

Vue3 工作坊

王孝弘

wang@ispan.com.tw

資展國際股份有限公司

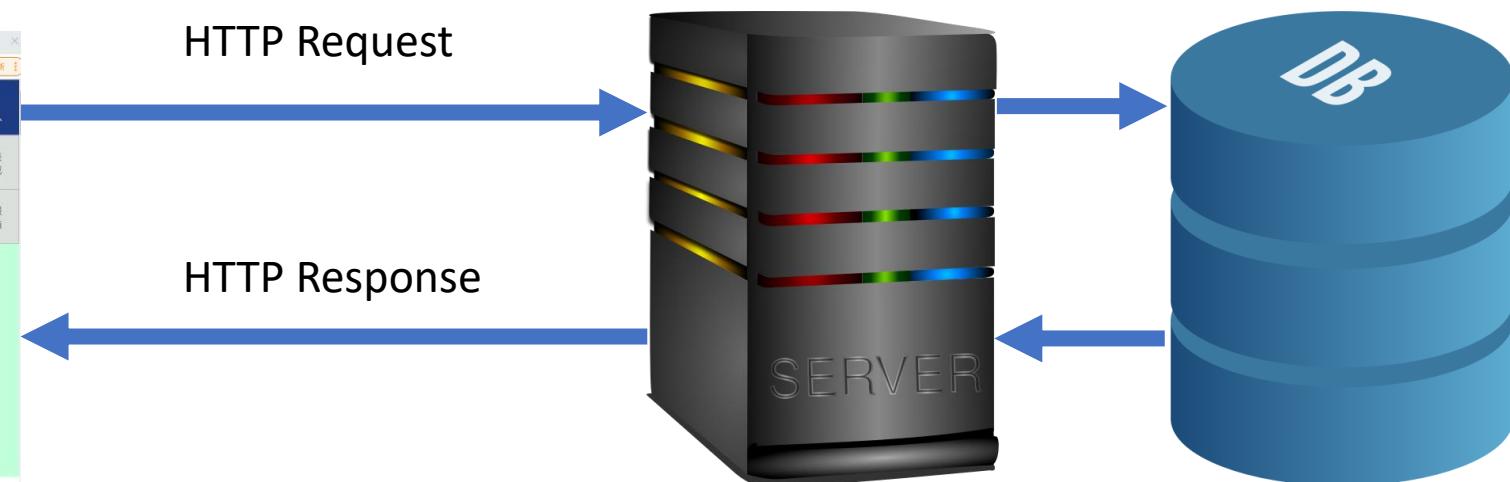
網頁開發架構

瀏覽器

<https://www.ispan.com.tw/>



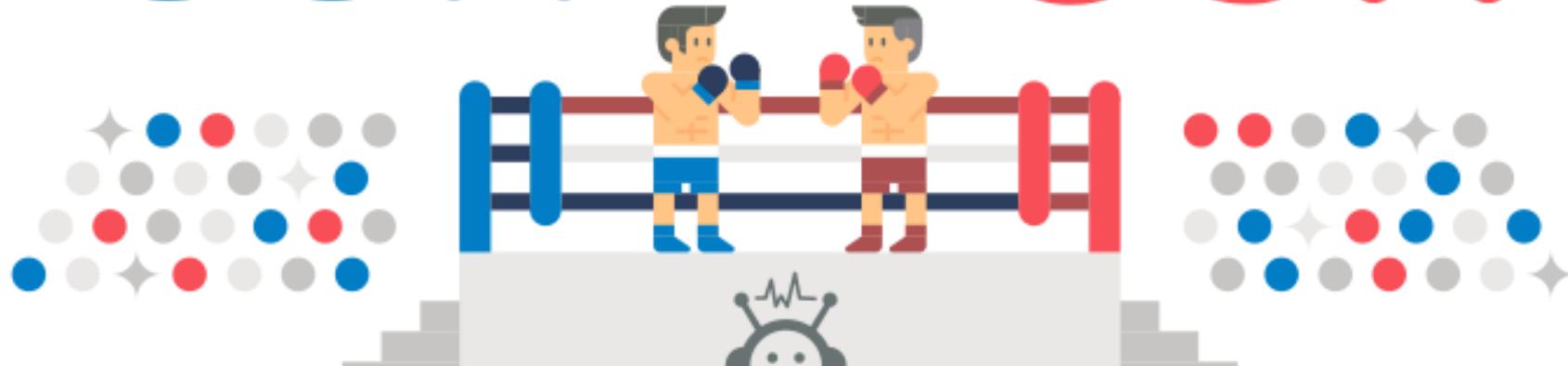
HTTP Request



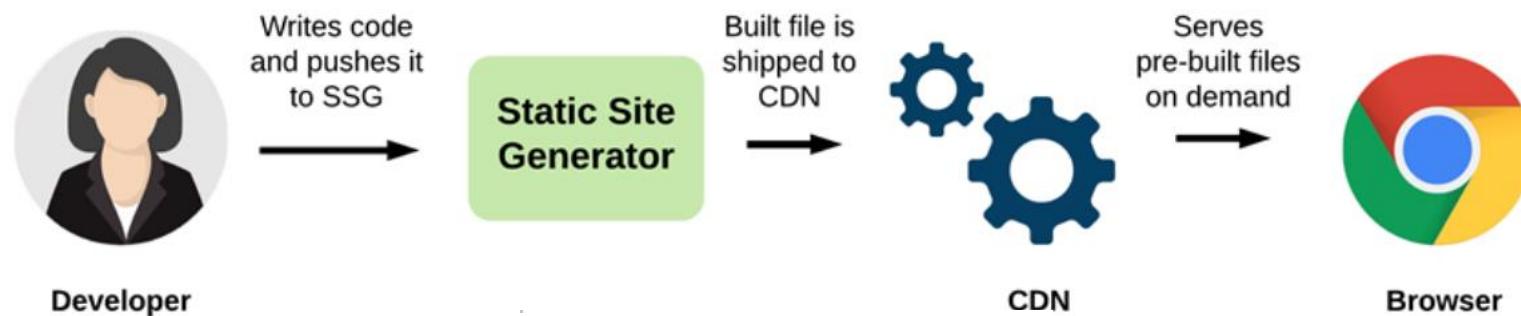
網站伺服器

資料庫

SSR vs CSR



圖片來源：<https://medium.com/walmartglobaltech/the-benefits-of-server-side-rendering-over-client-side-rendering-5d07ff2cefe8>

