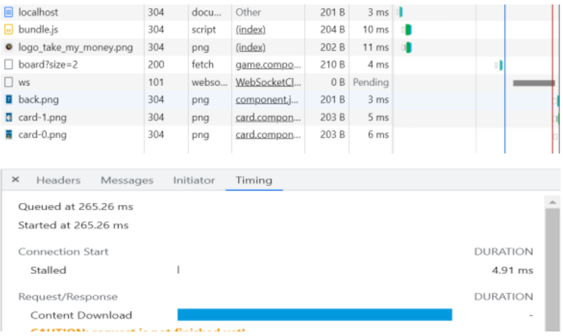
Rapport de Application Design

LI HAO - MIN2 - EPF

1. Play the whole game with size=2. By browsing the 3 views of the application, how many files did your browser download overall? How many time did it took to load them all?



localhost	304	docu...	Other	201 B	3 ms
bundle.js	304	script	(index)	204 B	10 ms
logo_take_my_money.png	304	png	(index)	202 B	11 ms
board?size=2	200	fetch	game.compo...	210 B	4 ms
ws	101	webso...	WebSocketCl...	0 B	Pending
back.png	304	png	component.j...	201 B	3 ms
card-1.png	304	png	card.compon...	203 B	5 ms
card-0.png	304	png	card.compon...	203 B	6 ms

Queued at 265.26 ms
Started at 265.26 ms

Connection Start — DURATION
Stalled — 4.91 ms

Request/Response — DURATION
Content Download — -

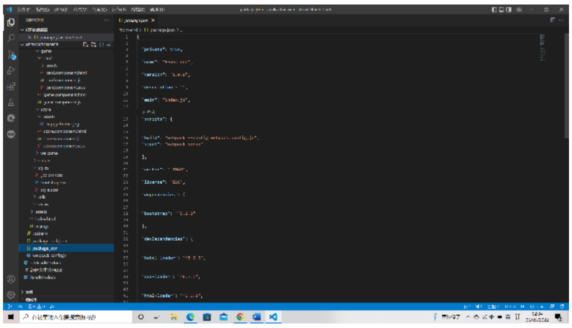
CAUTION: request is not finished yet!

a. It created 8 files in my browser download.
b. All loading needs to take 4.91 ms.

2. Component-oriented programming for the web is considered more maintainable. Why?

I think every module of the project only need to maintain one component. Because it is easier to pack. By the way, It is also more convenient if it is called in other. For example, I only use game.component.js to load in other chunks in the program. So that's why I think the web is considered more maintainable. It is more comfortable to establish a logical relationship between data between data.

3. As you can see, npm install command also generated a package-lock.json file along with package.json. What is the purpose of this file?



```
{
  "private": true,
  "main": "index.js",
  "scripts": {
    "build": "webpack --config webpack.config.js",
    "start": "webpack serve"
  },
  "license": "ISC",
  "dependencies": {
    "bootstrap": "5.1.3"
  },
  "devDependencies": {
```

a. Record the dependencies between the module of game
b. The version of the lock -up package
c. Speed up the download speed of the reinstalled installation.

For example:

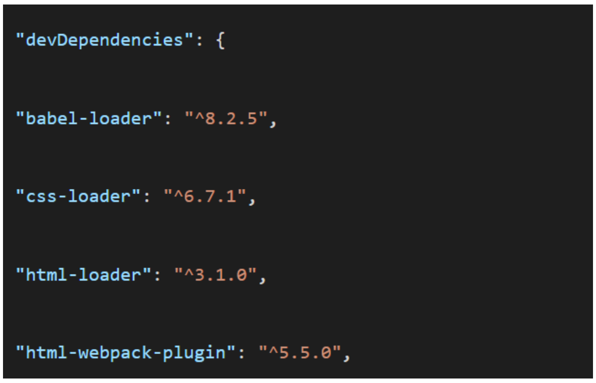
```
dependencies
```

This detail can set the parameters of the entire application

It can create a Dependency package and it can get rid of file-to-file dependencies.

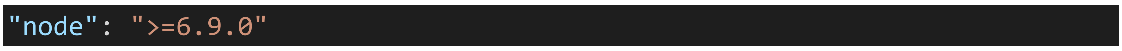
The purpose of this file tell me that which frameworks are externally dependent on.

