MODULE THREE - OBJECTS

INTRO TO OBJECTS

TODAY'S SCHEDULE

1. Cool Things
2. Review & Learning Objectives
3. So....what's the deal with objects?
4. An Intro to Objects
5. Object Literals
6. Getting & Setting Object Members
7. What is 'this'?
8. Recap & Weekly Tasks (Quiz Four, Assign One)

COOL THINGS

RESOURCES, LINKS TUTORIALS AND OTHER COOL THINGS…

- https://www.destroyallsoftware.com/talks/wat
- https://medium.com/dailyjs/the-why-behind-the-wat-an-explanation-of-javascripts-weird-type-system-83b92879a8db
- https://www.freecodecamp.org/news/web-development-2020/
- https://www.shopify.ca/partners/blog/web-design-trends

RESOURCES, LINKS TUTORIALS AND OTHER COOL THINGS…

- https://github.com/sorrycc/awesome-javascript
- https://bestofjs.org/projects/you-dont-know-js
- https://bestofjs.org/projects/react
- https://jamstack.wtf/
- https://snipcart.com/blog/jamstack
- http://procatinator.com/

LEARNING OBJECTIVES

1. construct a variety of programming structures including variables, constants, arrays, objects, functions, conditionals, and constructors
2. optimize code for increased functionality, performance, readability, and reusability;

WEEK 3 RECAP

- sometimes things go wrong in our code
- when there are errors in our code, we need to find them using problem-solving, deduction and various tools available
- we also need to handle errors beyond our control (in a graceful way)
- writing quality code is awesome & there are tools & best practices to follow that can help us do this

INTRO TO OBJECTS & OBJECT LITERALS

WHAT'S THE BIG DEAL WITH OBJECTS AND JS?

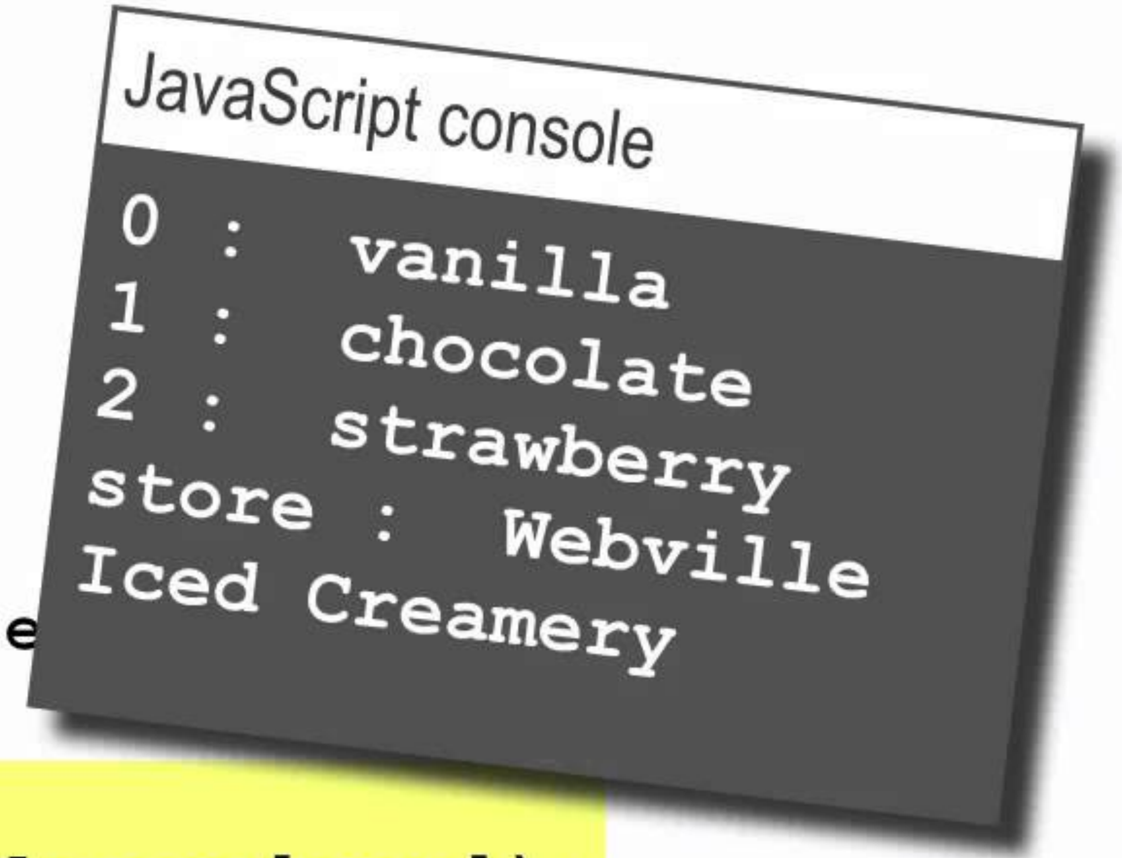
EVERYTHING (ALMOST) IS AN OBJECT

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

```
var flavors = [
    "vanilla",
    "chocolate",
    "strawberry"
];

flavors.store = "Webville Ice

for (var prop in flavors) {
   console.log(prop, ": ", flavors[prop]);
}
```



```
0 :  vanilla
1 :  chocolate
2 :  strawberry
store :  Webville
Iced Creamery
```

LET'S TALK ABOUT OBJECTS…

When we describe objects, we usually focus on their **characteristics** and **what they can do**.