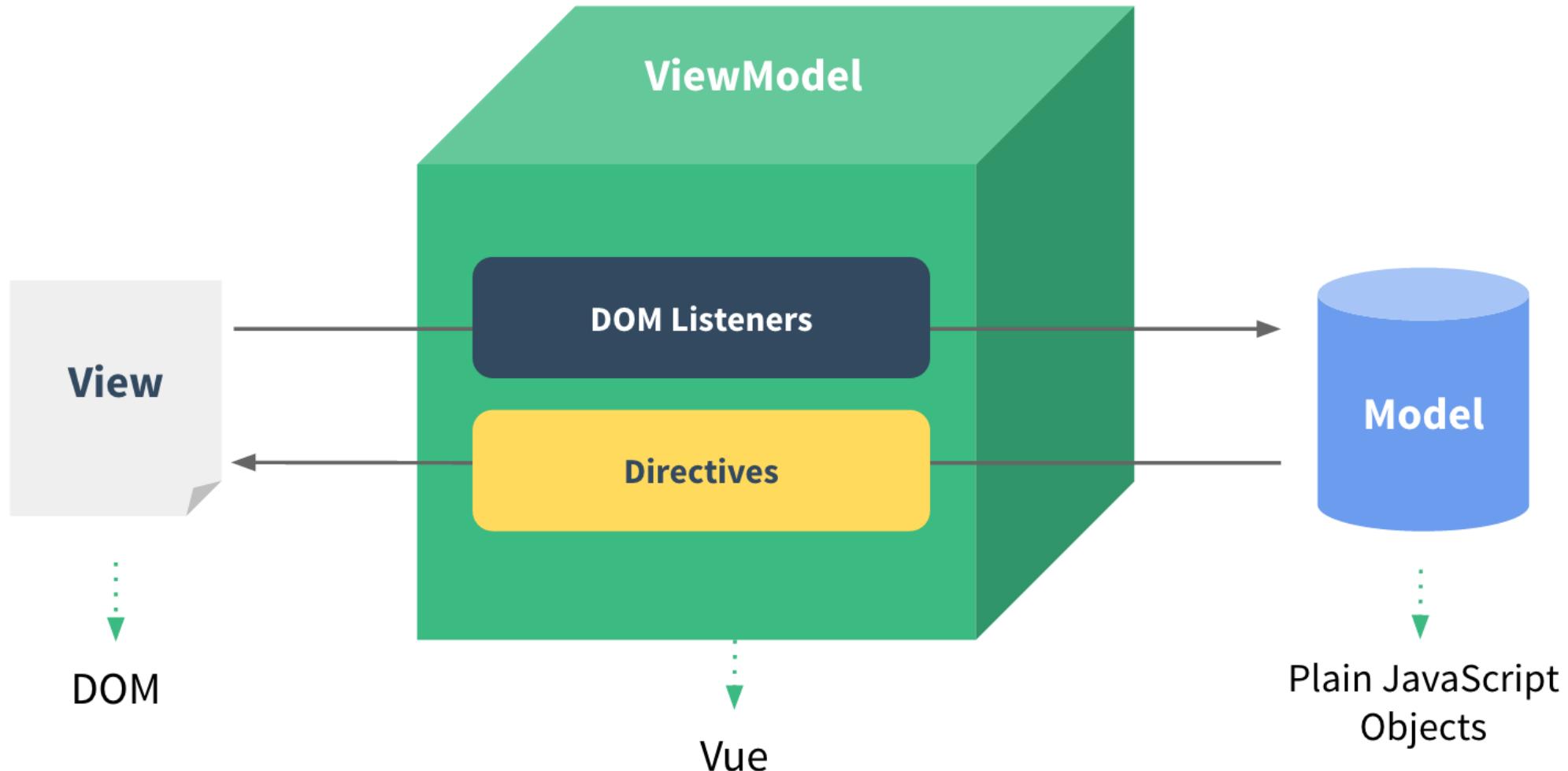


圖片來源：<https://www.partech.nl/nl/jamstack#>

SSG

ack-series-part-1-of-4-basics-of-

MVVM



Single Page Application

Single Page Applications Work Differently

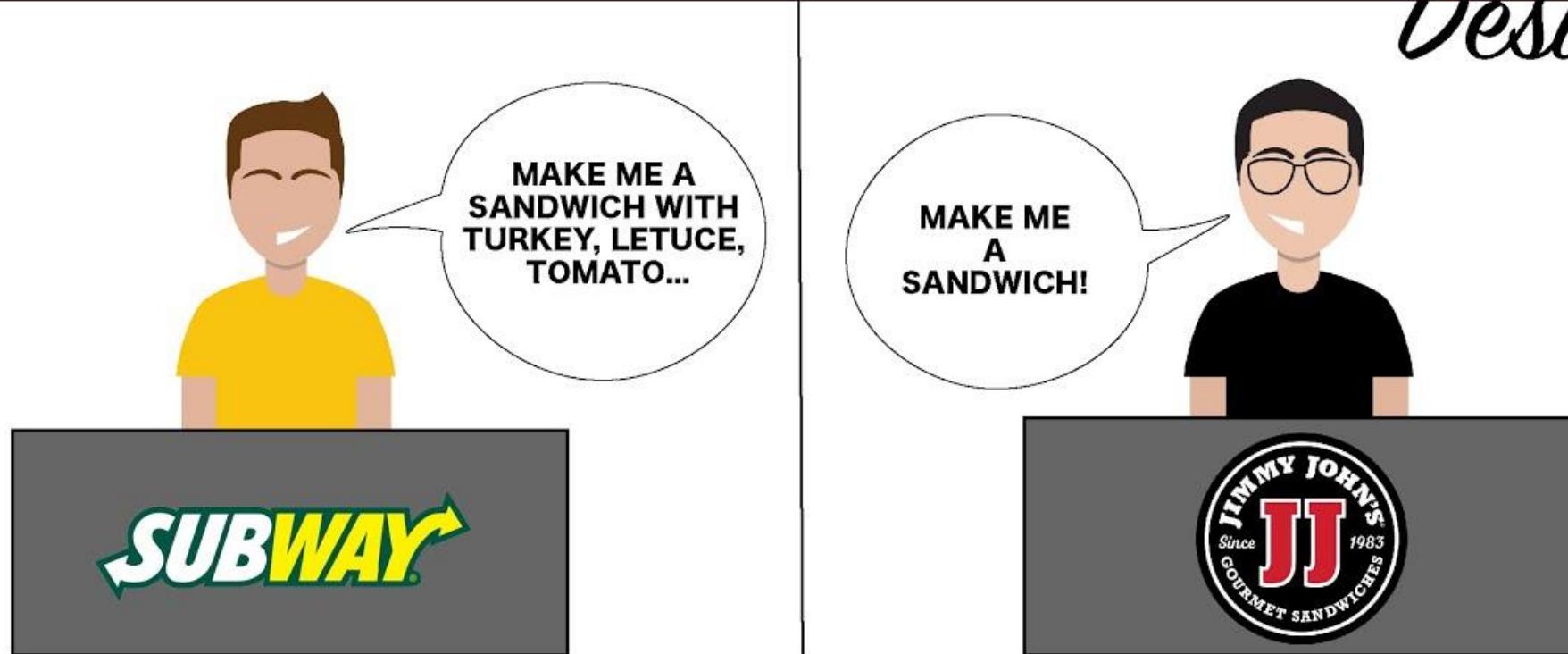


圖片來源：<https://www.excellentwebworld.com/what-is-a-single-page-application/>

Musings of

IMPERATIVE AND DECLARATIVE

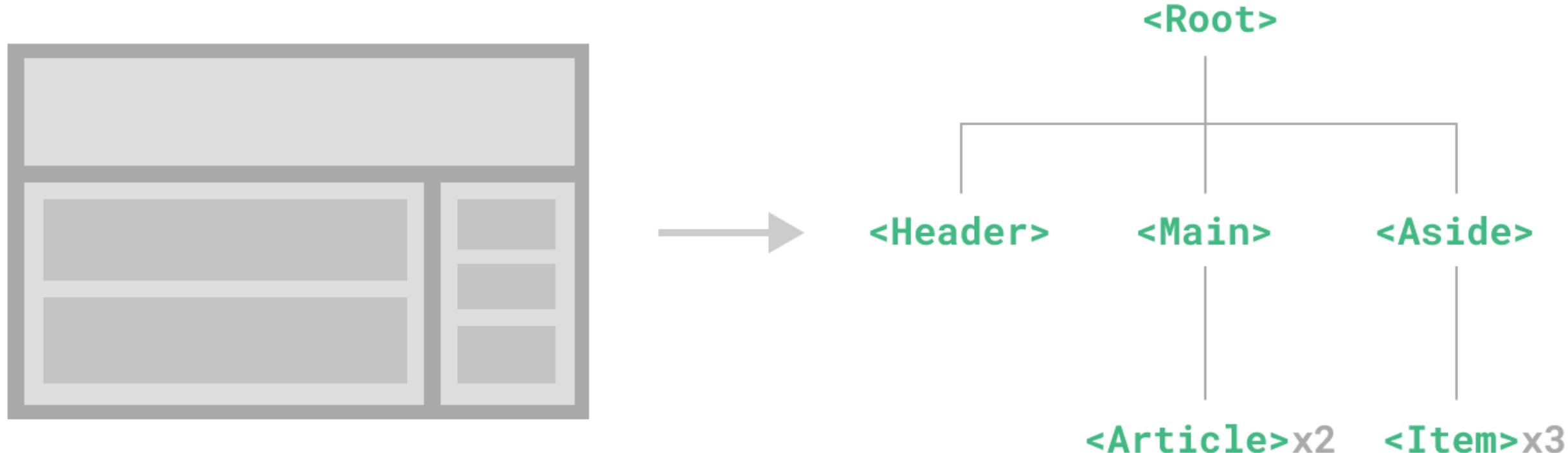
Design



圖片來源：https://www.youtube.com/watch?v=Xn5YVwTVC_I

Single File Component(SFC)

元件、組件



Vue3 生态进展和计划@尤雨溪_VueConf CN 2021



https://www.youtube.com/watch?v=On6V97n_iEE

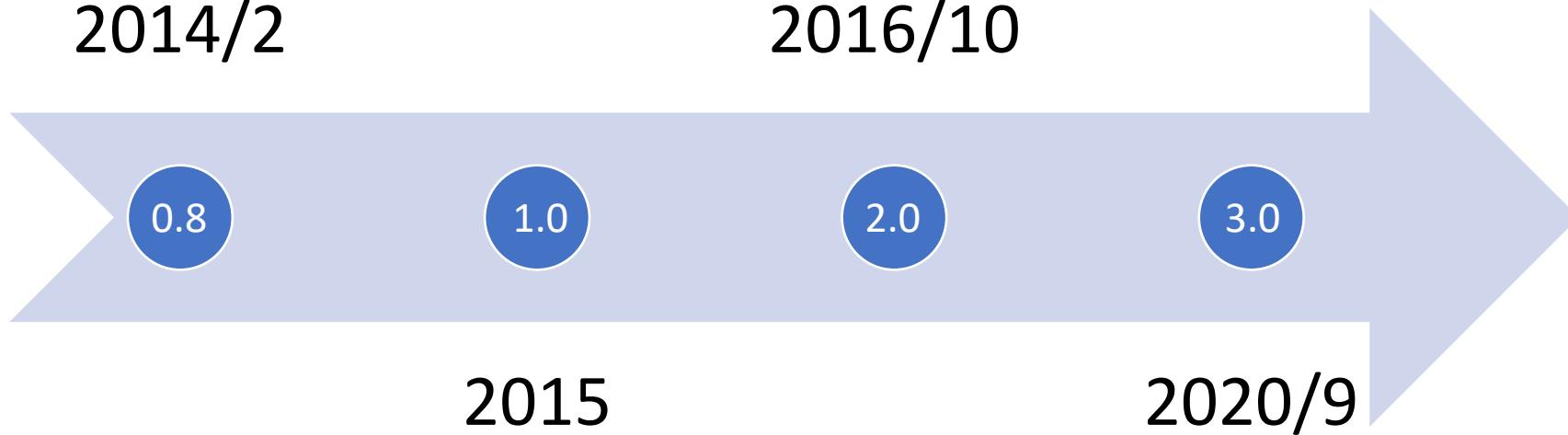
<https://vuejs.org/>

The Progressive JavaScript Framework

An approachable, performant and versatile framework for building web user interfaces.

