

W12-P1: call back hell DOM demo

The image shows a code editor on the left and a web browser on the right. The code editor displays a JavaScript file named `index.html` 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', () => {
9   setTimeout(() => {
10     heading1.style.color = 'red';
11     setTimeout(() => {
12       heading2.style.color = 'green';
13       setTimeout(() => {
14         heading3.style.color = 'blue';
15         setTimeout(() => {
16           heading4.style.color = 'yellow';
17         }, 500);
18       }, 1000);
19     }, 2000);
20   }, 1000);
21 });
```

The browser window shows the result of the code execution. The page title is "Asynchronous Javascript". The page content consists of four lines of text, each in a different color and enclosed in a colored box:

- hello world (red text, purple box)
- hello people (green text, green box)
- hello Javascript (blue text, blue box)
- hello Async JS (yellow text, yellow box)

A "click me" button is visible at the bottom of the page. The browser's DevTools console shows the following HTML structure:

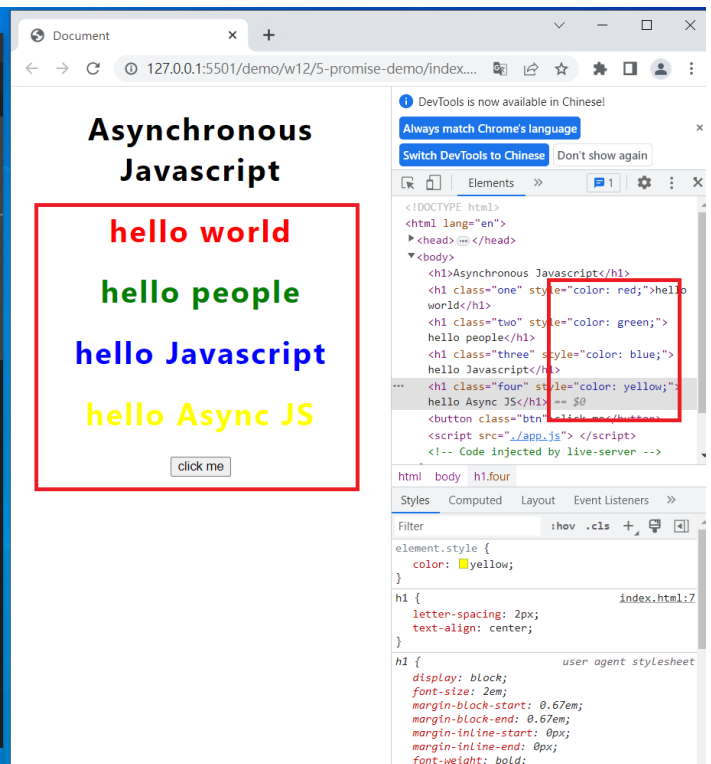
```
<!DOCTYPE html>
<html lang="en">
  <head>
  </head>
  <body>
    <h1>Asynchronous Javascript</h1>
    <h1 class="one" style="color: red;">hello world</h1>
    <h1 class="two" style="color: green;">hello people</h1>
    <h1 class="three" style="color: blue;">hello Javascript</h1>
    <h1 class="four" style="color: yellow;">hello Async JS</h1>
    <button class="btn">click me</button>
    <script src="/app.js"></script>
    <!-- Code injected by live-server -->
  </body>
</html>
```

The DevTools Styles panel shows the following styles for the `h1.three` element:

