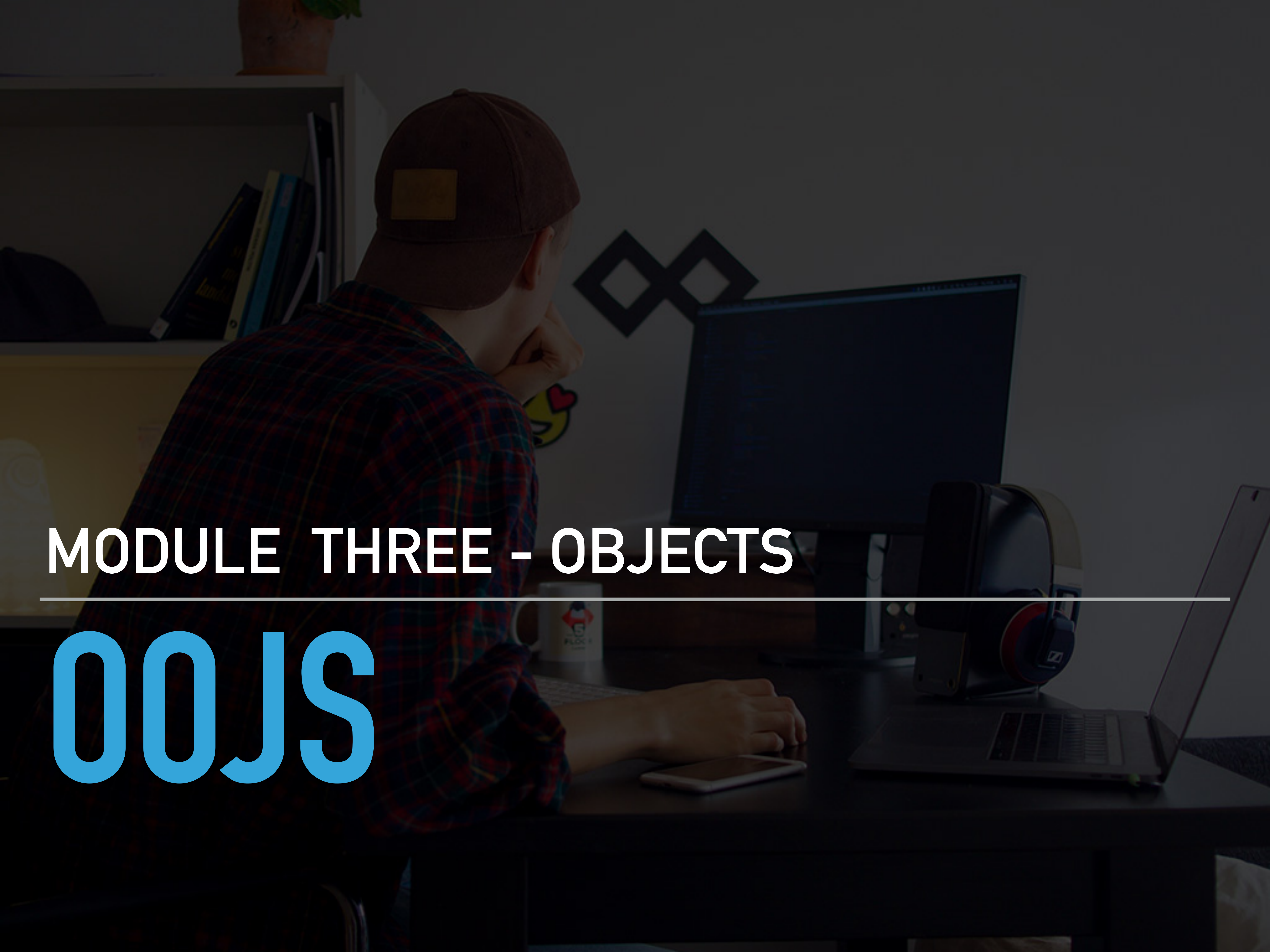
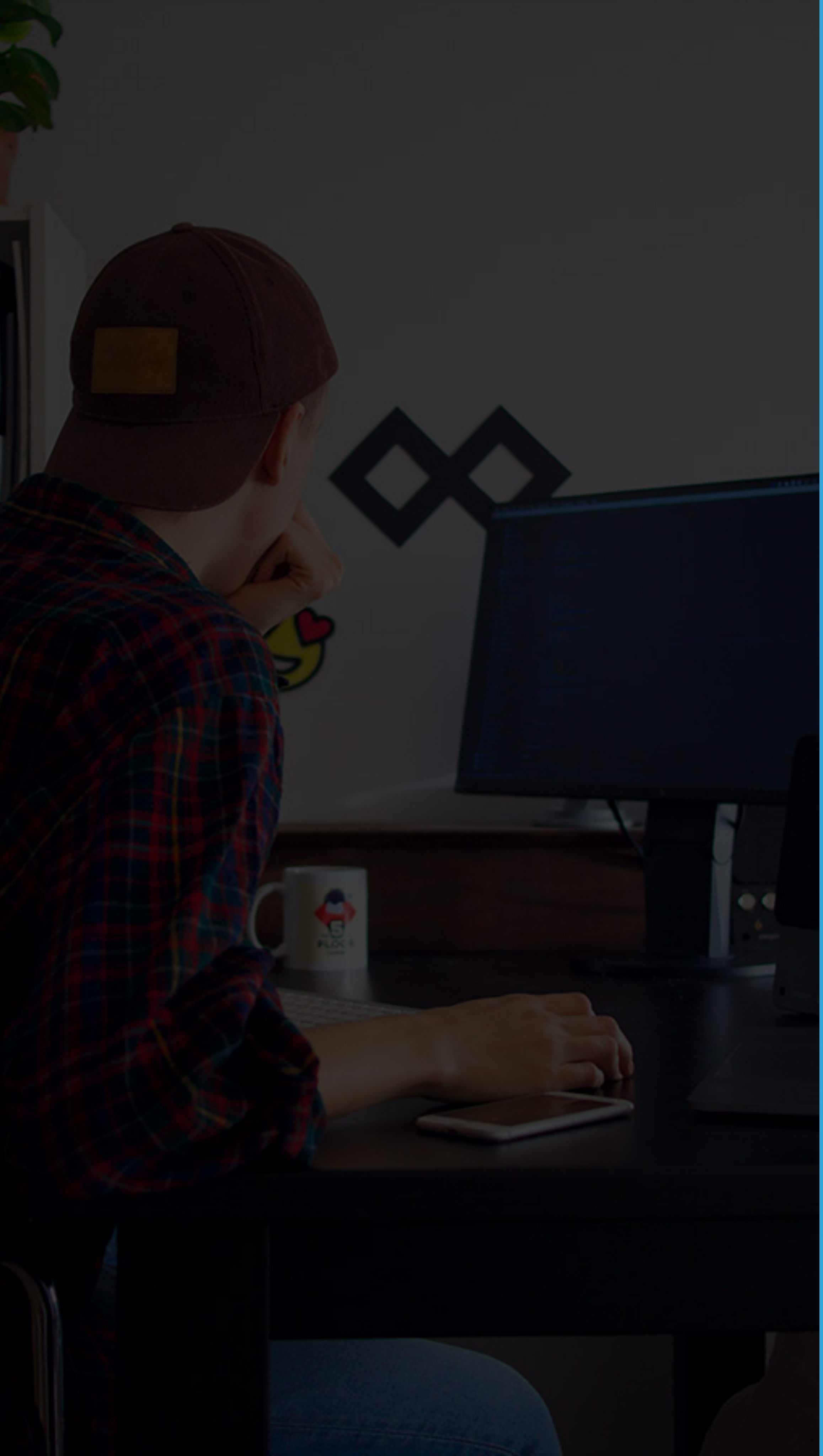


