Libraries / Frameworks

# Intro to ReactJS

access101

# Libraries / Frameworks

# Question…

## Can’t we build web apps without libraries / frameworks?

# Answer…

## Yes. But… It comes with it’s own complexities and it’s imperative.

# Imperative vs Declarative

```
// Vanilla JS (imperative)

button.onclick = function(e){
  document.getElementById("target-element").innerHTML = e.target.dataset.text
}
```

**React (Declarative)**

```
handleClick(e){
  this.setState({text: "NEW TEXT OR WHATEVER"})
}

render() {
  const results = this.state.activeTerms.map((term) => {
    return <div>{term}</div>
  })
  return (
    <div>
      <button onClick={this.handleClick}>Click Me</button>
      <div id="target-element">
        {this.state.text}
      </div>
    </div>
  );
}
```

# Libraries / Frameworks

It's an established piece of code that takes care of a lot of the grunt work in building a Web application so you can focus on the important pieces rather than re-inventing the wheel.

# Intro to **React**.

## React is a UI library developed and maintained by facebook.

# Getting started with **React**.

access101

# Prerequisites

- HTML5
- CSS3
- Javascript
- ES6

# Features of React

- Blazing fast as it uses virtual DOM
- Component structure
- Boosts productivity of the developer
- Easily to jump on react-native

# ReactDOM & JSX

# Props in react

- **props** is a special keyword in **React.**
- Used for passing down data from one component to another.
- **props** data is read-only.

# Array helper methods

- **Map** will loop through each item of array, same like **forEach** but Map returns the value of the array.
- **Filter** will return array based on the boolean of the comparison.
- **Find** will return the record if a particular element is found in the array.
- **Reduce** will is the most flexible helper, we can probably reimplement all the other helpers by just using reduce.

# State in react

- **State** is a built in object in **class components.**
- **State** object change triggers component re-render.
- **State** is mutable.

# Modular Export

- **Named export** means the key should have **name** . Hence a "named" export lol 😂.
- With a **Default export**, you don't need any name. Because you can name it whatever you want

# Lifecycle of Components

The three phases are: **Mounting**, **Updating**, and **Unmounting**.

- **Mounting Phase:** Mounting means putting elements into the **DOM.**
- **Updating Phase** means is when a component is updated.
- **Unmounting Phase** means when a component is removed from the DOM.

# 1.Mounting

Mounting means putting elements into the DOM.

- **Constructor()** method is called before anything else, when the component is initiated.
- **render** method is required, and is the method that actual outputs HTML to the DOM..
- **componentDidMount** method is called after the component is rendered..

# 2.Updating

Mounting A component is updated whenever there is a change in the component's **state** or **props**.

- **render** method method is required and will always be called.
- **componentDidUpdate** method is called after the component is updated.

