MODULE ONE - JAVASCRIPT FOUNDATIONS

ARRAYS, FUNCTIONS, LOOPS & CONDITIONAL STATEMENTS

TODAY'S SCHEDULE

1. Cool Things
2. Review/ Review Quiz One
3. Practice Questions
4. Awesome Arrays
5. Fun with Functions
6. Conditional Structures
7. Loops
8. Concept Review
9. Learning Activities (Quiz Two, Lab One)

LEARNING OBJECTIVES - WEEK 2

1. construct a variety of programming structures including variables, constants, arrays, objects, functions, conditionals, and constructors;

PART ONE – COOL THINGS

RESOURCES, LINKS TUTORIALS AND OTHER COOL THINGS...

- https://frontendmasters.com/books/front-end-handbook/2019/
- https://davidwalsh.name/javascript-tricks
- https://skillcrush.com/2018/06/18/projects-you-can-do-with-javascript/
- http://dressacat.com/

PART TWO – LET'S REVIEW

www.twiddla.com

LAST WEEK, WE LEARNED..

WHAT WE ALREADY KNOW ABOUT JS...

- JavaScript allows us to create dynamic and interactive websites/applications
- we can do things like change content, modify styles, react to user actions and much more

WHAT WE ALREADY KNOW ABOUT JS...

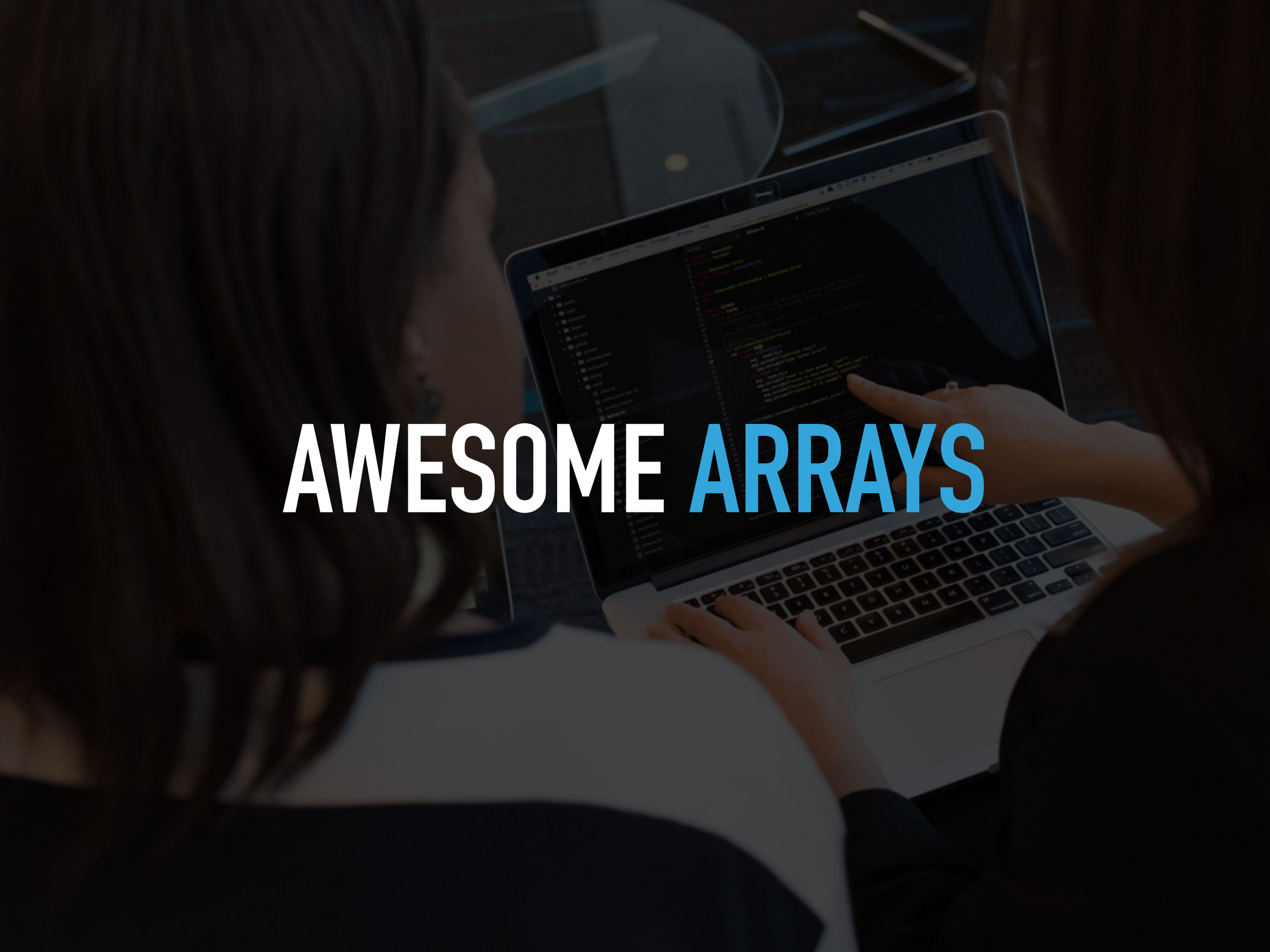
- JavaScript can be used client-side and server -side
- JavaScript is loosely-typed
- JavaScript is object based & everything in JavaScript is an object

