6/9.

npm = maven ?   libraries, development tools.

Node. Js , npm (node package manager)
  libraries / frame works
  development tools

ES6 $\xrightarrow[\text{BABEL}]{}$ ES5.   npm manage all these tools
  in command line interfaces

ES6 modules $\xrightarrow[\text{webPack.}]{}$ Bundle

Command Line.
  copy NUL test.js. / touch test.js.
  del test.js / rm test.js.
    for deleting folders : rm -r ⇐ recursively.   MAC
    ~~for deleting~~   rmdir /S test   Windows
  Start a file : {
    windows : start index.html
    mac : open index.html

Node.js   nodejs.org.

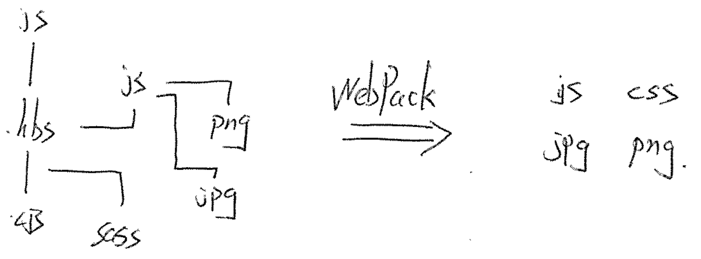
- npm init
- npm install webPack. --save-dev (save webPack as development pendency of our project)
- npm install jquery --save.

✱ -npm install will automatically install all the dependencies in package.json.

All the dependencies are stored in the node-modules folder.

-- npm uninstall jquery --save : delete packages.

-- npm install live-server --global : make the package accessible from all projects.



```
webpack.config.js
    module.exports = {
        entry : './src/js/index.js',    -- file(s) we want to bundle.
        output: {
                path.resolve(_dirname, 'dist/js'),
                filename: 'bundle.js'

        }
    };
```

npm script → package.json "scripts": { "dev": "webpack --mode development" }
~~put~~ "build": "webpack --mode production"

-- npm install webpack-cli --save-dev (webpack command line interface).

-- npm run dev // -- npm run build ⟸ the compressed code will be much cleaner.

auto - deployment
① -- npm install webpack-dev-server . --save-dev .

②. webpack.config.js ⟶ ~~devServer~~ module.exports = {
    devServer : {
        contentBase : './dist'
    }.

dist folder is like class, while src folder is just src.

③ package.json ⟶ "scripts" : { "start" : "webpack-dev-serve
    --mode development
    -- open .
    open browser automatically.

④ npm run start

auto-deployment is not working because everything is simply read from bundle.js . The devServer's contentBase should match the output path, which was dist/js before

move src/index.html (with no script tag) to dist and make the script injected automatically.

① --npm install html-webpack-plugin
② webpack.config.js   * Just Don't forget to import html-webpack-plugin

```
plugins: [
    new HtmlWebpackPlugins ({
        filename: 'index.html',
        template: './src/index.html'
    })
]
};
```

If I want to create files in dist, I need to use dev or build ?
command.
Anyway, these commands are defined in package.json → "scripts".

Babel. (Pay attention to the version)
① - npm install babel-core babel-preset-env. babel-loader --save-dev

② webpack.config.JS → module: {
```
                    rules: [
                        { test: /\.js$/,        this will filter all the JS files
exclude: /node-modules/,
                          use: {
                            loader: 'babel-loader'
                          }
                        }
                    ]
                  }
                 }.
```

③ create a babel config file .babelrc
```
    {
      "presets": [      piece of code apply actual transformations to the code.
        ["env", {
                 "targets": { "browsers: [
                              "last 5 versions", "ie >= 8"
                         ]]
                }
```
However, something like promise cannot be converted to ES5, so we need package babel-polyfill.

⊕ - npm install babel-polyfill . --save

This won't be a dev tool but real code goes into bundle .

webpack . config. js
entry : ['babel-polyfill',]

MVC Architecture.

Model Controller View

$$\begin{tikzcd} \text{Search.js} \arrow[r, bend right] & \text{index.js} & \text{searchView.js} \arrow[l, bend left] \end{tikzcd}$$

data & logic

display.