4. By convention, all NPM dependencies use a 3-digit format for version numbers. How do you call this?



```
"devDependencies": {

"babel-loader": "^8.2.5",

"css-loader": "^6.7.1",

"html-loader": "^3.1.0",

"html-webpack-plugin": "^5.5.0",
```

I think ˆ will match the latest major version of the dependencies.
For example, ˆ3.1.0 will match all 3.x.x packages, but not 4.0.0.



```
"node": ">=6.9.0"
```

>= means (greater than or equal to)
It must be greater than or equal to this version number

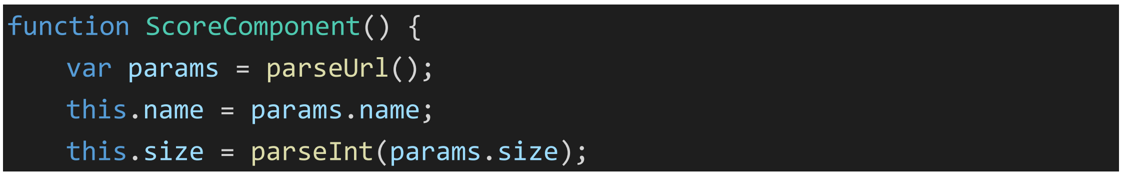
5. What is a devDependency exactly? What are the differences with a dependency?

Difference: devDependency is the development environment but depDependency is the production environment.
The package of devDependencies are what we depend on to run the code locally or in the development environment and it only needed for development and testing.

6. Can you think of at least 2 things that are possible with ES6 classes, but cannot be done with Java classes?

- fetch
- Promises

7. What are the differences between var and let?

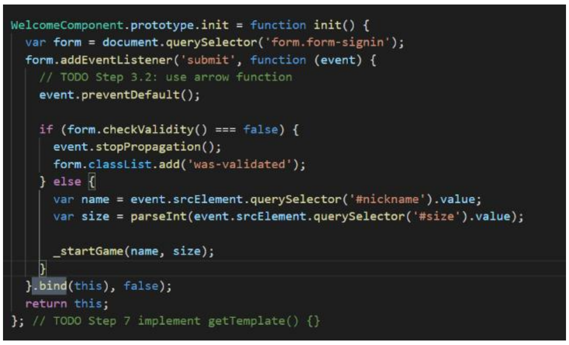


```
function ScoreComponent() {
    var params = parseUrl();
    this.name = params.name;
    this.size = parseInt(params.size);
```

var is accessible in the current (global) scope and descendant scopes, while let is only accessible in the current scope.

For example, Var is declared in the function and the entire function is effective. But Let is the scope of the block, if the variable defined in the block, such as in the for loop, it cannot be accessed outside.

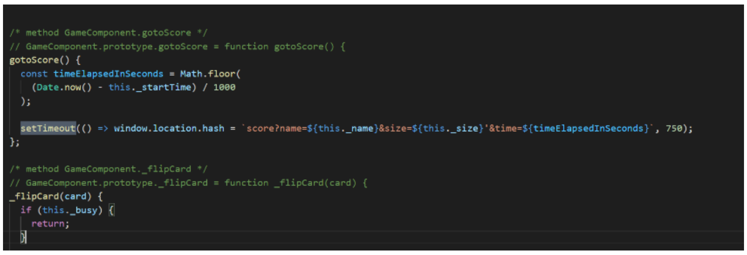
8. What is the .bind(this) stuff? What does happen if you delete it? Is it needed when using an arrow function ?



```
WelcomeComponent.prototype.init = function init() {
  var form = document.querySelector('form.form-signin');
  form.addEventListener('submit', function (event) {
    // TODO Step 3.2: use arrow function
    event.preventDefault();

    if (form.checkValidity() === false) {
      event.stopPropagation();
      form.classList.add('was-validated');
    } else {
      var name = event.srcElement.querySelector('#nickname').value;
      var size = parseInt(event.srcElement.querySelector('#size').value);

      _startGame(name, size);
    }
  }.bind(this), false);
  return this;
}; // TODO Step 7 implement getTemplate() {}
```

a. It can create a new function. When bind () is called, this is specified as the first parameter of Bind (), and the remaining parameters will be used as a parameter of the new function for use.
b. If I don't write .bind(this) I won't be able to get the relevant images loaded in the game
c. Arrow functions do not need to bind this

