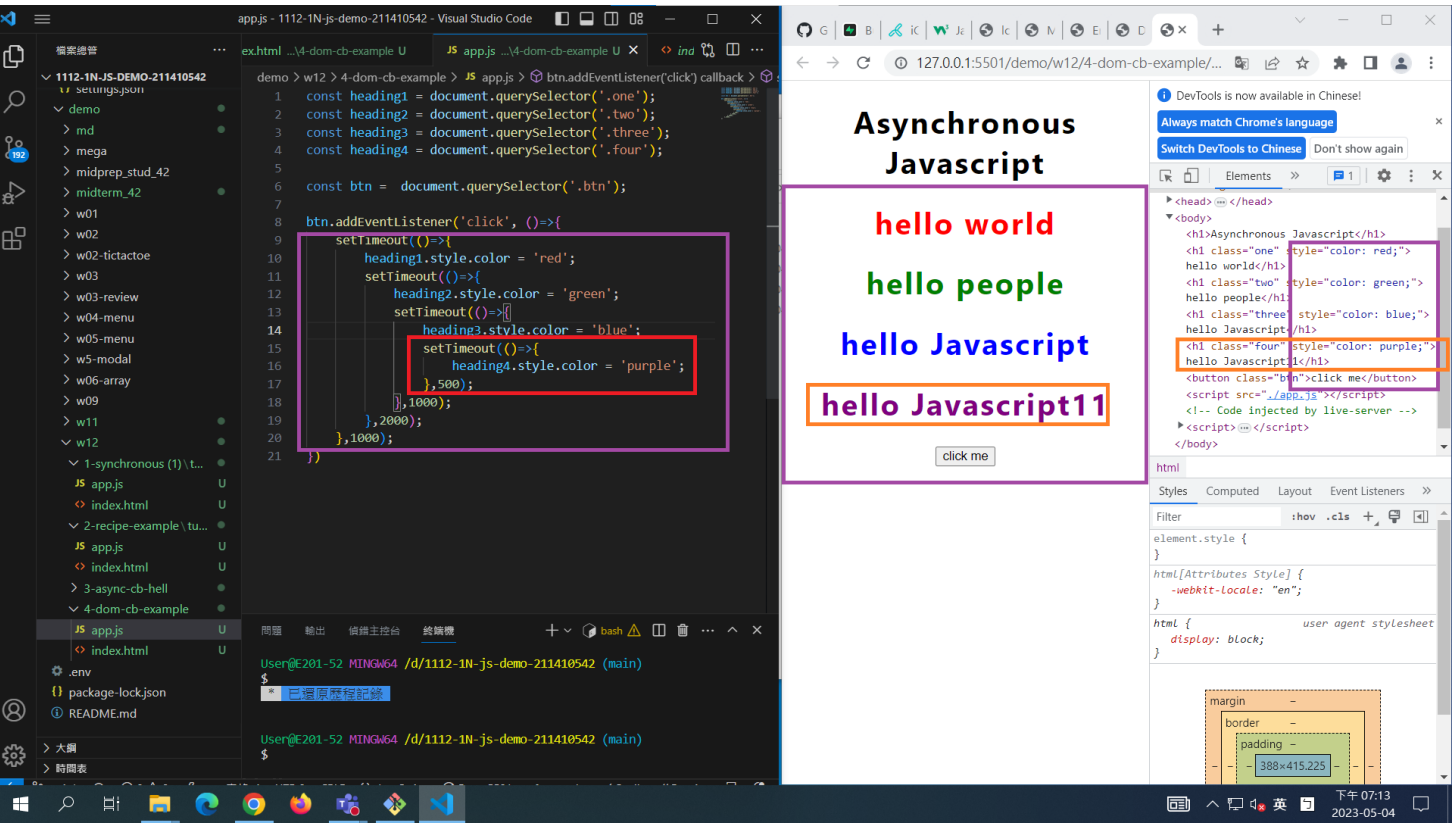


Github URL

W12-P1: call back hell DOM demo



10dc7ea George0113

Thu May 4 19:18:46 2023 +0800 call back hell DOM demo

W12-p2: use promise to solve the cb hell problem

The screenshot displays a development environment with Visual Studio Code on the left and a web browser on the right. The VS Code editor shows a file named `app.js` with the following JavaScript code:

```
1 const heading1 = document.querySelector('.one');
2 const heading2 = document.querySelector('.two');
3 const heading3 = document.querySelector('.three');
4 const heading4 = document.querySelector('.four');
5
6 const btn = document.querySelector('.btn');
7
8 btn.addEventListener('click', () => {
9   addColor(1000, heading1, 'red')
10    .then(() => addColor(2000, heading2, 'green'))
11    .then(() => addColor(1000, heading3, 'blue'))
12    .then(() => addColor(500, heading4, 'purple'))
13    .catch(error => console.log(error));
14 });
15
16
17 const addColor = (time, element, color) => {
18   return new Promise((resolve, reject) => {
19     if (element) {
20       setTimeout(() => {
21         element.style.color = color;
22         resolve();
23       }, time);
24     } else {
25       reject(new Error('There is no such element ${element}'));
26     }
27   });
28 }
29
```

