GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/JS-SF-12-resources` repo to your computer
2. Open the `06-objects-json > starter-code` folder in your code editor

JAVASCRIPT DEVELOPMENT

OBJECTS & JSON

LEARNING OBJECTIVES

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

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

AGENDA

- Objects review
- Lab: Translate real world scenarios into objects
- Lab: Create objects
- JSON
- Lab: Work with JSON

WEEKLY OVERVIEW

WEEK 4	Objects & JSON / Intro to DOM & jQuery
WEEK 5	Events & jQuery / Ajax & APIs
WEEK 6	Asynchronous JS & callbacks / Advanced APIs

EXIT TICKET QUESTIONS

1. Like: Catch phrase
2. Like: Coding in class

HOMEWORK — GROUP DISCUSSION

EXERCISE

TYPE OF EXERCISE

- Groups of 3

TIMING

6 min

1. Show off your bot! What can it do?
2. Share a challenge you encountered, and how you overcame it.
3. If you tried something that didn’t work, or wanted to add functionality but weren’t quite sure how, brainstorm with your group how you might approach it.

WARMUP EXERCISE

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

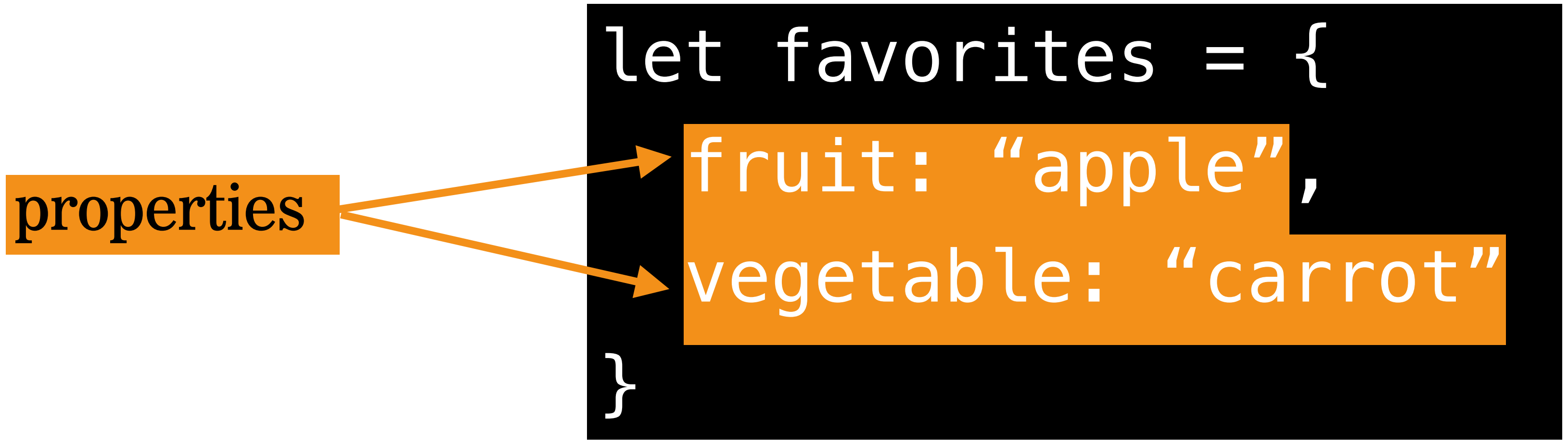
NUMBER

ARRAY

BOOLEAN

OBJECT

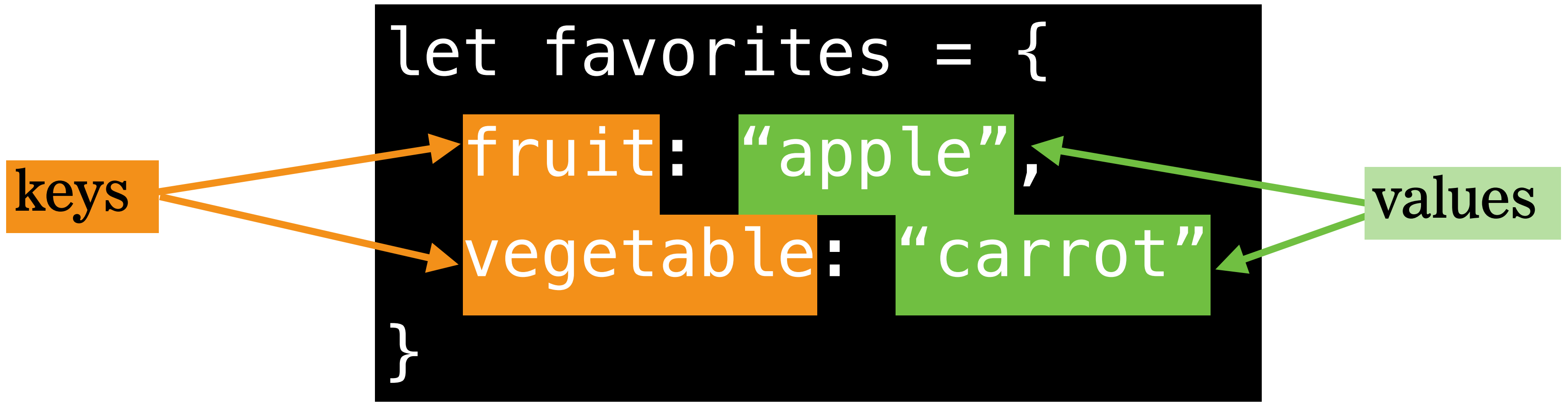
AN OBJECT IS A COLLECTION OF PROPERTIES



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

PROPERTY = KEY & VALUE

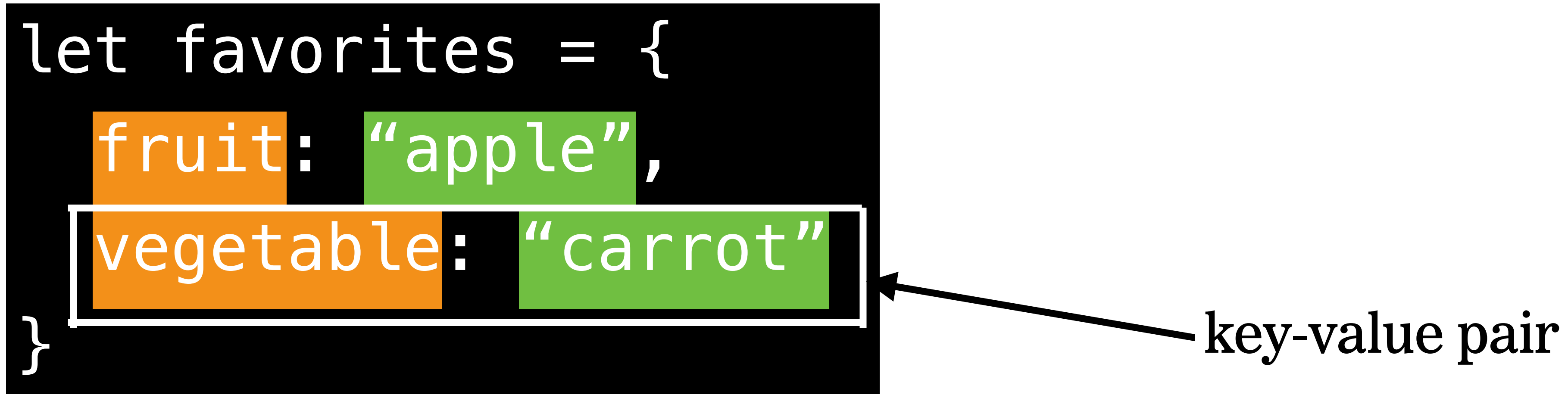
- A **property** is an association between a key and a value
  - **key**: name (often descriptive) used to reference the data
  - **value**: the data stored in that property
- A property is sometimes referred to as a **key-value pair**



