Asynchronous JavaScript is a programming paradigm that allows non-blocking execution of code, typically used in scenarios like handling asynchronous operations such as fetching data from an API, reading files, or waiting for user input. Here are four examples of asynchronous JavaScript code with explanations:

1. Callbacks:

```javascript
function fetchData(callback) {
  setTimeout(() => {
    const data = "Async data";
    callback(data);
  }, 1000);
}

function processData(data) {
  console.log("Processing:", data);
}

fetchData(processData);
```

Explanation:
- `fetchData` simulates an asynchronous operation using `setTimeout`.
- It takes a callback function (`processData`) as an argument.
- After the timeout, it invokes the callback with the fetched data.

2. Promises:

```javascript
function fetchData() {
  return new Promise((resolve) => {
    setTimeout(() => {
      const data = "Async data";
      resolve(data);
    }, 1000);
  });
}

fetchData().then((data) => {
  console.log("Data:", data);
});
```

Explanation:
- `fetchData` returns a Promise that resolves after a timeout.
- The `then` method is used to handle the resolved value when the Promise is fulfilled.

3. Async/Await:

```javascript
function fetchData() {
  return new Promise((resolve) => {
    setTimeout(() => {
      const data = "Async data";
      resolve(data);
    }, 1000);
  });
}

async function processData() {
  const data = await fetchData();
  console.log("Data:", data);
}

processData();
```

Explanation:
- `fetchData` returns a Promise.
- `processData` is an asynchronous function using `async/await` to simplify Promise handling.

4. Fetch API:

```javascript
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log('Data:', data))
  .catch(error => console.error('Error:', error));
```

Explanation:
- The `fetch` function is used to make an asynchronous network request.
- Chaining `then` is used to handle the response, and `catch` handles errors.
- This example fetches JSON data from a hypothetical API.

These examples demonstrate different approaches to handle asynchronous operations in JavaScript using callbacks, promises, async/await, and the Fetch API.