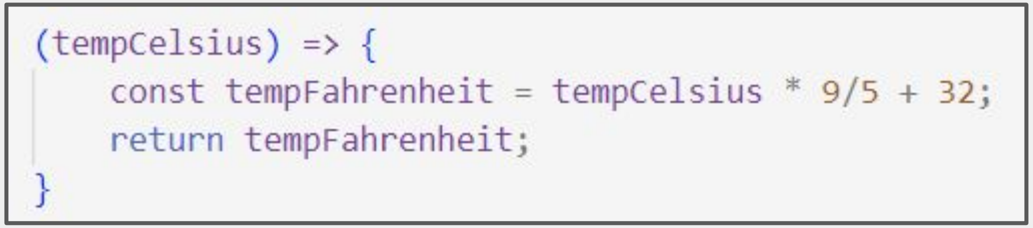
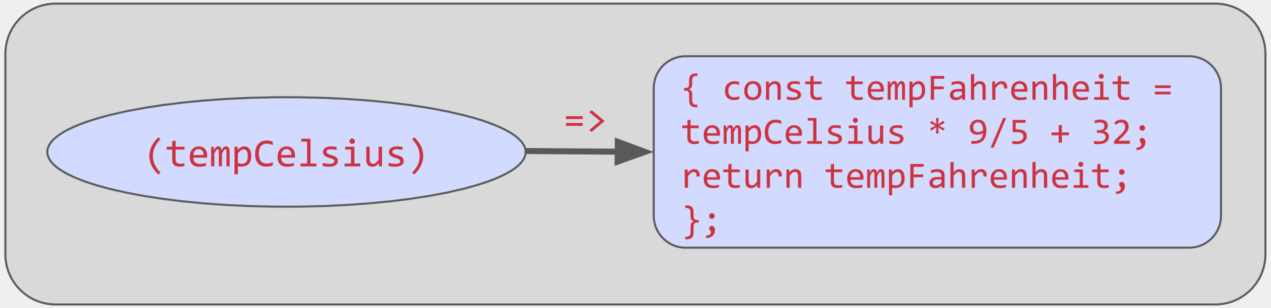
JS Cont: Callback Functions and Arrays

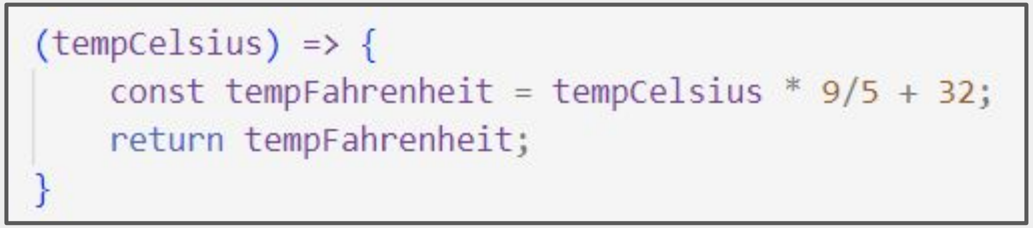
Abby Chou



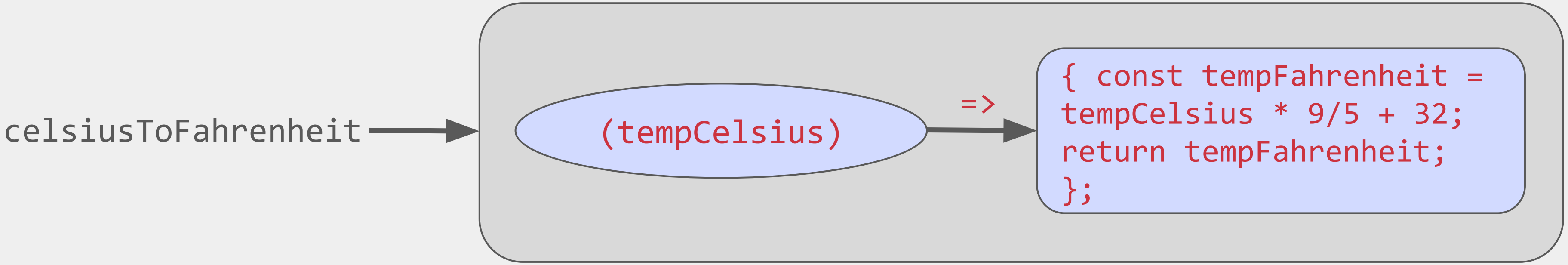
```
(tempCelsius) => {
    const tempFahrenheit = tempCelsius * 9/5 + 32;
    return tempFahrenheit;
}
```

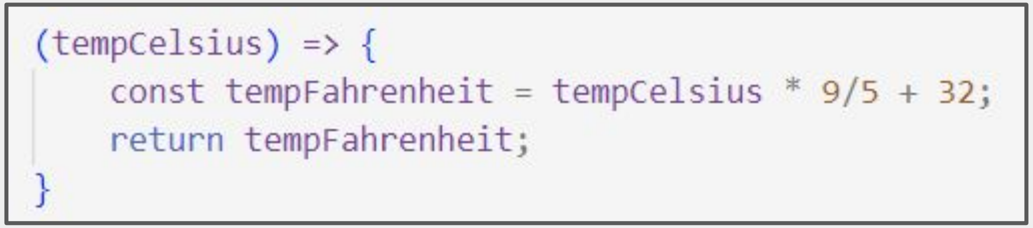


```
(tempCelsius)
=>
{ const tempFahrenheit =
tempCelsius * 9/5 + 32;
return tempFahrenheit;
};
```

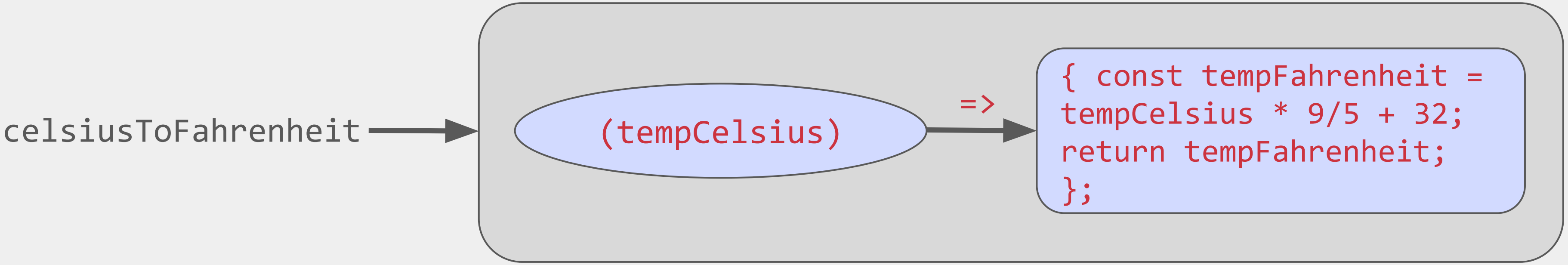


```
(tempCelsius) => {
    const tempFahrenheit = tempCelsius * 9/5 + 32;
    return tempFahrenheit;
}
```

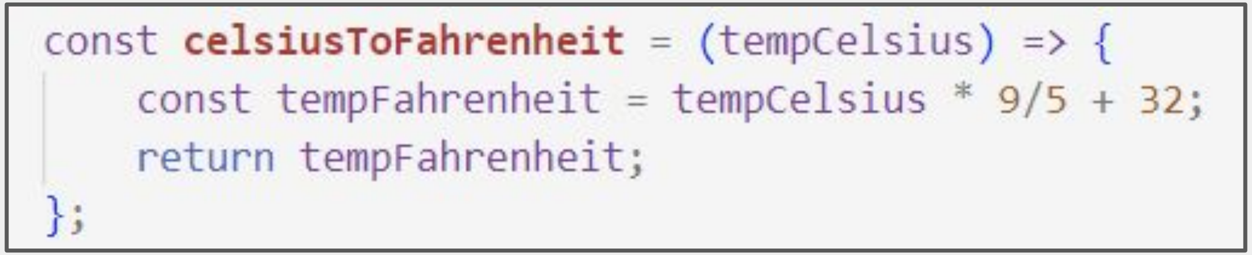

celsiusToFahrenheit
(tempCelsius)
=>
{ const tempFahrenheit = tempCelsius * 9/5 + 32; return tempFahrenheit; };



