

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/JS-SF-8-resources` repo to your computer
2. Open the `06-objects-json > starter-code` folder in your code editor

JAVASCRIPT DEVELOPMENT

OBJECTS AND JSON

LEARNING OBJECTIVES

At the end of this class, you will be able to

- › Identify likely objects, properties, and methods in real-world scenarios
- › Create JavaScript objects using object literal notation
- › Implement and interface with JSON data

AGENDA

- Objects, properties, and methods
- Lab: Translate real world scenarios into objects
- Lab: Create objects
- JSON
- Lab: Work with JSON

INTRO TO CRUD AND FIREBASE

WEEKLY OVERVIEW

WEEK 4

Slackbot Lab / Objects & JSON

WEEK 5

Intro to the DOM / Intro to jQuery

WEEK 6

Ajax & APIs / Asynchronous JavaScript & Callbacks

EXIT TICKET QUESTIONS

1. practical examples of hoisting and gotchas
2. Hubot installfest was complicated; is there a better alternative exercise for us? (paraphrase)
3. What's the difference between `msg.send` vs `msg.reply`? How do you work with APIs?
4. When would it be valuable to use hoist?
5. Do most people create hubots with the same script as `coffee.example` or `javascript`
6. How to store/use variables within `slackbot.js` file

EXIT TICKET QUESTIONS (CONTINUED)

7. How can I have a counter that increments every time the bot is called? How can I get this counter to de-increment over time?
8. How could I find out if two commas were used within a set number of characters (regEx?)

WARMUP EXERCISE



TYPE OF EXERCISE

▶ Pairs

TIMING

3 min

1. For the thing you've been assigned, make a list of attributes (descriptions) and actions (things it can do).

OBJECTS

OBJECTS ARE A SEPARATE DATA TYPE

STRING

NUMBER

ARRAY

BOOLEAN

OBJECT

AN OBJECT IS A COLLECTION OF PROPERTIES

properties

```
let favorites = {  
  fruit: "apple",  
  vegetable: "carrot"  
}
```

PROPERTY = KEY & VALUE

- A **property** is an association between a key and a value
 - **key**: name (often descriptive) used to reference the data
 - **value**: the data stored in that property
- A property is sometimes referred to as a **key-value pair**



KEY-VALUE PAIR

- A property is sometimes referred to as a **key-value pair**

```
let favorites = {  
  fruit: "apple",  
  vegetable: "carrot"  
}
```

key-value pair



AN OBJECT IS NOT ORDERED

```
0  [
1  "apple",
2  "pear",
   "banana"
]
```

```
{
  fruit: "apple",
  vegetable: "carrot",
  fungus: "trumpet mushroom"
}
```

A METHOD IS A PROPERTY WHOSE VALUE IS A FUNCTION

```
let favorites = {  
  fruit: "apple",  
  vegetable: "carrot",  
  declare: function() {  
    console.log("I like fruits and vegetables!");  
  }  
}
```

method

YOU REFERENCE A PROPERTY WITH DOT NOTATION

object

object name

referencing properties

```
let favorites = {  
  fruit: "apple",  
  veg: "carrot",  
  declare: function() {  
    console.log("I like fruit and veg");  
  }  
}
```

property name

```
favorites.fruit  
> "apple"  
favorites.veg  
> "carrot"
```

object name

calling a method

method name

```
favorites.declare()  
> "I like fruits and vegetables"
```

EXERCISE — OBJECTS



KEY OBJECTIVE

- ▶ Create JavaScript objects using object literal notation

TYPE OF EXERCISE

- ▶ Pairs (same pair as for previous exercise)

TIMING

3 min

1. On your desk or on the wall, write code to create a variable whose name corresponds to the thing you were assigned in the previous exercise (cloud, houseplant, nation, office chair, or airplane).
2. Write code to add a property to the object and specify a value for the property.
3. Write code to add a method to the object, and specify a value for the method (use a comment or `console.log()` statement for the function body).
4. **BONUS:** Rewrite your answers for 1-3 as a single JavaScript statement.

REAL WORLD SCENARIO

A user, browsing on a shopping website, searches for size 12 running shoes, and examines several pairs before purchasing one.