Recipe.js

List.js

Likes.js

recipeView.js

listView.js

likesView.js

Its always the case to make the model js file an upper case.

export can be anything like variables, even functions, however, you have to use the same name as import.

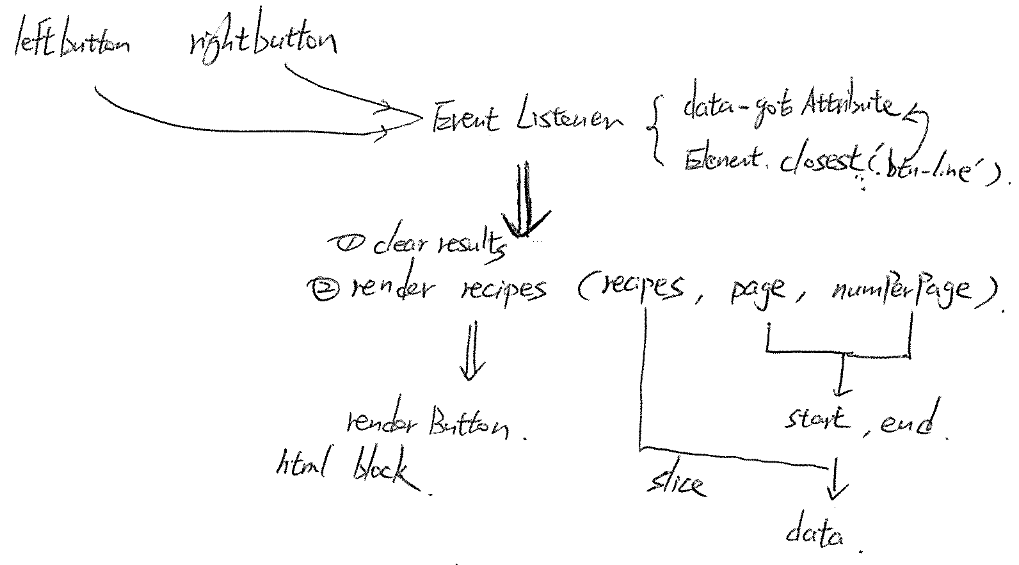
when we want to export one thing from a module:

export default '---';

In this case, you can give any name in import.

e.g. (If I want to use a different name).
import { add as a, multiply as m, ID } from './views/searchView'.

import * as searchView from './Views/searchView' pagination logic



All attributes in a tag should be surrounded with " ".

Ingredients.
Read the list of ingredients, parse each ingredient into the count unit and description.
new method: parseIngredients. { Uniform units
Remove parenthes
parse ingredients

For testing, we can just put some variable into window:
window.r = state.recipe;
Then we can access the recipe even if it is not exposed yet

*after replace, it must be assigned to the ~~value~~ variable.

parse Ingredient :

split with ' ' ⟶ findIndex(el2 => unitsShort.includes(el2));

if this condition doesn't match, it will return -1.

```
objIng = {
    count: parseInt(xx, 10),
    unit: '',
    ingredient: arrIng.slice(1).join(' ')   ① for No unit but
}                                              first element being number.
    ingredient (automatically creates ingredient: ingredient in ES6)
                                ② for No unit, no number.
```

③ "4 1/2 cup ~~of~~ water"

4.5 is what I want.   [4, 1/2, cup, water].slice[0, unitInd

⇓

eval([4, 1/2].join("+"))

eval will recognize "4+1/2" as code

⇓

4.5.

Likes. model. { [].
add, delete, isliked, getlikedNum.

controller. Recipe Area event handler.
matches. ('.recipe__love, .recipe__love *').

① states like exists ?
② curent recipe is liked ?
  No add. toggle button, UI.
  Yes delete. toggle button, UI

like Views
① toggle button button – svg — use href = "---"
  change the icon
setAttribute ('href', iconstring)

② when we come back ~~from~~ to an liked recipe, button should be liked.
likes.id is recipe's id ✓.
add isliked(id) as an argument to the render recipe function
and decides the specific part in string template

③ likes menu.
style. visibility. = visiable.

④. a[href*="${id}"]
* means all.

~~If~~ It is always a good idea to test if queryselector is successful