Front End
Software Development

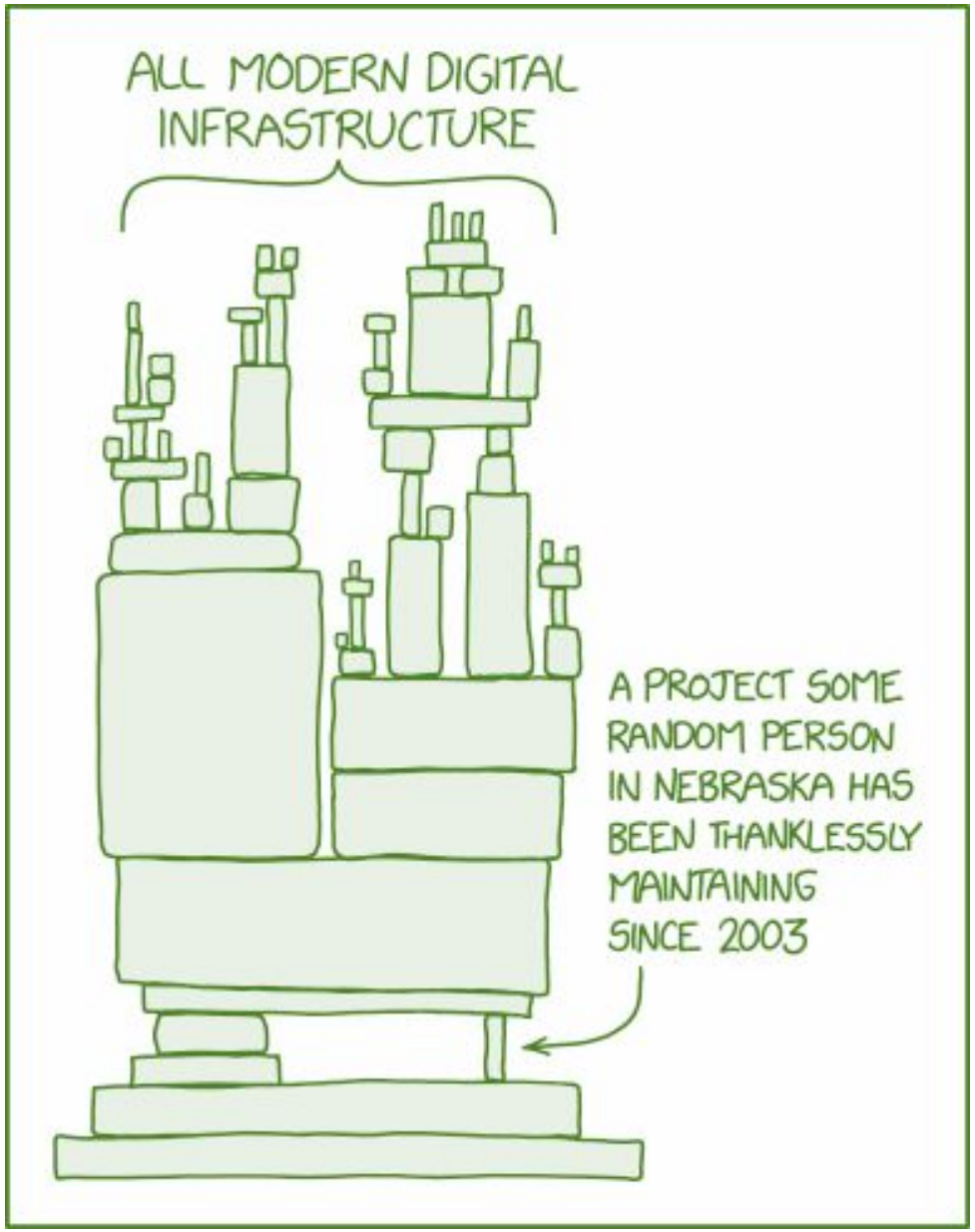
Web App Design w/React (weeks 13 - 18)

Week 03

Agenda

- Questions
- Web Services
  - REST vs SOAP
    - HTTP Methods
  - Tools / Resources
  - HTTP Requests with Fetch
- Functional Components
- Spread Operator
- Destructuring
- Hooks
- Async/Await

Questions


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

Web Services

- **H**yper**T**ext **T**ransfer **P**rotocol
- Web Service standards
  - **S**imple **O**bject **A**ccess **P**rotocol *(SOAP)*
    - Legacy, fallen into disfavor
    - Typically uses XML

```
<person id="11122565">
  <lastName>Smith</lastName>
  <firstName>John</firstName>
</person>
```

During the late 90s, early 2000s SOAP and XML were all the rage.

