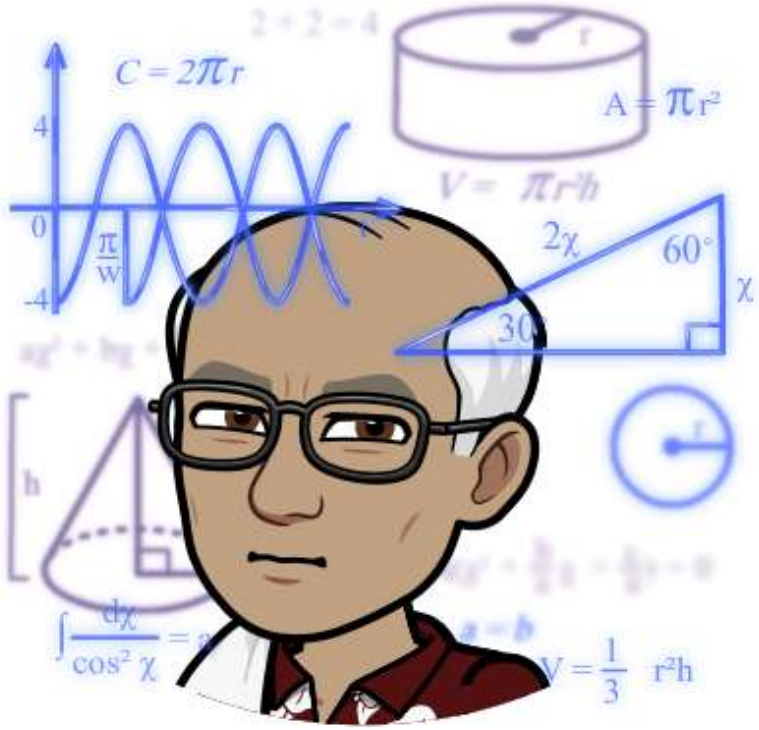
MODULE 3

Callback functions

TECH ELEVATOR
A Stride Company

What is a function?



ELEVATE YOURSELF

JavaScript functions

```
function multiplyBy(multiplicand, multiplier) {
    let result = multiplicand * multiplier;

    return result;
}
```

Name

Parameters

- **descriptive** - it should be clear what type of action or calculation the function performs when invoked
- **camelCase** - the first letter of the name is lowercase and the first letter of each subsequent word is uppercase
- **unique** - function names need to be unique across all JavaScript code that is loaded into the page. If a name conflicts with another function, the one that's loaded last will overwrite the other one

Function Parameters

```
function multiplyBy(multiplicand, multiplier) {
  let result = multiplicand * multiplier;


  return result;
}
```

- Can set defaults
- Where are the parameter data types?
- Parameters are optional!

We don’t need no stinkin’ parameters

- You can give a function as many parameters as you want.
- If no parameters are defined, you can still pass parameters.
- Use arguments array to access them.
- Have to use Array.from(arguments) to use any of the array functions (those are later).

```
function concatAll() { // No parameters defined, but we still might get some
  let result = '';
  for(let i = 0; i < arguments.length; i++) {
    result += arguments[i];
  }
  return result;
}
```

We don’t need no stinkin’ parameters

- JavaScript rest parameter
- ...<variable name>
- Takes all incoming variables and puts then in an array.
- **No need for Array.from()**

```
function concatAll(...incoming) {let result = '';
   for(let i = 0; i < incoming.length; i++) {
      result += incoming[i];
   }
   return result;
}
```

Anonymous Functions

Parameters

Fat Arrow

```
(multiplicand, multiplier) => {
    let result = multiplicand * multiplier;

    return result;
}
```

Anonymous Functions

```
let multiply = (multiplicand, multiplier) => {
    let result = multiplicand * multiplier;

    return result;
}

console.log( multiply(5, 2) ); // Prints `10` to the console
```

Anonymous Functions

```
// Filter an array of numbers so that we are only left with even numbers
let numbers = [1, 2, 3, 4];

let evenNumbers = numbers.filter( (number) => {
    return number % 2 === 0;
});

console.log( evenNumbers ); // Prints out `[2, 4]`
```