OBJECTS = NOUNS

A **user**, browsing on a **shopping website**, searches for size 12 running shoes, and examines **several pairs** before purchasing one.

implicit object:

shopping cart

PROPERTIES = ADJECTIVES

A user, browsing on a shopping website, searches for **size 12** **running** shoes, and examines several pairs before purchasing one.

implicit properties:

for each pair of shoes:

price
color

for the shopping cart:

contents
total
shipping
tax

METHODS = VERBS

A user, browsing on a shopping website, **searches** for size 12 running shoes, and examines several pairs before purchasing one.

implicit methods:

for each pair of shoes:

add to cart

for the shopping cart:

**calculate shipping
calculate tax
complete purchase
remove item**

PRACTICE: REAL WORLD SCENARIOS & OBJECTS

EXERCISE — REAL WORLD SCENARIOS & OBJECTS



EXERCISE

KEY OBJECTIVE

- ▶ Identify likely objects, properties, and methods in real-world scenarios

TYPE OF EXERCISE

- ▶ Groups of 3-4

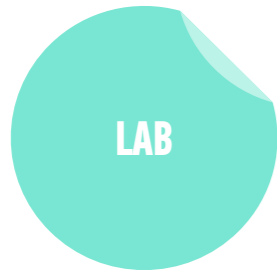
TIMING

10 min

1. Read through your scenario together.
2. Identify and write down likely objects, properties, and methods in your scenario. (Remember to consider implicit objects as well as explicit ones.)
3. Choose someone to report you results to the class.

PRACTICE: MONKEYS

LAB — OBJECTS



KEY OBJECTIVE

- ▶ Create JavaScript objects using object literal notation

TYPE OF EXERCISE

- ▶ Individual or pair

TIMING

20 min

1. Open starter-code > 1-object-exercise > monkey.js in your editor.
2. Create objects for 3 different monkeys each with the properties name, species, and foodsEaten, and the methods eatSomething(thingAsString) and introduce.
3. Practice retrieving properties and using methods with both dot notation and bracket syntax.

JSON IS A DATA FORMAT BASED ON JAVASCRIPT

object

```
let instructor = {
  firstName: 'Sasha',
  lastName: 'Vodnik',
  city: 'San Francisco',
  classes: [
    'JSD', 'FEWD'
  ],
  classroom: 8,
  launched: true,
  dates: {
    start: 20170906,
    end: 20171113
  }
}
```

JSON

```
{
  "firstName": "Sasha",
  "lastName": "Vodnik",
  "city": "San Francisco",
  "classes": [
    "JSD", "FEWD"
  ],
  "classroom": 8,
  "launched": true,
  "dates": {
    "start": 20170906,
    "end": 20171113
  }
}
```

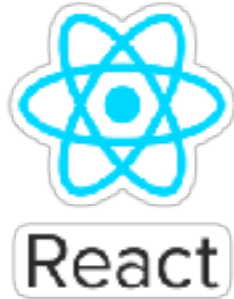
JSON

- ▶ Easy for humans to read and write
- ▶ Easy for programs to parse and generate

```
{
  "firstName": "Sasha",
  "lastName": "Vodnik",
  "city": "San Francisco",
  "classes": [
    "JSD", "FEWD"
  ],
  "classroom": 8,
  "launched": true,
  "dates": {
    "start": 20170906,
    "end": 20171113
  }
}
```

JSON IS NOT JAVASCRIPT-SPECIFIC

- Used across the web by programs written in many languages



JSON RULES

- Property names must be double-quoted strings.
- Trailing commas are forbidden.
- Leading zeroes are prohibited.
- In numbers, a decimal point must be followed by at least one digit.
- Most characters are allowed in strings; however, certain characters (such as ', ", \, and newline/tab) must be 'escaped' with a preceding backslash (\) in order to be read as characters (as opposed to JSON control code).
- All strings must be double-quoted.
- No comments!

EXERCISE — JSON



KEY OBJECTIVE

- ▶ Implement and interface with JSON data

TYPE OF EXERCISE

- ▶ Groups of 2-3

TIMING

3 min

1. Write JSON code that contains an error.
2. Write your code on the wall.
3. When everyone's code is done, we will look at the code together as a class and practice identifying errors.

WORKING WITH NESTED DATA STRUCTURES

- Parse the JSON to a JavaScript object (or array!)
- View the resulting data structure.
- Locate the data you want to reference within the data structure.
- Use dot syntax and/or square bracket notation to reference the next level down, then repeat for each level until you get to the data you're seeking.

