

## W12-P1: call back hell DOM demo

**Asynchronous Javascript**

hello world  
hello people  
hello Javascript  
hello Async JS

click me

```
const heading1 = document.querySelector('.one');
const heading2 = document.querySelector('.two');
const heading3 = document.querySelector('.three');
const heading4 = document.querySelector('.four');

const btn = document.querySelector('.btn');

btn.addEventListener('click', () => {
  setTimeout(() => {
    heading1.style.color = 'green';
    setTimeout(() => {
      heading2.style.color = 'red';
      setTimeout(() => {
        heading3.style.color = 'purple';
        setTimeout(() => {
          heading4.style.color = 'blue';
        }, 500);
      }, 1000);
    }, 2000);
  }, 1000);
});
```

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

9e763ac7 Jun206 Thu May 4 19:10:01 2023 +0800 0504

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

**Asynchronous Javascript**

hello world  
hello people  
hello Javascript  
hello Jun

click me

```
const heading1 = document.querySelector('.one');
const heading2 = document.querySelector('.two');
const heading3 = document.querySelector('.three');
const heading4 = document.querySelector('.four');

const btn = document.querySelector('.btn');

btn.addEventListener('click', () => {
  addColor(1000, heading1, 'red')
    .then(() => {
      return addColor(2000, heading2, 'green');
    })
    .then(() => {
      return addColor(1000, heading3, 'blue');
    })
    .then(() => {
      return addColor(500, heading4, 'purple');
    })