2014/2

2016/10



Vue 簡單用

- 在 head 標籤中引用 Vue 的 CDN

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

- 設定 Vue 要控制的 HTML 範圍

```
<div id="app">  
</div>
```

Vue 與 HTML 的綁定

- 透過 createApp 函式(function)建立 Vue 的物件
- 使用mount()方法，綁定 Vue 管理的 HTML 範圍

```
const { createApp } = Vue
const app = createApp({
  //Options api(Vue2 / Vue3) 或 Composition API(Vue3)
})
app.mount('#app')
```

Options api

Options: State

data
props
computed
methods
watch
emits
expose

Options: Rendering

template
render
compilerOptions

Options: Lifecycle

beforeCreate
created
beforeMount
mounted
beforeUpdate
updated
beforeUnmount
unmounted
errorCaptured
renderTracked
renderTriggered
activated
deactivated

Options: Misc

name
inheritAttrs
components
directives

Component Instance

\$data
\$props
\$el
\$options
\$parent
\$root

官方文件：<https://vuejs.org/api/>

Composition API

setup()

- Basic Usage
- Accessing Props
- Setup Context
- Usage with Render Functions

Reactivity: Core

- ref()
- computed()
- reactive()
- readonly()
- watchEffect()
- watchPostEffect()

Reactivity: Utilities

- isRef()
- unref()
- toRef()
- toRefs()
- isProxy()
- isReactive()
- is_READONLY()

Reactivity: Advanced

- shallowRef()
- triggerRef()
- customRef()

Lifecycle Hooks

- onMounted()
- onUpdated()
- onUnmounted()
- onBeforeMount()
- onBeforeUpdate()
- onBeforeUnmount()
- onErrorCaptured()
- onRenderTracked()
- onRenderTriggered()
- onActivated()
- onDeactivated()
- onServerPrefetch()

官方文件：<https://vuejs.org/api/>

試試看

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<div id="app"> {{ message }} </div>
<script>
const { createApp } = Vue
const app = createApp({
  data() {
    return {
      message: 'Hello Vue!'
    }
  }
})
app.mount('#app')
</script>
```

{} Mustache 兩個大括號

{} 裡面的內容表示一段會回傳結果的 JavaScript 程式碼



Hello Vue!

Vue 初體驗 - todos

Todos

想要做甚麼？

作業 1

X

作業 2

X

作業 3

X

尚有 3 個工作未完成

清除完成工作

範例參考：<https://todomvc.com/>

todos 資料顯示 - html

● 讀取陣列、物件中的資料，使用 v-for Directives

```
<ul class="list-group mt-3">
  <li v-for="todo in todos" class="list-group-item">
    <div class="d-flex justify-content-between">
      <div>
        <input class="form-check-input me-3" type="checkbox">
        <label class="form-check-label" for="firstCheckbox"> {{todo.title}} </label>
      </div>
      <button class="badge bg-danger rounded-pill border-0">X</button>
    </div>
  </li>
</ul>
```

Directives

- v-text
- v-html
- v-show
- v-if
- v-else
- v-else-if
- v-for
- v-on
- v-bind
- v-model
- v-slot
- v-pre
- v-once
- v-memo
- v-cloak

Directives文件參考：<https://vuejs.org/api/built-in-directives.html>

todos 資料顯示 - vue

```
<script>
  const { createApp } = Vue
  const app = createApp({
    data() {
      return {
        todos: [
          { "id": 1, "title": "作業1", "completed": false },
          { "id": 2, "title": "作業2", "completed": false },
          { "id": 3, "title": "作業3", "completed": false }],
      }
    }
  })
  app.mount('#app')
</script>
```

todo 資料新增 - html

● 雙向繫節 v-model、事件綁定 v-on(@)

```
<input type="text" v-model="newTodo" placeholder="想要做甚麼?" @keyup.enter="addTodo">
```

- .enter
- .tab
- .delete
- .esc
- .space
- .up
- .down
- .left
- .right

v-on:submit.prevent="onSubmit"



Name

Starts with v-
May be omitted when using shorthands

Argument

Follows the colon or shorthand symbol

Modifiers

Denoted by the leading dot

Value

Interpreted as JavaScript expressions

modifiers 文件參考：<https://vuejs.org/guide/essentials/event-handling.html#key-modifiers>

Directives
v-text
v-html
v-show
v-if
v-else
v-else-if
v-for
v-on
v-bind
v-model
v-slot
v-pre
v-once
v-memo
v-cloak

todo 資料新增 - vue

```
data() {
  return {
    todos: [{ "id": 1, "title": "作業1", "completed": false }, { "id": 2, "title": "作業2", "completed": false },{...}],
    newTodo: ""
  }
},
methods: {
  addTodo() {
    const value = this.newTodo && this.newTodo.trim();
    if (!value) {
      return;
    }
    this.todos.push({ "id": this.todos.length + 1, "title": value, "completed": false });
    this.newTodo = "";
  }
},
```

todo 資料刪除 – html & vue

● 事件綁定 v-on(@) Directives

```
<button @click="removeTodo(todo)" class="badge bg-danger rounded-pill border-0">X</button>
```

```
data() {
  return {
    todos: [{ "id": 1, "title": "作業1", "completed": false }, { "id": 2, "title": "作業2", "completed": false }],
    newTodo: ""
  },
  methods: {
    removeTodo(todo) {
      const index = this.todos.indexOf(todo);
      this.todos.splice(index, 1);
    },
  },
},
```

Directives

- v-text
- v-html
- v-show
- v-if
- v-else
- v-else-if
- v-for
- v-on
- v-bind
- v-model
- v-slot
- v-pre
- v-once
- v-memo
- v-cloak

勾選完成的 todo - html

- 雙向繫節 v-model Directives
- 樣式 class 繫節 v-bind()
 - todo.completed 為 true 就會套用 completed 類別樣式

```
<input v-model="todo.completed" class="form-check-input me-3" type="checkbox">
<label :class="{ completed: todo.completed }" class="form-check-label">{{todo.title}}</label>
```

```
<style>
.completed {
  color: #949494;
  text-decoration: line-through;
}
</style>
```



將 todos 儲存在 localStorage 中

- 使用 watch 屬性，用來監看某個值，值改變時就會執行
 - handler(newValue, oldValue){ 值改變時執行這裡的程式 }
 - Immediate屬性：設定成 true 表示載入元件就會執行
 - deep屬性：設定成 true 就能監看物件中的屬性是否改變

```
watch: {  
  todos: {  
    handler(todos) {  
      localStorage.setItem("todos", JSON.stringify(todos))  
    },  
    deep: true  
  }  
},
```

計算還有多少 todos 要做

- 使用 computed 屬性
 - computed 函式用到 data 的資料，這些資料改變時，這個 computed 函式就會被重新執行和求值
 - computed 中的函式必須要有一個回傳值
 - computed 中的函式無法傳入參數

```
<strong class="me-3">尚有 {{ remaining }} 個工作未完成</strong>
```

```
computed: {
  remaining() {
    let activeTodos = this.todos.filter(todo => !todo.completed)
    return activeTodos.length
  }
},
```

清除所有完成的 todos - html & vue

```
<button class="btn btn-warning me-3" @click="removeComplete">清除完成工作</button>
```

```
methods: {  
    removeComplete() {  
        this.todos = this.todos.filter(todo =>!todo.completed)  
    }  
},
```

todos 資料修改 - html

● 顯示畫面

```
<div v-if="editedTodo !== todo" class="d-flex justify-content-between align-items-center">
  <div>
    <input v-model="todo.completed" class="form-check-input me-3" type="checkbox">
    <label v-bind:class="{ completed: todo.completed }" @dblclick="editTodo(todo)"
      class="form-check-label"> {{todo.title}} </label>
  </div>
  <button @click="removeTodo(todo)" class="badge bg-danger rounded-pill">X</button>
</div>
```

● 編輯畫面

```
<input v-else type="text" v-model="todo.title" class="form-control" @blur="doneEdit(todo)"
  @keyup.enter="doneEdit(todo)" @keyup.escape="cancelEdit(todo)">
```

todos 資料修改 - vue