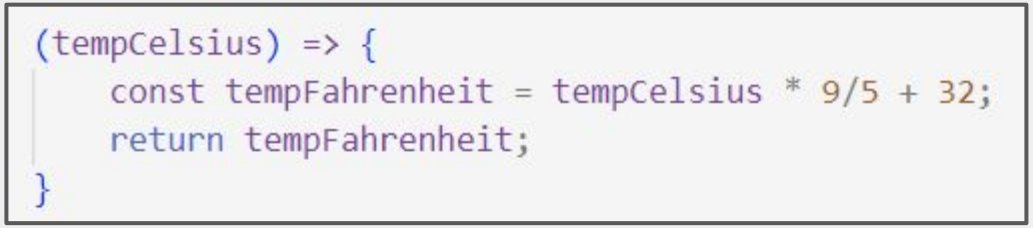
```
(tempCelsius) => {
    const tempFahrenheit = tempCelsius * 9/5 + 32;
    return tempFahrenheit;
}
```



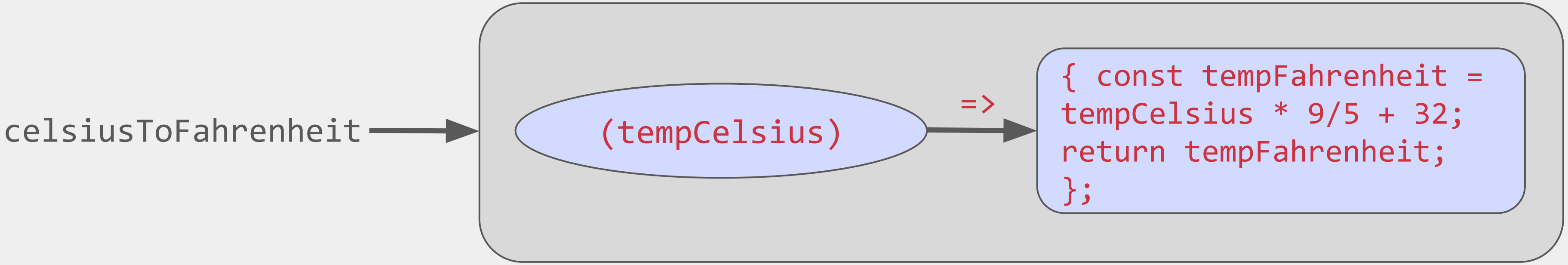
```
{ const tempFahrenheit =
tempCelsius * 9/5 + 32;
return tempFahrenheit;
};
```

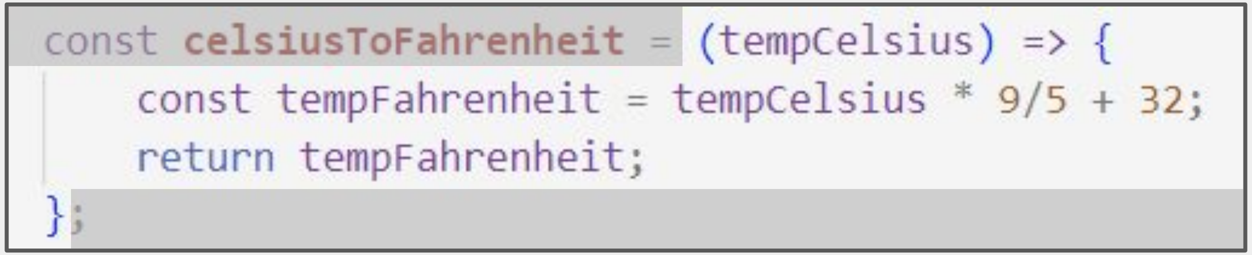


```
const celsiusToFahrenheit = (tempCelsius) => {
    const tempFahrenheit = tempCelsius * 9/5 + 32;
    return tempFahrenheit;
};
```



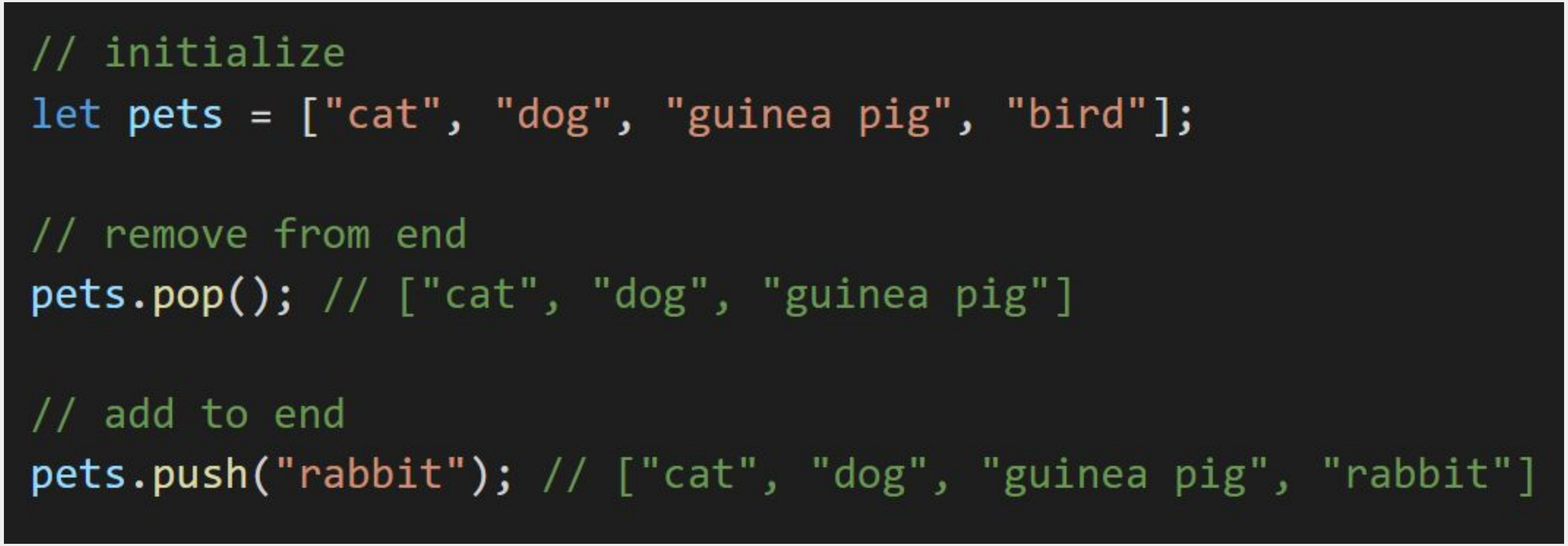
```
(tempCelsius) => {
    const tempFahrenheit = tempCelsius * 9/5 + 32;
    return tempFahrenheit;
}
```


celsiusToFahrenheit
(tempCelsius)
=>
{ const tempFahrenheit = tempCelsius * 9/5 + 32; return tempFahrenheit; };



```
const celsiusToFahrenheit = (tempCelsius) => {
    const tempFahrenheit = tempCelsius * 9/5 + 32;
    return tempFahrenheit;
};
```