WHAT WE ALREADY KNOW ABOUT JS...

- JavaScript is a single-threaded language
- JavaScript can be included in your web application using the <script></script>
- We need to implement script loading strategies to ensure our scripts are loaded at the right time

WHAT WE ALREADY KNOW ABOUT JS...

- JavaScript is standardized using the ECMAScript language
- Declare variables using let, var, const*
- 6 primitive data types, everything else is an object type
- Dynamically & loosely typed



AWESOME ARRAYS

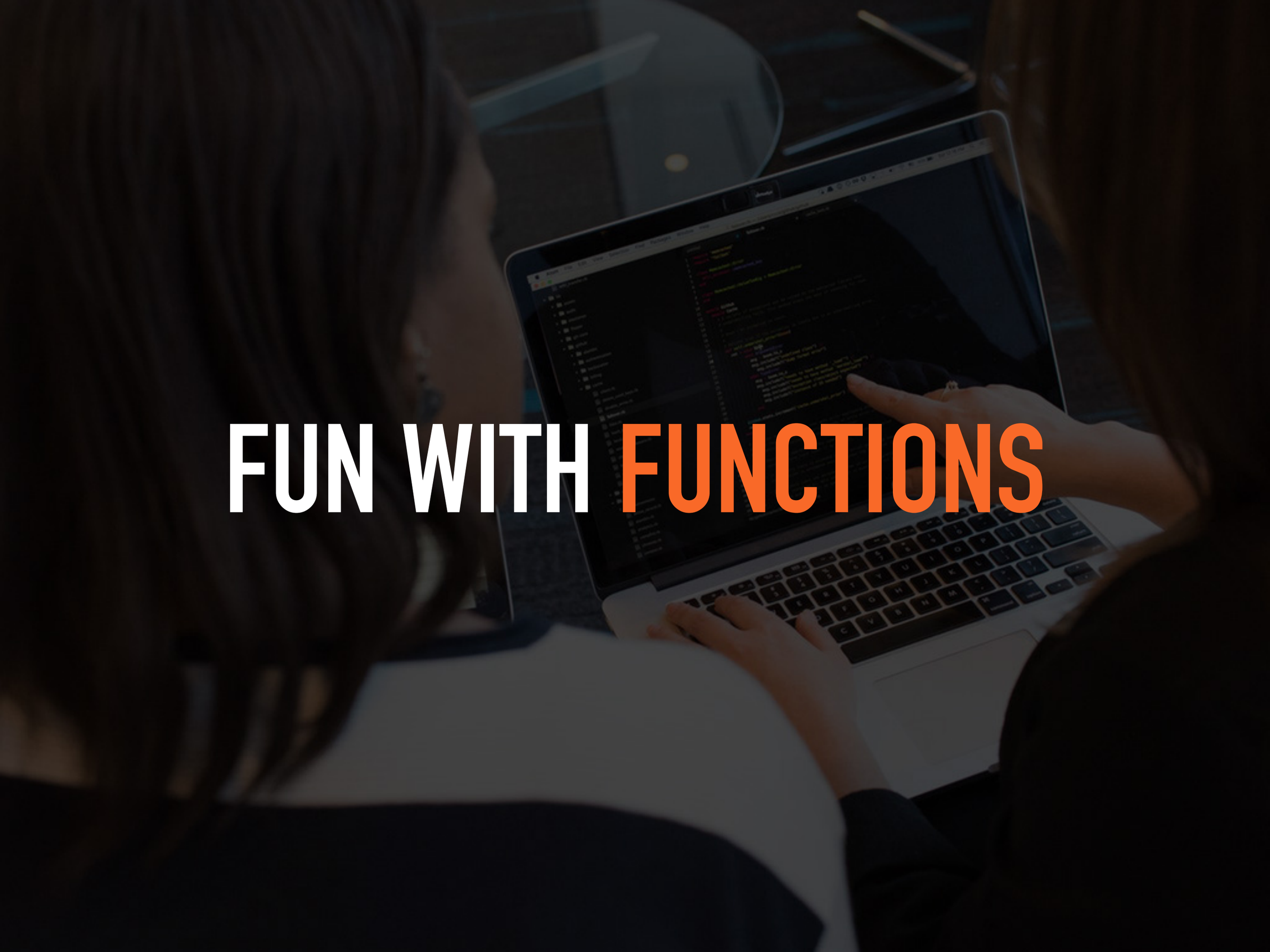
LET'S TALK ARRAYS...