9. What are the advantages of Promises?

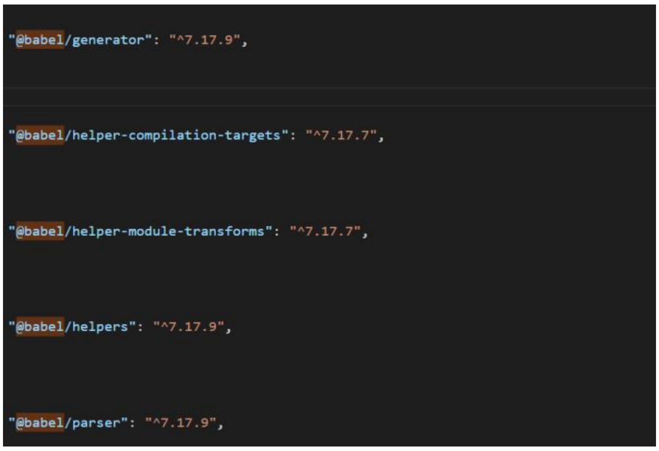


```
/* method GameComponent.gotoScore */
// GameComponent.prototype.gotoScore = function gotoScore() {
gotoScore() {
  const timeElapsedInSeconds = Math.floor(
    (Date.now() - this._startTime) / 1000
  );

  setTimeout(() => window.location.hash = `score?name=${this._name}&size=${this._size}'&time=${timeElapsedInSeconds}`, 750);
};

/* method GameComponent._flipCard */
// GameComponent.prototype._flipCard = function _flipCard(card) {
_flipCard(card) {
  if (this._busy) {
    return;
  }
```

In the original code of game, there are multiple requests in some places, and the latter request requires the return result of the last request. Such code readability and maintenance are not good.
1. I can specify the callback function after the asynchronous operation is completed
2. The code will become flat and read more comfortable.
3. It can capture better error.

10. What version of ECMAScript async / await was released in?
ECMAScript 8
It was released in June 2017.

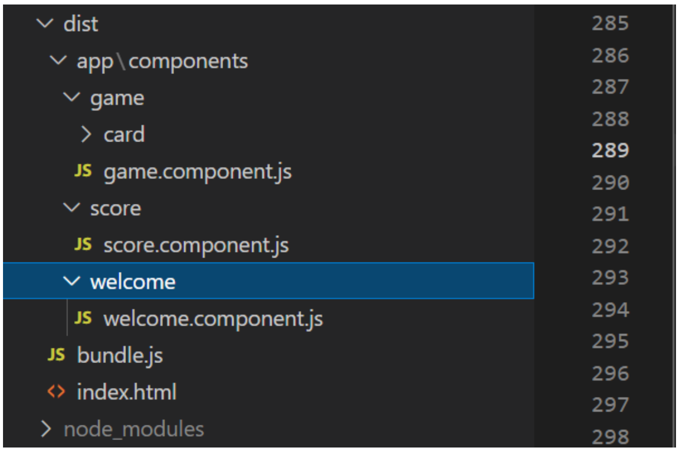
11. What does the @ symbol mean in @babel/***?



```
"@babel/generator": "^7.17.9",

"@babel/helper-compilation-targets": "^7.17.7",

"@babel/helper-module-transforms": "^7.17.7",

"@babel/helpers": "^7.17.9",

"@babel/parser": "^7.17.9",
```

@Is a scope area, it can also be said that the owner of this package can be prevented from occupying the package in advance.
For example, I want to define this thing as rooter and I want to refer to the method of this framework, I need to use @ symbol

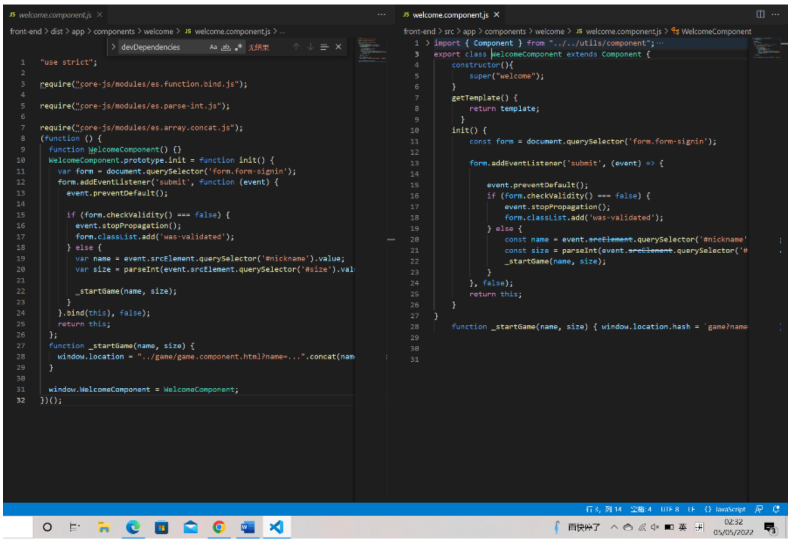
12. Look at the files produced within dist/ folder. How babel transpile your class WelcomeComponent?



```
∨ dist
  ∨ app\components
    ∨ game
      > card
      JS game.component.js
    ∨ score
      JS score.component.js
    ∨ welcome
      JS welcome.component.js
  JS bundle.js
  <> index.html
> node_modules
```