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

KEY-VALUE PAIR

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



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

AN OBJECT IS NOT ORDERED

```
  [
0   “apple”,
1   “pear”,
2   “banana”
  ]
```

ARRAY
ordered

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

OBJECT
not ordered

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

TWO WAYS TO GET/SET PROPERTIES

dot notation

square bracket notation

GETTING A PROPERTY VALUE WITH DOT NOTATION

object

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

object name

getting properties

property name

```
favorites.fruit
> “apple”
favorites.veg
> “carrot”
```

object name

calling a method

method name

```
favorites.declare()
> “I like fruit and veg”
```

SETTING A PROPERTY VALUE WITH DOT NOTATION

object

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

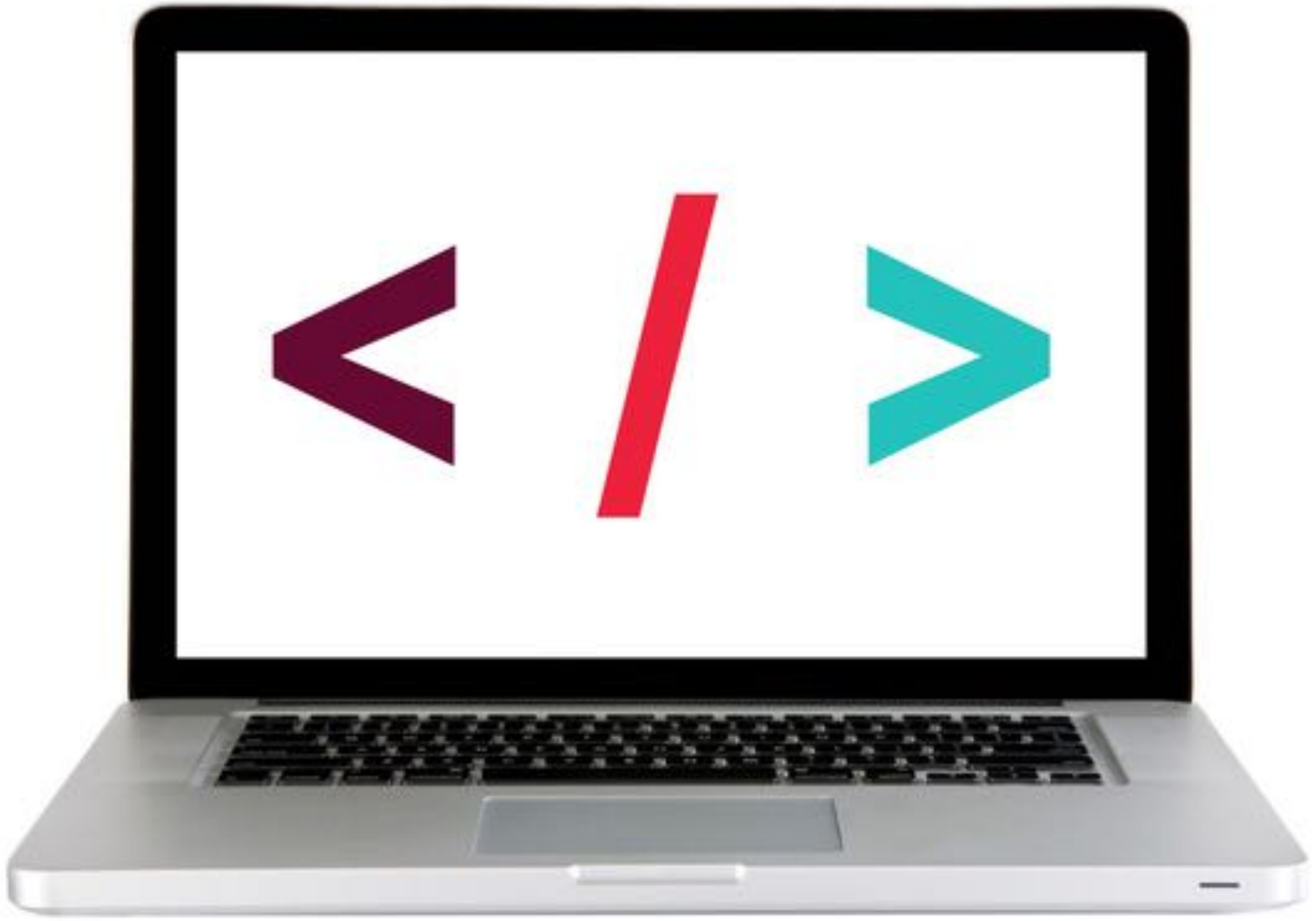
setting properties

```
favorites.fungus = ‘shiitake’;
favorites.pet = ‘hamster’;
```