The browser window shows the output of the code, displaying the text "hello world", "hello people", "hello Javascript", and "hello George" in different colors (red, green, blue, and purple respectively). The browser's developer tools are open, showing the HTML structure and the styles applied to the elements.

6236ce0 George0113

Thu May 4 20:21:38 2023 +0800

W12-p2: use promise to solve the cb hell problem

W12-P3: use async/await to solve the cb hell problem

The screenshot displays a Visual Studio Code editor and a web browser. The editor shows a file named `app.js` with the following code:

```
1 const heading1 = document.querySelector('.one');
2 const heading2 = document.querySelector('.two');
3 const heading3 = document.querySelector('.three');
4 const heading4 = document.querySelector('.four');
5
6 const btn = document.querySelector('.btn');
7
8 btn.addEventListener('click', async () => {
9   const result = await displayColor();
10  console.log('result', result);
11 });
12
13 const displayColor = async () => {
14   try {
15     await addColor(2000, heading1, 'red');
16     await addColor(2000, heading2, 'green');
17     await addColor(2000, heading3, 'blue');
18     await addColor(2000, heading4, 'purple');
19     console.log('success');
20   } catch (error) {
21     console.log(error);
22   }
23 }
24
25 const addColor = (time, element, color) => { ...
26
27
28
29 }
```

The browser shows the output of the code, displaying the text "hello world", "hello people", "hello Javascript", and "hello George" in different colors (red, green, blue, purple) and a "click me" button.

The terminal output shows the command `git log --pretty=format:"%h%x09%an%x09%ad%x09%ae" --after="2023-5-03"` and the commit message "W12-p2: use promise to solve the cb hell problem".

76bebea George0113

Thu May 4 20:42:14 2023 +0800

W12-P3: use async/await to solve the cb hell problem

W12-p4: xhr, get sample.txt

success reading

The image shows a web browser window on the right and a Visual Studio Code editor on the left. The browser window displays the title "AJAX" and a "show json" button. Below the button, there is a red box containing the text: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Dolores nihil ex atque. Asperiores modi delectus excepturi. Debitis est tenetur voluptas autem nulla consequuntur quae dolorem quisquam, assumenda reprehenderit adipisci aliquid odit hic fugit iste sed totam cporis nobis." The browser's DevTools console on the right shows the response of the XHR request, which is a JSON object: {"status": 200, "text": "OK", "state": 2}. The VS Code editor on the left shows the source code of the application. The code is in a file named "sample.txt" and it uses XMLHttpRequest (XHR) to fetch data from a server. The code is as follows:

```
1 const xhr = new XMLHttpRequest();
2
3 xhr.open('GET', './api/sample.txt');
4
5 xhr.onreadystatechange = function(){
6   console.log('xhr', xhr);
7   if (xhr.readyState === 4 && xhr.status === 200){
8     const text = document.createElement('p');
9     console.log('p', text);
10    text.textContent = xhr.responseText;
11    document.body.appendChild(text);
12  }else{
13    console.log({
14      status: xhr.status,
15      text: xhr.statusText,
16      state: xhr.readyState
17    });
18  }
19 }
20
21 xhr.send();
22
```