It will replace the class with a method and also replace the welcome component class with a function constructor. Then the constructor contains all the methods of this class

13. What is the weight of the transpiled sources compared to your original sources?



```
front-end > dist > app > components > welcome > welcome.component.js > ...
"use strict";

require("core-js/modules/es.function.bind.js");

require("core-js/modules/es.parse-int.js");

require("core-js/modules/es.array.concat.js");
(function () {
  function WelcomeComponent() {}
  WelcomeComponent.prototype.init = function init() {
    var form = document.querySelector('form.form-signin');
    form.addEventListener('submit', function (event) {
      event.preventDefault();

      if (form.checkValidity() === false) {
        event.stopPropagation();
        form.classList.add('was-validated');
      } else {
        var name = event.srcElement.querySelector('#nickname').value;
        var size = parseInt(event.srcElement.querySelector('#size').val

        _startGame(name, size);
      }
    }.bind(this), false);
    return this;
  };
  function _startGame(name, size) {
    window.location = "../game/game.component.html?name=...".concat(nam
  }

  window.WelcomeComponent = WelcomeComponent;
})();
```

```
front-end > src > app > components > welcome > welcome.component.js > WelcomeComponent
import { Component } from "../../utils/component"; ...
export class WelcomeComponent extends Component {
    constructor(){
        super("welcome");
    }
    getTemplate() {
        return template;
      }
    init() {
        const form = document.querySelector('form.form-signin');

        form.addEventListener('submit', (event) => {

            event.preventDefault();
            if (form.checkValidity() === false) {
                event.stopPropagation();
                form.classList.add('was-validated');
            } else {
                const name = event.srcElement.querySelector('#nickname'
                const size = parseInt(event.srcElement.querySelector('#
                _startGame(name, size);
            }
        }, false);
        return this;
    }
}
    function _startGame(name, size) { window.location.hash = `game?name
```

By comparison, the content of the code has increased a lot

Mainly added the guide package, and then replaced the class with the method through the steps of the function

14. What is the difference between import * from './utils' and import { parseUrl } from './utils'?

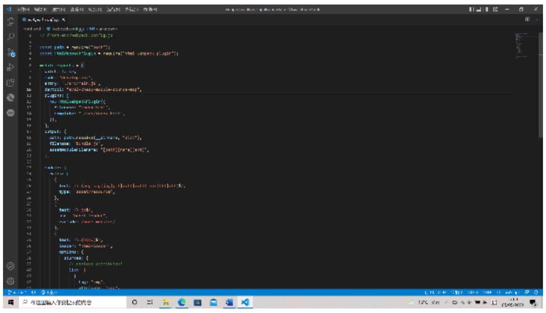
In the case of not using {} to refer to the module, the name of the import module is arbitrary.

After explicitly declaring the naming and exporting, when using {} to refer to the module in another js, the module naming during import is meaningful.

15. Why the utils.js will also be transpiled?

Because when the user inputs, it is automatically corrected.

16. What does the webpack -- config webpack.config.js do?



```
// front-end/webpack.config.js

const path = require("path");
const HtmlWebpackPlugin = require("html-webpack-plugin");

module.exports = {
  [illegible]
  [illegible]
  entry: "./src/main.js",
  devtool: "eval-cheap-module-source-map",
  plugins: [
    new HtmlWebpackPlugin({
      filename: "index.html",
      template: "./src/index.html",
    }),
  ],
  output: {
    path: path.resolve(__dirname, "dist"),
    filename: "bundle.js",
    assetModuleFilename: "[path][name][ext]",
  },

  module: {
    rules: [
      {
        test: [illegible]
        type: "asset/resource",
      },
      {
        test: /\.js$/,
        [illegible]
        [illegible]
      },
      {
        test: /\.(html)$/,
        loader: "html-loader",
        options: {
          sources: {
            // replace attributes!
            list: [
              {
                tag: "img",
```

Webpack.config.js is the default configuration file packed by Webpack, which is in the root directory of the Memory project.

For example, if I enter the webpack instruction in the console and do not follow any parameters, the webpack defaults to find the webpack.config.js file in the root directory, and load and perform the corresponding dependencies according to the file.

17. Can you guess how exactly style-loader works exactly?

I think style-loader helps me directly mount the content parsed by css-loader into the html page.

18. What does the _ prefix means on a sass file?

I feel like _ it is possible to make many files to organize the code and everything will be compiled under the same file.