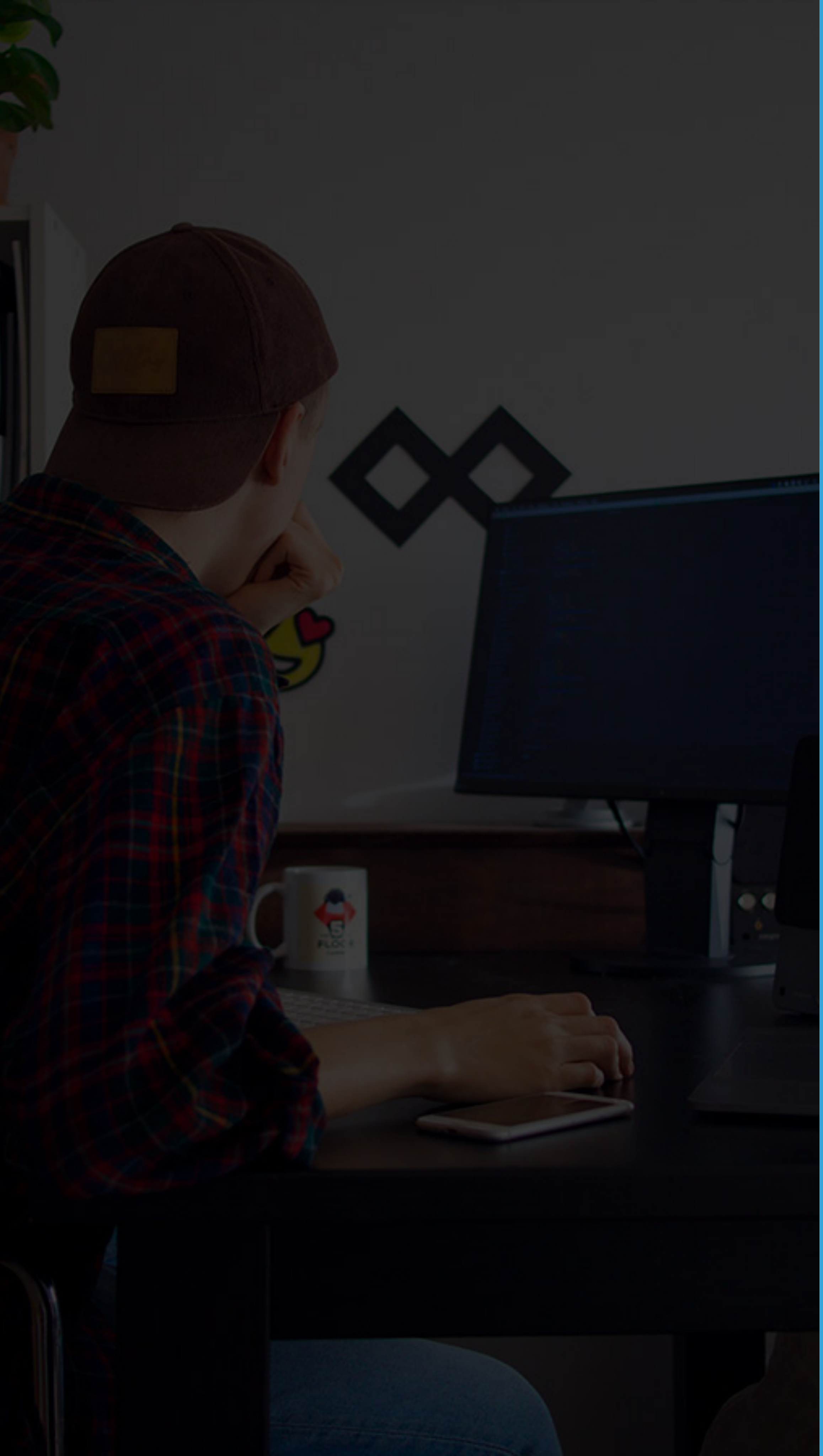
MODULE THREE - OBJECTS

OOJS



TODAY'S SCHEDULE

1. Cool Things
2. Midterm Review
3. Let's Review Objects
4. What is OOP?
5. Review - Objects!
6. OOJS / Constructor Functions
7. Not 'This', Again!
8. Other Methods to Create Object Instances
9. Recap & Next Week