    .catch((error) => console.log(error));
});

const addColor = (time, element, color) => {
  return new Promise((resolve, reject) => {
    if (element) {
      setTimeout(() => {
        element.style.color = color;
        resolve();
      }, time);
    } else {
      reject(new Error('There is no such e'));
    }
  });
};
```

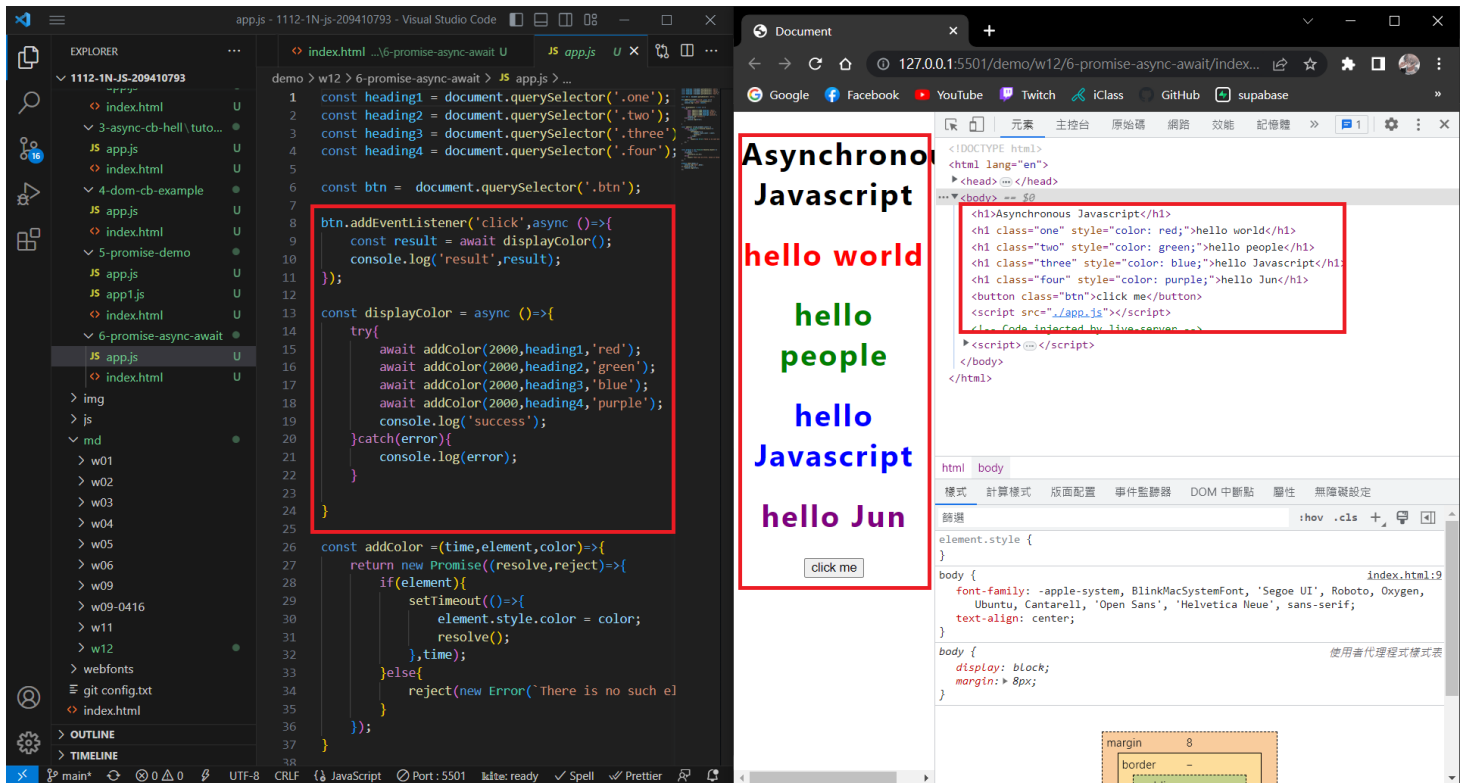
```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Asynchronous Javascript</title>
</head>
<body>
  <div>
    <h1 class="one" style="color: red;">hello world</h1>
    <h1 class="two" style="color: green;">hello people</h1>
    <h1 class="three" style="color: blue;">hello Javascript</h1>
    <h1 class="four"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

947ca3f1

Jun206 Thu May 4 20:54:19 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 development environment with Visual Studio Code on the left and a web browser on the right. The browser shows the rendered output of the code, which is a list of text elements: "Asynchronous Javascript", "hello world", "hello people", "hello Javascript", and "hello Jun". The code in the browser's developer tools is as follows:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Asynchronous Javascript</title>
    <script src="app.js"></script>
  </head>
  <body>
    <h1>Asynchronous Javascript</h1>
    <div class="one">hello world</div>
    <div class="two">hello people</div>
    <div class="three">hello Javascript</div>
    <div class="four">hello Jun</div>
    <button class="btn">click me</button>
  </body>
</html>
```