- stores a set of values that are related to each other
- elements in an array can be any data type, can even store arrays inside of an array (multidimensional array)
- two techniques to create in JS : array literal and array constructor
- An array is object, therefore it has methods and properties

```
// this is an array literal containing strings, a number, a boolean and even an array
let arrayLiteral = [['Jess', 'Gilfillan'], 35, 'professor', true, ];


// this is an array created using an array Constructor
let arrayConstructor = new Array ('jess', 'cats', true);


// to access items in the array, we can specify the index number. What would be displayed in the console?
console.log(arrayLiteral[1]);


// what would be displayed in the console?
console.log(arrayConstructor[0]);

/* arrays are objects (most things in JavaScript are & we'll talk about this in detail soon! )
Objects have methods and properties. We can use the length property to find out the
length of our array literal stored in arrayLiteral. What would be displayed in the console? */

console.log(arrayLiteral.length);
```

EXAMPLE TIME

Weekly Learning > Module 1> Code Examples



FUN WITH FUNCTIONS

HOW WOULD YOU DESCRIBE A FUNCTION ?

LET'S TALK FUNCTIONS...

- consists of a series of statements that are grouped together because they perform a specific task
- built-in browser functions, named functions, anonymous functions, immediately invoked function expressions (IIFE)
- functions with parameters
- function with return values

FUNCTIONS VS. FUNCTION EXPRESSIONS

- If you put a function where the interpreter would expect to see an expression, it's treated as an expression and known as a function expression
- Function is not processed until the interpreter gets to that statement

VARIABLE SCOPE ...

- the location where you declare a variable will affect where it can be used within your code

```
let myCoolVariable;
```

- local vs. global variables

GLOBAL VS. LOCAL VARIABLES

- interpreter creates local variables when the function is run and removes as soon as the function has finished its task

- global variables are stored in memory for as long as the web page is loaded in the browser

- global variables use more memory

- Use local variables whenever possible

BUILT-IN BROWSER FUNCTIONS

```
var myText = 'I am a string';
var newString = myText.replace('string', 'cat');
```

NAMED FUNCTIONS

```
//declare
function superCoolFunction {
    alert('This function is super cool!');
}

//invoke
superCoolFunction();
```

FUNCTION RETURN VALUES

```
//declare with parameters
function getArea(width, height) {
    return width * height;
}

//invoke with arguments
getArea(4, 6);
```

