JAVASCRPT FETCH

Content

- Functions vs callbacks
- Synchronous vs asynchronous
- async & await
- Promises
  - .then
  - .catch
- Static vs dynamic web pages
- Backend & Frontend
- Get vs Post
- Fetch
  - Client initiative
  - Server initiative

Functions vs callbacks

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Current Time Display</title>
  <style>
    body {
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
      font-family: Arial, sans-serif;
      background-color: #f0f0f0;
    }
    #time {
      font-size: 3em;
      color: #333;
    }
  </style>
</head>
```

```
<body>
  <div id="time"></div>

  <script>
    function updateTime() {
      const timeElement = document.getElementById('time');
      const currentTime = new Date().toLocaleTimeString();
      timeElement.textContent = currentTime;
    }

    // Update the time immediately
    updateTime();

    // Update the time every second
    setInterval(updateTime, 1000);
  </script>
</body>
</html>
```

Synchronous vs asynchronous

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Async Fetch Example</title>
</head>
<body>
  <h1>Async Fetch Example</h1>
  <button id="fetch-1">Fetch  Data 1</button>
  <button id="fetch-2">Fetch  Data 2</button>
  <p id="result"></p>
  <script>
...
  </script>
</body>
</html>
```

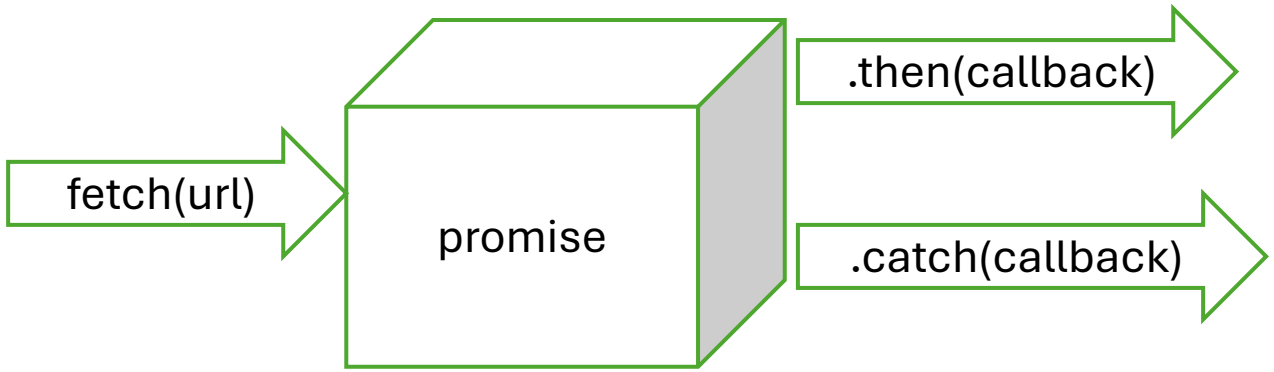
```
document.getElementById('fetch-1').addEventListener('click', async () => {
    try {
      const response = await fetch('https://jsonplaceholder.typicode.com/posts/1');
      const data = await response.text();
      console.log(`Async Fetch Result: ${data}`)

    } catch (error) {
      console.error('Error fetching data asynchronously:', error);
    }
   console.log('End 1');
});

document.getElementById('fetch-2').addEventListener('click', () => {
    const fetchPromise = fetch('https://jsonplaceholder.typicode.com/posts/1')
      .then(response => response.text())
      .then(data => {
        console.log(`Async Fetch Result: ${data}`)
      })
      .catch(error => {
        console.error('Error fetching data :', error);
      });

    console.log('End 2');
});
```

Promises




```
const fetchPromise = fetch('https://jsonplaceholder.typicode.com/posts/1')
        .then(response => response.text())
        .then(data => {console.log(` Async Fetch Result: ${data}` ) })
        .catch(error => {console.error('Error fetching data :', error); }
);
```

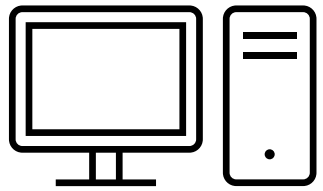
Static (frontend) vs dynamic (frontend + backend) web pages

Frontend

- HTML+ JavaScript +CSS
- The process runs in the browser (client)
- Graphics or immediate calculus
- Example: Calculator

Backend

- Python or others
- The process is in the server
- Database operations
- Example: Google search



Get vs Post

Get

- Data is sent in the URL
  - https://www.google.com/search?q=udfjc
- Limited amount of data can be sent.

Post

- Data is sent in the request body, not in the URL.
  - Upload or download a file
- Can send a large amount of data.

Fetch

```
import socket
import time
import board
import digitalio
import adafruit_requests as requests
from adafruit_esp32spi import adafruit_esp32spi
from adafruit_esp32spi import adafruit_esp32spi_wifimanager

# Setup ESP32 as Wi-Fi module
esp32_cs = digitalio.DigitalInOut(board.D10)
esp32_ready = digitalio.DigitalInOut(board.D9)
esp32_reset = digitalio.DigitalInOut(board.D5)
esp = adafruit_esp32spi.ESP_SPIcontrol(board.SPI(), esp32_cs, esp32_ready, esp32_reset)
wifi = adafruit_esp32spi_wifimanager.ESPSPI_WiFiManager(esp, secrets)
wifi.connect()

# Set up the server
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
server = socket.socket()
server.bind(addr)
server.listen(1)
print('Listening on', addr)

# Handle requests
def handle_client(client):
    request = client.recv(1024)
    print(request)

    # Parse the GET request (if needed)
    request = str(request, 'utf-8')
    request_line = request.split(' ')[1]  # Get the request path

    if request_line == '/data':
        response = {
            'message': 'Hello from Raspberry Pi Pico!',
            'timestamp': time.time()
        }

        # Create the HTTP response
        client.send('HTTP/1.1 200 OK\n')
        client.send('Content-Type: application/json\n')
        client.send('Connection: close\n\n')
        client.sendall(json.dumps(response))
    else:
        client.send('HTTP/1.1 404 Not Found\n')
        client.send('Content-Type: text/html\n')
        client.send('Connection: close\n\n')
        client.sendall('<h1>404 Not Found</h1>')

    client.close()

while True:
    client, addr = server.accept()
    print('Client connected from', addr)
    handle_client(client)
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Raspberry Pi Pico Frontend</title>
</head>
<body>
    <h1>Data from Raspberry Pi Pico</h1>
    <p id="data">Loading...</p>

    <script>
        async function fetchData() {
            try {
                const response = await fetch('/data');
                const data = await response.json();
                document.getElementById('data').innerHTML = `Message: ${data.message}, Timestamp: ${new Date(data.timestamp * 1000)}`;
            } catch (error) {
                document.getElementById('data').innerHTML = 'Error fetching data';
                console.error('Error:', error);
            }
        }

        fetchData();
    </script>
</body>
</html>
```