LEARNING OBJECTIVES

1. understand what objects are, the basics of OOP and how to do JS in an object-oriented way
2. understand what a constructor function is and how to use it
3. understand and demonstrate other ways to create objects

COOL THINGS!

RESOURCES, LINKS TUTORIALS AND OTHER COOL THINGS…

- https://medium.com/javascript-scene/top-javascript-frameworks-and-topics-to-learn-in-2020-and-the-new-decade-ced6e9d812f9
- https://css-tricks.com/become-a-front-end-master-in-2020-with-these-10-project-ideas/
- https://watchandcode.com/
- https://hackernoon.com/jokes-programmers-will-understand-23d484d8bef8

RESOURCES, LINKS TUTORIALS AND OTHER COOL THINGS…

- https://technostacks.com/blog/front-end-development-tools/
- https://github.com/h5bp/Front-end-Developer-Interview-Questions
- https://www.freecodecamp.org/news/cracking-the-front-end-interview-9a34cd46237/
- http://endless.horse/
- https://trypap.com/

LET'S REVIEW – MIDTERM EXAM

LET'S REVIEW - OBJECTS

WHAT IS AN OBJECT?

- a collection of related data and/or functionality
- objects are made up of multiple members, each of which has a name and value
- Each name/value pair must be separated by a comma and the name and value in each case are separated by a colon

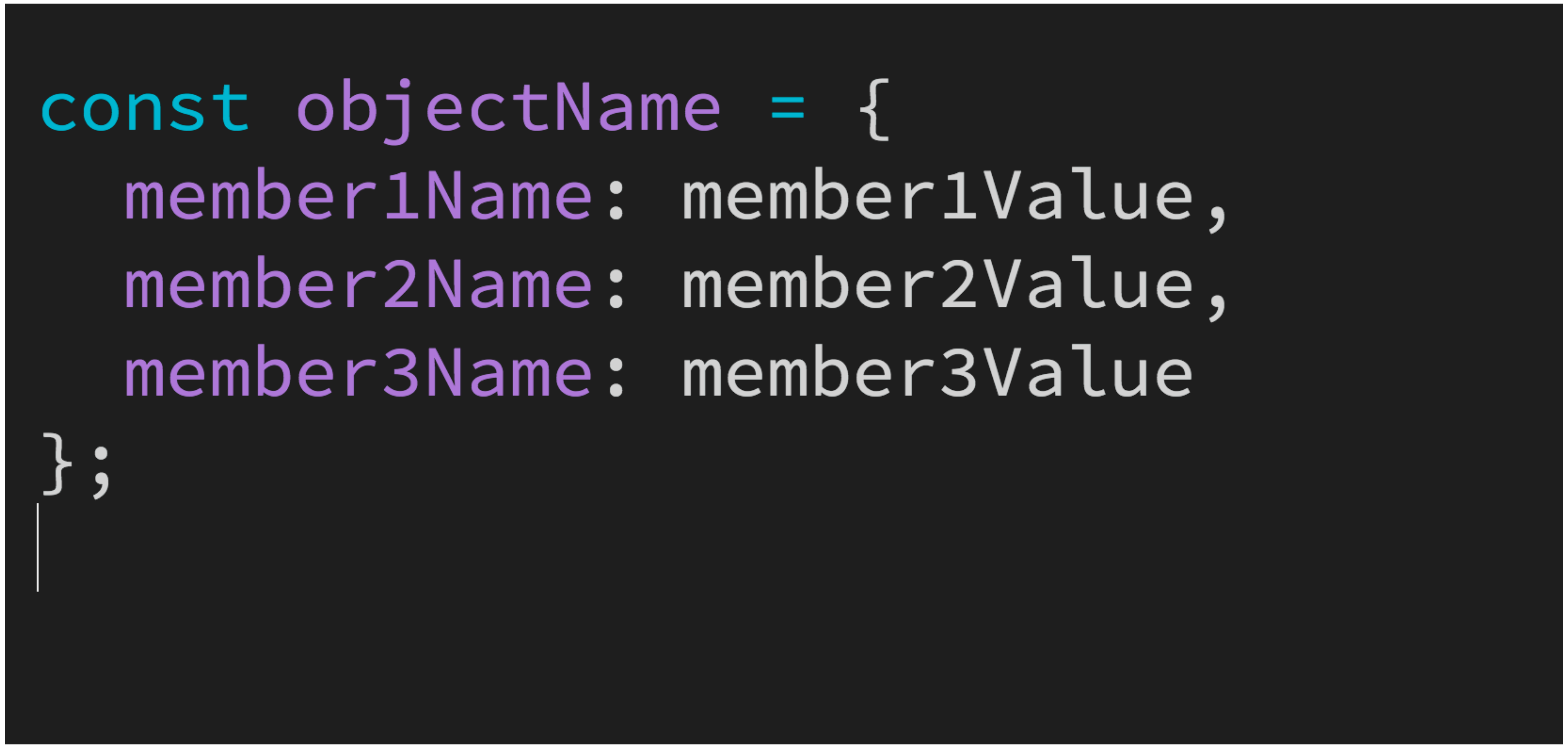
IN JAVASCRIPT, EVERYTHING (ALMOST) IS AN OBJECT

(And if it’s not an object, it still might act like one)