```
data() {  
  return {  
    todos: JSON.parse(localStorage.getItem("todos") || '[]'),  
    newTodo: "",  
    editedTodo: null,  
    beforeEditCache: ""  
  },
```

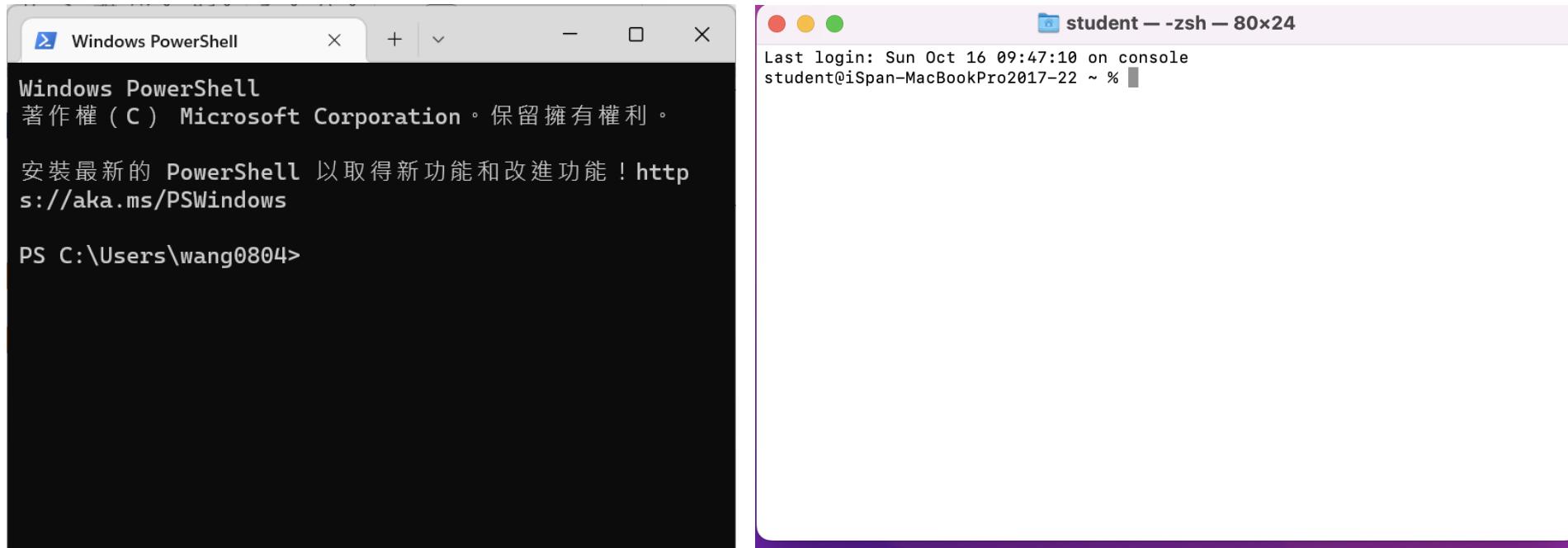
```
methods: {  
  editTodo(todo) {  
    this.beforeEditCache = todo.title  
    this.editedTodo = todo  
  },  
  doneEdit(todo) {  
    todo.title = todo.title.trim()  
    this.editedTodo = null  
  },  
  cancelEdit(todo) {  
    this.editedTodo = null  
    todo.title = this.beforeEditCache  
  },  
},
```

建立 Vue 應用程式

- 熟悉終端機指令的基本操作
- 安裝 Node.js 16.0 以上的版本

預備知識

- 熟悉終端機指令的基本操作



指令參考：https://ftp.kh.edu.tw/Linux/Redhat/en_6.2/doc/gsg/ch-doslinux.htm

- 安裝 Node.js 16.0 以上的版本

安裝 node.js



下載

最新版: 16.18.0 (包含 npm 8.19.2)

下載適合您平台的 Node.js 原始碼或安裝套件。立刻開始使用 Node.js。

LTS 建議大部分使用者使用	目前版本 最新功能	
 Windows 安裝程式 <small>node-v16.18.0-x64.msi</small>	 macOS 安裝程式 <small>node-v16.18.0.pkg</small>	 原始碼 <small>node-v16.18.0.tar.gz</small>

下載網址：<https://nodejs.org/zh-tw/download/>

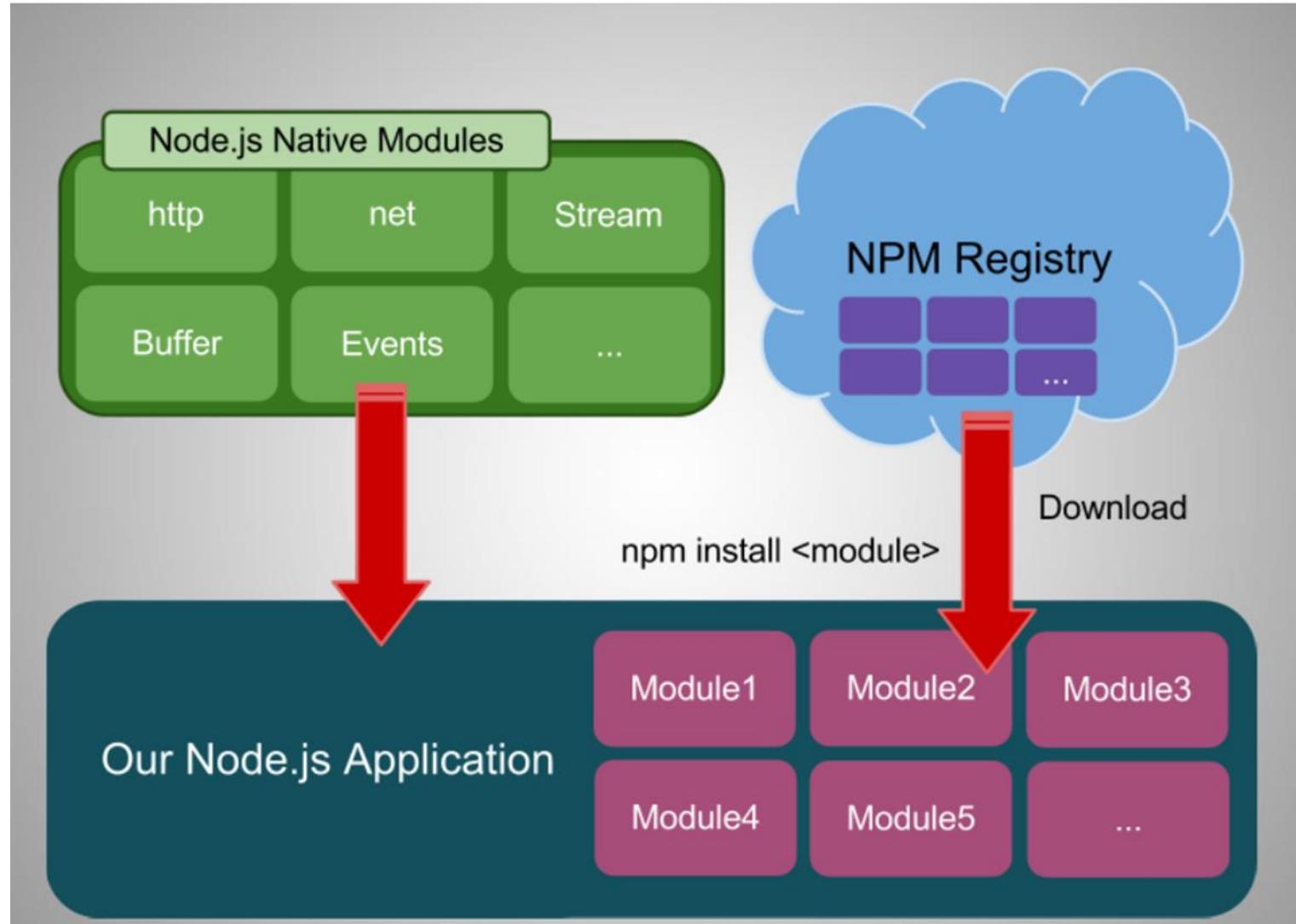
安裝完成後的測試

- 輸入 node -v 查看 node.js 的版本
- 輸入 npm -v 查看 npm 的版本



```
[student@iSpan-MacBookPro2017-22 Documents % node -v
v16.17.1
[student@iSpan-MacBookPro2017-22 Documents % npm -v
8.15.0
student@iSpan-MacBookPro2017-22 Documents % ]
```

Module 和 Package



NPM 套件管理工具

- Node Package Manager(npm)是Node.js的套件(package)管理工具。
 - 可以用來搜尋、安裝新的套件，也可用來列出已安裝的套件
- Node.js在0.6.3的版本之後開始內建npm



<https://www.npmjs.org/>

npm help

```
文件 --zsh-- 80x24
[student@iSpan-MacBookPro2017-22 Documents % npm help
npm <command>

Usage:

npm install           install all the dependencies in your project
npm install <foo>    add the <foo> dependency to your project
npm test              run this project's tests
npm run <foo>        run the script named <foo>
npm <command> -h      quick help on <command>
npm -l                display usage info for all commands
npm help <term>       search for help on <term>
npm help npm          more involved overview

