POST Request in JS:

In JavaScript, a POST request can be made using the XMLHttpRequest object (XHR) or the Fetch API.

Using the XMLHttpRequest object:

```
let xhr = new XMLHttpRequest();
xhr.open("POST", "http://example.com/submit", true);
xhr.setRequestHeader("Content-Type", "application/json");
xhr.onreadystatechange = function() {
  if (xhr.readyState === 4 && xhr.status === 200) {
    console.log(xhr.responseText);
  }
};
let data = { name: "John Doe", age: 25 };
xhr.send(JSON.stringify(data));
```

This example creates a new XMLHttpRequest object, opens a connection to the specified URL and sets the request method to "POST". The request headers are set to "Content-Type: application/json" and the data is sent as a stringified JSON object.

Using the Fetch API:

```
let data = { name: "John Doe", age: 25 };
fetch("http://example.com/submit", {
  method: "POST",
  headers: {
    "Content-Type": "application/json"
  },
  body: JSON.stringify(data)
})
.then(response => response.json())
.then(data => {
  console.log(data);
});
```

Async and Await in JS:

In JavaScript, async and await are two keywords that are used to handle asynchronous code in a more synchronous and readable way. They are often used together to make working with promises and asynchronous code simpler.

async is a keyword that is used to declare a function as asynchronous. An asynchronous function is a function that returns a promise and can be paused with the await keyword.

```
async function example() {
  // asynchronous code here
}
```

await is a keyword that is used to pause the execution of an asynchronous function until a promise is resolved. It can only be used within an async function. The await keyword is placed before the promise that we want to wait for.

```
async function example() {
  let response = await fetch("http://example.com/data.json");
  let data = await response.json();
  console.log(data);
}
```

In this example, the await keyword is used to wait for the promise returned by the fetch function to be resolved, and then it waits for the promise returned by the response.json() method to be resolved. This allows the code to be written in a more synchronous and readable way.

It's worth noting that when an error occurs in a function that contains the await keyword, the error will be caught by the nearest enclosing try-catch block or rejected by the returned promise.

async and await are useful for making asynchronous code easier to work with and they make the code more readable and maintainable. They are supported in most modern browsers, and they are widely used in JavaScript libraries and frameworks.