  - **RE**presentational **S**tate **T**ransfer *(REST)*
    - Roy Fielding; Dissertation, 2000 https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
    - Typically uses JSON (**J**ava**S**cript **O**bject **N**otation)

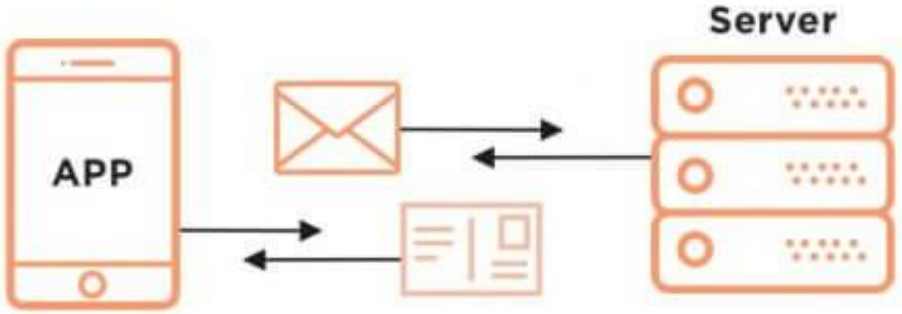
```
{
  id: "11122565",
  lastName: "Smith",
  firstName: "John"
}
```

REST vs SOAP

SOAP vs. REST APIs

SOAP IS LIKE USING AN ENVELOPE

Extra overhead, more bandwidth required, more work on both ends(sealing and opening).



REST IS LIKE A POSTCARD

Lighterweight, can be cached, easier to update

Source: https://assets-global.website-files.com

HTTP Verb Methods

- POST - **C**reate
- GET - **R**ead
- PUT - **U**pdate
- DELETE – **D**elete
- Others (not common)
  - PATCH
  - HEAD