All commands:

access, adduser, audit, bin, bugs, cache, ci, completion,
config, dedupe, deprecate, diff, dist-tag, docs, doctor,
edit, exec, explain, explore, find-dupes, fund, get, help,
hook, init, install, install-ci-test, install-test, link,
ll, login, logout, ls, org, outdated, owner, pack, ping,
pkg, prefix, profile, prune, publish, rebuild, repo,
restart, root, run-script, search, set, set-script,
shrinkwrap, star, stars, start, stop, team, test, token,
```

npm 常用指令 1

- npm help [指令名稱]：查詢指令詳細用法
- npm init : 建立 package.json
- npm init -y : 不用詢問建立 package.json
- npm install [套件名稱] -g : -g 表示全域安裝
- npm install [套件名稱] : 安裝套件在目前專案中
 - 安裝套件記錄在 package.json 的 dependencies 中
- npm install [套件名稱] --save-dev(縮寫-D)
 - 安裝套件記錄在 package.json 的 devDependencies 中

npm 常用指令 2

- npm uninstall [套件名稱] : 移除套件
- npm uninstall [套件名稱] -g : 移除全域套件
- npm search [套件名稱] : 搜尋套件
- npm home [套件名稱] : 套件所在網址
- npm info [套件名稱] version : 套件最新版本
- npm list : 列出已安裝套件
- npm update : 升級所有套件
- npm update [套件名稱] : 升級指定套件
- npm view [套件名稱] version : 檢視某個套件的版本

Package.json

- 描述 nodejs 專案的相關資訊，專案安裝了甚麼套件也會記錄在這個檔案中

```
{  
  "name": "application-name",  
  "version": "0.0.1",  
  "private": true,  
  "scripts": {  
    "start": "node app.js"  
  },  
  "dependencies": {  
    "express": "3.4.8",  
    "jade": "*",  
    "socket.io": "latest",  
    "mongoose": ">=2.5.3"  
  }  
}
```

建立 package.json



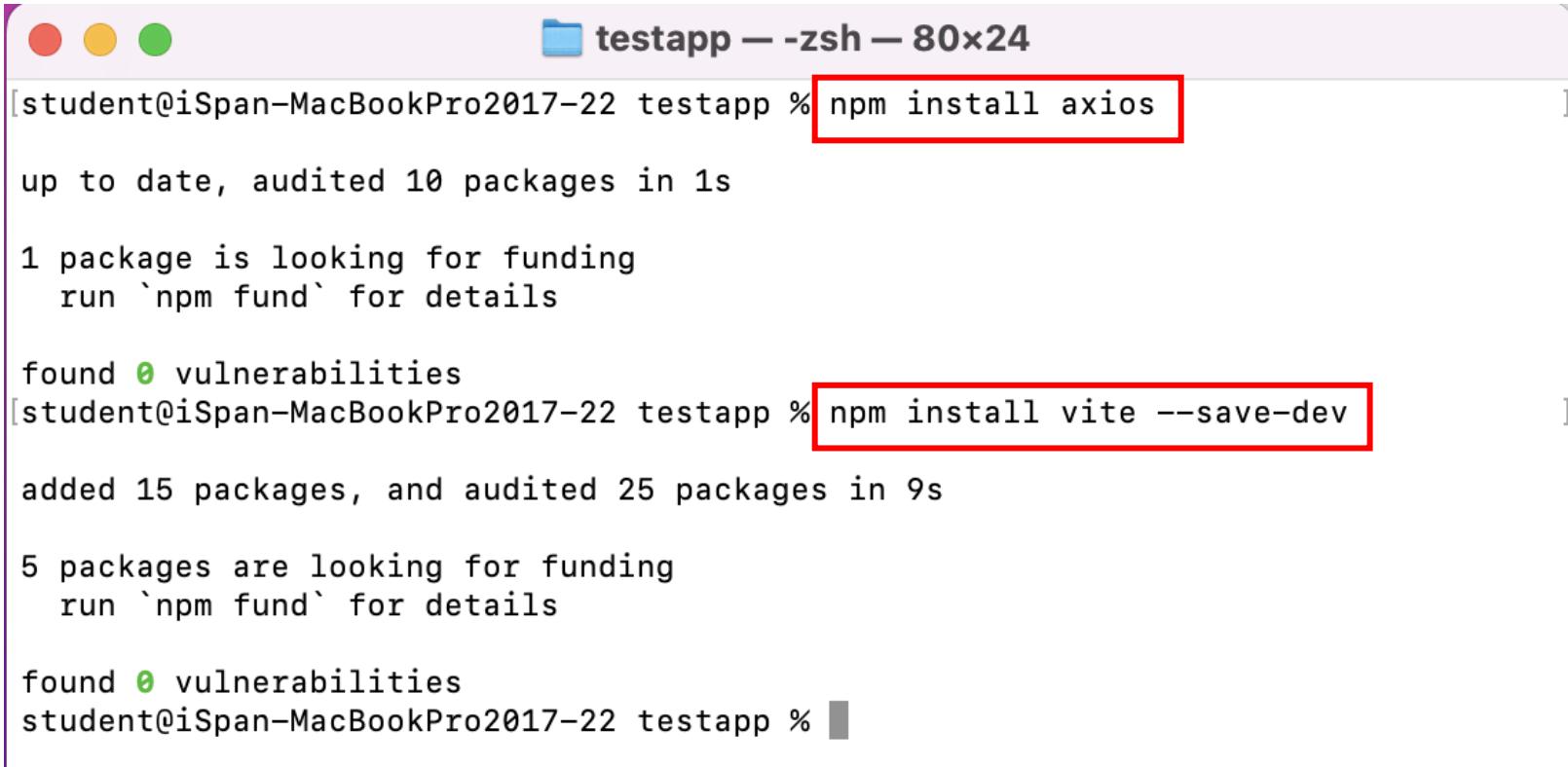
A screenshot of a terminal window titled "testapp — zsh — 80x24". The window shows a command-line session where a user creates a new directory "testapp" in their "documents" folder, changes into it, and runs "npm init -y" to generate a "package.json" file. The generated file is then displayed.

```
[student@iSpan-MacBookPro2017-22 ~ % pwd
/Users/student
[student@iSpan-MacBookPro2017-22 ~ % cd documents
[student@iSpan-MacBookPro2017-22 documents % mkdir testapp
[student@iSpan-MacBookPro2017-22 documents % cd testapp
[student@iSpan-MacBookPro2017-22 testapp % npm init -y
Wrote to /Users/student/Documents/testapp/package.json:

{
  "name": "testapp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

安裝套件

- npm install axios
- npm install vite --save-dev

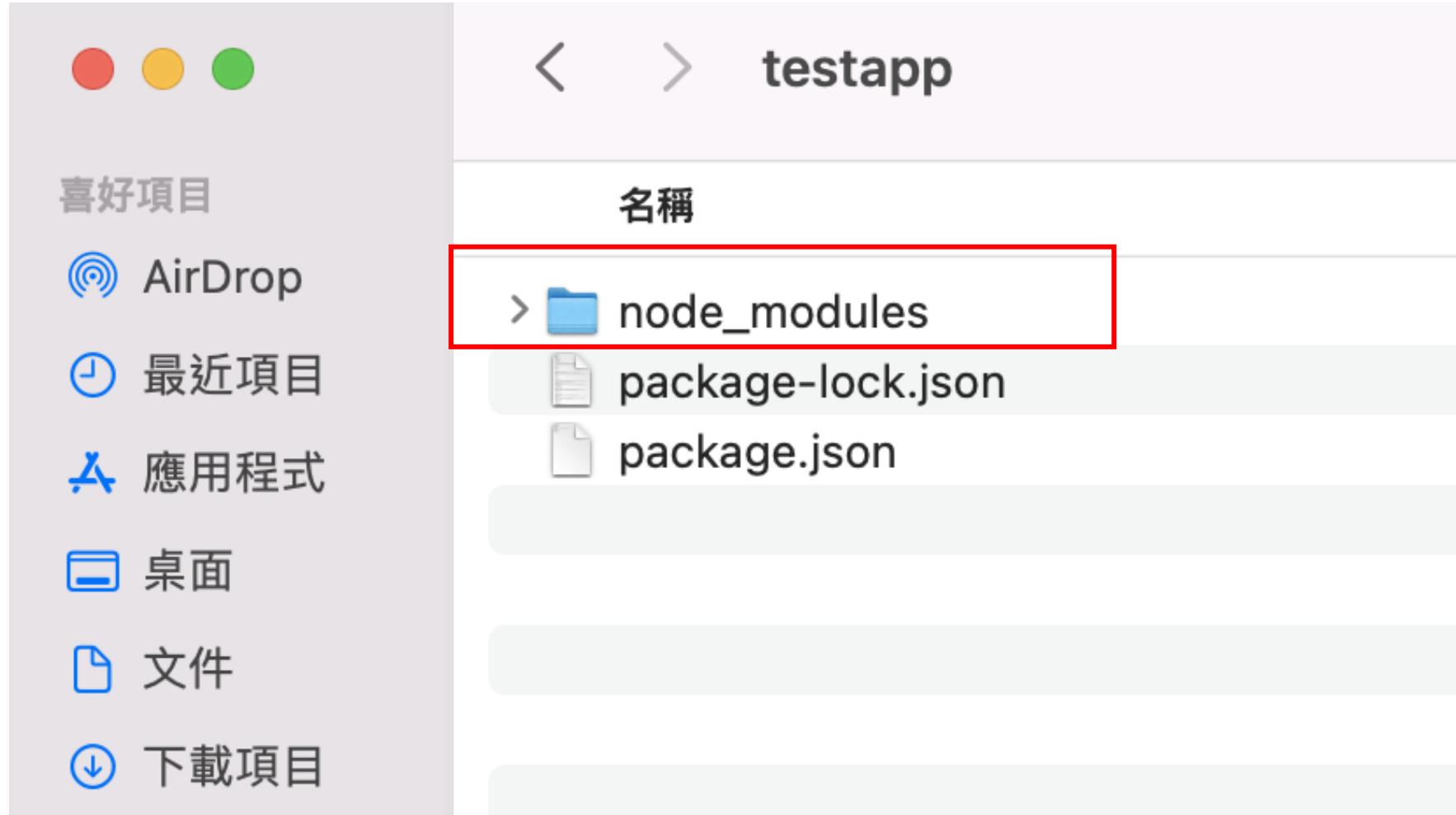


```
[student@iSpan-MacBookPro2017-22 testapp % npm install axios
up to date, audited 10 packages in 1s
1 package is looking for funding
  run `npm fund` for details
found 0 vulnerabilities
[student@iSpan-MacBookPro2017-22 testapp % npm install vite --save-dev
added 15 packages, and audited 25 packages in 9s
5 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
student@iSpan-MacBookPro2017-22 testapp % ]
```