Anonymous Functions

```
// Filter an array of numbers so that we are only left with even numbers
let numbers = [1, 2, 3, 4];

let evenNumbers = numbers.filter( (number) => {
    return number % 2 === 0;
});

console.log( evenNumbers ); // Prints out `[2, 4]`
```

What we want to do

What to operate on

What rules to apply

filter

```
let wordsToFilter = [“Answer”,”Always”,”Basically”,”Bravo”];

let wordsStartingWithA = wordsToFilter.filter( (word) => {
   // Only keep words starting with A
   return word.startsWith(“A”);
});

console.log(wordsStartingWithA);
```

find

```
let wordsToFind = [“Answer”,”Always”,”Basically”,”Bravo”];

let wordContainingS = wordsToFind.find( (word) => {
   // Find a word with the letter s in it
   return word.includes(‘s’);
});

console.log(wordContainingS);
```

forEach

```
let numbers = [1, 2, 3, 4];

numbers.forEach( (number) => {
   console.log(`This number is ${number}` );
});
```

every

```
let wordsToFind = [“Answer”,”Always”,”Basically”,”Bravo”];

let doTheyAllHaveAnS = wordsToFind.every( (word) => {
   // Is there an S in it?
   return word.includes(‘s’);
});

console.log(doTheyAllHaveAnS);
```

some

```
let wordsToFind = [“Answer”,”Always”,”Basically”,”Bravo”];

let isThereAnS = wordsToFind.some( (word) => {
   // Is there an S in it?
   return word.includes(‘s’);
});

console.log(isThereAnS);
```

map

```
let numbersToSquare = [1, 2, 3, 4];

let squaredNumbers = numbersToSquare.map( (number) => {
   return number * number;
});

console.log(squaredNumbers);
```

reduce

```
let nameParts = ['bosco', 'p.', 'soultrain'];

let fullName = nameParts.reduce( (reducer, part) => {
   return reducer + ' ' + part.substring(0, 1).toLocaleUpperCase() + part.substring(1);
}, ''); // <--- The empty quotes is the value of the reducer for the first element

console.log(fullName.trim());
```

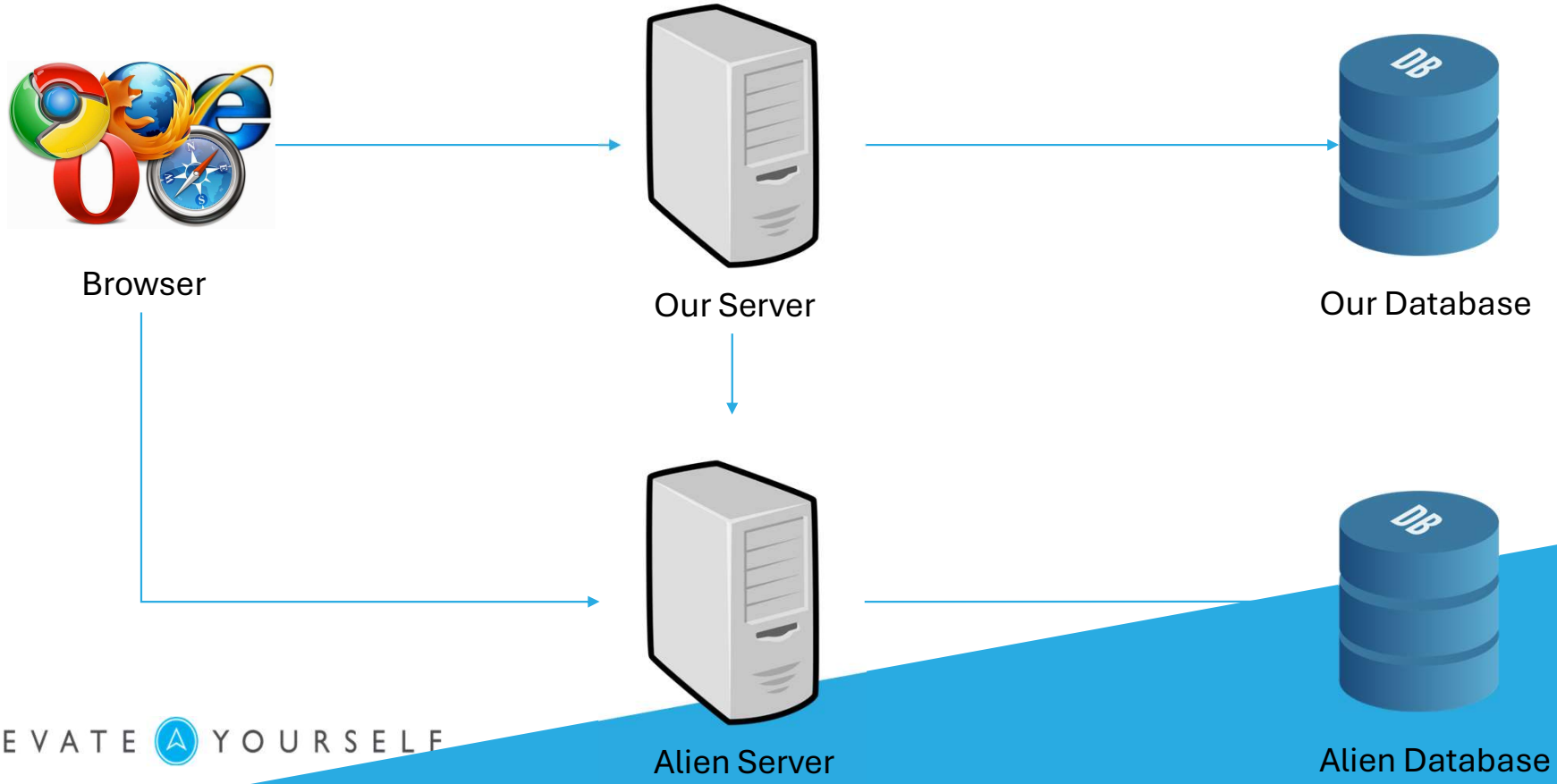
A visual to remember...

```
map([🐮, 🌭, 🐔, 🌽], cook)
=> [🍔, 🍟, 🍗, 🍿]

filter([🍔, 🍟, 🍗, 🍿], isVegetarian)
=> [🍟, 🍿]

reduce([🍔, 🍟, 🍗, 🍿], eat)
=> 💩
```