OBJECT LITERALS

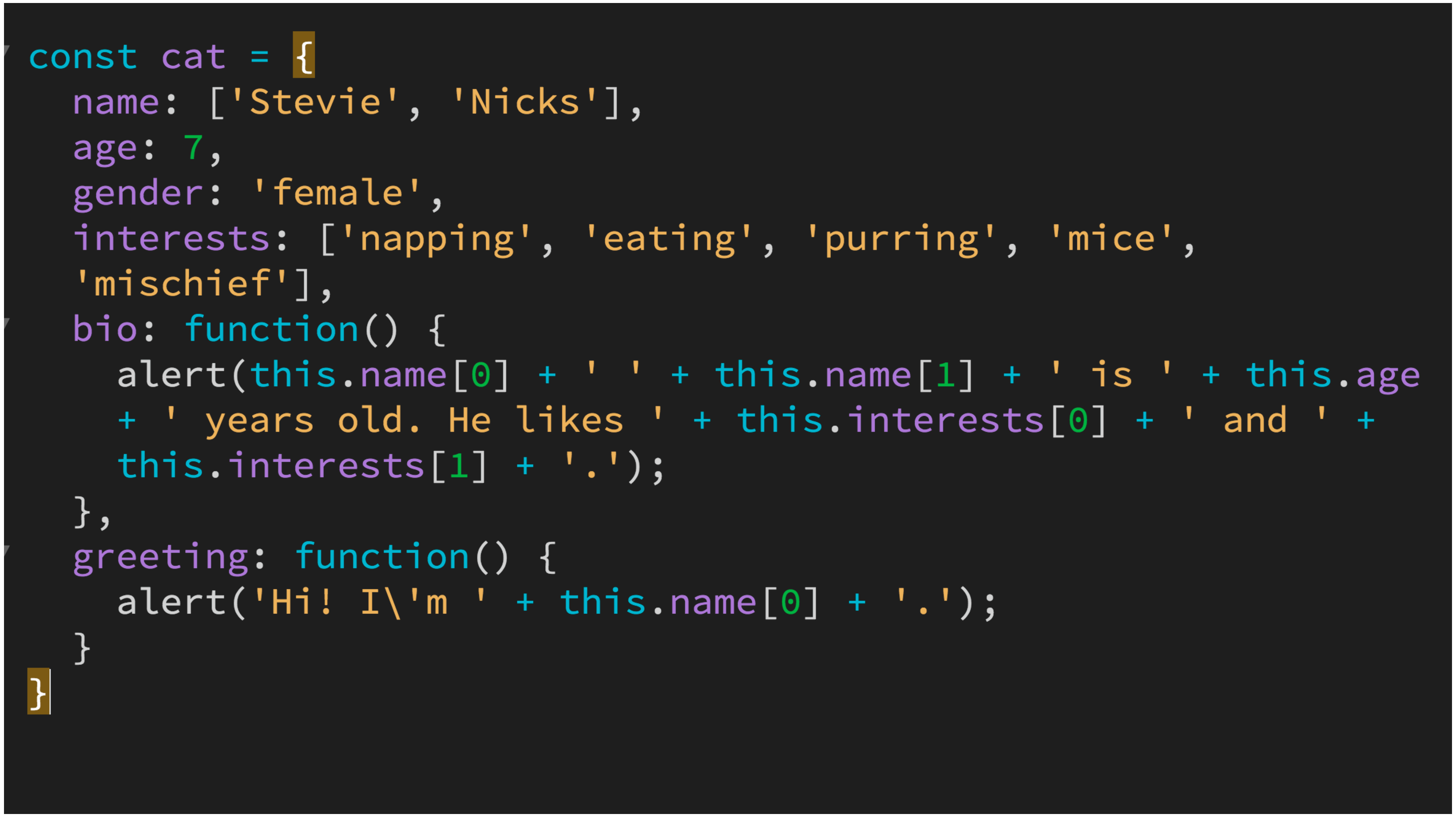
We call objects that we create object literals, they are different than objects instantiated from a class.

OBJECT LITERAL SYNTAX



```javascript
const objectName = {
  member1Name: member1Value,
  member2Name: member2Value,
  member3Name: member3Value
};
```

OBJECT LITERAL EXAMPLE



```
const cat = {
  name: ['Stevie', 'Nicks'],
  age: 7,
  gender: 'female',
  interests: ['napping', 'eating', 'purring', 'mice',
  'mischief'],
  bio: function() {
    alert(this.name[0] + ' ' + this.name[1] + ' is ' + this.age
    + ' years old. He likes ' + this.interests[0] + ' and ' +
    this.interests[1] + '.');
  },
  greeting: function() {
    alert('Hi! I\'m ' + this.name[0] + '.');
  }
}
```