安裝的套件會記錄在 package.json 檔中

```
1  {
2    "name": "testapp",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \\\"Error: no test specified\\\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "axios": "^1.1.3"
14   },
15   "devDependencies": {
16     "vite": "^3.1.8"
17   }
18 }
19 |
```

套件程式碼會裝在node_modules資料夾中



練習看看

- 建立一個node.js專案
 - 建立資料夾
 - 產生package.json檔
 - npm init -y
- 在 VS Code 中開啟專案
 - 建立一個index.js檔
 - 透過 node index.js 執行

```
\Documents>cd workspace  
\Documents\workspace>mkdir node-project  
\Documents\workspace>cd node-project  
\Documents\workspace\node-project>npm init -y
```

```
"name": "project",  
"version": "1.0.0",  
"description": "",  
"main": "index.js",  
"scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
},  
"keywords": [],  
"author": "",  
"license": "ISC"
```

透過 node index.js 執行

- Index.js

```
console.log("Hello Node.js")
```

- 執行 Index.js 檔

```
\workspace\node-project> node index.js
```

- 結果

```
Hello Node.js
```

安裝 axios 套件

```
npm install axios
```

```
"dependencies": {  
    "axios": "^1.1.3"  
},
```

- 改寫 Index.js 檔並執行看看

```
const axios = require('axios')  
  
axios.get('https://jsonplaceholder.typicode.com/todos/1')  
.then(response => console.log(response.data))
```

```
{ userId: 1, id: 1, title: 'delectus aut autem', completed: false }
```

安裝 json-server 套件

```
npm install json-server --save-dev
```

- 新增 db.json 檔

```
{  
  "todos": [  
    {  
      "id": 1,  
      "title": "作業1",  
      "completed": false  
    },  
    ]  
}
```

```
"dependencies": {  
  "axios": "^1.1.3"  
},  
"devDependencies": {  
  "json-server": "^0.17.0"  
}
```

- 啟動 json-server

啟動 json-server 套件

- 啟動 json-server

```
json-server db.json
```

- 改寫 Index.js 並執行看看

```
axios.get('http://localhost:3000/todos')
  .then(response => console.log(response.data))
```

```
\{^_^\}/ hi!
Loading db.json
Done

Resources
http://localhost:3000/todos

Home
http://localhost:3000
```

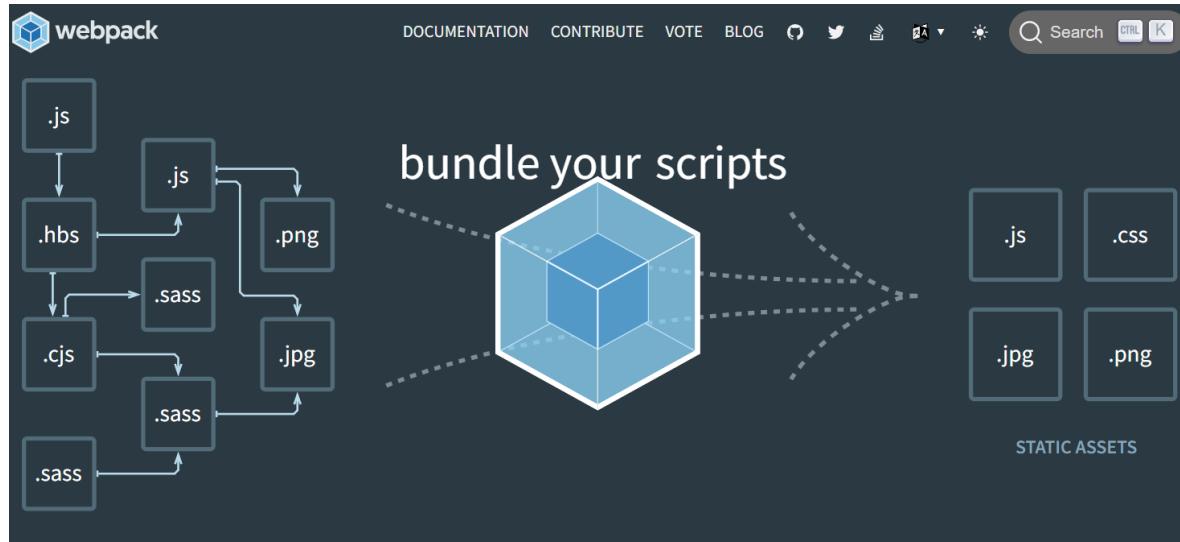
```
[{
  id: 1, title: '作業1', completed: false },
  { id: 2, title: '作業2', completed: true },
  { id: 3, title: '作業3', completed: false }]
```

建立 Vue Application

建立 Vue Application

```
> npm init vue@latest
```

- 這段指令會安裝及執行 create-vue
- 專案建置使用新一代的模組打包工具 vite，之前使用的是 webpack



<https://webpack.js.org/>

Vite
Next Generation
Frontend Tooling

Get ready for a development environment that
can finally catch up with you.

[Get Started](#)

[Why Vite?](#)

[View on GitHub](#)

執行 npm init vue@latest

```
[student@iSpan-MacBookPro2017-22 documents % npm init vue@latest
```

```
Vue.js – The Progressive JavaScript Framework
```

```
[✓] Project name: vue-app
[✓] Add TypeScript? No Yes
[✓] Add JSX Support? No Yes
[✓] Add Vue Router for Single Page Application development? No Yes
[✓] Add Pinia for state management? No Yes
[✓] Add Vitest for Unit Testing? No Yes
[✓] Add Cypress for both Unit and End-to-End testing? No Yes
[✓] Add ESLint for code quality? No Yes
```

```
Scaffolding project in /Users/student/Documents/vue-app...
```

```
Done. Now run:
```

```
cd vue-app
npm install
npm run dev
```

```
student@iSpan-MacBookPro2017-22 documents %
```

安裝及執行 vue 的專案

```
[student@iSpan-MacBookPro2017-22 documents % cd vue-app
[student@iSpan-MacBookPro2017-22 vue-app % npm install
  added 33 packages, and audited 34 packages in 5s
  4 packages are looking for funding
    run `npm fund` for details
  found 0 vulnerabilities
[student@iSpan-MacBookPro2017-22 vue-app % npm run dev
  > vue-app@0.0.0 dev
  > vite
  VITE v3.1.8 ready in 377 ms
  → Local: http://127.0.0.1:5173/
  → Network: use --host to expose
```

http://localhost:5173

Vite App



You did it!

You've successfully created a project with [Vite + Vue 3](#).

Documentation

Vue's [official documentation](#) provides you with all information you need to get started.

Tooling

This project is served and bundled with [Vite](#). The recommended IDE setup is [VSCode + Volar](#). If you need to test your components and web pages, check out [Cypress](#) and [Cypress Component Testing](#). More instructions are available in `README.md`.

Ecosystem

Get official tools and libraries for your project: [Pinia](#), [Vue Router](#), [Vue Test Utils](#), and [Vue Dev Tools](#). If you need more resources, we suggest paying [Awesome Vue](#) a visit.

Community

Got stuck? Ask your question on [Vue Land](#), our official Discord server, or [StackOverflow](#). You should also subscribe to [our mailing list](#) and follow the official [@vuejs](#) twitter account for latest news in the Vue world.