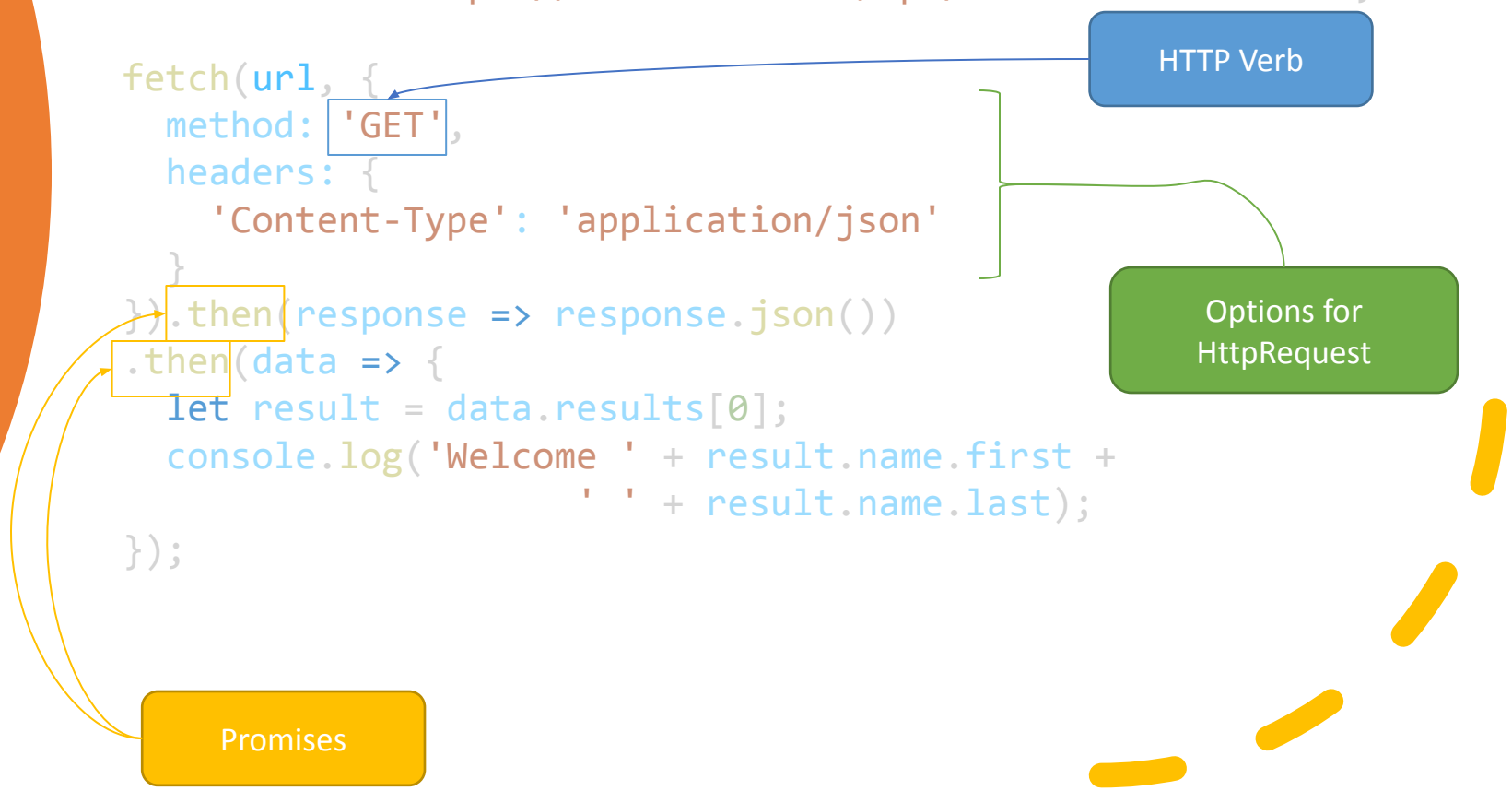
Tools / Resources

- Client Tools
  - Postman
    - http://www.getpostman.com/
  - SoapUI
    - https://www.soapui.org/downloads/soapui/
    - Useful for SOAP based services
- Browser Addons
  - JSONView
- Command Line
  - `wget`
  - `curl`
    Included with Windows 10 (version 1803 or later)
    - (Pre-Build 1803) https://curl.haxx.se/windows/

HTTP Requests with Fetch

- Random User - https://randomuser.me/documentation

```
const url = "https://randomuser.me/api/?results=1&noinfo";

fetch(url, {
  method: 'GET',
  headers: {
    'Content-Type': 'application/json'
  }
}).then(response => response.json())
.then(data => {
  let result = data.results[0];
  console.log('Welcome ' + result.name.first +
                        ' ' + result.name.last);
});
```



Functional Components

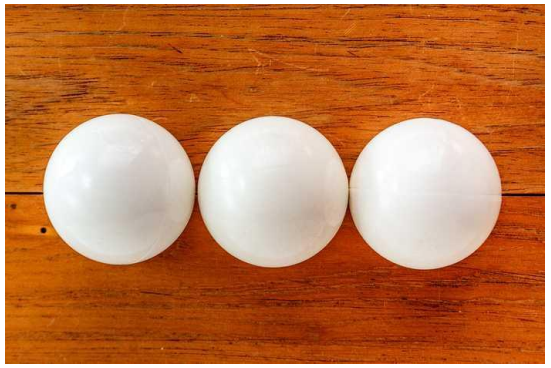
- Stateless Component
  - No state.
  - No lifecycle.
  - Simply a function that returns JSX


```
function Avatar(props) {
  const { avatar, name } = props;
  return(
    <div class="avatar">
      <img src={ avatar } />
      <span>{ name }</span>
    </div>
  );
}
```

Function Name (CamelCase)

JSX

Spread Operator



- `...variable`
  - Array / string (Iterable)
    - Allows value to be expanded in places where zero or more arguments are expected.