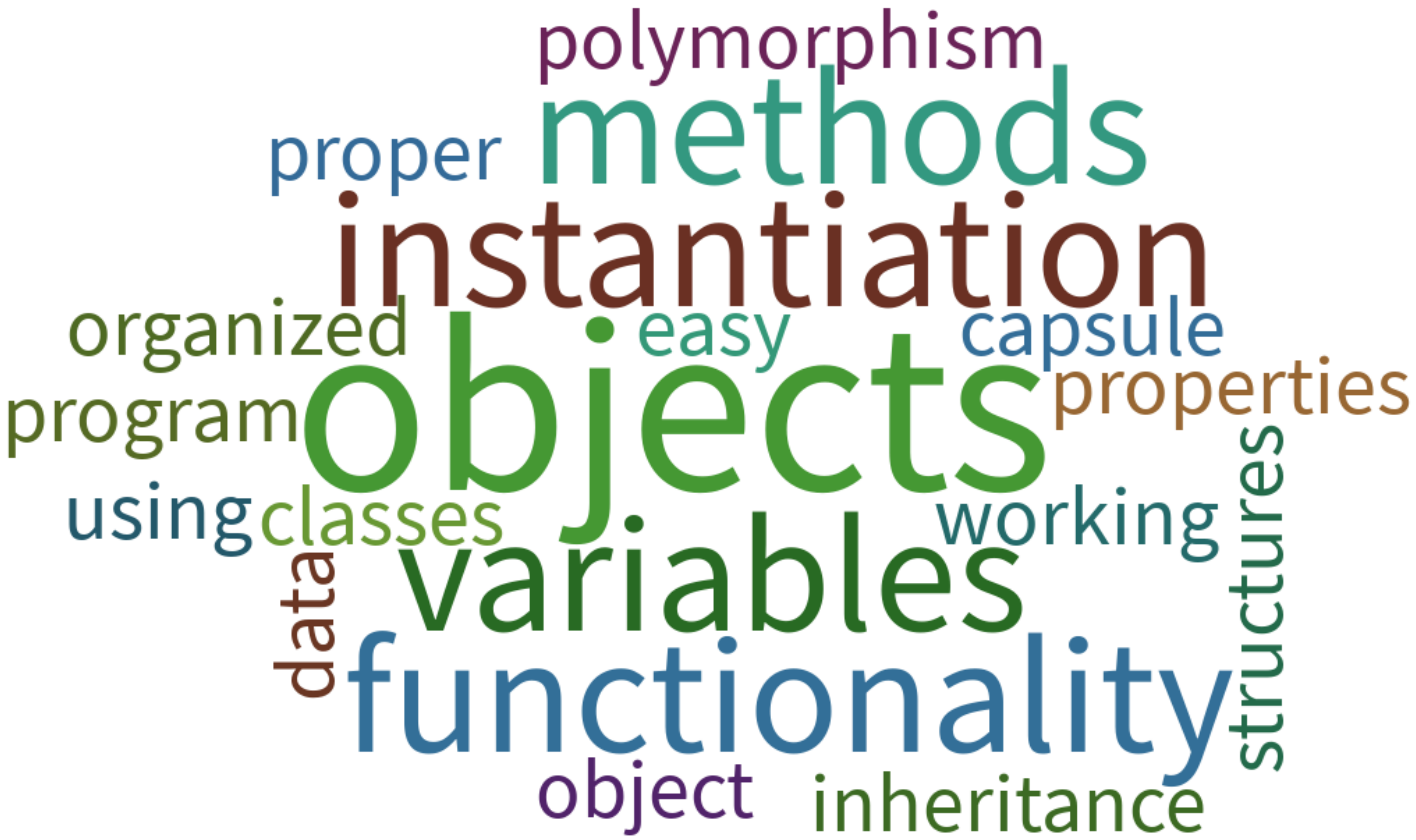
OBJECT LITERALS – WHEN TO USE?

- Need to transfer a series of structured, related data (i.e. sending a request to a server)
- more efficient than sending items
- easier to work with than an array

OOP BASICS

Object-oriented Programming in 7 minutes

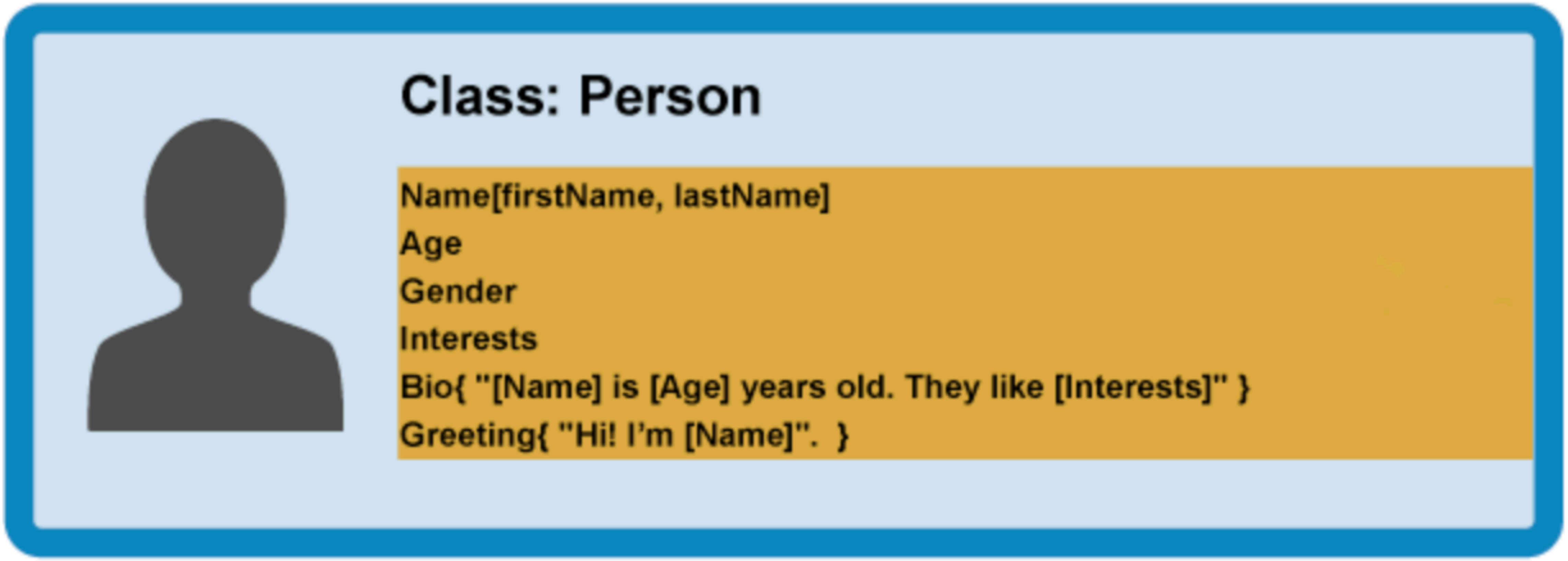
What is OOP?