setting a method

```
favorites.beAmbivalent = function() {
  console.log(“I like other things”);
};
```

LET'S TAKE A LOOK



GETTING A PROPERTY VALUE WITH SQUARE BRACKET NOTATION

object

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

object name

getting properties

```
favorites[fruit]
> “apple”
favorites[veg]
> “carrot”
```

property name

SETTING A PROPERTY VALUE WITH SQUARE BRACKET NOTATION

object

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

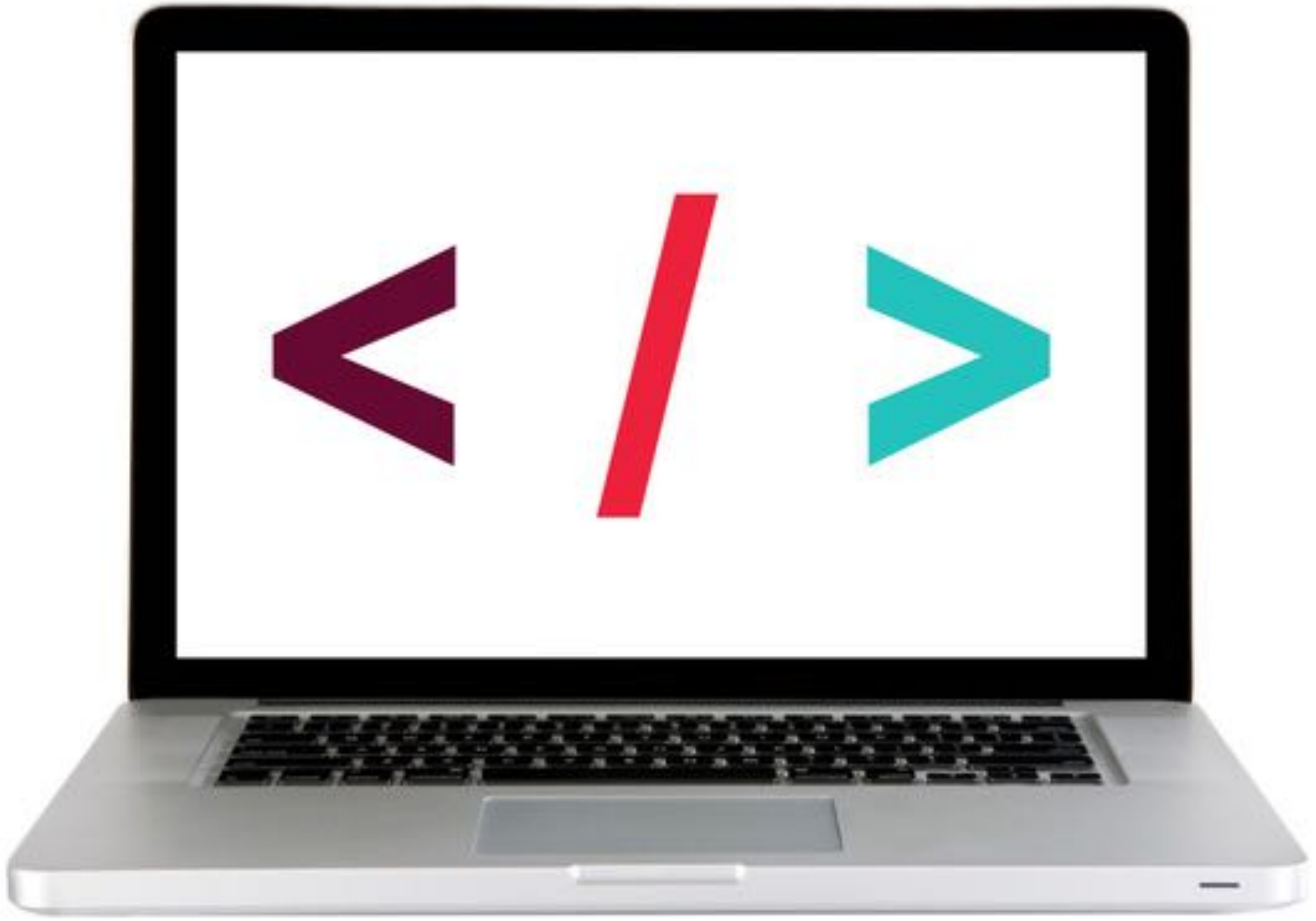
setting properties

```
favorites[fungus] = ‘shiitake’;
favorites[pet] = ‘hamster’;
```

