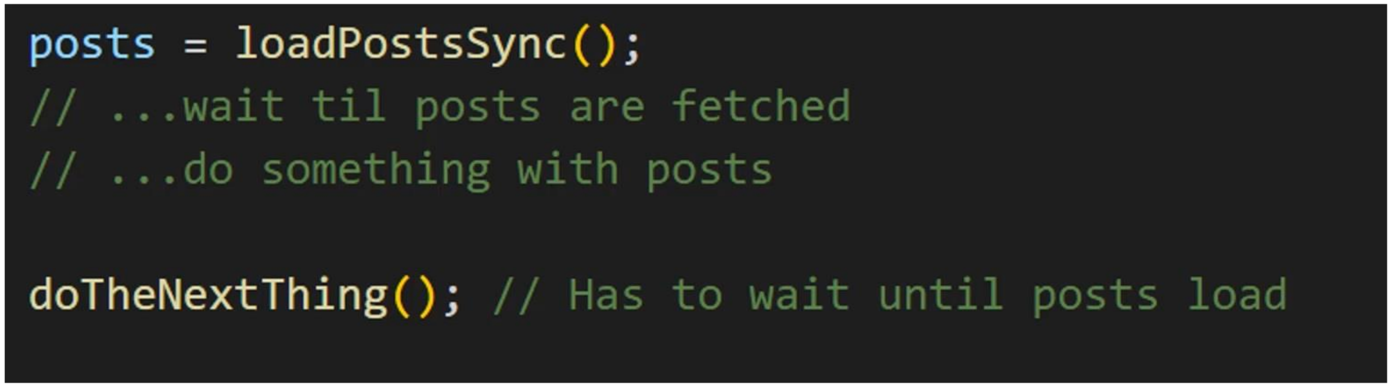
JS Asynchronous Programming

Synchronous Code



```
posts = loadPostsSync();
// ...wait til posts are fetched
// ...do something with posts

doTheNextThing(); // Has to wait until posts load
```

Asynchronous Code

```
loadPostsAsync(function () {
  // ...wait til posts are fetched
  // ...do something with posts
});

doTheNextThing(); // Doesn't have to wait until posts load
```

Browser/Server APIs

Most Async code you work with will be part of an API or library

- XMLHttpRequest & Fetch
- jQuery Ajax, Axios, other HTTP libraries
- Node.js fs (filesystem) module

JS Handling Async Code

There are a few ways to work with Async code

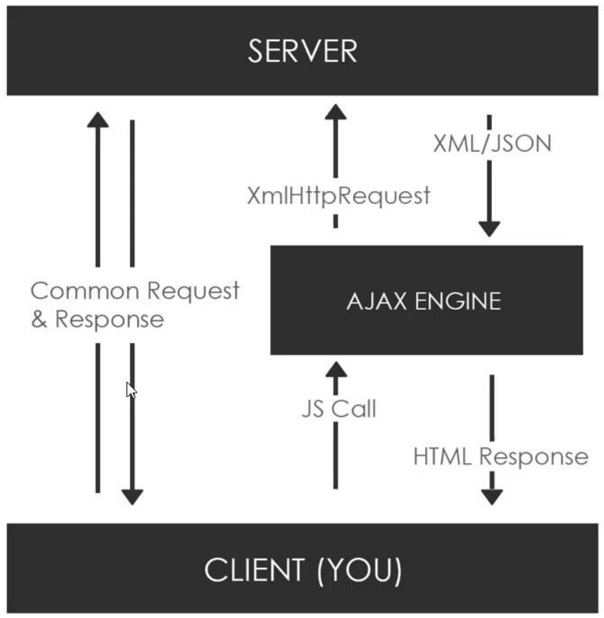
- Callbacks
- Promises
- Async/Await

What Is Ajax?

- ✓Asynchronous JavaScript & XML
- ✓ Set of web technologies
- ✓Send & Receive data asynchronously
- ✓Does not interfere with the current page
- ✓JSON has replaced XML for the most part

What Is Ajax?

- ✓ Make async requests in the background
- ✓ No page reload / refresh
- ✓ Fetch data
- ✓ Very interactive



JS XmlHttpRequest (XHR) Object

- ✓ API in the form of an object
- ✓ Provided by the browsers JS environment
- ✓ Methods transfer data between client / server
- ✓ Can be used with other protocols than HTTP
- ✓ Can work with data other than XML (JSON, plain text)

Libraries & Other Methods

- ✓Fetch API
- ✓Axios
- ✓Superagent
- ✓jQuery
- ✓Node HTTP

What Is An API?

- ✓**Application Programming Interface**
- ✓Contract provided by one piece of software to another
- ✓Structured request and response
- ✓We just worked with an API that takes a request and responds with jokes

REST APIS

- ✓ **Representational State Transfer**
- ✓ Architecture style for designing networked applications
- ✓ Relies on a stateless, client-server protocol, almost always HTTP
- ✓ Treats server objects as resources that can be created or destroyed
- ✓ Can be used by virtually any programming language
- ✓ All APIS have their own rules and structure

JS HTTP Requests

- ✓ **GET: Retrieve data from a specified resource**
- ✓ **POST: Submit data to be processed to a specified resource**
- ✓ **PUT: Update a specified resource**
- ✓ **DELETE: Delete a specified resource**

- ✓ HEAD: Same as get but does not return a body
- ✓ OPTIONS: Returns the supported HTTP methods

API Endpoints

GET	https://someurl.com/api/users	// Get all users
GET	https:// someurl.com/api/users/1	// Get single user
POST	https:// someurl.com/api/users	// Add user
PUT	https:// someurl.com/api/users/1	// Update user
DELETE	https:// someurl.com/api/users/1	// Delete user