MODULE 4

Web Services (GET)

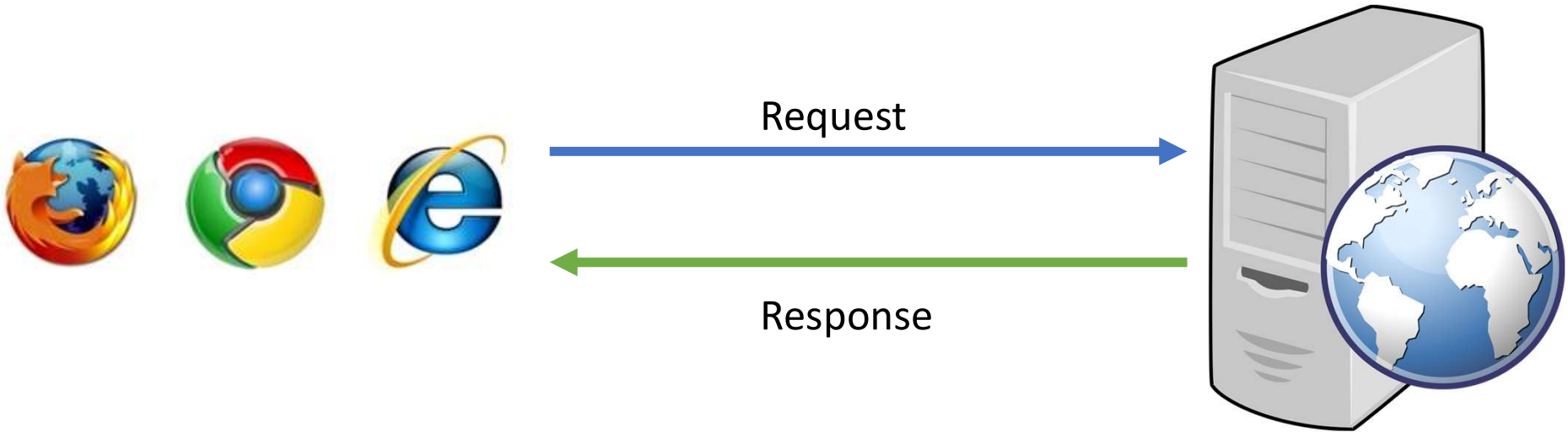
Web Services and APIs

- Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks.

- An API (Application Programming Interface) is a set of features and rules that exist inside a software program (the application) enabling interaction with it through software

HTTP and HTTPS

- HTTP: Hypertext transfer protocol
  - How browsers and servers communicate with each other
  - Defines a simple request/response protocol



Key Elements of the Response Stream

- HTTP Status code
  - 1xx Informational
  - 2xx Success (200 OK)
  - 3xx Redirect (301 Moved Permanently)
  - 4xx Client Error (400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found)
  - 5xx Server Error (500 Internal Server Error, 501 Not Implemented)

Creating and Consuming

- **Creating** a web service is simply exposing methods and properties from classes
- **Consuming** a web service is calling those APIs and getting the data.

Response Data

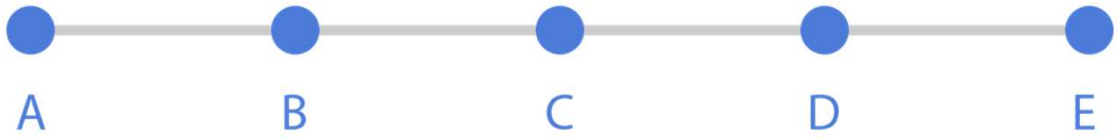
- JSON (JavaScript Object Notation) is a lightweight data-interchange format.

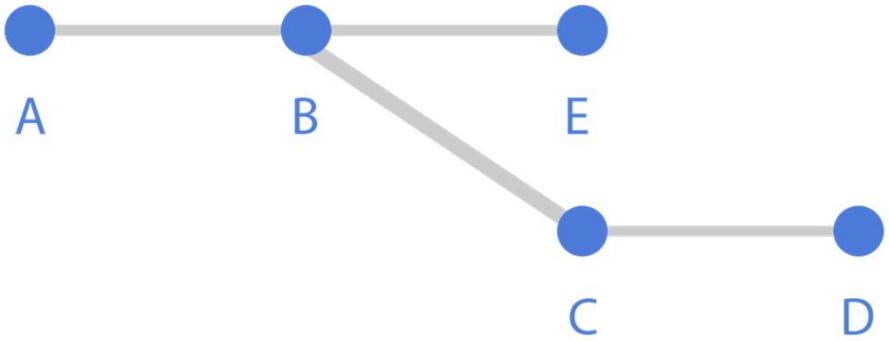

```
{
  "id": 1,
  "name": "Aloft Cleveland",
  "address": {
     "id": "69006b81-7f58-4acc-a10e-f9f87affae5f",
     "address": "1111 W 10th St",
     "address2": "",
     "city": "Cleveland",
     "state": "Ohio",
     "zip": "44113"
  },
  "stars": 3,
  "roomsAvailable": 48,
  "costPerNight": 274,
  "coverImage": "aloft-cleveland.webp"
}
```

Asynchronous Programming

- Our programs: Synchronous

Asynchronous Programming





Axios

- Promise based HTTP client for the browser and node.js
- Another Framework that makes async a little easier

Getting Axios in Vue.js

- Need to add package:
  - npm install axios
- Need to import the code
  - Import axios from ‘axios’
- That’s it!

Axios Syntax

```
axios.get('/user?ID=12345')
  .then(function (response) {
    // handle success
    console.log(response);
  })
  .catch(function (error) {
    // handle error
    console.log(error);
  })
```

```
axios.post('/user', {
    firstName: 'Fred',
    lastName: 'Flintstone'
  })
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

Promises, promises.

- I promise I will return!
- Three states:
  - Pending: initial state, neither fulfilled nor rejected.
  - Fulfilled: meaning that the asynchronous operation completed successfully.
  - Rejected: meaning that the asynchronous operation failed.
- Use .then() to access functions when promise returns
- Use .catch() for errors

LET'S CODE!

WHAT QUESTIONS DO YOU HAVE?

Reading for tonight:

Web Services POST