In programming, we can think of characteristics as **PROPERTIES** and the things the object can do as **METHODS**.

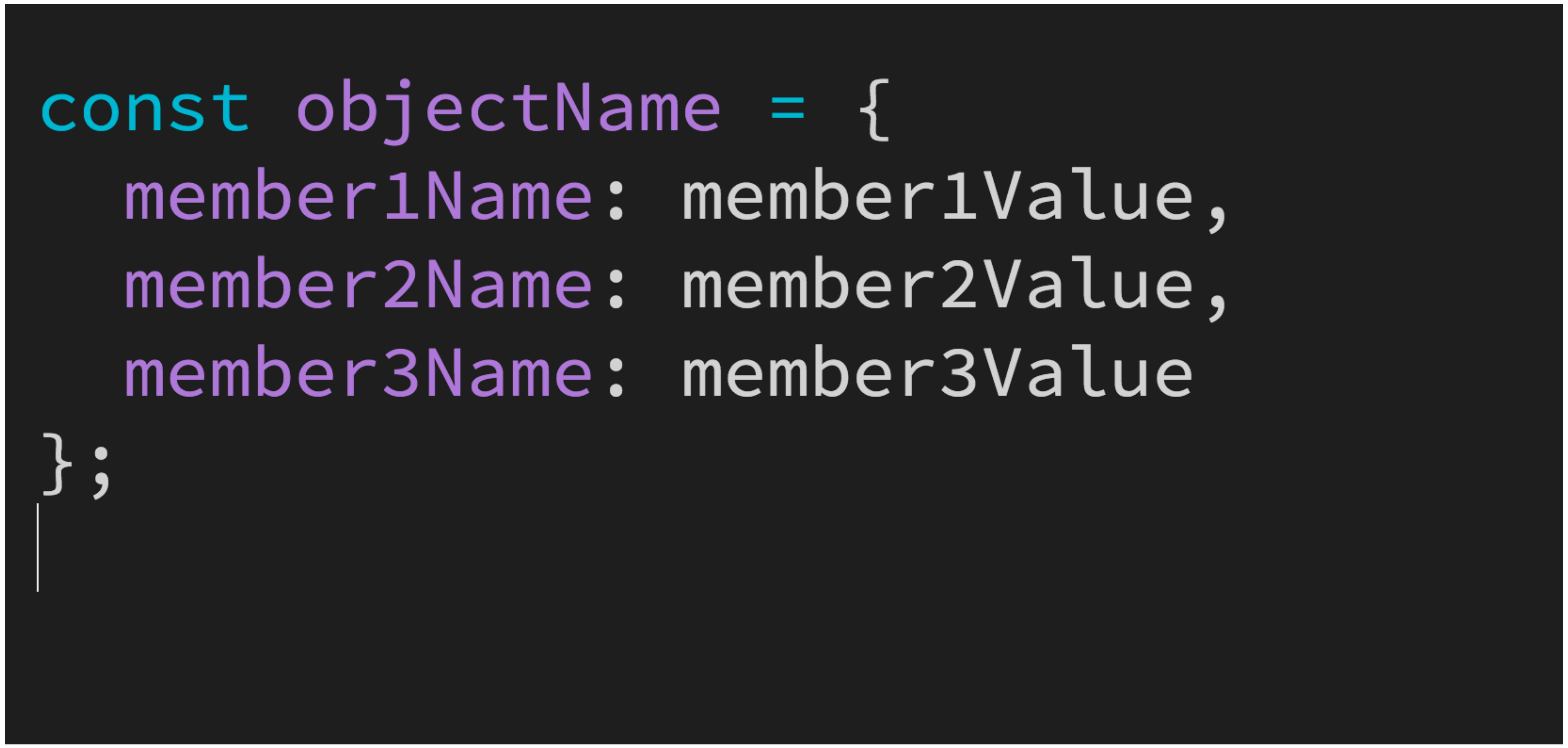
WHAT IS AN OBJECT?