Review: Push and Pop

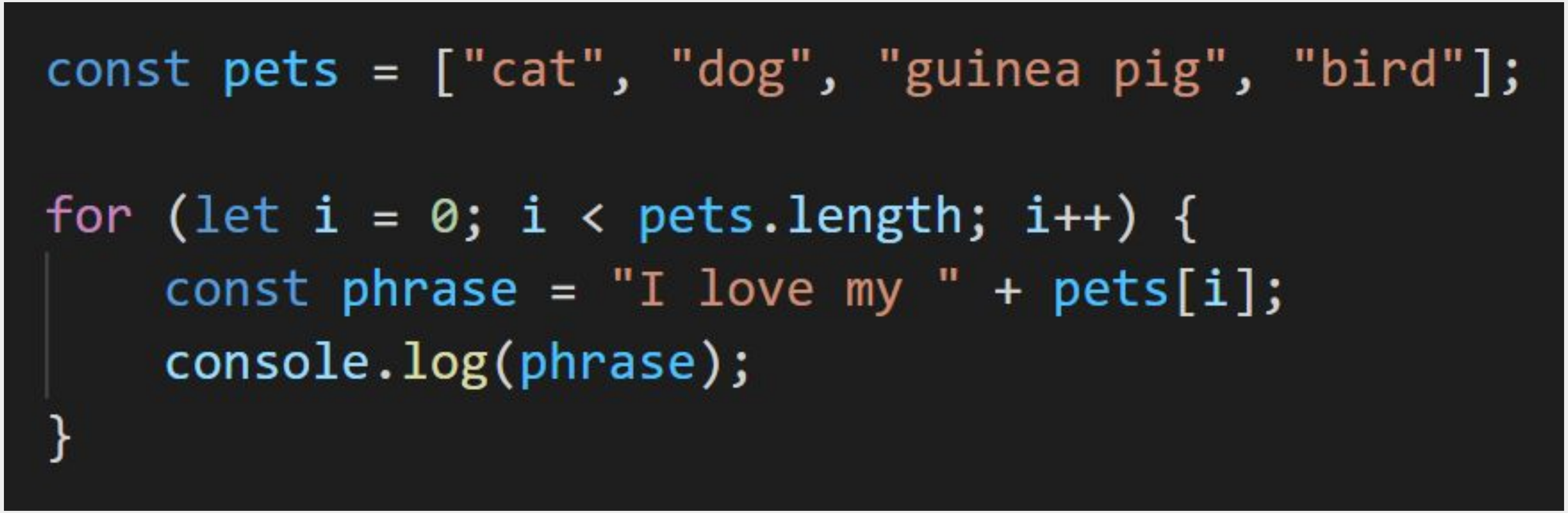


```
// initialize
let pets = ["cat", "dog", "guinea pig", "bird"];

// remove from end
pets.pop(); // ["cat", "dog", "guinea pig"]

// add to end
pets.push("rabbit"); // ["cat", "dog", "guinea pig", "rabbit"]
```

Review: Looping Through an Array



```
const pets = ["cat", "dog", "guinea pig", "bird"];

for (let i = 0; i < pets.length; i++) {
    const phrase = "I love my " + pets[i];
    console.log(phrase);
}
```

Practice!

Write a function that takes in an array of temperatures (in Celsius) and outputs an array with all the temperatures converted to Fahrenheit.

DO NOT mutate (change) the input array!

Paste your code into the browser console (or write it directly there) to test.

tempF = tempC * 9/5 + 32

These slides (to reference syntax): weblab.is/jscont
Yesterday's JS slides: weblab.is/js

Adding more…

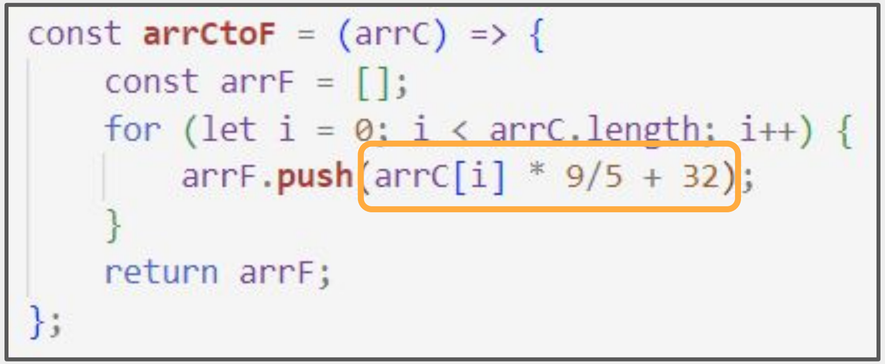
Let’s say we now want three functions:

- Converting an array of Celsius temps to Fahrenheit (what we just did)
- Converting an array of Fahrenheit temps to Celsius
- Converting an array of Celsius temps to Kelvin

Adding more…

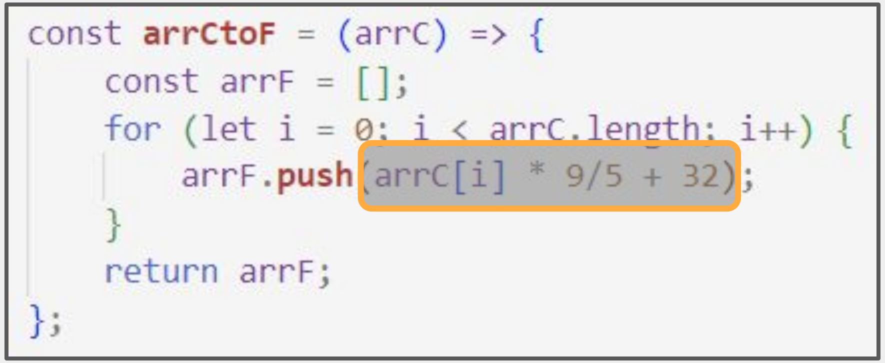
Let's say we now want three functions:

- Converting an array of Celsius temps to Fahrenheit (what we just did)
- Converting an array of Fahrenheit temps to Celsius
- Converting an array of Celsius temps to Kelvin



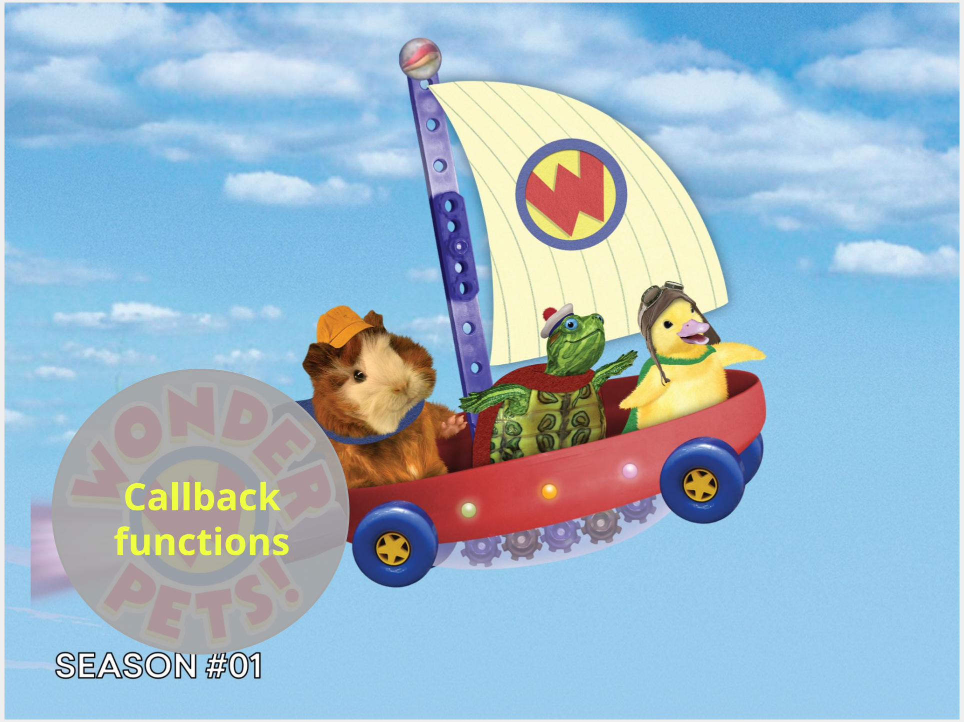
```
const arrCtoF = (arrC) => {
    const arrF = [];
    for (let i = 0; i < arrC.length; i++) {
        arrF.push(arrC[i] * 9/5 + 32);
    }
    return arrF;
};
```

We want to reuse all this code *except* the the part highlighted in orange.



```
const arrCtoF = (arrC) => {
    const arrF = [];
    for (let i = 0; i < arrC.length; i++) {
        arrF.push(arrC[i] * 9/5 + 32);
    }
    return arrF;
};
```

Callback functions to the rescue!