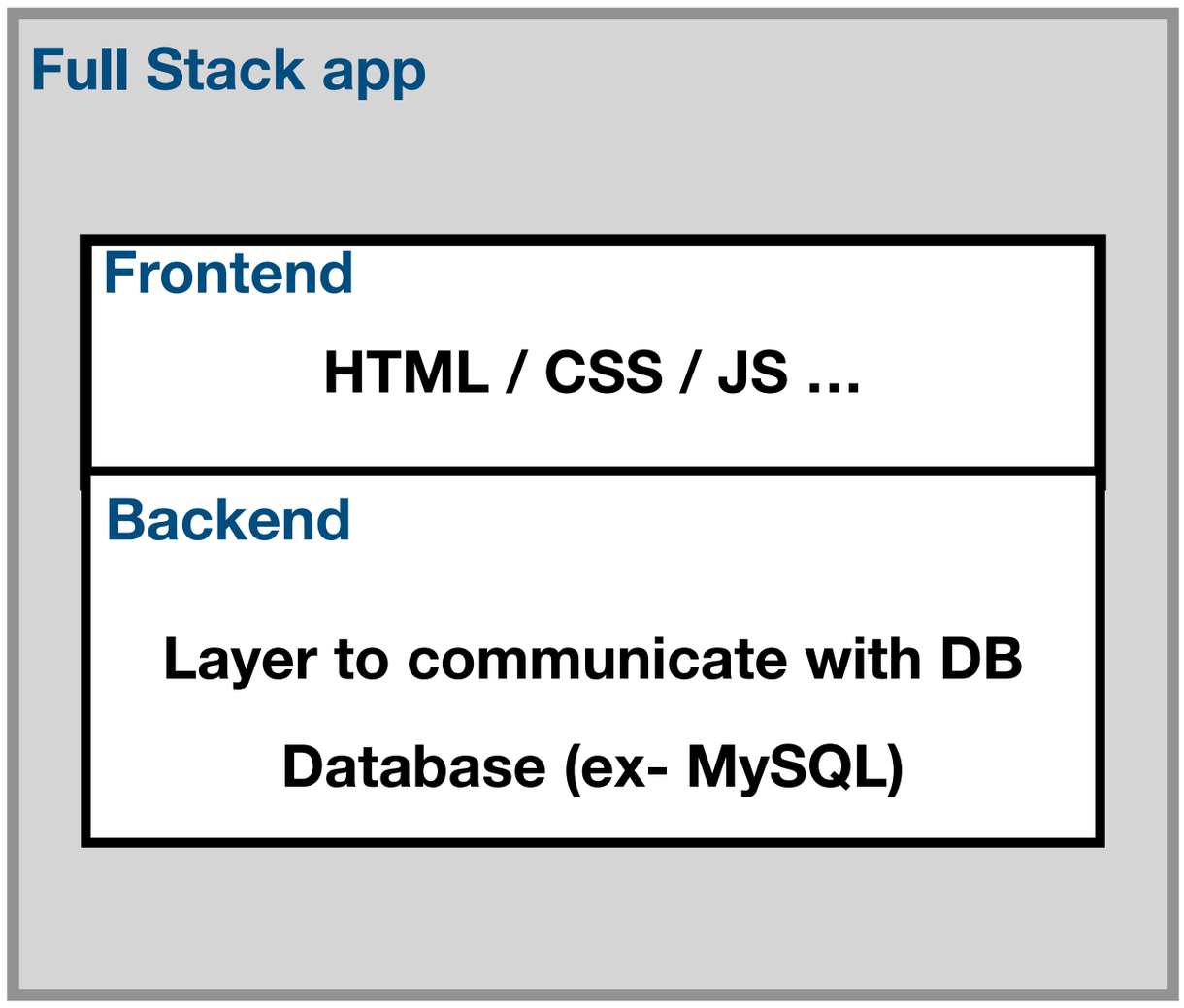
# 3. Unmounting

- **componentWillUnmount** method is called when the component is about to be removed from the DOM.