```
element.style {
  color: blue;
}
h1 {
  letter-spacing: 2px;
  text-align: center;
}
h1 {
  display: block;
  font-size: 2em;
  margin-block-start: 0.67em;
  margin-block-end: 0.67em;
  margin-inline-start: 0px;
  margin-inline-end: 0px;
}
```

W12-P2: use promise to solve the cb hell problem

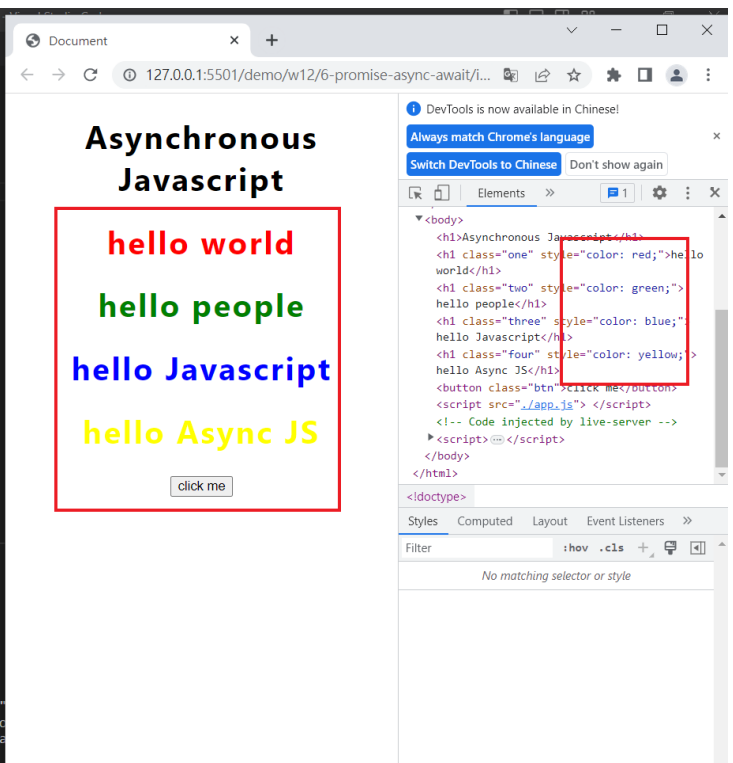
```
demo > w12 > 5-promise-demo > JS appjs > addColor
1 const heading1 = document.querySelector('.one');
2 const heading2 = document.querySelector('.two');
3 const heading3 = document.querySelector('.three');
4 const heading4 = document.querySelector('.four');
5
6 const btn = document.querySelector('.btn');
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, 'yellow'))
13  .catch(error=>console.log(error))
14 });
15
16 const addColor = (time, element, color) => {
17   return new Promise((resolve, reject) => {
18     if (element) {
19       setTimeout(() => {
20         element.style.color = color;
21         resolve();
22       }, time);
23     } else {
24       reject(new Error('There is no such element ${element}')); // 返回catch
25     }
26   });
27 }
28
29 // btn.addEventListener('click', ()=>{
30 //   setTimeout(() => {
31 //     heading1.style.color = 'red';
32 //     setTimeout(() => {
33 //       heading2.style.color = 'green';
34 //       setTimeout(() => {
35 //         heading3.style.color = 'blue';
36 //         setTimeout(() => {
37 //           heading4.style.color = 'yellow';
38 //         }, 500);
39 //       }, 1000);
40 //     }, 2000);
41 //   }, 1000);
42 // });
```



36f3290 21141003901~ Thu May 4 20:19:48 2023 +0800 W12-P2: use promise to solve the cb hell problem

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

```
demo > w12 > 6-promise-async-await > JS appjs > ...
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(1000, heading1, 'red');
16     await addColor(2000, heading2, 'green');
17     await addColor(1000, heading3, 'blue');
18     await addColor(500, heading4, 'yellow');
19   } catch (error) {
20     console.log(error);
21   }
22 }
23
24 }
```



W12-P4: xhr, get sample.txt

success reading

The image shows a web application running in a browser, demonstrating a successful XMLHttpRequest (XHR) to retrieve data from a server. The application is titled "AJAX" and has a "show json" button.

JavaScript Code (Left Panel):