Changing Architecture?



ELEVATE YOURSELF

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

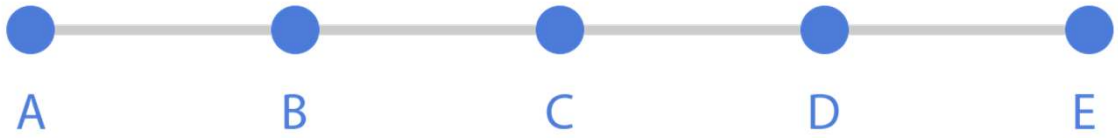
Creating and Consuming

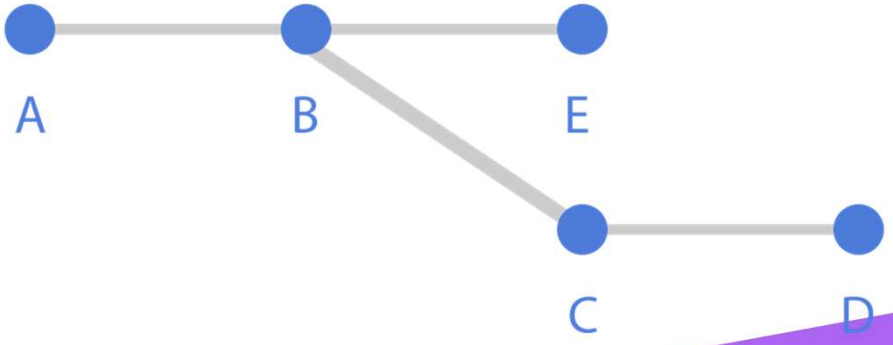
- **Creating** a web service is simply exposing methods and properties from classes
- **Consuming** a web service is calling those APIs and getting the data.

Asynchronous Programming

- Our programs: Synchronous

Asynchronous Programming





Axios – HTTP Client

- Axios provides an interface for accessing and manipulating parts of the HTTP pipeline
- Promise<Response> axios(input[, init]);

```
axios.get(‘https://catfact.ninja/fact')
.then( (response) => {
      document.getElementById('results').innerText =
response.data.fact; }
   });
```

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:
The DOM