FUNCTIONS WITH PARAMETERS

```
//declare with parameters
function getArea(width, height) {
    return width * height;
}

//invoke with arguments
getArea(4, 6);
```

ANONYMOUS FUNCTIONS

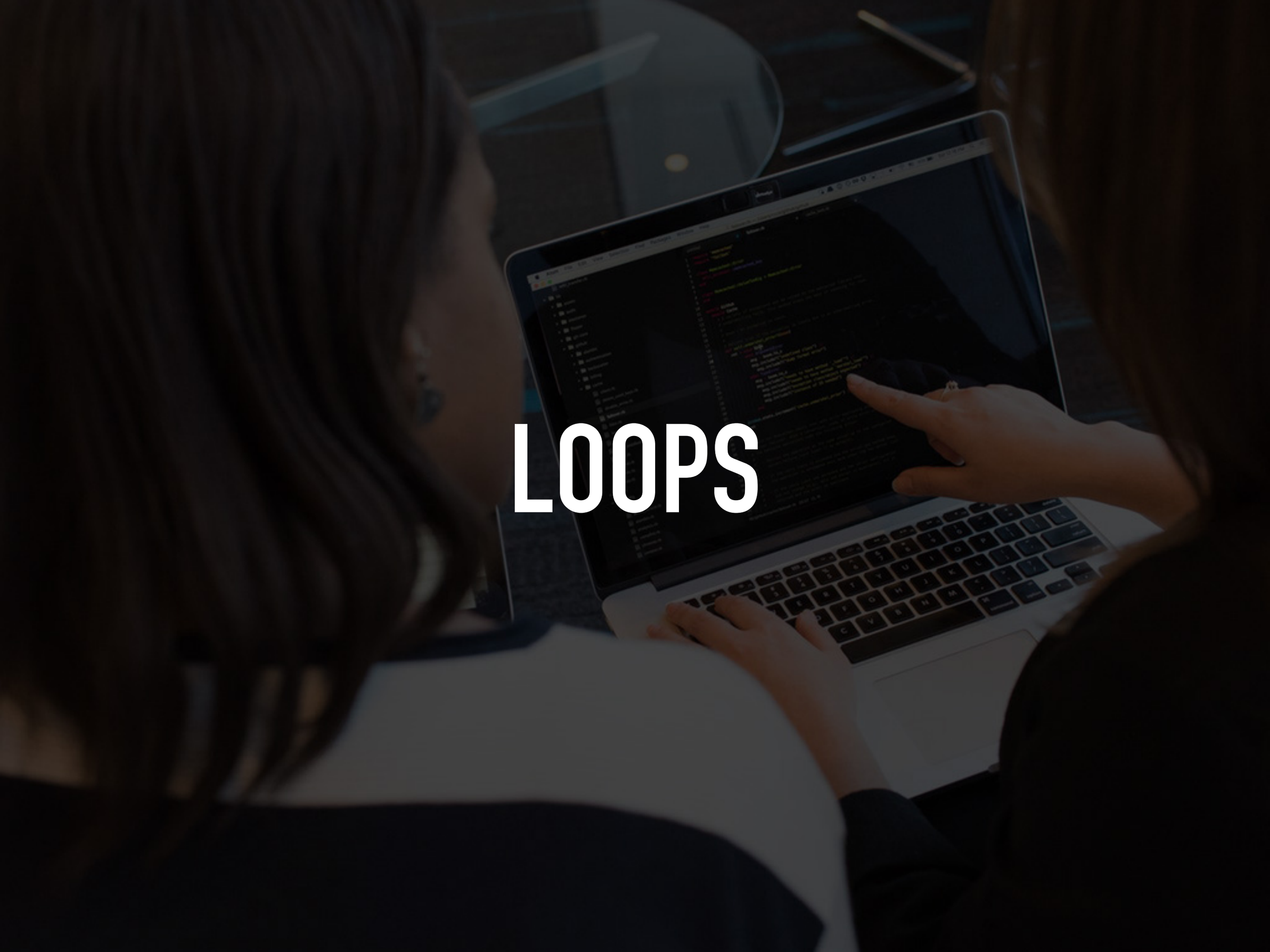
```
var myButton = document.querySelector('button');

myButton.onclick = function() {
    alert('hello');
}
```