[Support Vue](#)

vue 專案的 Package.json

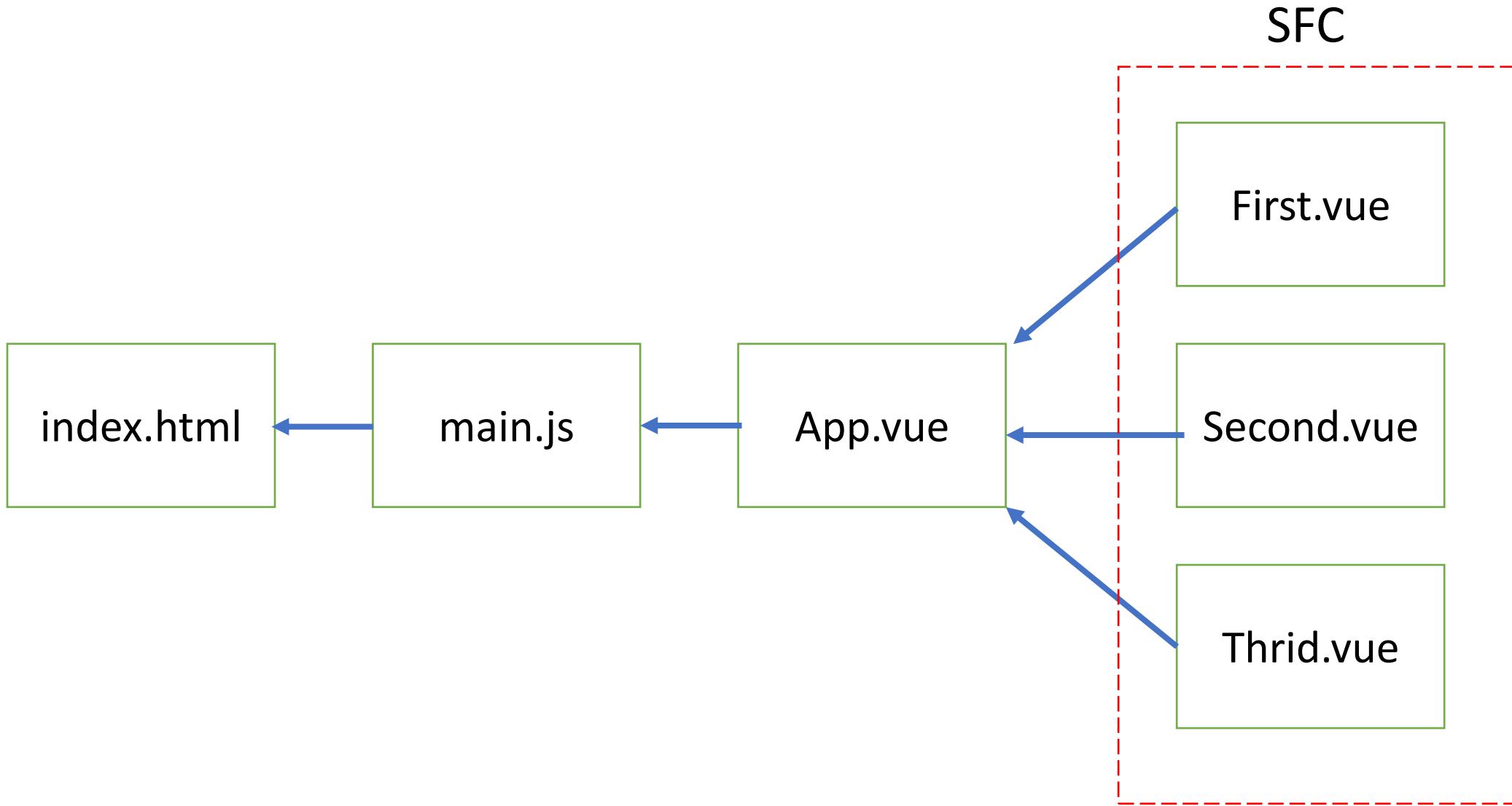
```
{  
  "name": "vue-app",  
  "version": "0.0.0",  
  ▷ 偵錯  
  "scripts": {  
    "dev": "vite",  
    "build": "vite build",  
    "preview": "vite preview --port 4173"  
  },  
  "dependencies": {  
    "vue": "^3.2.38"  
  },  
  "devDependencies": {  
    "@vitejs/plugin-vue": "^3.0.3",  
    "vite": "^3.0.9"  
  }  
}
```

```
15  "dependencies": {  
16    "pinia": "^2.0.21",  
17    "vue": "^3.2.38",  
18    "vue-router": "^4.1.5"  
19  },  
20  "devDependencies": {  
21    "@rushstack/eslint-patch": "^1.1.4",  
22    "@types/jdom": "^20.0.0",  
23    "@types/node": "^16.11.56",  
24    "@vitejs/plugin-vue": "^3.0.3",  
25    "@vitejs/plugin-vue-jsx": "^2.0.1",  
26    "@vue/eslint-config-prettier": "^7.0.0",  
27    "@vue/eslint-config-typescript": "^11.0.0",  
28    "@vue/test-utils": "^2.0.2",  
29    "@vue/tsconfig": "^0.1.3",  
30    "cypress": "^10.7.0",  
31    "eslint": "^8.22.0",  
32    "eslint-plugin-cypress": "^2.12.1",  
33    "eslint-plugin-vue": "^9.3.0",  
34    "jdom": "^20.0.0",  
35    "npm-run-all": "^4.1.5",  
36    "prettier": "^2.7.1",  
37    "start-server-and-test": "^1.14.0",  
38    "typescript": "~4.7.4",  
39    "vite": "^3.0.9",  
40    "vitest": "^0.23.0",  
41    "vue-tsc": "^0.40.7"
```

Vue 專案結構

VUE-APP	
> .vscode	
> node_modules	● 套件程式碼放在這裡
✓ public	● 公開檔案資料夾
★ favicon.ico	●
✓ src	● 原始檔資料夾，vue的程式都在這裡
> assets	● vue用的靜態檔案資料夾
> components	● 元件預設都放在這裡
▼ App.vue	● 主要整合Vue元件的檔案
JS main.js	● 載入 App.vue 的內容掛載到 index.html
↳ .gitignore	
↳ index.html	● vue 專案的唯一 HTML 文件(Single Page Application)
{ } package-lock.json	
{ } package.json	● 紀錄 vue 專案安裝了那些套件，專案的執行方式等地專案資訊
(i) README.md	
JS vite.config.js	

網站結構



Single-File Component(SFC)

- Script

```
<script setup></script>
```

- template

```
<template>
  <h1>My First Vue Component</h1>
</template>
```

- style

```
<style scoped></style>
```

Component的建立

Components/HelloVue.vue

```
<script setup>
  import { ref } from "@vue/reactivity"
  const msg = ref('')
</script>
```

```
<template>
  <h3>Hello, {{ msg }}!!</h3>
  <input type="text" v-model="msg" />
</template>
```

```
<style scoped>
  h3{ color:blue; }
</style>
```

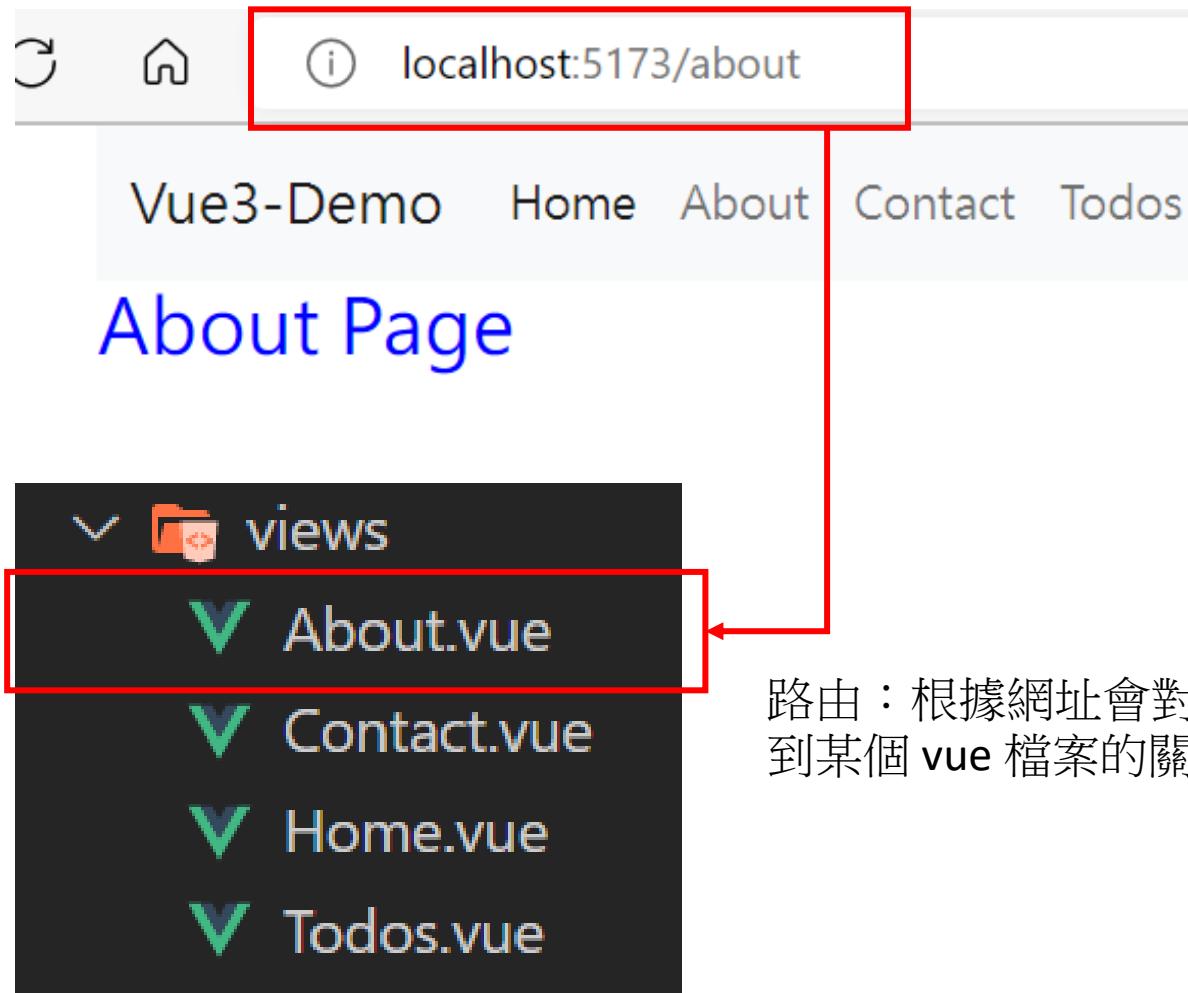
Component 的使用

App.vue

```
<script setup>
  import HelloVue from '@/components/HelloVue.vue'
</script>
```

```
<template>
  <HelloVue />
  <hello-vue />
</template>
```