setting a method

```
favorites[beAmbivalent] = function() {
  console.log(“I like other things”);
};
```

LET'S TAKE A LOOK



EXERCISE — OBJECTS

EXERCISE

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 SCENARIOS

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

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

5 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 your results to the class.

LAB — OBJECTS

LAB

KEY OBJECTIVE

- Create JavaScript objects using object literal notation

TYPE OF EXERCISE

- Individual or pair

TIMING

10 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

JSON IS A DATA FORMAT BASED ON JAVASCRIPT

object

```
let instructor = {
  firstName: ‘Sasha’,
  lastName: ‘Vodnik’,
  city: ‘San Francisco’,
  classes: [
    ‘JSD’, ‘FEWD’
  ],
  classroom: 7,
  launched: true,
  dates: {
    start: 20180205,
    end: 20180406
  },
};
```

JSON

```
{
  “firstName”: “Sasha”,
  “lastName”: “Vodnik”,
  “city”: “San Francisco”,
  “classes”: [
    “JSD”, “FEWD”
  ],
  “classroom”: 7,
  “launched”: true,
  “dates”: {
    “start”: 20180205,
    “end”: 20180406
  }
}
```

JSON

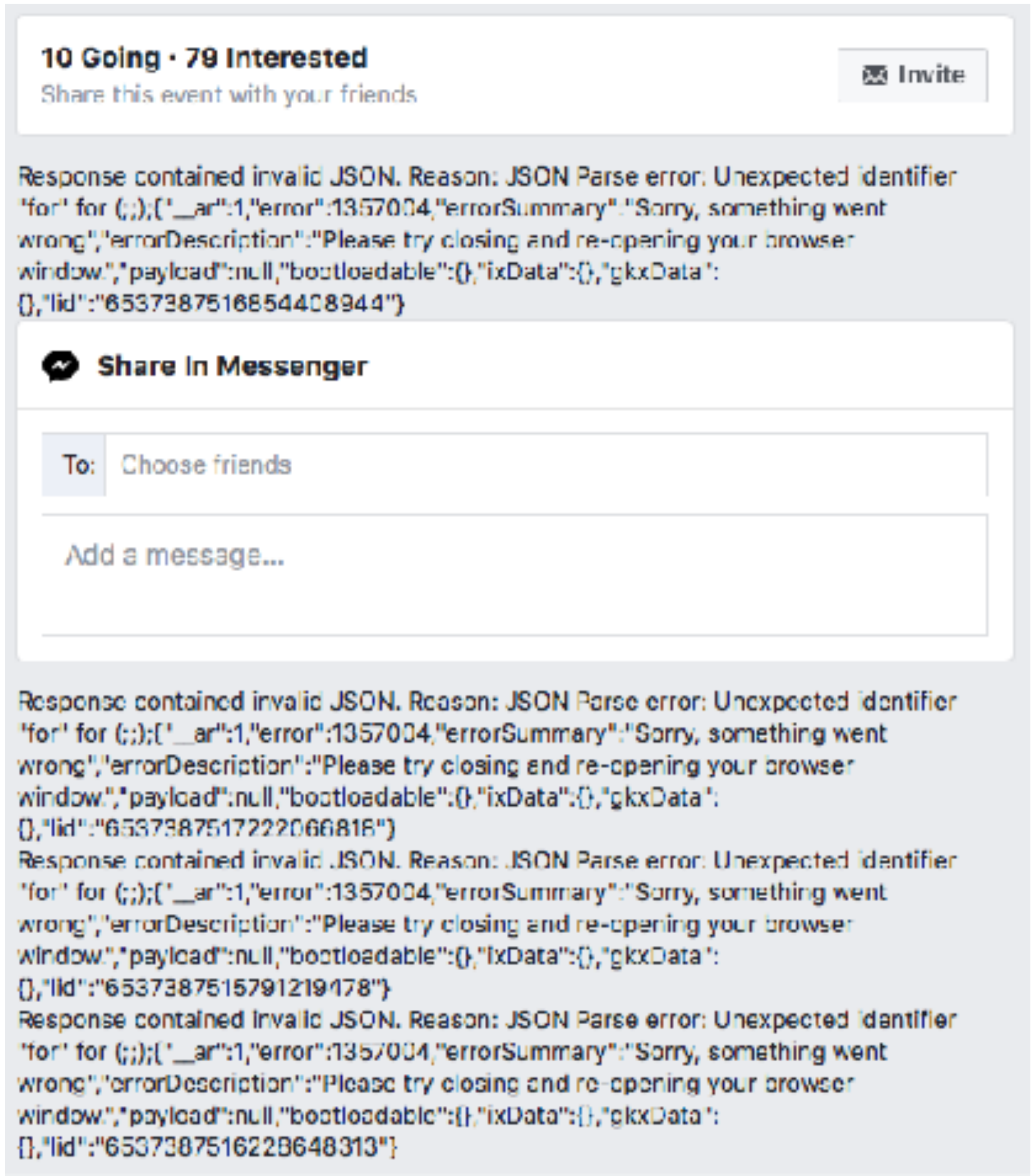
- Easy for humans to read and write
- Easy for programs to parse and generate

```
{
  “firstName”: “Sasha”,
  “lastName”: “Vodnik”,
  “city”: “San Francisco”,
  “classes”: [
    “JSD”, “FEWD”
  ],
  “classroom”: 7,
  “launched”: true,
  “dates”: {
    “start”: 20180205,
    “end”: 20180406
  }
}
```

JSON IS NOT JAVASCRIPT-SPECIFIC