fail reading

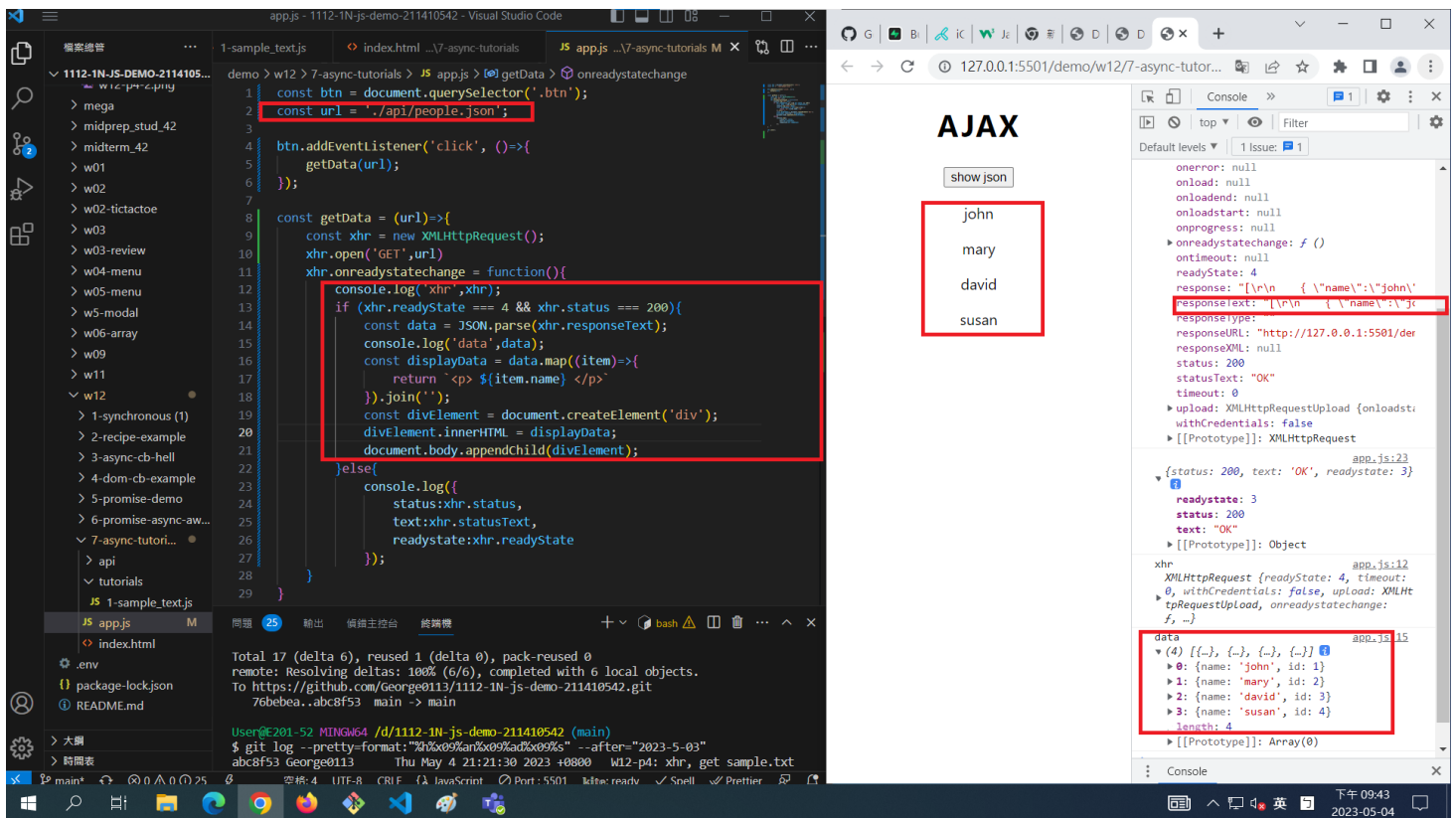
The image shows a development environment with Visual Studio Code on the left and a web browser on the right. In VS Code, the file `sample2.txt` is open, showing an XMLHttpRequest (XHR) setup. The URL `./api/sample2.txt` is highlighted in red. The `onreadystatechange` event listener is also highlighted, showing the status and response handling logic. The terminal at the bottom shows the command `npm run dev` and the output of the development server.

The browser on the right shows the page `127.0.0.1:5501/demo/w12/7-async-tutor...` with the title **AJAX**. The console shows the error: `GET http://127.0.0.1:5501/demo/w12/7-async-tutorials/api/sample2.txt 404 (Not Found)`. The console also shows the XHR object details, including the status `404` and the text `'Not Found'`.

abc8f53 George0113

Thu May 4 21:21:30 2023 +0800 W12-p4: xhr, get sample.txt

W12-P5: xhr, get people.json, and show names in browser



da97d98 George0113 Thu May 4 21:46:08 2023 +0800 W12-P5: xhr, get people.json, and show names in browser

```
$ git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2023-5-03"
```

da97d98 George0113	Thu May 4 21:46:08 2023 +0800	W12-P5: xhr, get people.json, and show names in browser
abc8f53 George0113	Thu May 4 21:21:30 2023 +0800	W12-p4: xhr, get sample.txt
76bebea George0113	Thu May 4 20:42:14 2023 +0800	W12-P3: use async/await to solve the cb hell problem
6236ce0 George0113	Thu May 4 20:21:38 2023 +0800	W12-p2: use promise to solve the cb hell problem
10dc7ea George0113	Thu May 4 19:18:46 2023 +0800	call back hell DOM demo