IMMEDIATELY INVOKED FUNCTION EXPRESSIONS (IIFE)

```
//grouping operators are parentheses that tell the interpreter to treat
this as an expression
let area = (function() {
let width = 3;
let height = 2;
return width * height;
}());
//final parentheses tell the interpreter to call the function
immediately
console.log(area);
```

EXAMPLE TIME

Weekly Learning > Module 1> Code Examples



LOOPS

LOOPS

- Allow us to do things over and over again (iteration)
- usually have the following features : a counter, an exit condition and an iterator
- for loops, while loops and do ... while loops

THE BASIC FOR LOOP

```
for (let i = 0; i < cats.length; i++) {
    if (i === cats.length - 1) {
        info += 'and ' + cats[i] + '.';
    } else {
        info += cats[i] + ', ';
    }
}
```

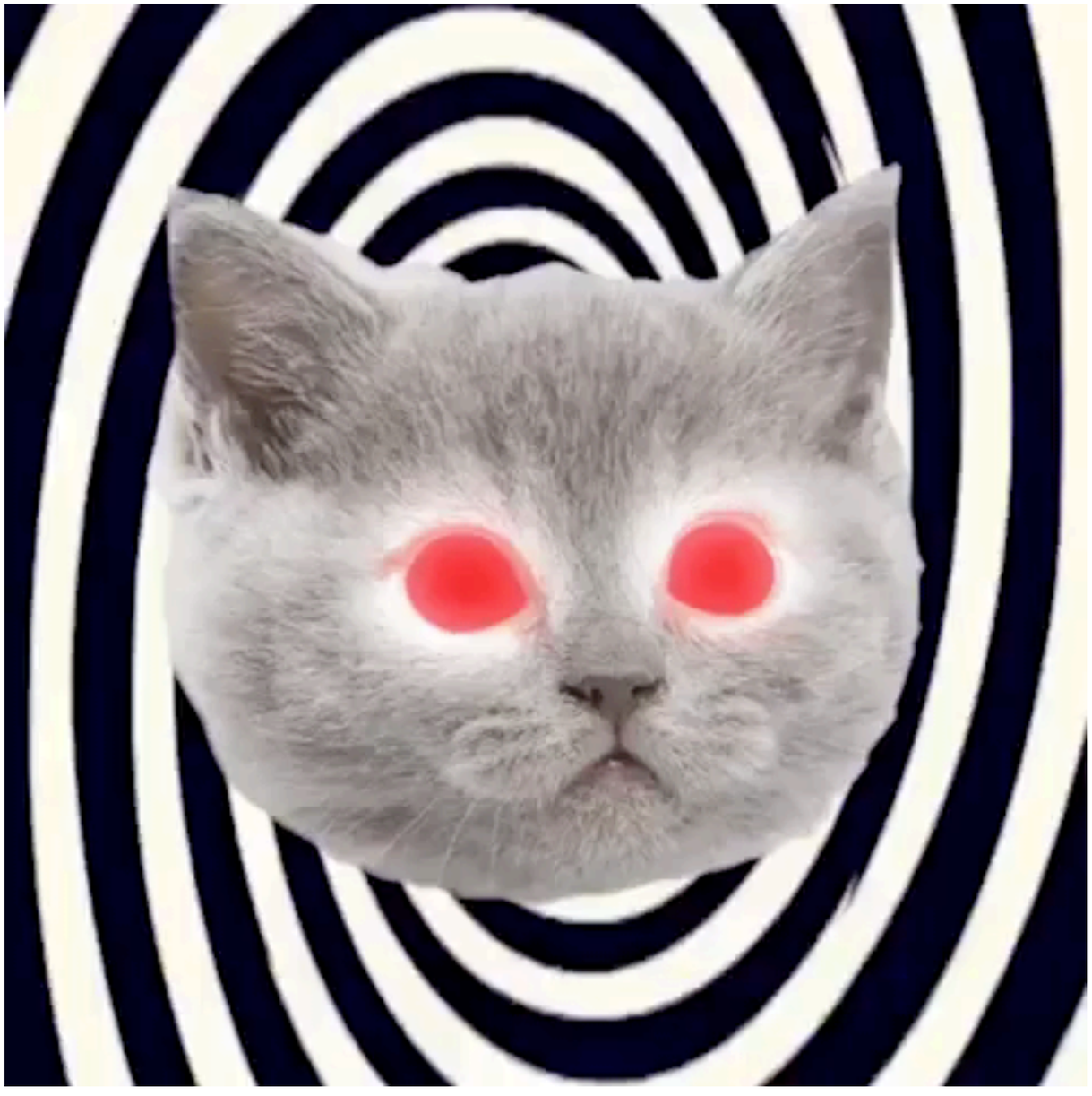
WHILE LOOP

```
let i = 0;

while (i < cats.length) {
  if (i === cats.length - 1) {
    info += 'and ' + cats[i] + '.';
  } else {
    info += cats[i] + ', ';
  }

  i++;
}
```