The code in the browser's developer tools is as follows:

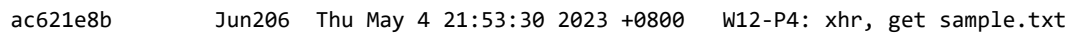
```
body {
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen,
  Ubuntu, Cantarell, 'Open Sans', 'Helvetica Neue', sans-serif;
  text-align: center;
}
body {
  display: block;
  margin: 8px;
}
```

4a0b0eed

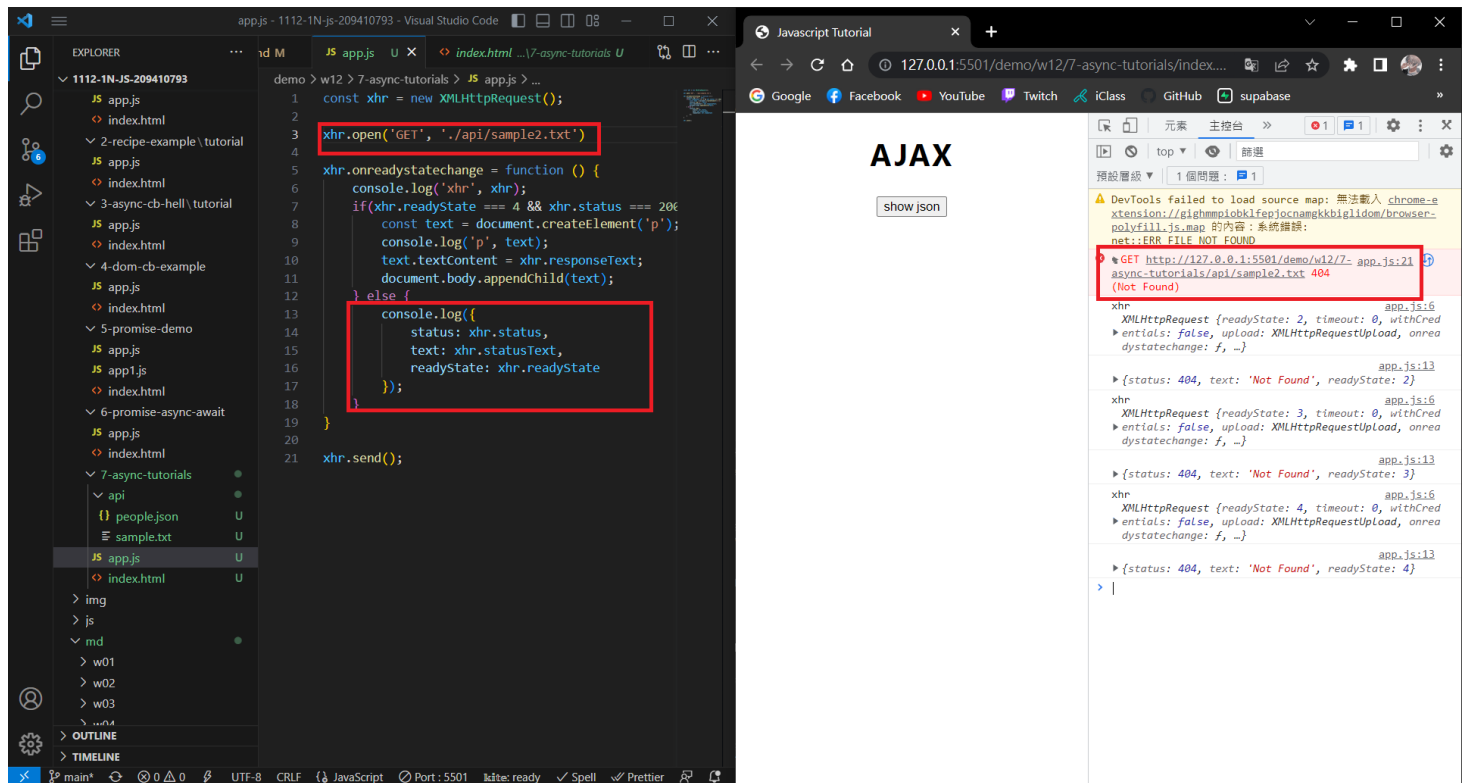
Jun206 Thu May 4 20:59:39 2023 +0800

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

## success reading

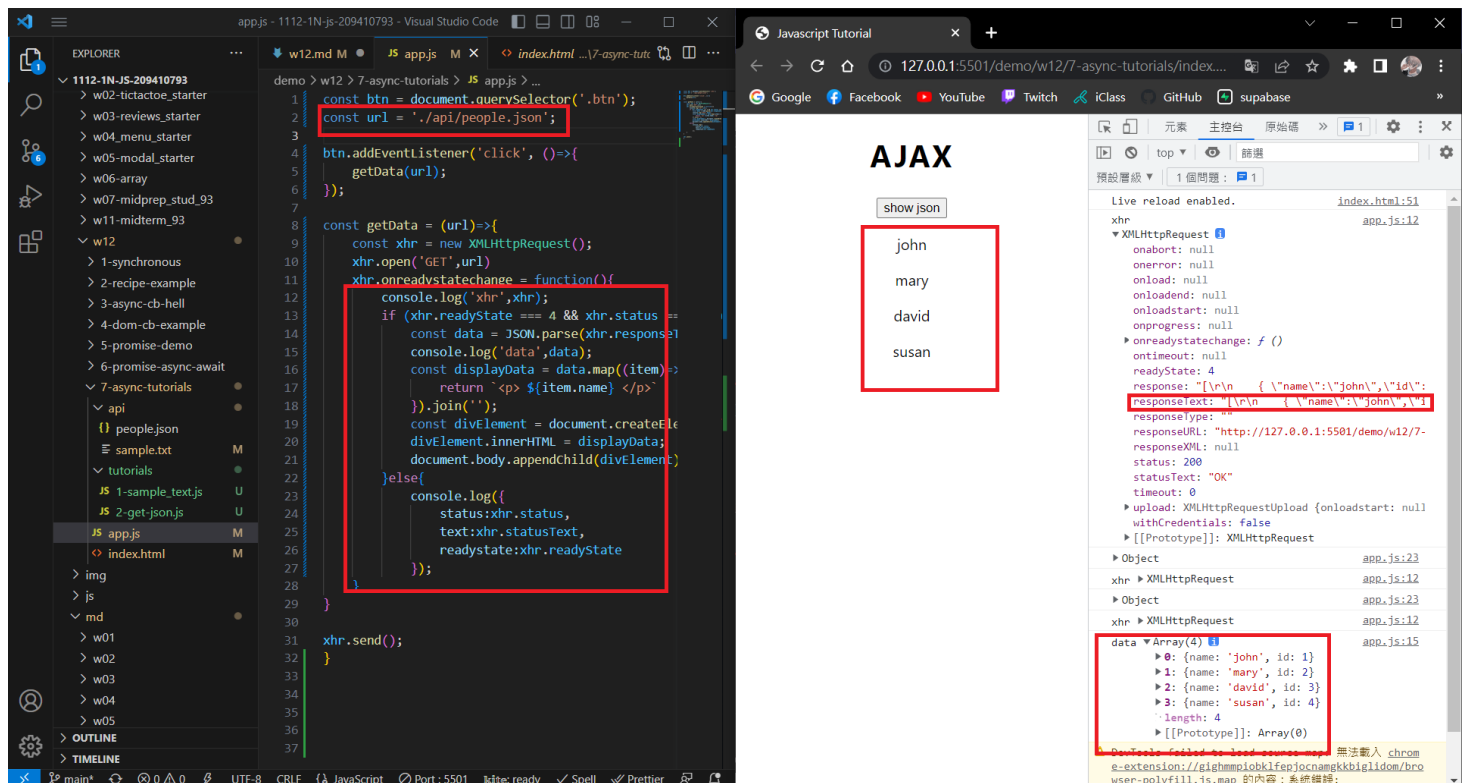


# fail reading



dc65a484 Jun206 Thu May 4 21:54:57 2023 +0800 W12-P4: xhr, get sample.txt #### fail reading

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



205cf84b	Jun206	Thu May 4 22:03:26 2023 +0800	W12-P5: xhr, get people.json, and show names in browser
----------	--------	-------------------------------	---

## W12-logs

```
User@E201-02 MINGW64 /d/1112-1N-js-209410793 (main)
$ git log --pretty=format:"%h%x09%an%x09%ad%x09%s" --after="2023-5-03"