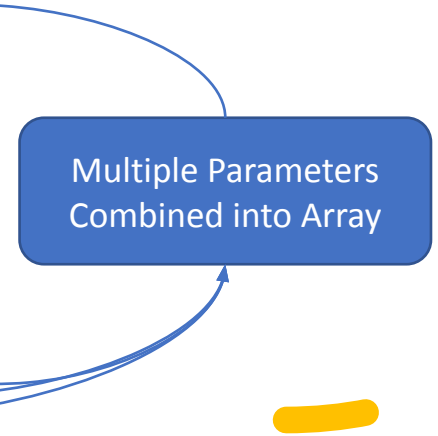
```
function add(a, b, c) {
  return a + b + c;
}
const numbers = [1, 2, 3];

add(...numbers); // add(1, 2, 3);
```

  - Object
    - Expands object properties where zero or more key-value pairs are expected.

```
const dog = { name: 'Fido', age: 1 };
const clonedDog = { ...dog, color: 'black' };
// { name: 'Fido', age: 1, color: 'black' }
```

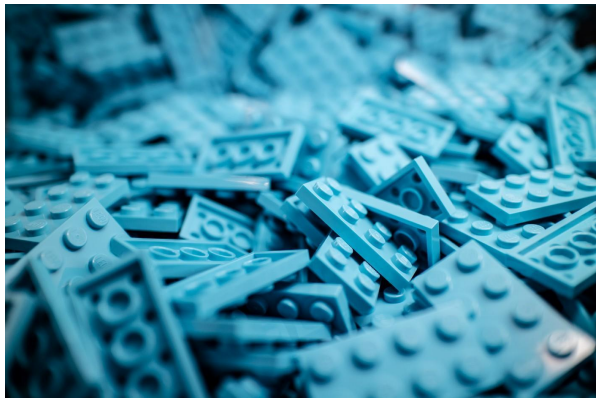
  - Rest

```
function bark(...dogs) {
  for(let dog of dogs) {
    dog.bark();
  }
}
barK(fido, spot, lassie);
```



Destructuring

- Allows array elements and/or object properties to be extracted into separate variables.
  - Arrays



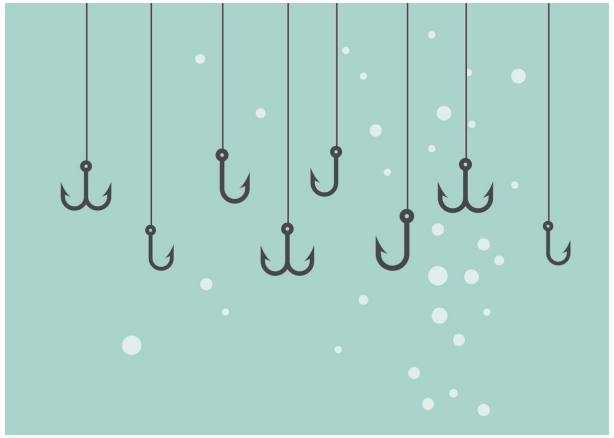
```
let [ X, Y, Z ] = [ 1, 2, 3, 4, 5 ];
```

Ignored

  - Objects

```
let person = {
  first: 'John',
  last: 'Smith',
  age: 47
}
let { first, last } = person;
```

Hooks



- What are?
  - A special function that lets you “hook into” React features.
  - Added in React 16.8
  - Not usable inside classes.
- `useState` – *Access state*

```
function LikeButton() {
  const [ like, setLike ] = React.useState(false);

  return(
    <div className="display-1">
      <i className={`bi bi-hand-thumbs-${like ? 'up' : 'down'}`}
         onClick={() => setLike((prev) => ! prev)}></i>
    </div>
  );
}
```

- `useEffect` – *Perform side effects*

Async/Await

- "Replaces" *callbacks* and Promises (`.then`/`.catch`)
  - But **why**?!?!
    - Easier to read!
    - More like other programming languages
- `async` / `await`
  - `await` calls **MUST** be located in functions marked with `async` keyword

```
const url = "https://randomuser.me/api/?results=1&noinfo";

const getUser = async () => {
  const response = await fetch(url);
  const users = await response.json();
  console.log(users.results[0]);
};
getUser();
console.log('What is taking you so long...');
```

Asynchronous method / keyword

Pause / Wait for Return

DEMO

APIs, The ❤️ of your Application