- Used across the web by programs written in many languages

JSON IS EVERYWHERE!



```
Response contained invalid JSON. Reason: JSON Parse error: Unexpected identifier
"for" for (;;);{"__ar":1,"error":1357004,"errorSummary":"Sorry, something went
wrong","errorDescription":"Please try closing and re-opening your browser
window.","payload":null,"bootloadable":{},"ixData":{},"gkxData":
{},"lid":"6537387516854408944"}
```

```
Response contained invalid JSON. Reason: JSON Parse error: Unexpected identifier
"for" for (;;);{"__ar":1,"error":1357004,"errorSummary":"Sorry, something went
wrong","errorDescription":"Please try closing and re-opening your browser
window.","payload":null,"bootloadable":{},"ixData":{},"gkxData":
{},"lid":"6537387517222066816"}
Response contained invalid JSON. Reason: JSON Parse error: Unexpected identifier
"for" for (;;);{"__ar":1,"error":1357004,"errorSummary":"Sorry, something went
wrong","errorDescription":"Please try closing and re-opening your browser
window.","payload":null,"bootloadable":{},"ixData":{},"gkxData":
{},"lid":"6537387515791219478"}
Response contained invalid JSON. Reason: JSON Parse error: Unexpected identifier
"for" for (;;);{"__ar":1,"error":1357004,"errorSummary":"Sorry, something went
wrong","errorDescription":"Please try closing and re-opening your browser
window.","payload":null,"bootloadable":{},"ixData":{},"gkxData":
{},"lid":"6537387516228648313"}
```

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!