- 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

OBJECT LITERALS

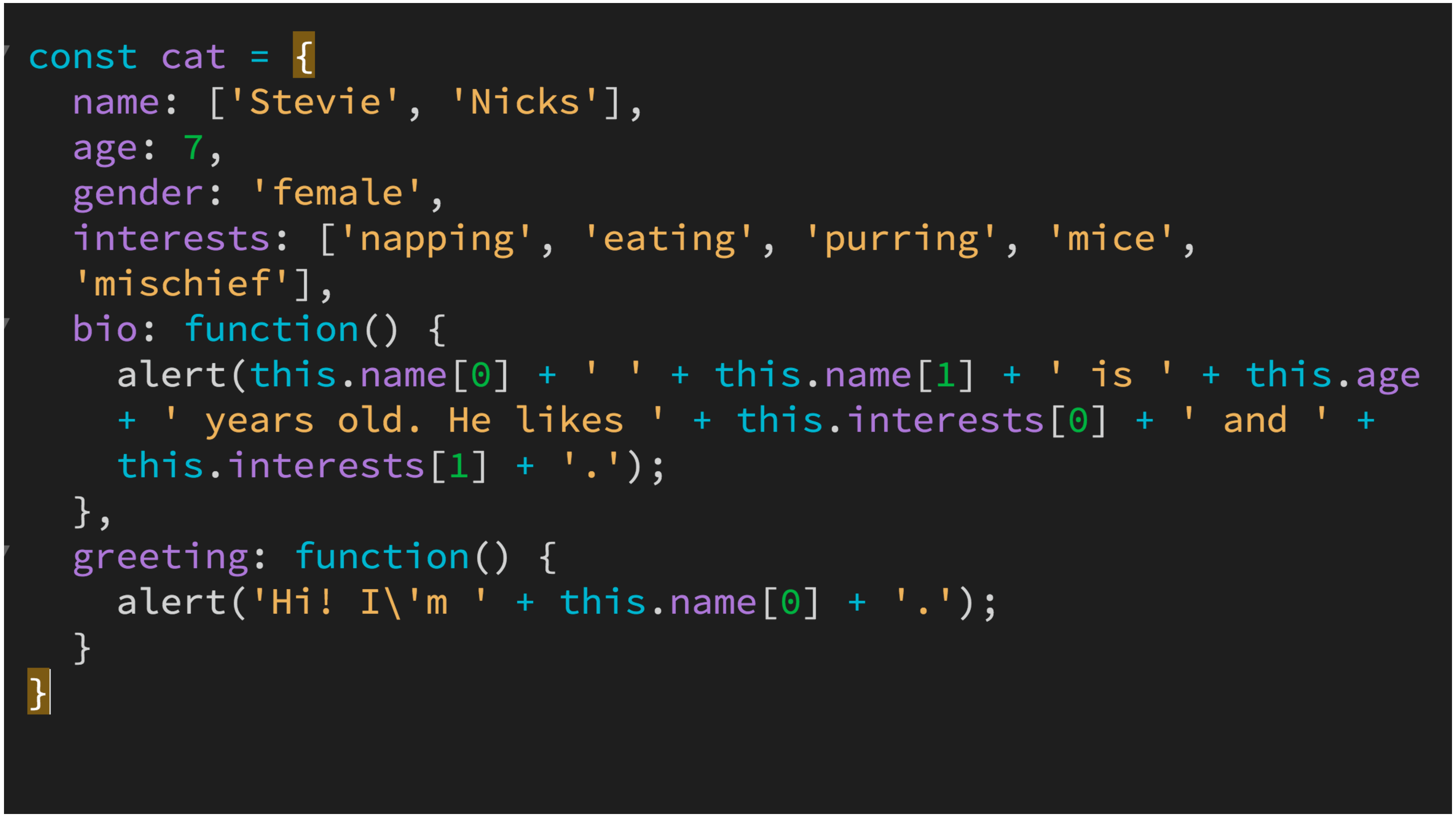
We call objects that we create object literals, they are different than objects instantiated from a class.

OBJECT LITERAL SYNTAX



```
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)
- <u>more efficient</u> to sending items
- <u>easier</u> to work with than an array

LET'S CREATE AN OBJECT LITERAL …

ACCESSING OBJECT MEMBERS

- We can access object members **encapsulated** inside our objects using dot or bracket notation
- the object name goes first and acts as a **namespace**

ACCESSING OBJECT MEMBERS - DOT NOTATION

```
//we can access object literal methods and propeties using dot notation

console.log(cat.name[0]);
cat.bio();
```

ACCESSING OBJECT MEMBERS - BRACKET NOTATION

```
// we can access object literal properties using
bracket notation
console.log(cat['name'][1]);
```

BRACKET NOTATION OR DOT NOTATION?

- **dot notation** is generally considered better as we can access both properties and methods
- **bracket notation** allows access to properties
- **bracket notation** can allow you to dynamically set a member name whereas dot notation does not

SETTING OBJECT MEMBERS

- Can **set** (update) the value of object members
- **declare** the member you want to set (using dot or bracket notation)
- can also **create completely new members**

GETTING & SETTING OBJECT MEMBERS

WHAT IS ‘THIS’?

LET'S TALK ABOUT 'THIS'

- refers to the current object the code is being written inside
- useful because it ensures the proper values are used when the member context changes

PAIR PROGRAMMING TIME!

Woot! Woot!

RECAP & NEXT WEEK

IN WEEK 4, WE LEARNED

- How to create an object literal
- Why we use object literals
- How to access and update member properties
- What is this?
- We've been using objects all along

WEEKLY LEARNING: QUIZ FOUR & ASSIGNMENT ONE

NEXT WEEK:
INTERACTING WITH THE DOM

SOURCES:

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