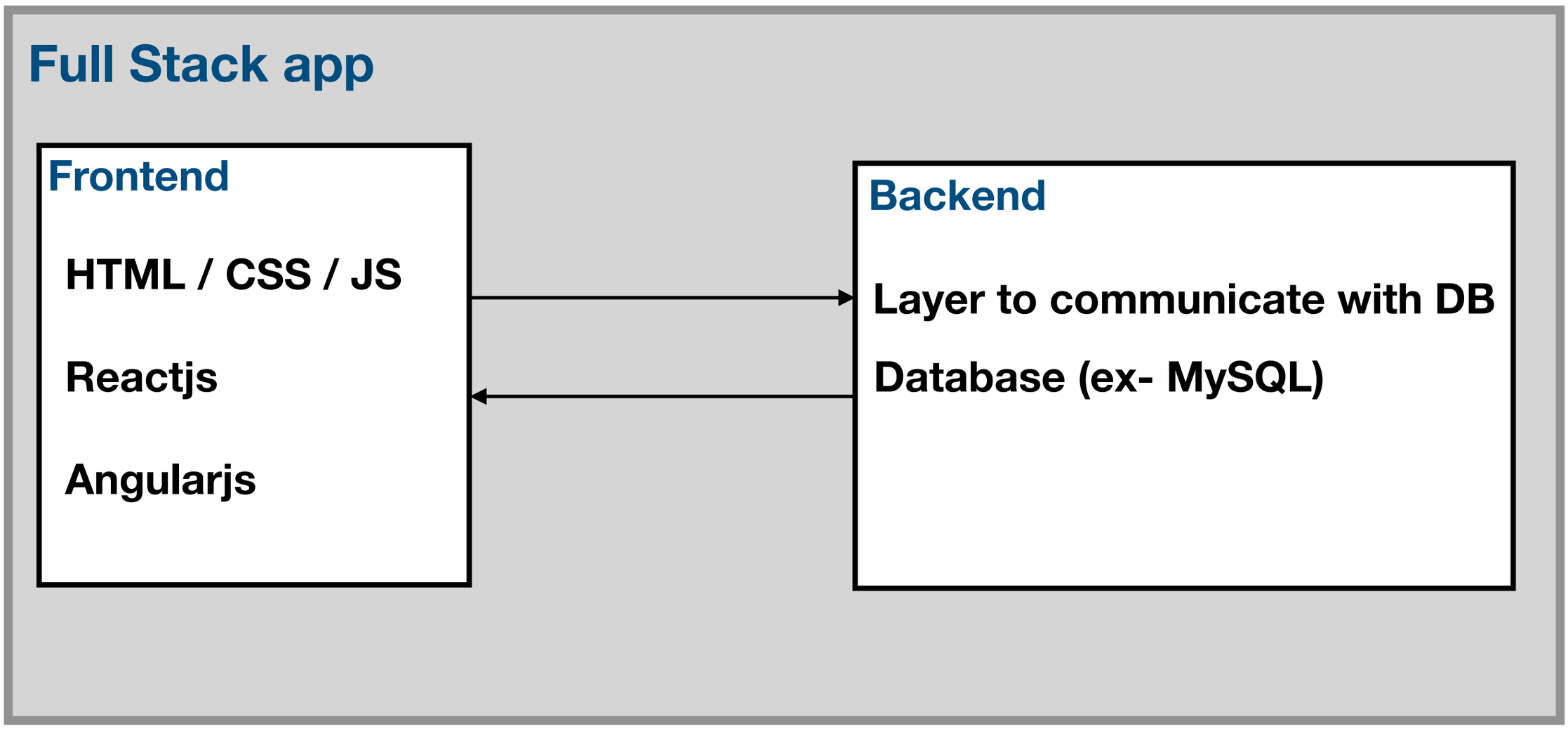
# Introducing **RESTful API**

# (a.k.a **REST API**)

# Full stack application - Early days



# Full stack application - Nowadays



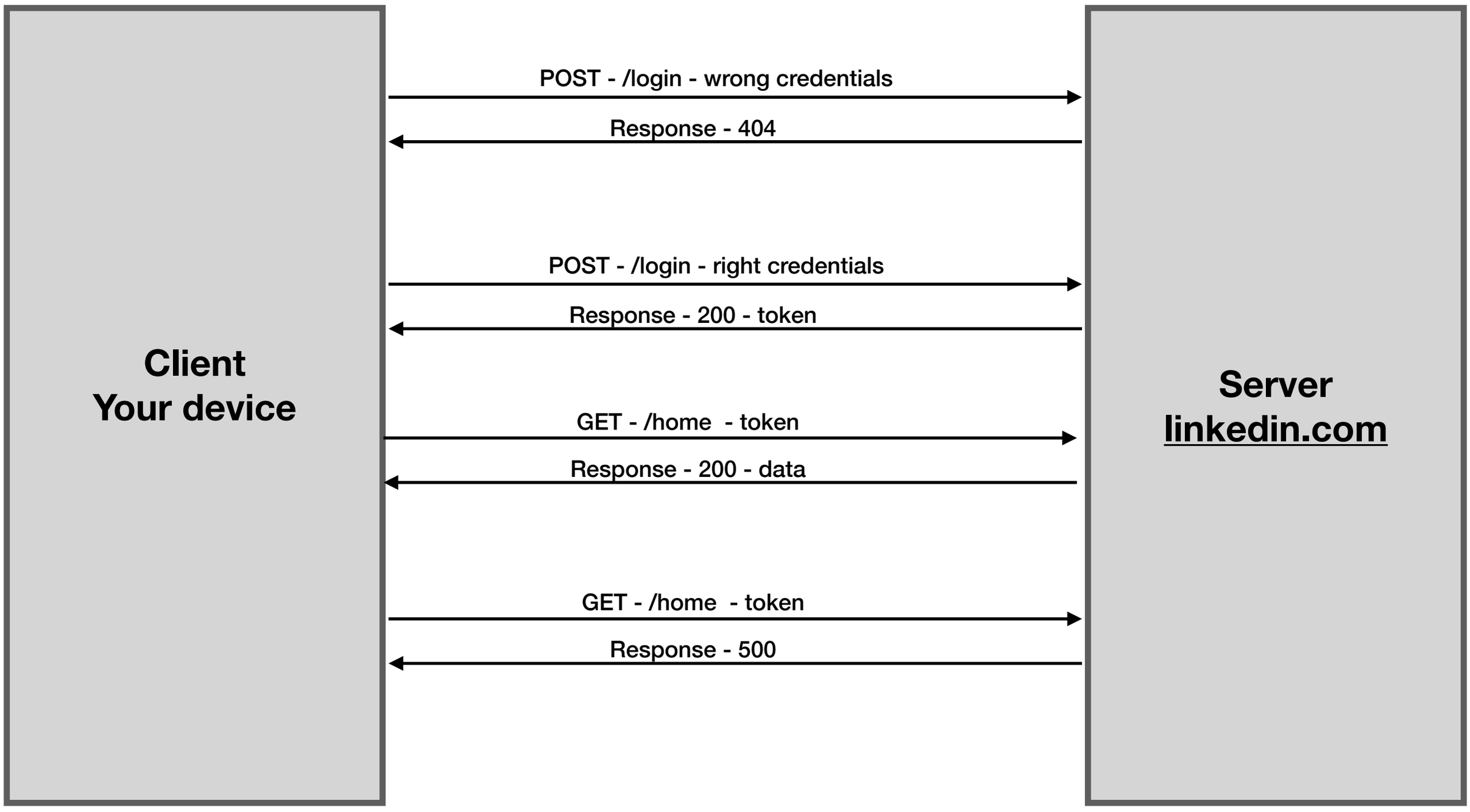
# **HTTP Methods** - Conventions (rules)

- **GET** - Retrieves data from server
- **POST** - Posts data to the server
- **PUT / PATCH** - Updates the record
- **DELETE** - Deletes the record

# **HTTP** - Error codes

- **4XX** - 400, 401, 404... - Client side errors
- **5XX** - 500, 501, 502... - Server side errors
- **3XX** - 300, 301, 302... - Redirection
- **2XX** - 200, 201, 202... - Success
- 1**XX** - 100, 101, 102... - Informational

# Let's simulate



# Promise and Fetch

# Promise

A **promise** is an object that may produce a single value some time in the future.

It has 3 states

- Resolved
- Reject
- Pending

# Fetch

The **Fetch API** is a promise-based Javascript API for making asynchronous HTTP requests in the browser.

# Let's build a **news app**...

# Announcement