```
demo > w12 > 7-async-tutorials > JS app.js > onreadystatechange
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();
```

Browser Console (Right Panel):

The console shows the following log messages:

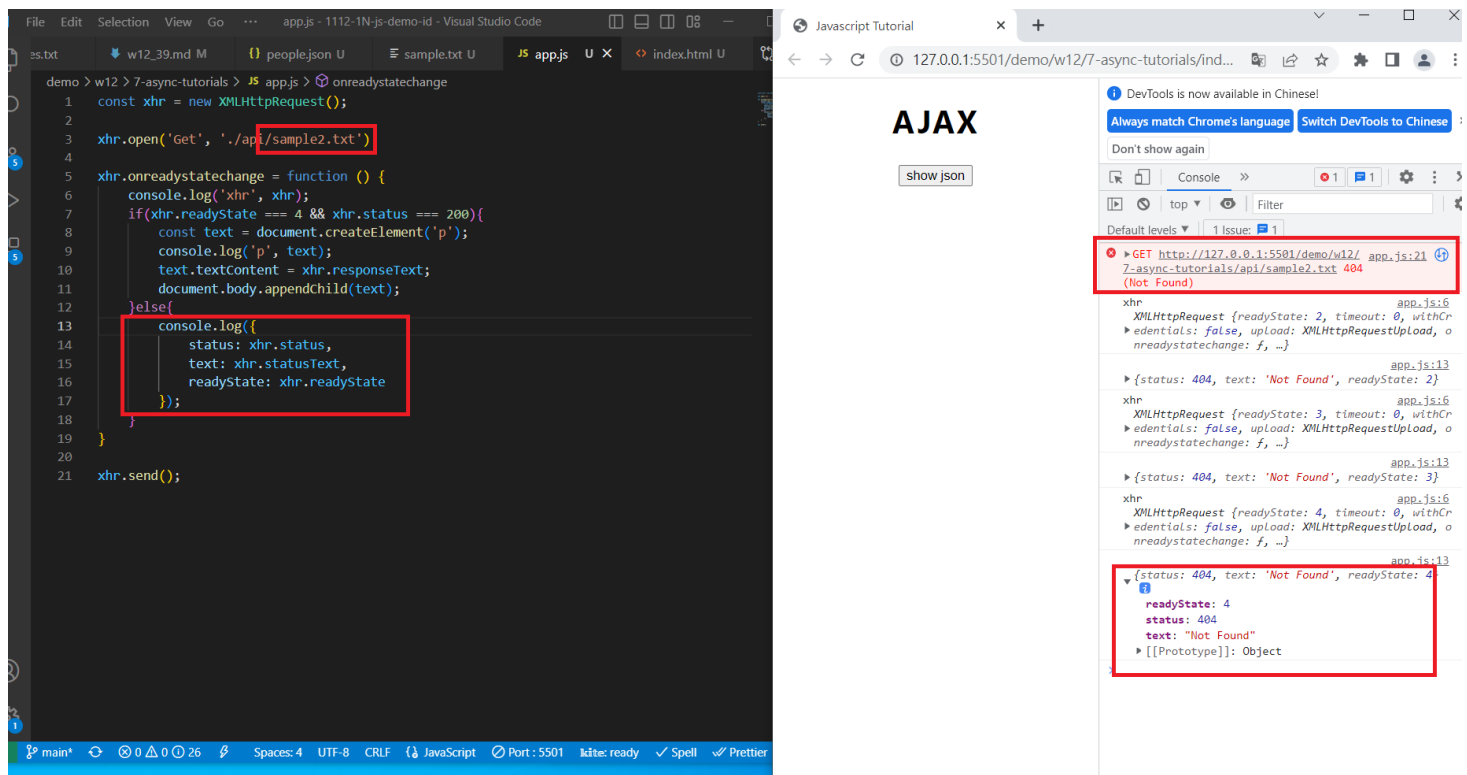
- `onreadystatechange: f ()`
- `readyState: 4`
- `response: "Lorem ipsum dolor sit amet consectetur adipiscing elit. Omnis error tempora itaque aperiam quam, vitae laudantium ut culpa iure enim adipisci nostrum, quaerat quas dignissimos quidem perspiciatis quod maxime ratione!"`
- `responseText: "Lorem ipsum dolor sit amet consectetur adipiscing elit. Omnis error tempora itaque aperiam quam, vitae laudantium ut culpa iure enim adipisci nostrum, quaerat quas dignissimos quidem perspiciatis quod maxime ratione!"`
- `status: 200`
- `statusText: "OK"`
- `timeout: 0`
- `upload: XMLHttpRequestUpload {onloadstart: null, withCredentials: false}`
- `[[Prototype]]: XMLHttpRequest`

DOM (Right Panel):

The DOM shows a new paragraph element (`<p>`) added to the body, containing the text:

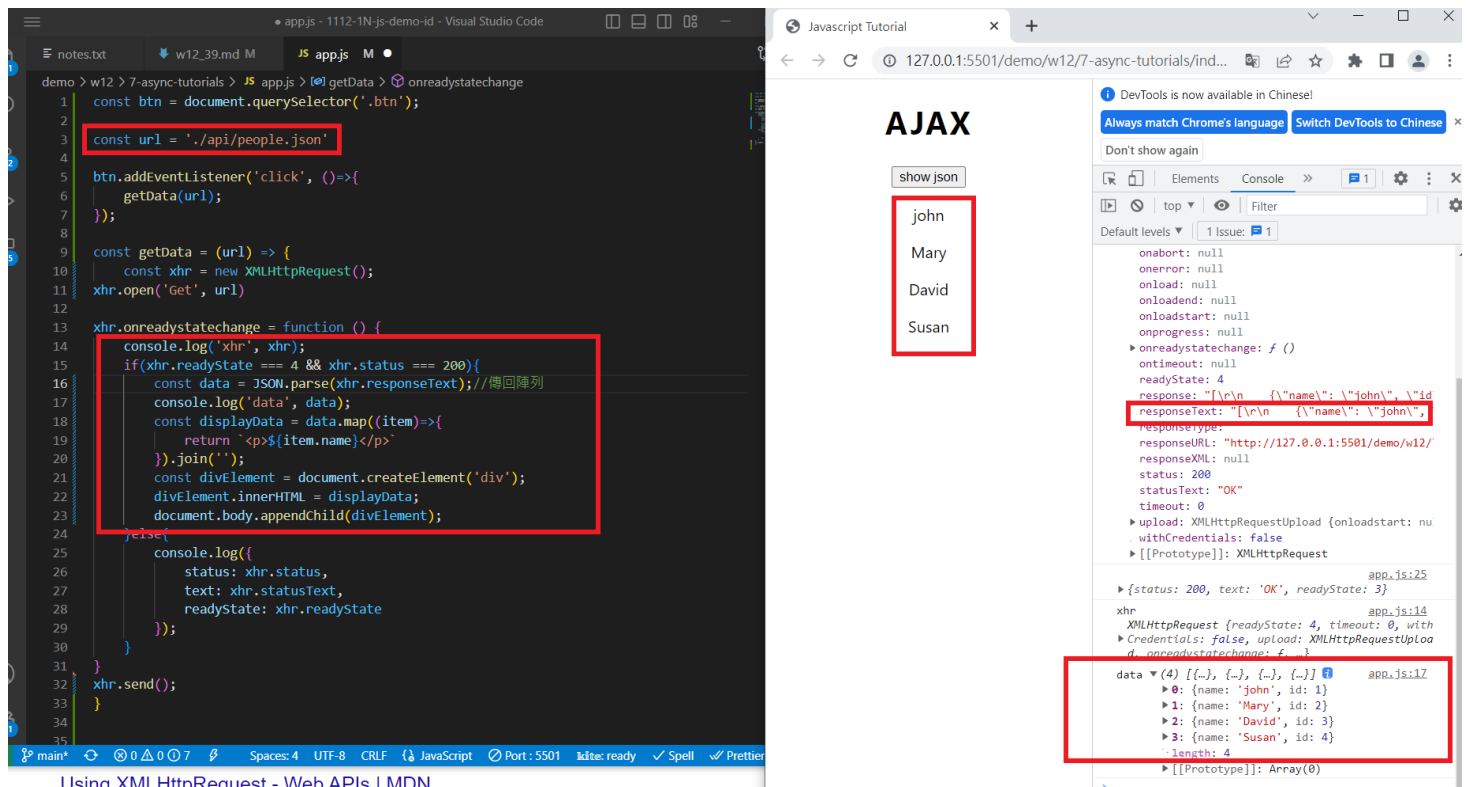
"Lorem ipsum dolor sit amet consectetur adipiscing elit. Omnis error tempora itaque aperiam quam, vitae laudantium ut culpa iure enim adipisci nostrum, quaerat quas dignissimos quidem perspiciatis quod maxime ratione!"

fail reading



53a1ea8 21141003901~ Thu May 4 21:24:20 2023 +0800 W12-P4: xhr, get sample.txt

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



3fa1869 21141003901~ Thu May 4 21:46:40 2023 +0800 W12-P5: xhr, get people.json, and show names in browser

W12-logs

