**COMPILING - TRANSPILING**

Contents

[1. **WEBPACK** 2](#_Toc77499831)

[ **Entry:** 2](#_Toc77499832)

[ **Output:** 2](#_Toc77499833)

[ **Loaders:** 2](#_Toc77499834)

[ **Plugins:** 2](#_Toc77499835)

[ **Mode:** 2](#_Toc77499836)

[ **Hash:** 2](#_Toc77499837)

1. **WEBPACK**

* **Entry:**
* the entry point is the module that webpack uses to start building its internal dependency graph.
* Thông thường sẽ trỏ đến src file.
* By default: ./src/index.js
* Thêm vendor file: có thể thêm **nhiều entry** vào dạng object(preferred)

module.exports = {

entry: {

main: './path/to/my/entry/file.js',

vendor: './vendor.js',

},

};

* Hoặc có thể thêm dạng array

module.exports = {

entry: ['./src/file\_1.js', './src/file\_2.js'],

output: {

filename: 'bundle.js',

},

};

* **Output:**
* the output property instructs webpack where to emit the bundle(s) and what name to use for the file(s).
* Đầu ra của webpack. Nơi để nó emit cái file bundle ra.
* By default: ./dist/main.js for the main bundle
* and ./dist for other generated files — such as images



* 2 props chính là filename, path
* **Filename**: tên file cần xuất ra
* **Path**: đường dẫn tuyệt đối và tên thư mục webpack sẽ bundle.
* **Loaders:**
* Là cái sẽ đc đưa vào **trước** khi compile 1 file (Pre-process).
* **Loaders** work at the individual file level **during** or **before** the bundle is generated
* Thường là sẽ cần loaders cho những file non-js. Và chuyển nó thành dạng module vd: Typescript language to JavaScript, inline images, import CSS files directly from our JavaScript
* Theo webpack: mún process thì tất cả các file là dạng module và phải là ngôn ngữ js
* Cách viết rules loaders trong webpack: vì tất cả là module nên có thể thấy là module: {} và bên trong sẽ là rules



* **use**: 1 array và là nơi khai báo các loaders cần thiết.
* **Note**: nó sẽ chạy kiểu pipeline **right to left / bottom to top**)
* **Use**: các loaders có thể có **options object**: use: [ {loaders:””, options: {...} }]
* **Plugins:**
* **plugins** are used for any other task that **loaders** can’t do (mạnh hơn loaders).
* **deeply integrate** into webpack because they can register hooks and access (and modify) the compiler
* **Plugins** có thể tích hợp sâu vào bất cứ quá trình nào khi bundle
* Vd: asset management, bundle minimization, optimization
* Các Plugins thông dụng:
  + HtmlWebpackPlugin (tạo file html khi bundle dựa trên [contentHash])
* **Mode:**
* Thường có 3 môi trường: **development, staging, production**.
* Trong webpack thì thông thường chỉ có 2: “**development”, “production”**
* This allows webpack to use built-in optimizations corresponding to each environment. Khi có define mode thì webpack sẽ dùng những cái có sẵn theo từng môi trường để tối ưu bundle.
* **Hash:**
* Dùng để cache lại các file đã build/bundle.
* Nếu có thay đổi trong file thì sẽ cache bursting (mất cache) và sẽ build lại file mới.
* **Square brackets / Template strings**
* Those are placeholders that Webpack will replace with their actual value
* Thường dành cho phần **output**
* **[hash]:** The hash of the module identifier.
* **[contenthash]:** hashes generated from the generated content
* **[chunkhash]**: The hash of the chunk content.
* **[name]**: The module name.
* **[id]**: The module identifier.
* **[query]**: The module query, i.e., the string following ? in the filename.