Using Callbacks

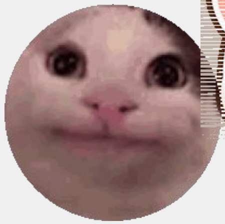
Step 1: Write a general modifyArray function that takes in 2 inputs: an array and a callback function, and returns a new array, the result of applying the callback function to each element of the original.

```
const modifyArray = (arr, transformFunc) => {
    // apply transformFunc to all elements of arr, return
    // the result
};
```

```
const modifyArray = (arr, transformFunc) => {
    // apply transformFunc to all elements of arr, return
    // the result
};
```

modifyArray([

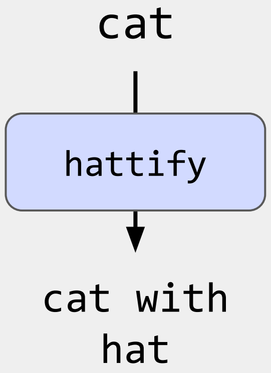
,



,

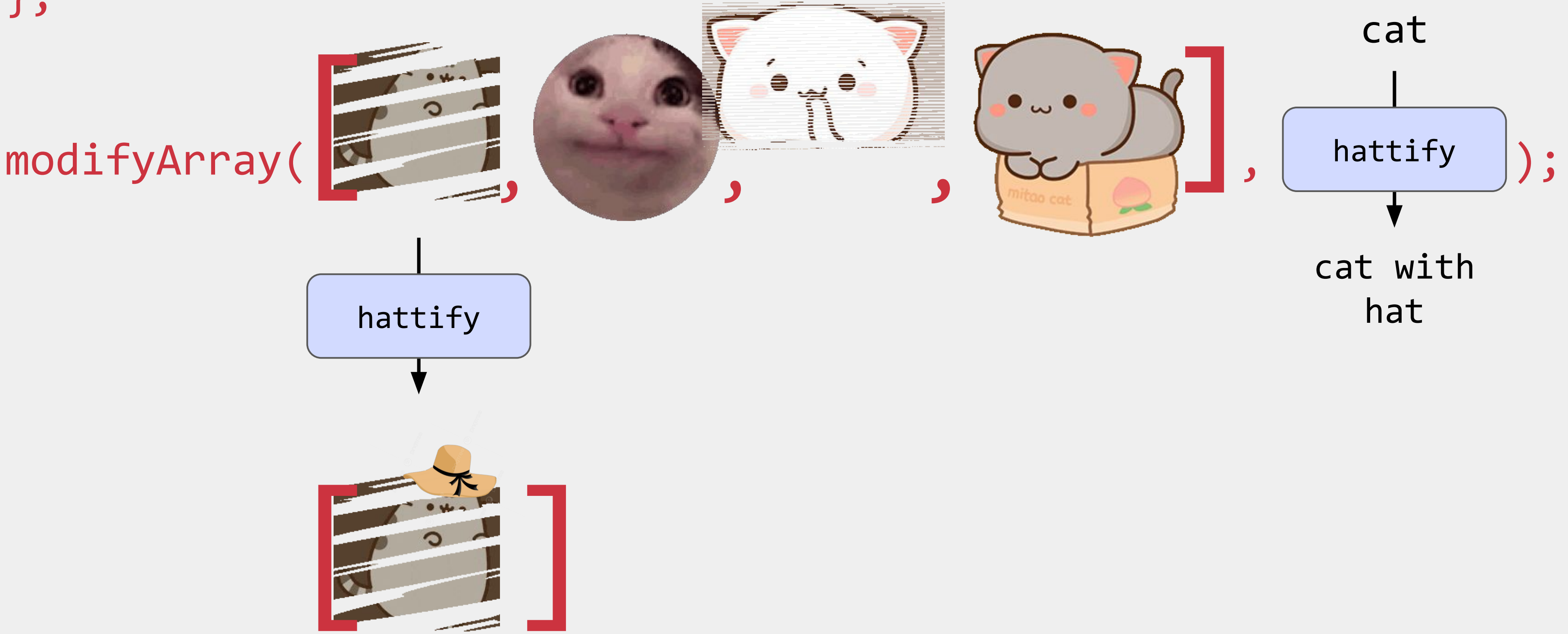
,

],

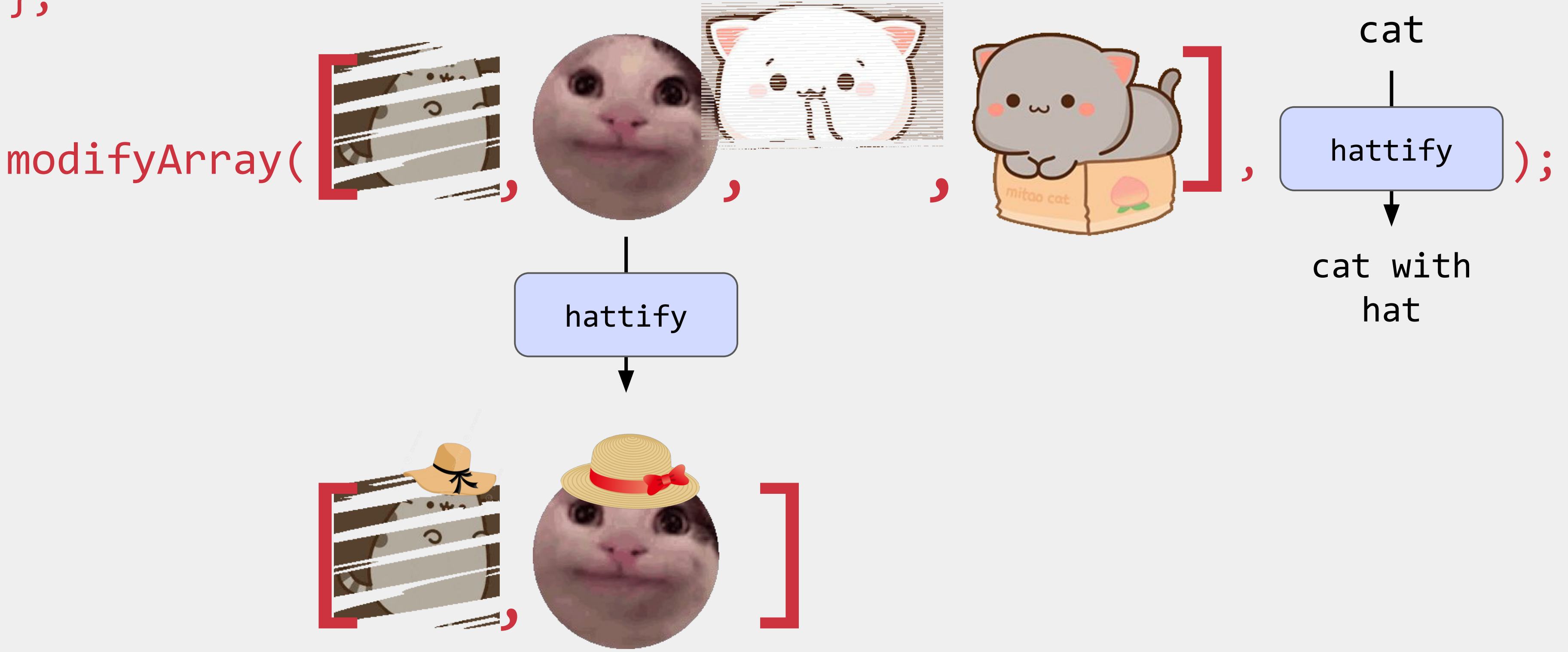


);