DO...WHILE LOOP

```
let i = 0;

do {
  if (i === cats.length - 1) {
    info += 'and ' + cats[i] + '.';
  } else {
    info += cats[i] + ', ';
  }

  i++;
} while (i < cats.length);
```

WATCH OUT FOR INFINITE LOOPS!



EXAMPLE TIME

Weekly Learning > Module 1> Code Examples

CONDITIONAL STRUCTURES

CONDITIONALS

- we often need to make decisions and execute code that depends on the decision
- conditional statements allow us to represent this decision making

IF ... ELSE

```
if (condition) {
    code to run if condition is true
} else {
    run some other code instead
}
```

NESTED IF ... ELSE

```
if (choice === 'sunny') {
  if (temperature < 86) {
    para.textContent = 'It is ' + temperature + ' degrees
    outside — nice and sunny. Let\'s go out to the beach,
    or the park, and get an ice cream.';
  } else if (temperature >= 86) {
    para.textContent = 'It is ' + temperature + ' degrees
    outside — REALLY HOT! If you want to go outside, make
    sure to put some sunscreen on.';
  }
}

//taken from https://developer.mozilla.org/en-
US/docs/Learn/JavaScript/Building_blocks/conditionals
```

LOGICAL OPERATORS

- && — AND; all conditions must be met to evaluate to true .
- || — OR; one or more of your conditions must be met to evaluate to true

LOGICAL OPERATORS

```
if (choice === 'sunny' && temperature < 86) {
  para.textContent = 'It is ' + temperature + ' degrees
  outside — nice and sunny. Let\'s go out to the beach,
  or the park, and get an ice cream.';
} else if (choice === 'sunny' && temperature >= 86) {
  para.textContent = 'It is ' + temperature + ' degrees
  outside — REALLY HOT! If you want to go outside, make
  sure to put some sunscreen on.';
}

// taken from https://developer.mozilla.org/en-
US/docs/Learn/JavaScript/Building_blocks/conditionals
```

TERNARY (CONDITIONAL) STATEMENT

```
// often used as a shortcut for an if/else statement
function getFee(isMember) {
  return (isMember ? '$2.00' : '$10.00');
}
// three operands - the condition, the code to execute if true and the code to execute if false

console.log(getFee(true));
```

SWITCH STATEMENTS

```
switch (expression) {
  case choice1:
    run this code
    break;

  case choice2:
    run this code instead
    break;

  // include as many cases as you like

  default:
    actually, just run this code
}
```

EXAMPLE TIME

Weekly Learning > Module 1> Code Examples

RECAP TIME

RECAP

- functions are blocks of reusable code
- there are built-in browser functions and we can create our own custom functions
- we can give our functions names or use anonymous functions
- functions can include parameters

RECAP

- Loops allow us to do things over and over again
- there are different types of loops available in JavaScript
- we can also use decision or control structures like a switch statement or if statement in order to make our code run only if a certain condition is met

TO DO THIS WEEK: QUIZ TWO & LAB ONE

NEXT WEEK :TROUBLESHOOTING, DEBUGGING, TOOLS & BEST PRACTICES