What is ***Rule*** in ***Webpack***?

An array of Rules which are matched to requests when modules are created. These rules can modify how the module is created. They can apply loaders to the module, or modify the parser.

A Rule can be separated into three parts — ***Conditions***, ***Results*** and ***nested Rules***.

So our project is most likely about the conditions.

We have two kinds of conditions :

1. The resource: An absolute path to the file requested. It's already resolved according to the **resolve** rules.

   Resolve : Configure how modules are resolved. For example, when calling `import 'lodash'` in ES2015, the `resolve` options can change where webpack goes to look for `'lodash'`.

```
module.exports = {
  //...
  resolve: {
    // configuration options
  },
};
```

   we have **extension option** in our webpack.configure.js**,** which the files are realized by its own format.

```
module.exports = {
  output: {
    publicPath: "http://localhost:3000/",
  },

  resolve: {
    extensions: [".tsx", ".ts", ".jsx", ".js", ".json"],
  },
```

   which is what enables users to leave off the format when importing:

```
import File from '../path/to/file';
```

2. The issuer: An absolute path to the file of the module which requested the resource. It's the location of the import.

   Example: When we `import './style.css'` within ***app.js***, the resource is ***/path/to/style.css*** and the issuer is ***/path/to/app.js***.

   In a Rule the properties `test`, `include`, `exclude` and `resource` are matched with the resource and the property `issuer` is matched with the issuer.

   Example:

   In the example below, **style-loader** is applied a CSS file is captured through JavaScript:

```
const config = {
  test: /\.css$/,
  rules: [
    { issuer: /\.js$/, use: "style-loader" },
    { use: "css-loader" },
  ],
};
```

Add SVG:

Install

```
npm install --save-dev @svgr/webpack
# or use yarn
yarn add --dev @svgr/webpack
```

Usage

webpack.config.js

```
module.exports = {
  module: {
    rules: [
      {
        test: /\.svg$/i,
        issuer: /\.[jt]sx?$/,
        use: ['@svgr/webpack'],
      },
    ],
  },
}
```

Your code

```
import Star from './star.svg'

const Example = () => (
  <div>
    <Star />
  </div>
)
```

This method gives you the ability to use svg files as a component.

IF

you want to bring svg in an img tag, not as a component, it's like the other files in webpack:

```
{
  test: /\.(png|jpe?g|gif|mp4|mp3|svg)$/i,
  use: [
    {
      loader: 'file-loader',
    },
  ],
},
```

And :

```
import Svg from "./assets/svg.svg"
const App = () => {
  return (
    <>
      <img src={Svg} alt="" />
```

Sass in webpack:

First, install Sass in your project:

`npm i sass`

Second, install sass-loader

```
npm install sass-loader sass webpack --save-dev
```

Third:

On your webpack.configure.js there's a mistake by default:

```
{
  test: /\.(css|s[ac]ss)$/i,
  use: ["style-loader", "css-loader", "postcss-loader"],
},
```

Just remove scss from this test, it makes a conflict error in webpack. Although for sass you need to apply sass-loader, which is totally separate from it and we do it in the next step:

Fourth:

Add this to your webpack.config.js

```
  rules: [
    {
      test: /\.s[ac]ss$/i,
      use: [
        // Creates `style` nodes from JS strings
        "style-loader",
        // Translates CSS into CommonJS
        "css-loader",
        // Compiles Sass to CSS
        "sass-loader",
      ],
    },
  ],
},
```

Fifth:

Go back to your project. Make a .scss file and do something there:

```
.test {
    color: #f01321;
}
```

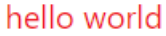
Import it on your project:

```
import "./scss/main.scss";

const App = () => {
  return (
    <>

      <p className="test">
        hello world
      </p>
```

And you have the result:



Good luck !