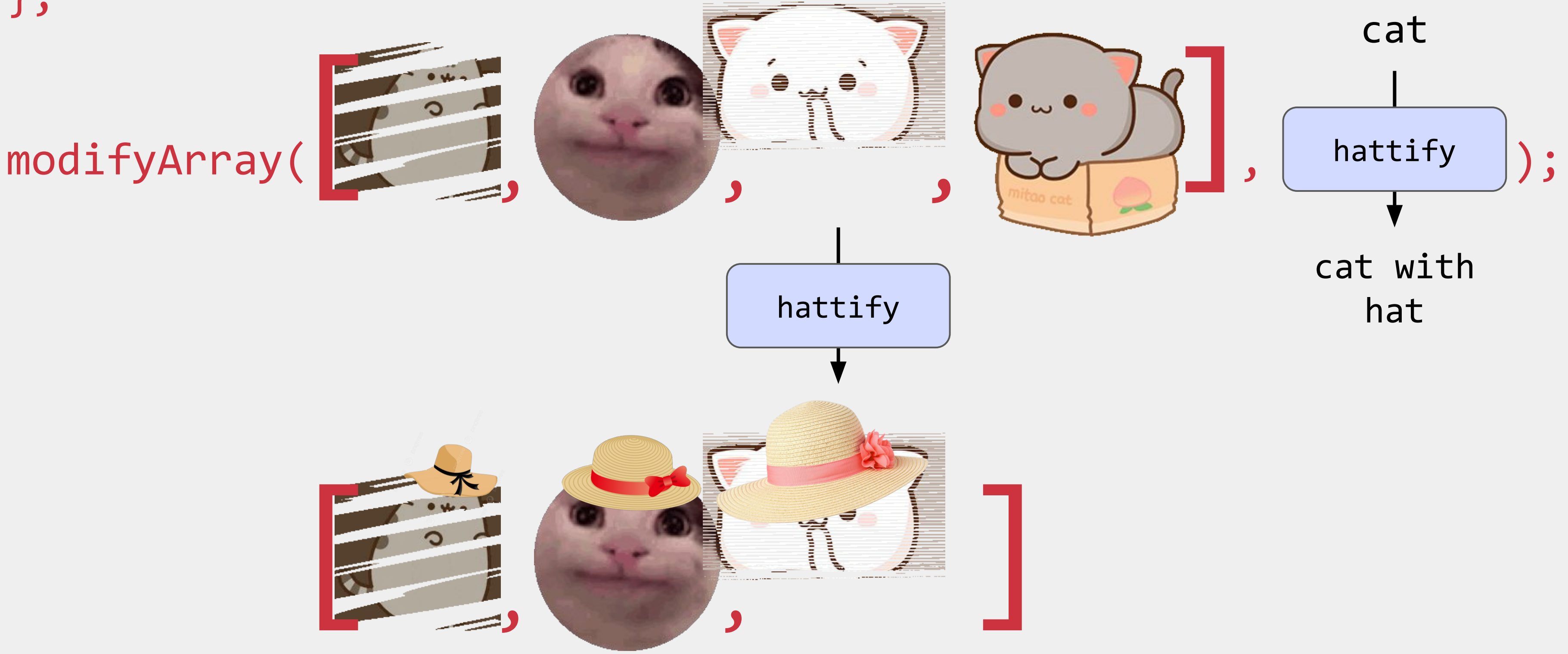
```
const modifyArray = (arr, transformFunc) => {
    // apply transformFunc to all elements of arr, return
    // the result
};
```



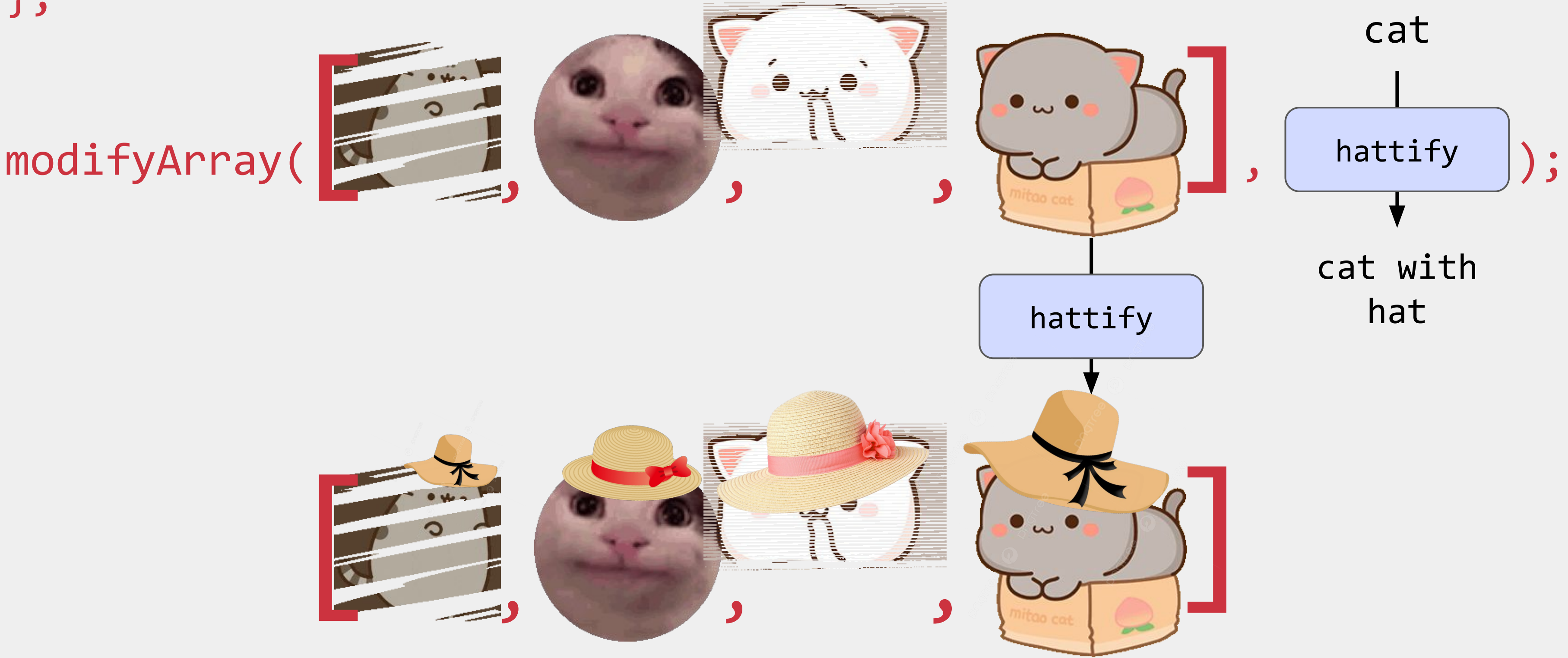
```
const modifyArray = (arr, transformFunc) => {
    // apply transformFunc to all elements of arr, return
    // the result
};
```



```
const modifyArray = (arr, transformFunc) => {
    // apply transformFunc to all elements of arr, return
    // the result
};
```

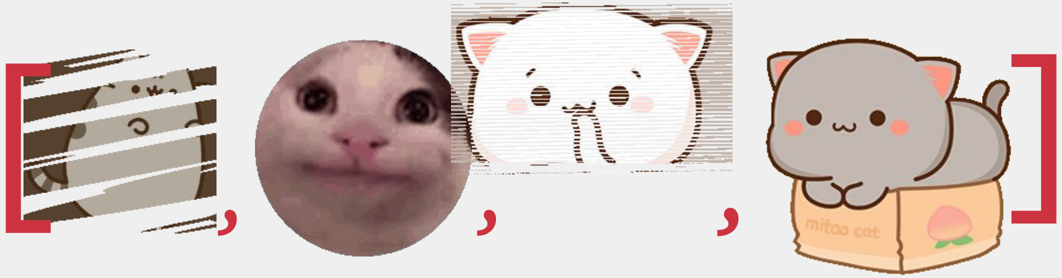


```
const modifyArray = (arr, transformFunc) => {
    // apply transformFunc to all elements of arr, return
    // the result
};
```

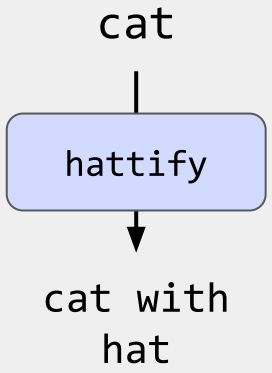


```
const modifyArray = (arr, transformFunc) => {
    // apply transformFunc to all elements of arr, return
    // the result
};
```

modifyArray(



,



);

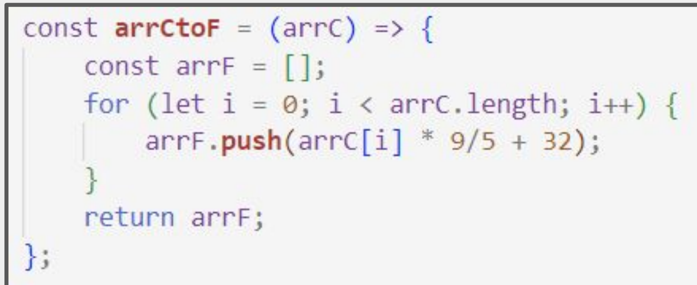
Let’s generalize :D

Step 1: Write a general modifyArray function that takes in 2 inputs: an array and a callback function, and returns a new array, the result of applying the callback function to each element of the original.