polymorphism
proper
methods
instantiation
organized
easy
capsule
program
objects
properties
using
classes
working
data
variables
structures
functionality
object
inheritance

LET'S TALK ABOUT OOP

- we use objects to model real world things in our programs/ provide easy access to functionality
- objects contain related data and code which represent info or functionality
- our object is a nice, neat package for our data and functionality that is easy to access

DEFINING AN OBJECT TEMPLATE (CLASS)

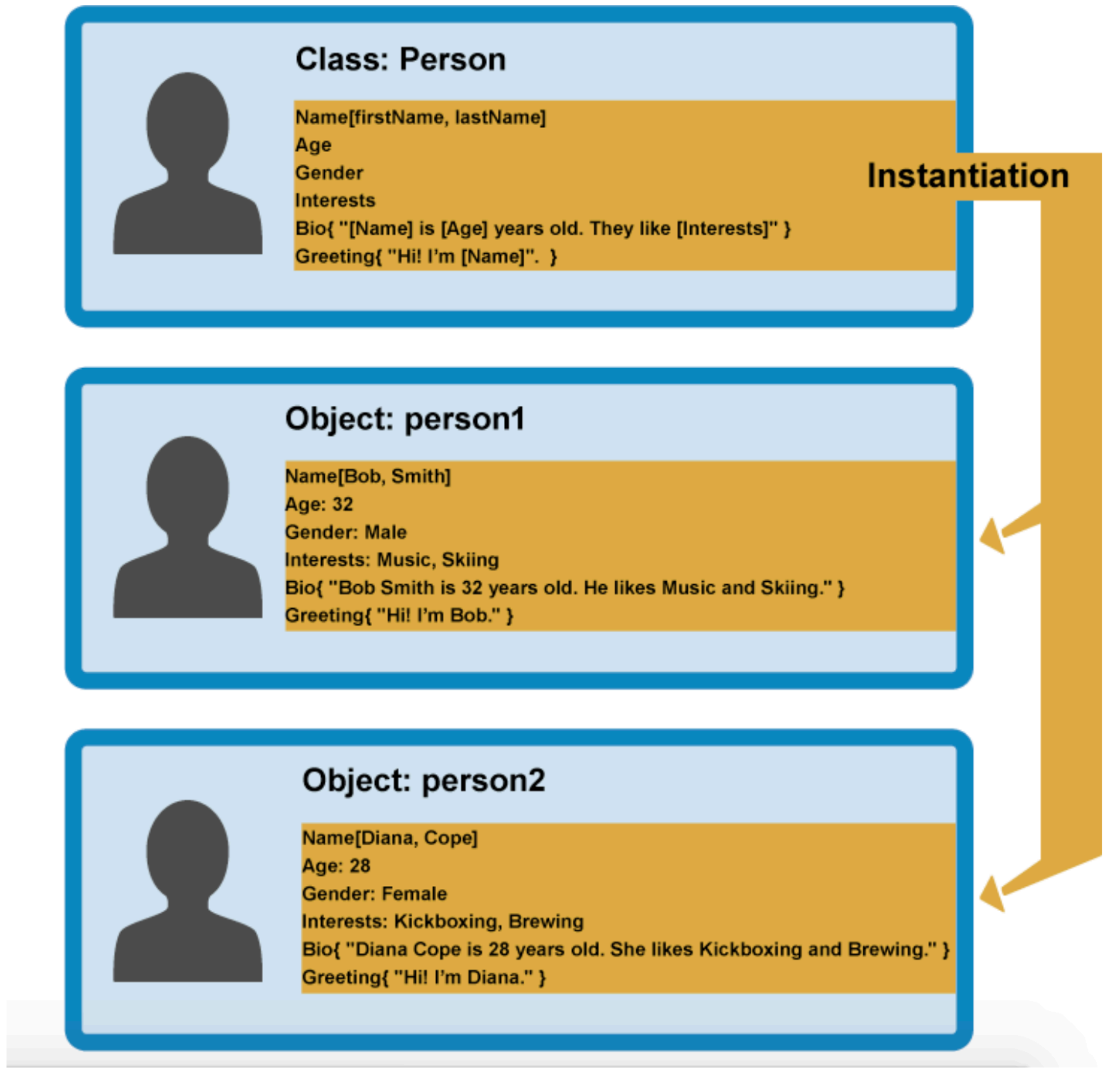


Class: Person

Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]" }
Greeting{ "Hi! I'm [Name]". }