WORKING WITH NESTED DATA STRUCTURES

1. PARSE THE JSON TO A JAVASCRIPT OBJECT (OR ARRAY!)

2. VIEW THE RESULTING DATA STRUCTURE

3. LOCATE THE DATA YOU WANT TO REFERENCE

4. USE DOT SYNTAX OR SQUARE BRACKET NOTATION TO MOVE DOWN A LEVEL, THEN REPEAT

WORKING WITH NESTED DATA STRUCTURES

1. PARSE THE JSON TO A JAVASCRIPT OBJECT (OR ARRAY!)

```
let person = '{"firstName":  
"Sasha","lastName": "Vodnik","city":  
"San Francisco","classes": ["JSD",  
"FEWD"],"classroom": 8,"launched":  
true,"dates": {"start": 20170906,"end":  
20171113}}';
```




```
let personObject = JSON.parse(person);
```

WORKING WITH NESTED DATA STRUCTURES

2. VIEW THE RESULTING DATA STRUCTURE

```
let personObject = JSON.parse(person);  
console.log(personObject);  
>
```



```
city: "San Francisco"  
▼ classes: Array(2)  
  0: "JSD"  
  1: "FEWD"  
  length: 2  
  ► __proto__: Array(0)  
classroom: 8  
▼ dates:  
  end: 20171113  
  start: 20170906  
  ► __proto__: Object  
firstName: "Sasha"  
lastName: "Vodnik"  
launched: true
```

WORKING WITH NESTED DATA STRUCTURES

3. LOCATE THE DATA YOU WANT TO REFERENCE

```
city: "San Francisco"
▼ classes: Array(2)
  0: "JSD"
  1: "FEWD"
  length: 2
  ► __proto__: Array(0)
classroom: 8
▼ dates:
  end: 20171113
  start: 20170906
  ► __proto__: Object
firstName: "Sasha"
lastName: "Vodnik"
launched: true
```


WORKING WITH NESTED DATA STRUCTURES

4. USE DOT SYNTAX OR SQUARE BRACKET NOTATION TO MOVE DOWN A LEVEL, THEN REPEAT

direct property:

```
console.log(personObject.city);  
> "San Francisco"
```

```
city: "San Francisco"  
▼ classes: Array(2)  
  0: "JSD"  
  1: "FEWD"  
  length: 2  
  ▶ __proto__: Array(0)  
classroom: 8  
▼ dates:  
  end: 20171113  
  start: 20170906  
  ▶ __proto__: Object  
firstName: "Sasha"  
lastName: "Vodnik"  
launched: true
```



WORKING WITH NESTED DATA STRUCTURES

4. USE DOT SYNTAX OR SQUARE BRACKET NOTATION TO MOVE DOWN A LEVEL, THEN REPEAT

```
city: "San Francisco"
▼ classes: Array(2)
  0: "JSD"
  1: "FEWD"
  length: 2
  ► __proto__: Array(0)
classroom: 8
▼ dates:
  end: 20171113
  start: 20170906
  ► __proto__: Object
firstName: "Sasha"
lastName: "Vodnik"
launched: true
```

direct property > array element

```
console.log(personObject.classes[0]);
> "JSD"
```

WORKING WITH NESTED DATA STRUCTURES

4. USE DOT SYNTAX OR SQUARE BRACKET NOTATION TO MOVE DOWN A LEVEL, THEN REPEAT

```
city: "San Francisco"
▼ classes: Array(2)
  0: "JSD"
  1: "FEWD"
  length: 2
  ► __proto__: Array(0)
classroom: 8
▼ dates:
  end: 20171113
  start: 20170906
  ► __proto__: Object
firstName: "Sasha"
lastName: "Vodnik"
launched: true
```

direct property > nested object property

```
console.log(personObject.dates.start);
> 20170906
```

EXERCISE — NESTED DATA STRUCTURES



EXERCISE

KEY OBJECTIVE

- ▶ Implement and interface with JSON data

TYPE OF EXERCISE

- ▶ Groups of 2-3

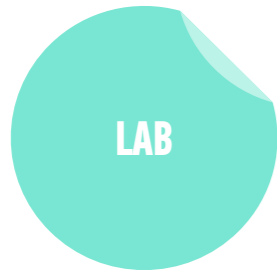
TIMING

3 min

1. Write JSON code containing a nested data structure (at least one property whose value is an array or an object).
2. Write JavaScript code to reference at least one value in the nested data structure.
3. Share your code on the Slack channel for today.

PRACTICE: JSON

LAB — JSON



KEY OBJECTIVE

- ▶ Implement and interface with JSON data

TYPE OF EXERCISE

- ▶ Individual or pair

TIMING

until 9:20

1. Open `starter-code > 3-json-exercise > app.js` in your editor.
2. Follow the instructions to write code that produces the stated output.

LEARNING OBJECTIVES – REVIEW

- Identify likely objects, attributes, and methods in real-world scenarios
- Create JavaScript objects using object literal notation
- Implement and interface with JSON data

NEXT CLASS PREVIEW

Intro to the DOM

- Identify differences between the DOM and HTML.
- Explain the methods and use the DOM in JavaScript.
- Use DOM manipulation to add elements to the browser window and modify existing elements.

Exit Tickets!

(Class #6)

Q&A