Step 2: Use your modifyArray function to do the following:

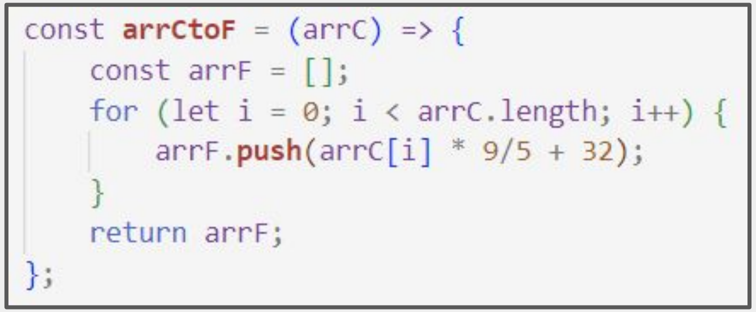
- Convert an array of Celsius temps to Fahrenheit (what we just did)
- Convert an array of Fahrenheit temps to Celsius
  - tempC = (tempF - 32) * 5/9
- Convert an array of Celsius temps to Kelvin
  - tempK = tempC + 273.15

Check out weblab.is/callbackshint if you’re stuck!

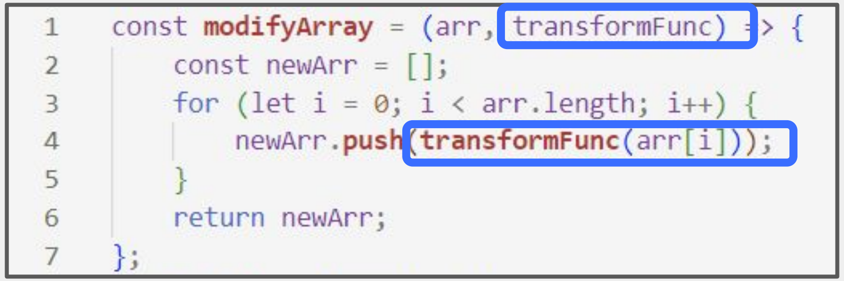


```
const arrCtoF = (arrC) => {
    const arrF = [];
    for (let i = 0; i < arrC.length; i++) {
        arrF.push(arrC[i] * 9/5 + 32);
    }
    return arrF;
};
```

Solution

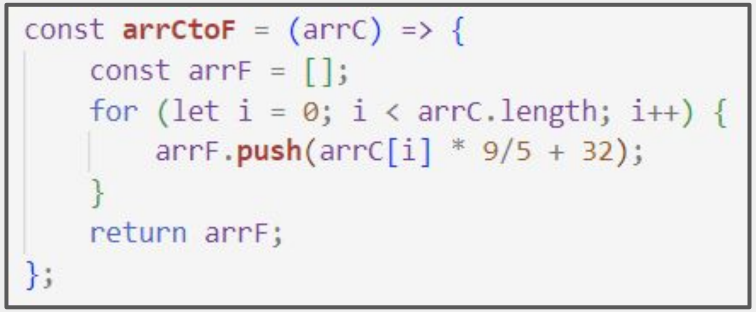


```
const arrCtoF = (arrC) => {
    const arrF = [];
    for (let i = 0; i < arrC.length; i++) {
        arrF.push(arrC[i] * 9/5 + 32);
    }
    return arrF;
};
```

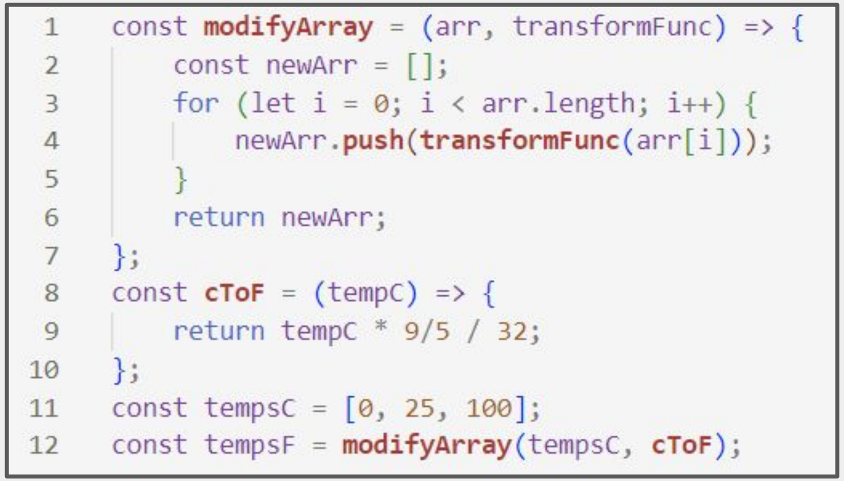


```
const modifyArray = (arr, transformFunc) => {
    const newArr = [];
    for (let i = 0; i < arr.length; i++) {
        newArr.push(transformFunc(arr[i]));
    }
    return newArr;
};
```

Solution



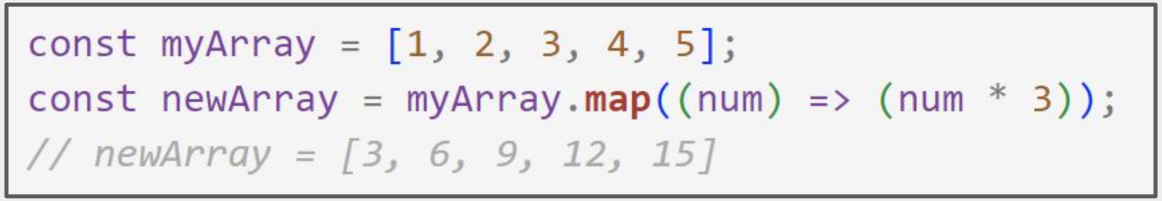
```javascript
const arrCtoF = (arrC) => {
    const arrF = [];
    for (let i = 0; i < arrC.length; i++) {
        arrF.push(arrC[i] * 9/5 + 32);
    }
    return arrF;
};
```



```javascript
const modifyArray = (arr, transformFunc) => {
    const newArr = [];
    for (let i = 0; i < arr.length; i++) {
        newArr.push(transformFunc(arr[i]));
    }
    return newArr;
};
const cToF = (tempC) => {
    return tempC * 9/5 / 32;
};
const tempsC = [0, 25, 100];
const tempsF = modifyArray(tempsC, cToF);
```

map(...)

Creates a new array by applying the callback function to every element of the starting array.

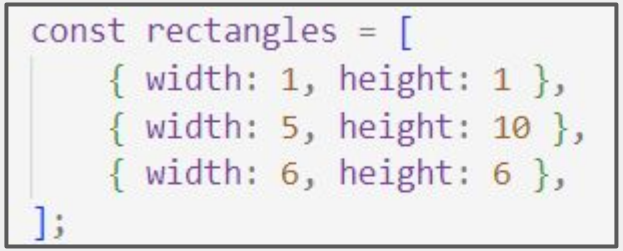


```
const myArray = [1, 2, 3, 4, 5];
const newArray = myArray.map((num) => (num * 3));
// newArray = [3, 6, 9, 12, 15]
```

Map Practice

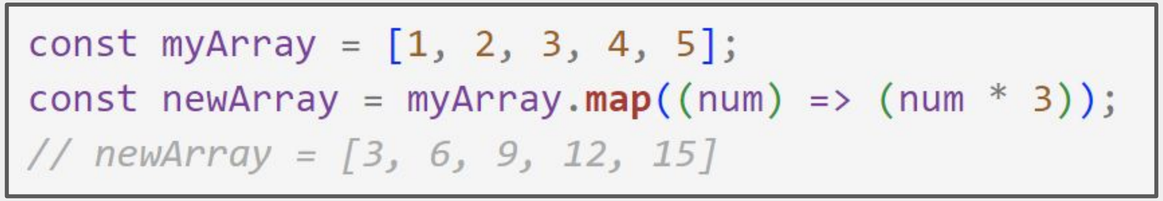
This is an array of JavaScript objects!

Copy this array into your JS file, then use map to create an array of the areas of the rectangles.