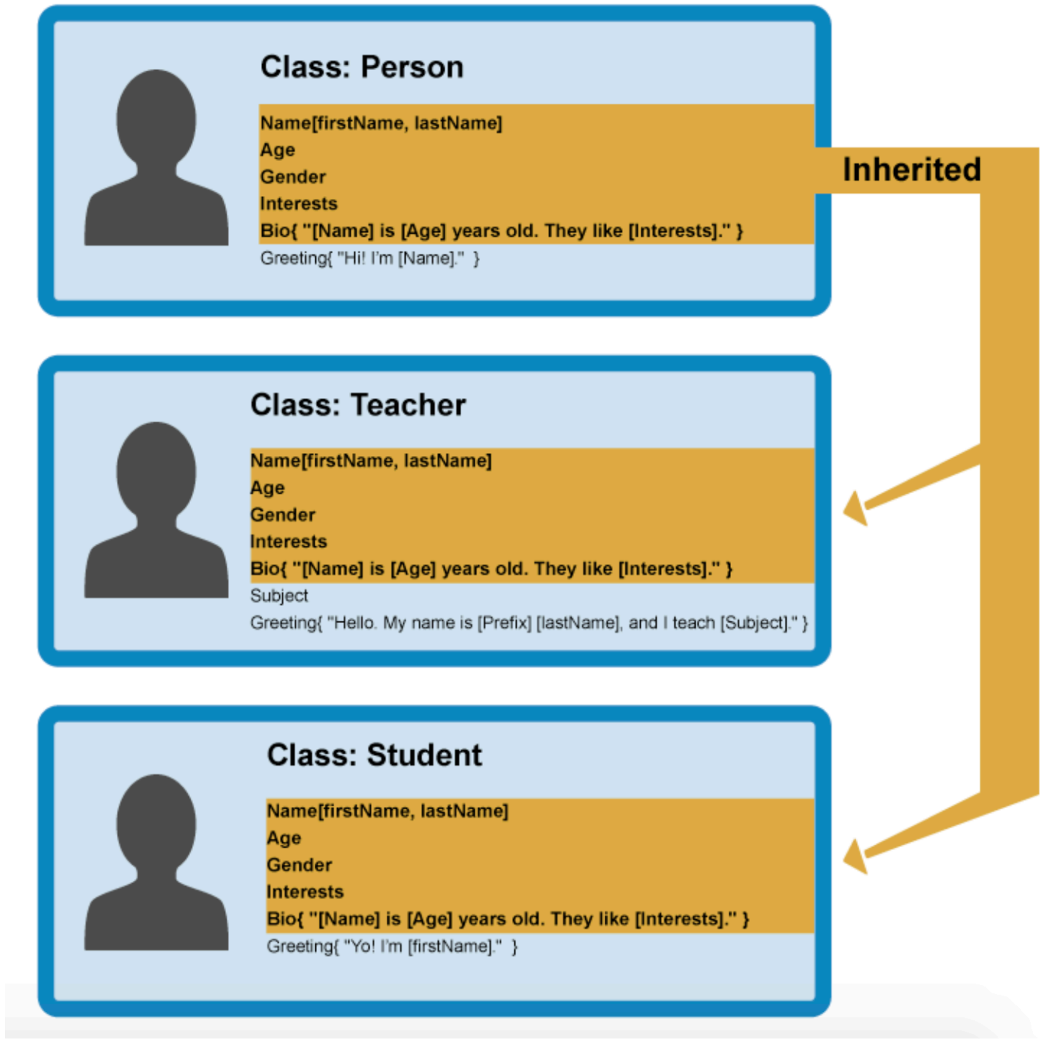
SOURCE: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object-oriented_JS

CREATING ACTUAL OBJECTS (INSTANTIATION)


Class: Person
Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]" }
Greeting{ "Hi! I'm [Name]". }
Instantiation
Object: person1
Name[Bob, Smith]
Age: 32
Gender: Male
Interests: Music, Skiing
Bio{ "Bob Smith is 32 years old. He likes Music and Skiing." }
Greeting{ "Hi! I'm Bob." }
Object: person2
Name[Diana, Cope]
Age: 28
Gender: Female
Interests: Kickboxing, Brewing
Bio{ "Diana Cope is 28 years old. She likes Kickboxing and Brewing." }
Greeting{ "Hi! I'm Diana." }

SOURCE: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object-oriented_JS

SPECIALIST CLASSES


Class: Person
Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]." }
Greeting{ "Hi! I'm [Name]." }
Inherited
Class: Teacher
Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]." }
Subject
Greeting{ "Hello. My name is [Prefix] [lastName], and I teach [Subject]." }
Class: Student
Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]." }
Greeting{ "Yo! I'm [firstName]." }

SOURCE: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object-oriented_JS

!IMPORTANT OOP CONCEPTS

- **Abstraction** : creating a simple model of a more complex thing
- **Encapsulation** : our object becomes a container or capsule for properties and methods
- **Instantiation**: creating an object instance from a class
- **Polymorphism**: the ability for multiple objects to implement the same functionality

OOJS

OOP + JS = OOJS

LET'S EXPLORE CREATING CLASSES VIA CONSTRUCTORS & CREATING OBJECT INSTANCES

Weekly Learning > Week 8 > Code Examples >

Example 01

METHOD ONE :
USING A PLAIN OL' FUNCTION

WHAT'S GOING ON HERE?

1. defining a function called createNewCat with a parameter called name
2. inside the function, we're creating an empty object
3. adding members to our object
4. returning the object so we can use it outside the function
5. call the function to create a new instance of this object

COOL, BUT...

we have to create an empty object each time and return, which is not as efficient (or cool) as we would like...

METHOD TWO : USING A CONSTRUCTOR FUNCTION

WHAT'S GOING ON HERE?

1. creating a class using a constructor function
2. creating properties and methods
3. instantiating new instances of this object with the new keyword and calling the constructor function with the required arguments
4. to access object instances, use the object namespace

```javascript
function Cat(name) {
  //capitalize your function name to show that it's a
  constructor function
  this.name = name; //new keyword, who dis?
  this.greeting = function() {
    let newPara = document.createElement('p');
    newPara.textContent = 'Meow there! I am ' + name +
    '.';
    main.appendChild(newPara);
  };
}
```

CONSTRUCTOR FUNCTIONS

The constructor function allows us to create a class.