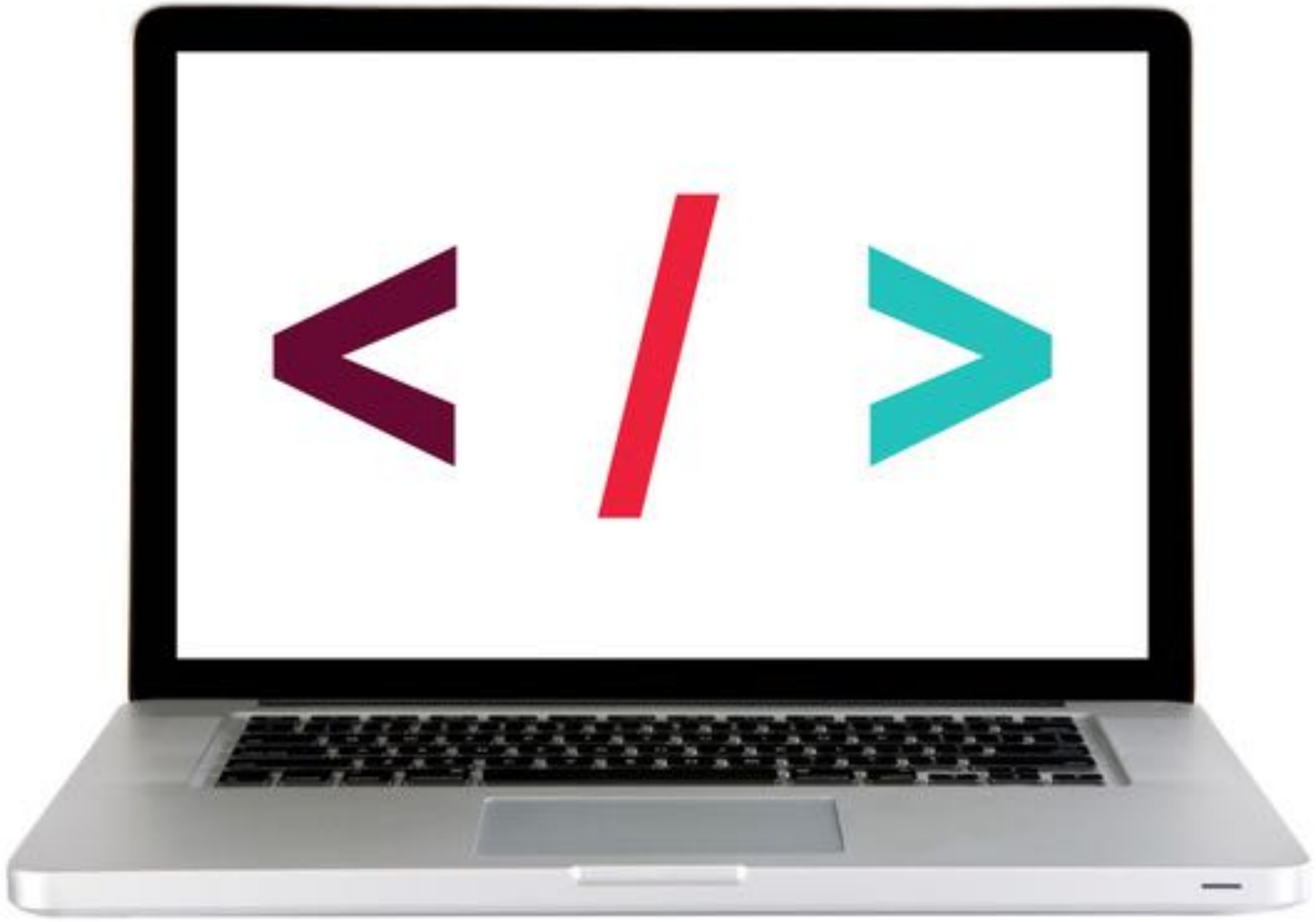
TO CONVERT AN OBJECT TO JSON

```
JSON.stringify(object);
```

TO CONVERT JSON TO AN OBJECT

```
JSON.parse(json);
```

LET'S TAKE A LOOK



EXERCISE — JSON

EXERCISE

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.