路由 vue-router



路由：根據網址會對應到某個 vue 檔案的關係

```
src > views > ▼ Home.vue > ...
1  <template>
2  |   <h3>Home Page</h3>
3  </template>
4
5  <script setup>
6
7  </script>
8
9  <style scoped>
10
11 </style>>
12
13
```

路由設定及使用

- 安裝路由套件

```
npm install vue-router@4
```

- <router-view> Vue Component 會顯示的地方
- <router-link> 產生超連結的標籤

App.vue

```
<template>
  <router-link to="/">Home</router-link>
  <router-link to="/about">About</router-link>
  <hr />
  <router-view />
</template>
```

在JS檔中設定路由

router.js

```
import { createWebHistory, createRouter } from "vue-router";
import Home from "@/views/Home.vue";
import About from "@/views/About.vue";
const routes = [
  {
    path: "/",
    name: "Home",
    component: Home,
  },
  {
    path: "/about",
    name: "About",
    component: About,
  },
];
```

router.js

```
const router = createRouter({
  history: createWebHistory(),
  routes,
});

export default router;
```

套用路由

main.js

```
import { createApp } from 'vue'  
import App from './App.vue'  
import router from './router'  
  
// createApp(App).mount('#app')  
createApp(App).use(router).mount('#app')
```

改寫 Todo

Todos.vue

Todos

想要做甚麼?

- 作業 1 X
- 作業 2 X
- 作業 3 X

TodoList.vue

尚有 ? 個工作未完成

TodoFooter.vue

清除完成工作

The screenshot shows a Vue.js application interface for managing a todo list. At the top, a blue header bar contains the text "Todos.vue". The main content area is titled "Todos" and features a search input field with the placeholder "想要做甚麼?". Below this is a list of three items, each with a checkbox and a red circular "X" button. The items are labeled "作業 1", "作業 2", and "作業 3". A blue button labeled "TodoList.vue" is positioned between the first and second items. At the bottom, a footer section contains the text "尚有 ? 個工作未完成" and a yellow button labeled "清除完成工作". A blue button labeled "TodoFooter.vue" is positioned below the footer text. The entire list area and the footer section are highlighted with red boxes.

資料的傳遞(父 > 子)

Child Component Parent Component

```
<script setup>
  import HelloVue from '@/components/HelloVue.vue'
```

```
<template>
  <HelloVue />
  <hello-vue />
</template>
```

HelloVue.vue 子組件

```
<script setup>
  const props = defineProps({
    message: String,
  });
</script>
```

```
<template>
  <h3>{{ message }}!!</h3>
  <input type="text" v-model="message" />
</template>
```

App.vue 父組件

```
<script setup>
  import HelloVue from '@/components/HelloVue.vue'
  import { ref } from "vue"
  let message = ref('Hello, Vue');
</script>
```

```
<template>
  <HelloVue :message="message" />
</template>
```

Hello, Vue!!

Hello, Vue

父-傳遞 todos 資料到子組件

Todos.vue

```
<script setup>

import { reactive, ref } from "vue";
const todos = reactive({
  "data": [
    { "id": 1, "title": "作業1", "completed": false },
    { "id": 2, "title": "作業2", "completed": true },
    { "id": 3, "title": "作業3", "completed": false }
  ]
})
</script>
```

```
<template>
  <TodoList :todos="todos.data" />
</template>
```

子-接收及顯示 todos 資料

TodoList.vue

```
<script setup>
const props = defineProps({
  todos: Array
})
</script>
```

```
<template>
  <li v-for="(todo,index) in props.todos" class="list-group-item" :key="index">
    <div class="d-flex justify-content-between">
      <div>
        <input class="form-check-input me-3" type="checkbox">
        <label class="form-check-label">{{todo.title}}</label>
      </div>
      <button class="badge bg-danger rounded-pill">X</button>
    </div>
  </li></template>
```

Todos 資料新增

Todos.vue

```
<script setup>
  import { reactive, ref } from "vue";
  const newTodo = ref("")
  const addTodo = () => {
    if (!newTodo.value) {
      return;
    }
    todos.data.push({ "id": todos.data.length + 1, "title": newTodo.value, "completed": false });
    newTodo.value = "";
  }
</script>
```

```
<template>
  <input type="text" v-model.trim="newTodo" class="form-control" autofocus autocomplete="off"
    placeholder="想要做甚麼?" @keyup.enter="addTodo" />
</template>
```

父 - 刪除todo資料

Todos.vue

```
<script setup>
const removeTodo = todo => {
  const index = todos.data.indexOf(todo);
  todos.data.splice(index, 1);
}
</script>
```

```
<template>
  <TodoList :todos="todos" @removeTodo="removeTodo" />
</template>
```

子 - 刪除 todo 資料

TodoList.vue

```
<script setup>

  const emit = defineEmits(["removeTodo"])

  const handleRemove = todo => {
    emit("removeTodo", todo)
  }

</script>
```

```
<template>
  <button @click="handleRemove(todo)"
         class="badge bg-danger rounded-pill">X</button>
</template>
```

勾選完成的 todo

TodoList.vue

```
<div class="d-flex justify-content-between">
  <div>
    <input v-model="todo.completed" class="form-check-input me-3" type="checkbox">
    <label v-bind:class="{ completed: todo.completed }" class="form-check-label">{{todo.title}}</label>
  </div>
  <button @click="handleRemove(todo)" class="badge bg-danger rounded-pill">X</button>
</div>
```

```
<style scoped>
  .completed {
    color: #949494;
    text-decoration: line-through;
  }
</style>
```

將 todos 儲存在 localStorage 中

- 從 localStorage 中讀出資料

```
const todos = reactive({ "data": JSON.parse(localStorage.getItem("todos") || '[]') })
```

- 將資料儲存在 localStorage 中

```
watch(() => todos, (newValue, oldValue) => {
  localStorage.setItem("todos", JSON.stringify(newValue.data))
}, { deep: true })
```

父 - 計算還有多少 todos 要做

Todos.vue

```
const remaining = computed(() => {
  let activeTodos = todos.data.filter(todo => !todo.completed)
  return activeTodos.length
})
```

```
<TodoFooter :remaining="remaining" />
```

子 - 計算還有多少 todos 要做

TodoFooter.vue

```
<script setup>  
  
  const props = defineProps({ remaining: Number })  
  
</script>
```

```
<template>  
  
  <div class="mt-3 d-flex justify-content-between">  
    <strong class=" me-3">尚有 {{props.remaining}} 個工作未完成</strong>  
    <button class="btn btn-warning me-3">清除完成工作</button>  
  </div>  
  
</template>
```

父 - 清除所有完成的 todos

Todos.vue

```
const removeComplete = () => {
  for (let i = todos.data.length - 1; i >= 0; i -= 1) {
    if (todos.data[i].completed) {
      todos.data.splice(i, 1);
    }
  }
}
```

```
<TodoFooter @removeComplete="removeComplete" :remaining="remaining" />
```

子 - 清除所有完成的 todos

TodoFooter.vue

```
<script setup>
const props = defineProps({ remaining: Number })
const emit = defineEmits(["removeComplete"])
const handleRemoveComplete = () => {
  emit("removeComplete")
}
</script>
```

```
<template>
  <div class="mt-3 d-flex justify-content-between">
    <strong class=" me-3">尚有 {{props.remaining}} 個工作未完成</strong>
    <button @click="handleRemoveComplete" class="btn btn-warning me-3">清除完成工作</button>
  </div>
</template>
```

todos 資料修改 - template

● 顯示畫面

TodoList.vue

```
<div v-if="editedTodo !== todo" class="d-flex justify-content-between align-items-center">
  <div>
    <input v-model="todo.completed" class="form-check-input me-3" type="checkbox">
    <label v-bind:class="{ completed: todo.completed }" @dblclick="editTodo(todo)"
      class="form-check-label"> {{todo.title}} </label>
  </div>
  <button @click="removeTodo(todo)" class="badge bg-danger rounded-pill">X</button>
</div>
```

● 編輯畫面

```
<input v-else type="text" v-model="todo.title" class="form-control" @blur="doneEdit(todo)"
  @keyup.enter="doneEdit(todo)" @keyup.escape="cancelEdit(todo)">
```

todos 資料修改 - script

TodoList.vue

```
import { ref, watch } from 'vue'
const editedTodo = ref(null)
const beforeEditCache = ref("")
const editTodo = todo => {
  beforeEditCache.value = todo.title
  editedTodo.value = todo
}
const cancelEdit = todo => {
  editedTodo.value = null
  todo.title = beforeEditCache.value
}
const DoneEdit = todo => {
  todo.title = todo.title.trim()
  editedTodo.value = null
}
```

Thank you