It’s pretty much like a standard function, except we aren’t creating an empty object or returning it.

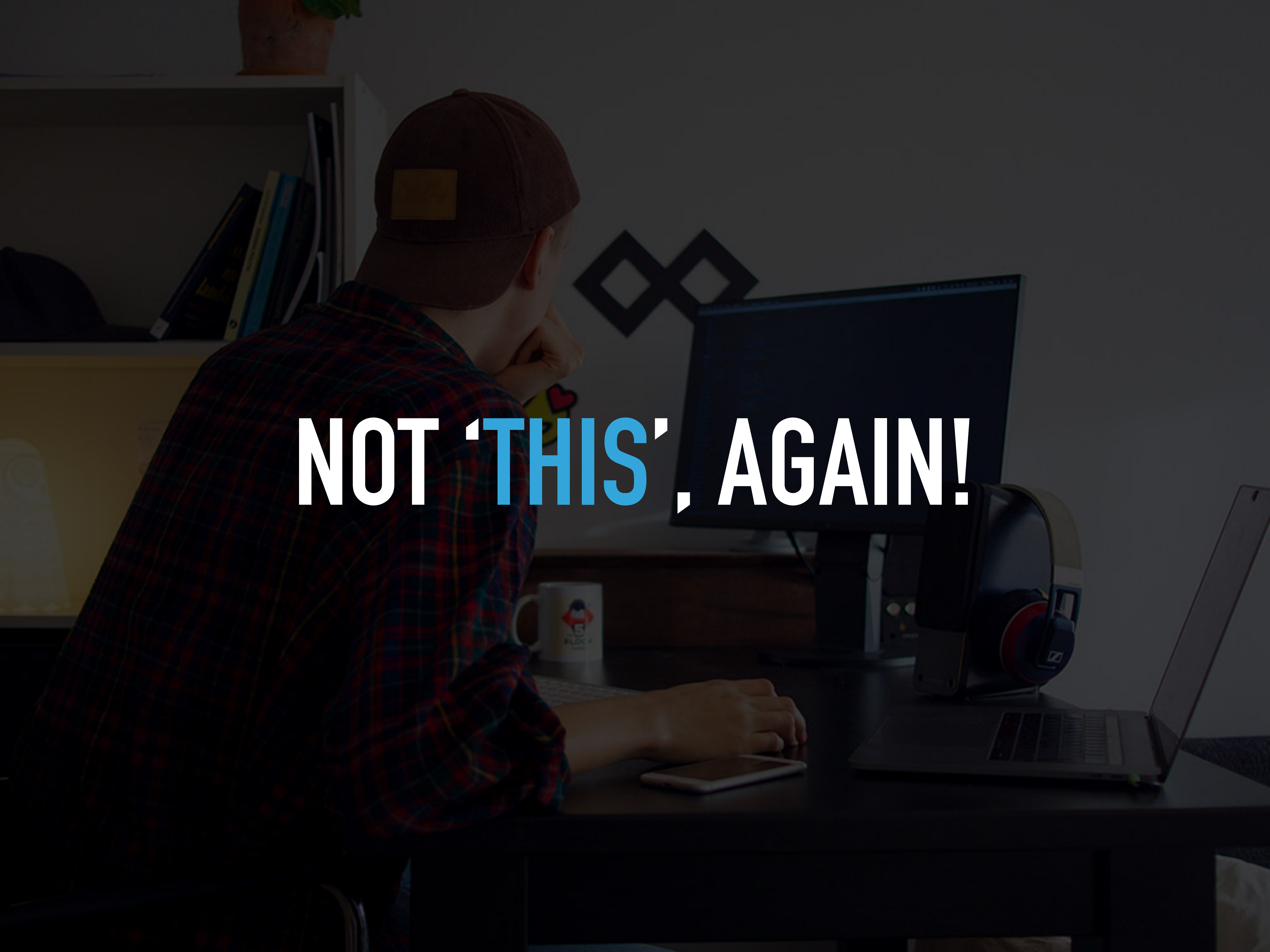
CONSTRUCTOR FUNCTIONS

Constructors help to give our code order. We can create constructors in one place, then create object instances when we need them.

COOL, BUT…

- one small drawback is that every time we call our constructor function, we are defining greeting( ) every time

- Don't worry, we'll improve on this when we talk about prototypes and inheritance.



NOT ‘THIS’, AGAIN!

WHAT IS THIS?

https://www.youtube.com/watch?v=2qMKjWf1KdE&t=74s

WHAT IS 'THIS'

- this keyword is often used inside of functions and objects
- where it's declared alters it's meaning
- always refers to one object, usually the object in which the function operates

WHAT DOES THIS REPRESENT IN DIFFERENT CONTEXTS ?

Weekly Learning > Week 8 > Code Examples >

Example 02

OTHER WAYS TO CREATE OBJECT INSTANCES

OTHER WAYS TO CREATE OBJECT INSTANCE – OBJECT CONSTRUCTOR

- object constructor
- pass object literal to object constructor

OTHER WAYS TO CREATE OBJECT INSTANCE - CREATE METHOD

- can create object instances without first creating constructor
- Built-in create method
- create new object based on existing object

Weekly Learning > Week 8 > Code Examples >

Example 03

RECAP & NEXT WEEK

OOP

- we use objects to model real world things in our programs/ provide easy access to functionality
- objects contain related data and code which represent info or functionality

OOJS

- We can create define object templates using a constructor function (like a class in JS)
- ‘this’ represents different things according to context.

NEXT WEEK - PROTOTYPES & INHERITANCE, A CLASSIER WAY & JSON

THANKS TO (SOURCES):

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object-oriented_JS

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object_prototypes