (ví dụ: SASS)

1. **Cài đặt**
   1. **Khởi tạo các file cơ bản:**

* Đầu tiên, em sẽ init npm vào project và tạo ra các file như sau:

****

* File ***index.html***chỉ có head và body rỗng, đồng thời import file ***bundle.js*** mà lát nữa em sẽ tạo sau thông qua webpack.
* File ***math.js*** em sẽ viết một hàm cơ bản trả về 1 số và export nó ra:

const sum = (a, b) => {

    return a + b;

}

export default sum;

* File ***index.js*** sẽ import file ***math.js*** và sẽ đưa kết quả tính toán vào trong trang html:

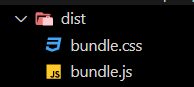
import sum from "./math";

const total = sum(3, 5);

document.write(total);

console.log(total);

* Bây giờ em sẽ cài đặt webpack và viết file config cho nó
  1. **Thiết lập file config**
* Em sẽ sử dụng các module sau:
  + Babel: dùng để chuyển đổi ES6, 7, 8,.. thành ES5
  + styleLoader, cssLoader, MiniCssExtractPlugin: tải và tách file css trong webpack
* Em sẽ tạo file tên là ***webpack.config.js*** trong đó sẽ chứa thông tin các cài đặt cho webpack, các module, plugins của webpack. Ở đây em sẽ tạo tên folder chứa file là *dist* và file được tạo ra sau khi chạy webpack là ***bundle.js***
* Em sẽ chạy lệnh webpack và nó sẽ tạo ra folder như sau



* Trong đó bundle.css là những cài đặt về css trong project và bundle.js là tất cả xử lý logic cũng như khai báo trong js của project.

1. **React với webpack**

* Tương tự với cách làm trên, em sẽ áp dụng webpack vào project em đã tạo với React sử dụng Ant Design.
* Để sử dụng được Ant Design, em sẽ thêm vào file ***.babelrc*** và ***webpack.config.js*** các khai báo như sau:
* ***.babelrc:***

{

  "presets": ["@babel/preset-env", "@babel/preset-react"],

  "plugins": [["import", { "libraryName": "antd", "style": true }]],

  "env": {

    "development" : {

      "compact": false

    }

  },

  "compact": true,

}

* ***webpack.config.js:*** em sẽ thêm plugin less loader để sử dụng được các file css trong Ant Design:

{

loader: "less-loader", // compiles Less to CSS

  options: {

   lessOptions: {

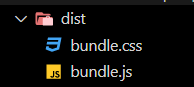
   javascriptEnabled: true,

   },

  },

},

* Sau khi chạy lệnh webpack em được folder như sau :