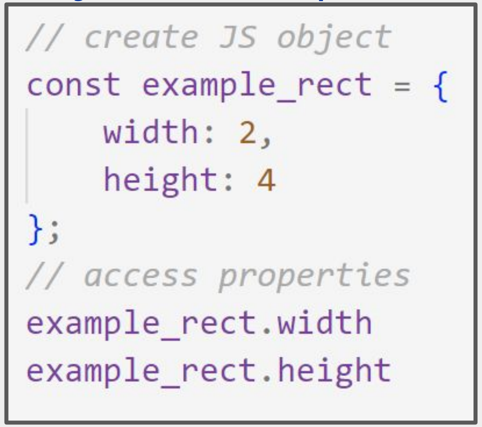
```
const rectangles = [
    { width: 1, height: 1 },
    { width: 5, height: 10 },
    { width: 6, height: 6 },
];
```

Map example:



```
const myArray = [1, 2, 3, 4, 5];
const newArray = myArray.map((num) => (num * 3));
// newArray = [3, 6, 9, 12, 15]
```

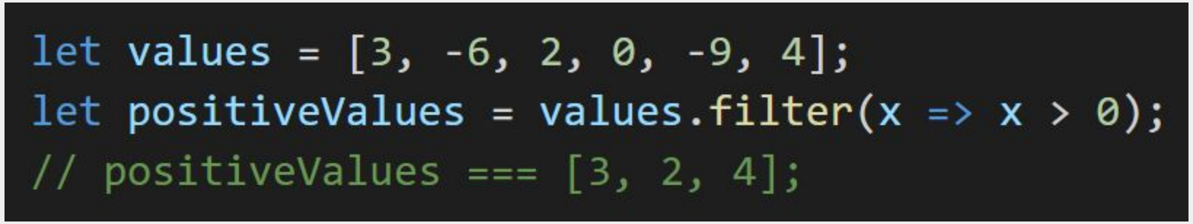
Objects example:



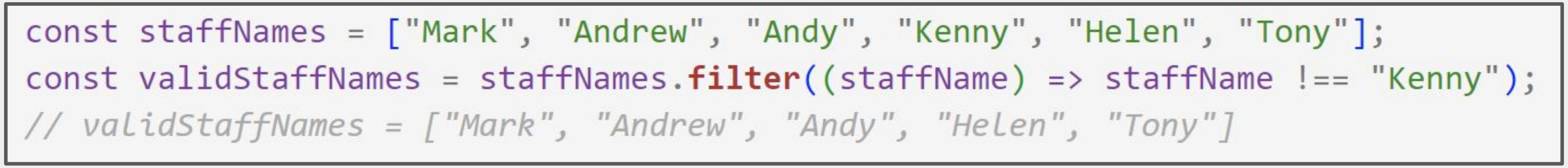
```
// create JS object
const example_rect = {
    width: 2,
    height: 4
};
// access properties
example_rect.width
example_rect.height
```

filter(...)

Creates a new array by selecting the elements in the starting array which pass the given “test” (i.e. *filtering out* the “bad” elements and keeping the “good” ones).



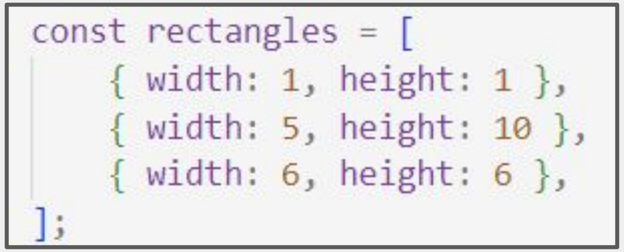
```javascript
let values = [3, -6, 2, 0, -9, 4];
let positiveValues = values.filter(x => x > 0);
// positiveValues === [3, 2, 4];
```



```javascript
const staffNames = ["Mark", "Andrew", "Andy", "Kenny", "Helen", "Tony"];
const validStaffNames = staffNames.filter((staffName) => staffName !== "Kenny");
// validStaffNames = ["Mark", "Andrew", "Andy", "Helen", "Tony"]
```

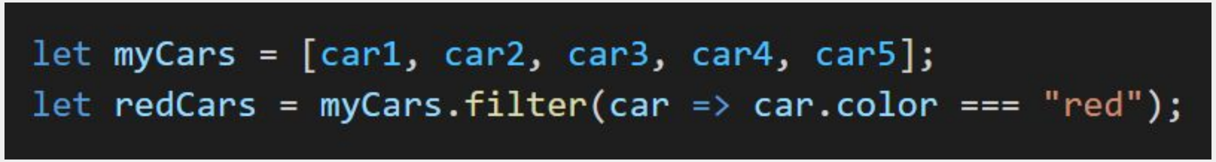
Filter Practice

Using the rectangles array from earlier, use filter to create an array that only contains rectangles with perimeter > 10.



```
const rectangles = [
    { width: 1, height: 1 },
    { width: 5, height: 10 },
    { width: 6, height: 6 },
];
```

Example:

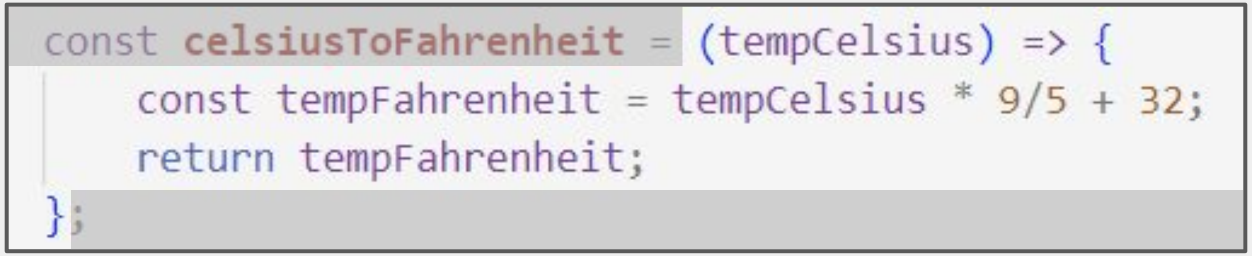


```
let myCars = [car1, car2, car3, car4, car5];
let redCars = myCars.filter(car => car.color === "red");
```

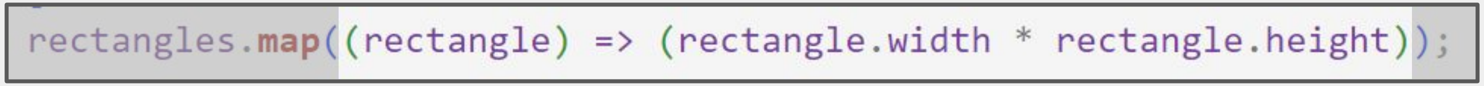
Recap: Functions!

```
(inputs) => { do stuff (and optionally return an output) }
```

```
(inputs) => (output)
```



```
const celsiusToFahrenheit = (tempCelsius) => {
    const tempFahrenheit = tempCelsius * 9/5 + 32;
    return tempFahrenheit;
};
```



```
rectangles.map((rectangle) => (rectangle.width * rectangle.height));
```

Why do we use callback functions?

1. Reusability (map and filter)
   a. So we don't have to write code to loop through an array over and over.
2. Abstraction
   a. "When x happens, do this"

And let other smart ppl (the writers of JavaScript libraries) write the code for this!

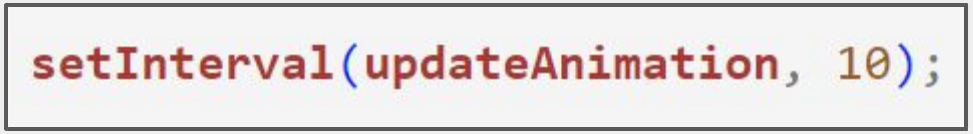
We can write the code for this...

Callback Functions in Web Dev

Let’s say we wrote a function, `updateAnimation()`, that we want to call every 10 milliseconds.

Writing code to manage timers is annoying, so the JavaScript authors did it for us!