```
User@E201-10 MINGW64 /d/1112-1N-js-demo-id (main)
$ git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2023-05-03"
3fa1869 21141003901~ Thu May 4 21:46:40 2023 +0800 W12-P5: xhr, get people.json, and show names in browser
53a1ea8 21141003901~ Thu May 4 21:24:20 2023 +0800 W12-P4: xhr, get sample.txt
37e498a 21141003901~ Thu May 4 20:41:15 2023 +0800 W12-P3: use astnc/await to solve the cb hell problem
36f3290 21141003901~ Thu May 4 20:19:48 2023 +0800 W12-P2: use promise to solve the cb hell problem
4476cc9 21141003901~ Thu May 4 19:18:22 2023 +0800 W12-P1: call back hell DOM demo
```

41a2c4d 21141003901~ Thu May 4 21:48:36 2023 +0800 W12-logs

```
$ git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2023-05-03"
```

41a2c4d 21141003901~ Thu May 4 21:48:36 2023 +0800 W12-logs

3fa1869 21141003901~ Thu May 4 21:46:40 2023 +0800 W12-P5: xhr, get people.json, and show names in browser

53a1ea8 21141003901~ Thu May 4 21:24:20 2023 +0800 W12-P4: xhr, get sample.txt

37e498a 21141003901~ Thu May 4 20:41:15 2023 +0800 W12-P3: use astnc/await to solve the cb hell problem

36f3290 21141003901~ Thu May 4 20:19:48 2023 +0800 W12-P2: use promise to solve the cb hell problem

4476cc9 21141003901~ Thu May 4 19:18:22 2023 +0800 W12-P1: call back hell DOM demo