205cf84b      Jun206  Thu May 4 22:03:26 2023 +0800    W12-P5: xhr, get people.json, and show names in browser
dc65a484      Jun206  Thu May 4 21:54:57 2023 +0800    W12-P4: xhr, get sample.txt ##### fail reading
ac621e8b      Jun206  Thu May 4 21:53:30 2023 +0800    W12-P4: xhr, get sample.txt
4a0b0eed      Jun206  Thu May 4 20:59:39 2023 +0800    W12-P3: use async/await to solve the cb hell problem
947ca3f1      Jun206  Thu May 4 20:54:19 2023 +0800    W12-P2: use promise to solve the cb hell problem
9e763ac7      Jun206  Thu May 4 19:10:01 2023 +0800    0504
```

205cf84b	Jun206	Thu May 4 22:03:26 2023 +0800	W12-P5: xhr, get people.json, and show names in browser
dc65a484	Jun206	Thu May 4 21:54:57 2023 +0800	W12-P4: xhr, get sample.txt ##### fail reading
ac621e8b	Jun206	Thu May 4 21:53:30 2023 +0800	W12-P4: xhr, get sample.txt
4a0b0eed	Jun206	Thu May 4 20:59:39 2023 +0800	W12-P3: use async/await to solve the cb hell problem
947ca3f1	Jun206	Thu May 4 20:54:19 2023 +0800	W12-P2: use promise to solve the cb hell problem
9e763ac7	Jun206	Thu May 4 19:10:01 2023 +0800	0504