“When the timer goes off, call updateAnimation”



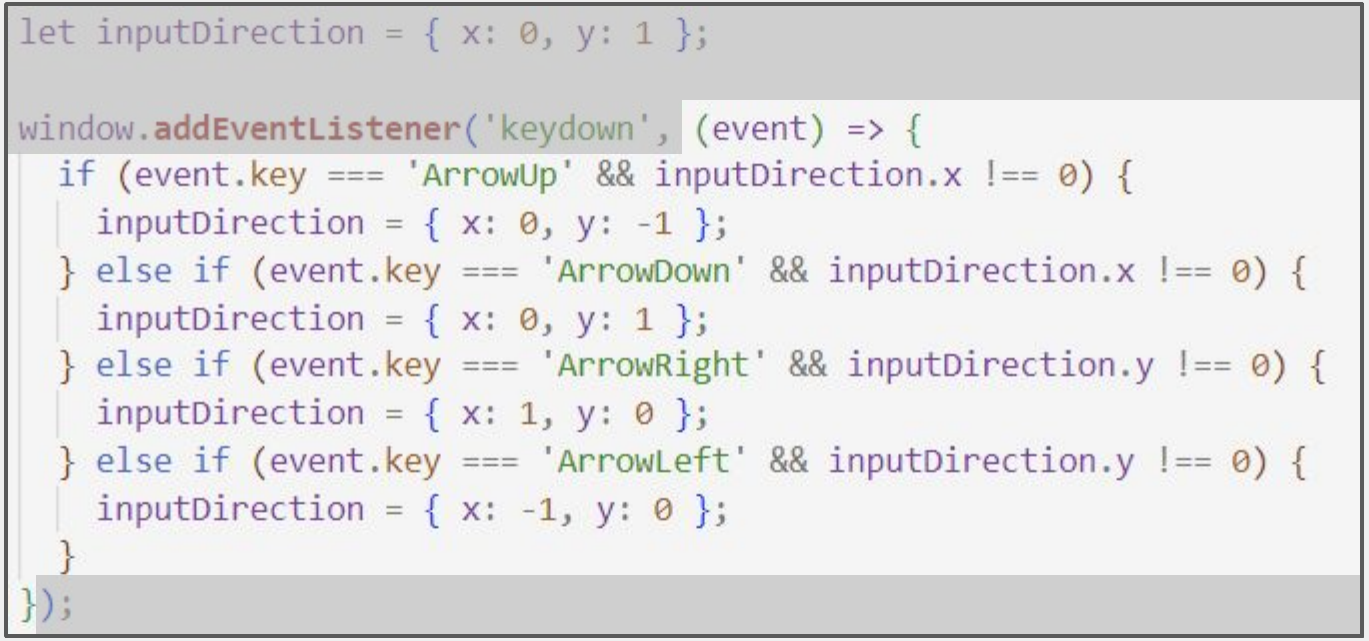
```
setInterval(updateAnimation, 10);
```

Callback Functions in Web Dev

Snake Workshop: Responding to keyboard inputs!

Callback Functions in Web Dev

Snake Workshop: Responding to keyboard inputs!
“When you get a keypress, do this with it”

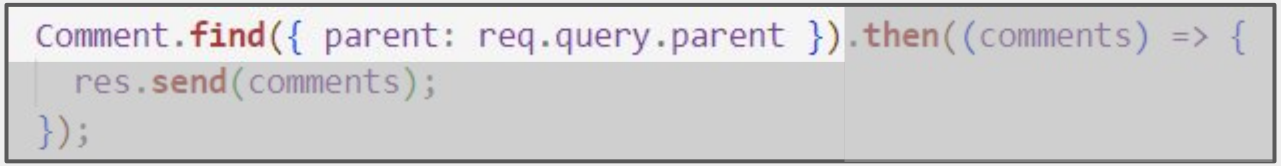


```
let inputDirection = { x: 0, y: 1 };

window.addEventListener('keydown', (event) => {
  if (event.key === 'ArrowUp' && inputDirection.x !== 0) {
    inputDirection = { x: 0, y: -1 };
  } else if (event.key === 'ArrowDown' && inputDirection.x !== 0) {
    inputDirection = { x: 0, y: 1 };
  } else if (event.key === 'ArrowRight' && inputDirection.y !== 0) {
    inputDirection = { x: 1, y: 0 };
  } else if (event.key === 'ArrowLeft' && inputDirection.y !== 0) {
    inputDirection = { x: -1, y: 0 };
  }
});
```

Callback Functions in Web Dev

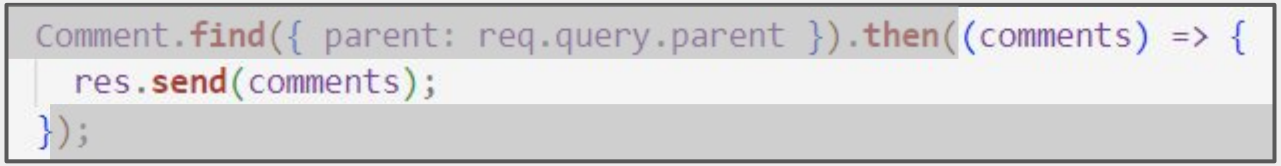
Later: Database query returns a promise, and we want to do something when that promise resolves (i.e. when the database gives us the info we need)



```
Comment.find({ parent: req.query.parent }).then((comments) => {
  res.send(comments);
});
```

Callback Functions in Web Dev

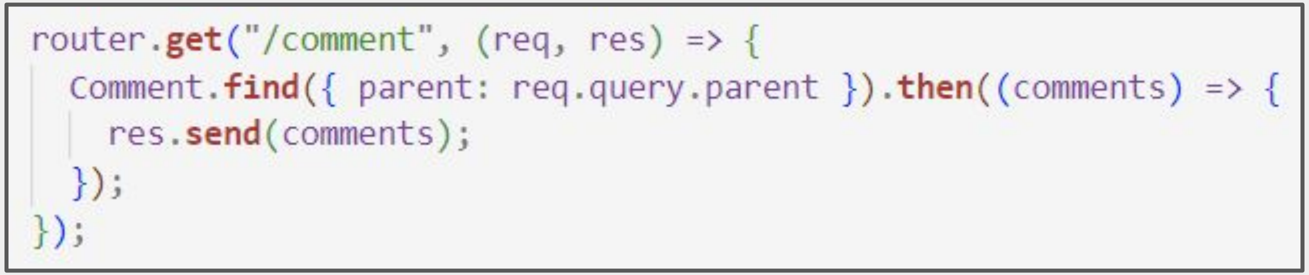
Later: Database query returns a promise, and we want to do something when that promise resolves (i.e. when the database gives us the info we need)



```
Comment.find({ parent: req.query.parent }).then((comments) => {
  res.send(comments);
});
```

Callback Functions in Web Dev

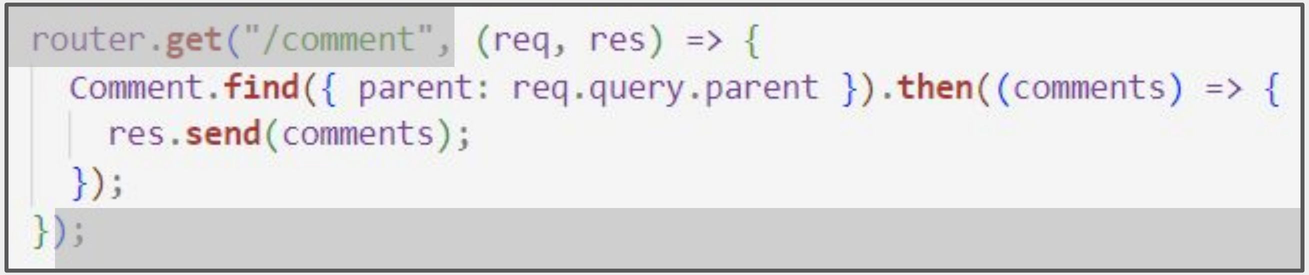
When we receive an http GET request to the /comment endpoint, we want to send a response with the comments we got from the database.



```
router.get("/comment", (req, res) => {
  Comment.find({ parent: req.query.parent }).then((comments) => {
    res.send(comments);
  });
});
```

Callback Functions in Web Dev

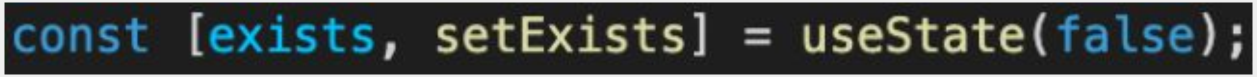
When we receive an http GET request to the /comment endpoint, we want to send a response with the comments we got from the database.



```
router.get("/comment", (req, res) => {
  Comment.find({ parent: req.query.parent }).then((comments) => {
    res.send(comments);
  });
});
```

Food for thought…

This is not a callback function, but what is it??



```
const [exists, setExists] = useState(false);
```