Functional Programming with JavaScript

- JavaScript supports **functional programming** because **JavaScript functions are first class citizens.**
- **This means that functions can do the same things that variables can do.**
- The latest JavaScript syntax adds language improvements that can beef up your functional programming techniques, including **arrow functions, promises, and the spread operator.**

- **In JavaScript, functions can represent data in your application**. You may have noticed that **you can declare functions with the var, let, or const keywords the same way you can declare strings, numbers, or any other variables:**

```
var log = function(message) {
console.log(message);
};
log("In JavaScript, functions are variables");
// In JavaScript, functions are variables
```

- Since functions are variables, we can add them to objects:

```
const obj = {
message: "They can be added to objects like variables",
log(message) {
console.log(message);
}
};
obj.log(obj.message);
// They can be added to objects like variables
```

- We can also add functions to arrays in JavaScript:

```
const messages = ["They can be inserted into arrays", message =>
console.log(message), "like variables", message => console.log(message)
];

messages[1](messages[0]); // They can be inserted into arrays
messages[3](messages[2]); // like variables
```

- Functions can be sent to other functions as arguments, just like other variables:

```
const insideFn = logger => {
logger("They can be sent to other functions as arguments");
};
insideFn(message => console.log(message));
// They can be sent to other functions as arguments
```

- They can also be returned from other functions, just like variables:

```
const createScream = function(logger) {
return function(message) {
logger(message.toUpperCase() + "!!!");
};
};
const scream = createScream(message => console.log(message));
scream("functions can be returned from other functions");
scream("createScream returns a function");
scream("scream invokes that returned function");
// FUNCTIONS CAN BE RETURNED FROM OTHER FUNCTIONS!!!
// CREATESCREAM RETURNS A FUNCTION!!!
// SCREAM INVOKES THAT RETURNED FUNCTION!!!
```

- The last two examples were of higher-order functions: functions that either take or return other functions.

- We could describe the same createScream higher-order function with arrows:

```
const createScream = logger => message => {
logger(message.toUpperCase() + "!!!");
};
```

- If you see more than one arrow used during a function declaration, this means that you're using a higher-order function.