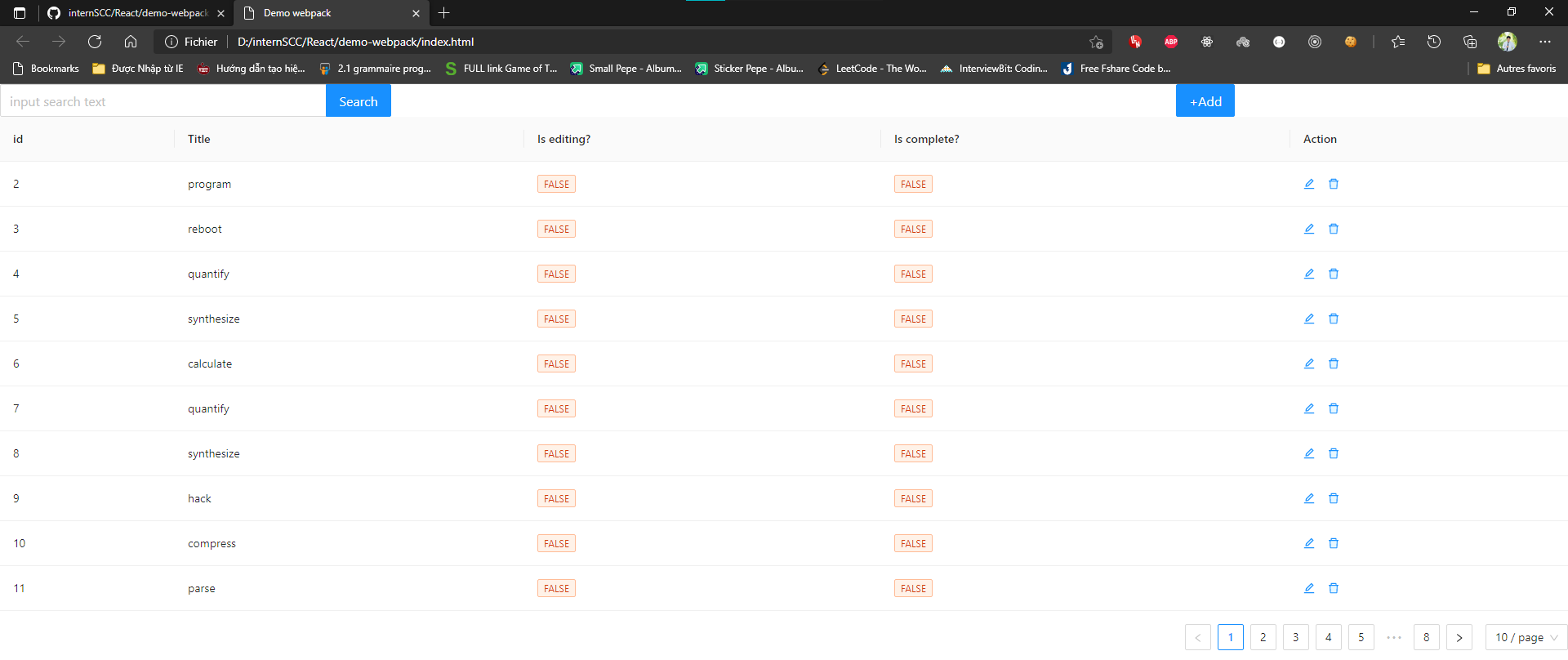
1. **Source code**

* Tất cả source code nằm trong link github sau:

[internSCC/React/demo-webpack at main · uytran36/internSCC (github.com)](https://github.com/uytran36/internSCC/tree/main/React/demo-webpack)

1. **Demo**

* Khi em mở file index.html, em đã có thể sử dụng web giống như chạy server



1. **Jest**
   1. **Giới thiệu**

* Jest là công cụ dùng để unit testing trong React. Em sẽ sử dụng công cụ này để test các component trong React.
* Em sẽ viết các test case với yêu cầu như sau:
  + Kiểm tra xem các component đã render đúng chưa
  + Trong form đăng nhập, mật khẩu có ít nhất 6 kí tự và tối đa 20 kí tự
* Em sẽ sử dụng Jest với Enzyme để test component tiện hơn
  1. **Cài đặt và sử dụng Jest với Enzyme**
* Em sẽ cài đặt các thư viện sau:
  + @testing-library/jest-dom
  + Enzyme
  + enzyme-adapter-react-16
  + @babel/preset-env
  + @babel/preset-react
* Em sẽ tạo thư mục *\_\_test\_\_* để chứa file test ***FormLogin.test.js***, đồng thời em sẽ tạo file ***babel.config.js*** và ***jest.config.js*** để khai báo các cài đặt
* Trong file ***FormLogin.test.js*** em sẽ tạo 3 test cho 3 yêu cầu ở trên:
  + Kiểm tra xem component đã render đúng chưa:

describe("Form login component", () => {

  it("should render form", () => {

    const wrapper = shallow(<FormLogin />);

    const form = wrapper.find(".form-login");

    expect(form.hasClass("login-form-button"));

  });

});

* + Mật khẩu có ít nhất 6 kí tự :

describe("Password min 6 chars", () => {

  it("should have min 6 chars", () => {

    const wrapper = mount(

      <StaticRouter>

        <FormLogin />

      </StaticRouter>

    );

    let password = wrapper.find("input#normal\_login\_password");

    password.simulate("change", {

      target: { value: "12345" },

    });

    password = wrapper.find("input#normal\_login\_password")

    expect(password.props().value).toEqual("12345");

  });

});

* + Mật khẩu có tối đa 20 kí tự :

describe("Password max 20 chars", () => {

  it("should have max 20 chars", () => {

    const wrapper = mount(

      <StaticRouter>

        <FormLogin></FormLogin>

      </StaticRouter>

    );

    let password = wrapper.find("input#normal\_login\_password");

    password.simulate("change", {

      target: { value: "1234512312312123123131231231231231" },

    });

    password = wrapper.find("input#normal\_login\_password");

    expect(password.props().value).toEqual(

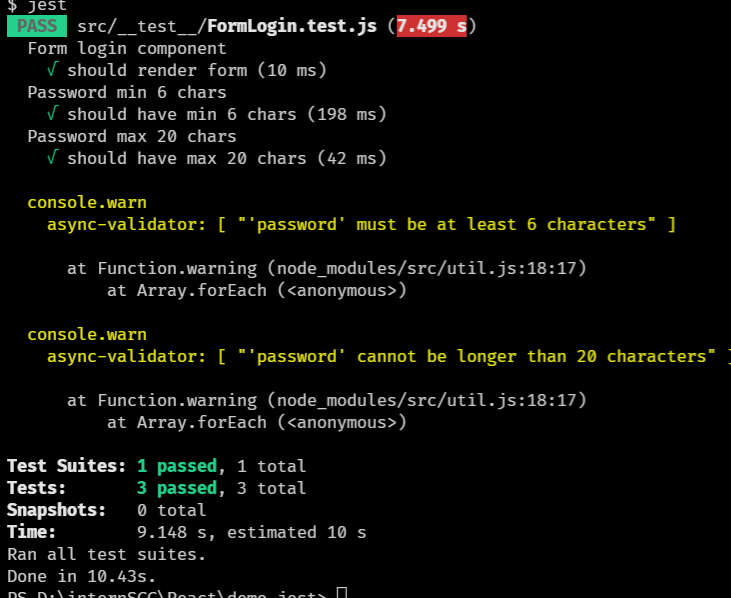
      "1234512312312123123131231231231231"

    );

  });

});

* 1. **Kết quả**

****