LAB — JSON

LAB

KEY OBJECTIVE

- Implement and interface with JSON data

TYPE OF EXERCISE

- Individual or pair

TIMING

10 min

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

WORKING WITH NESTED DATA STRUCTURES

YAY, I GOT SOME DATA!

```
let person = '{"firstName":
"Sasha","lastName": "Vodnik","city":
"San Francisco","classes": ["JSD",
"FEWD"],"classroom": 7,”launched":
true,"dates": {"start": 20180205,"end":
20180406}}’;
```

WAIT, WHAT?!

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!)

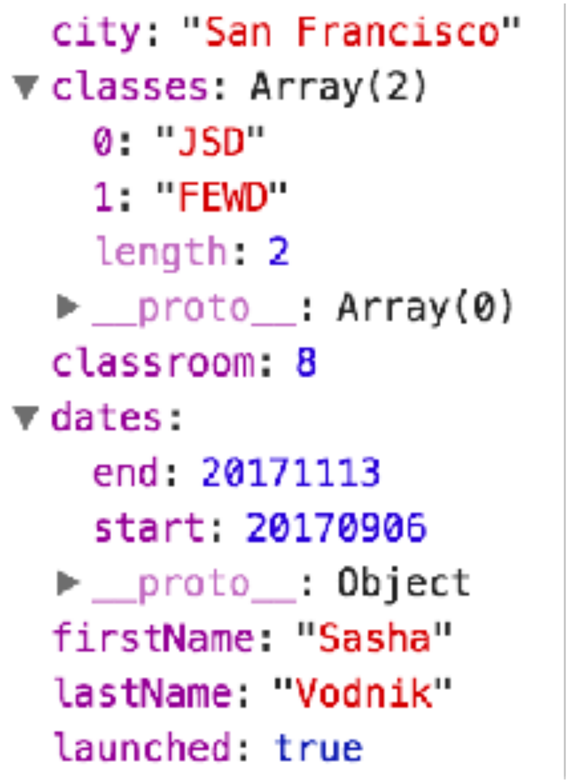
```
let person = '{"firstName":
"Sasha","lastName": "Vodnik","city":
"San Francisco","classes": ["JSD",
"FEWD"],"classroom": 7,”launched":
true,"dates": {"start": 20180205,"end":
20180406}}’;
```

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

WORKING WITH NESTED DATA STRUCTURES

2. VIEW THE RESULTING DATA STRUCTURE

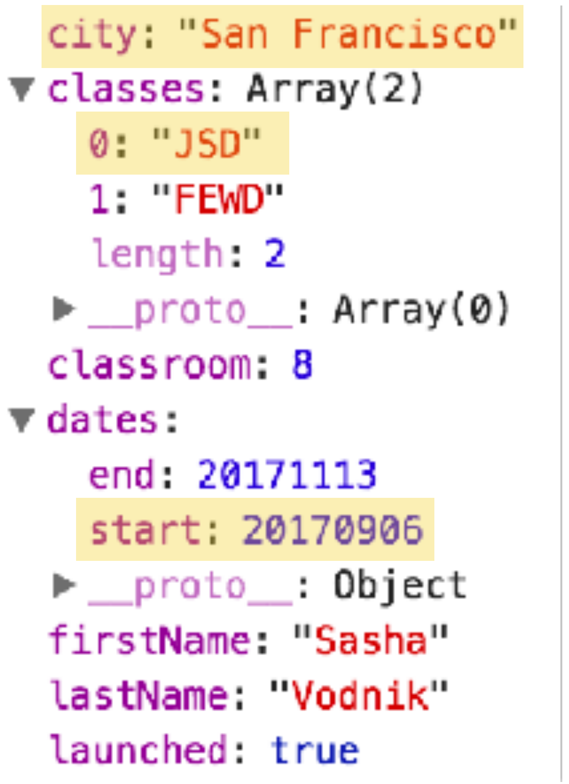
```
let personObject = JSON.parse(person);
console.log(personObject);
>
```



```
   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

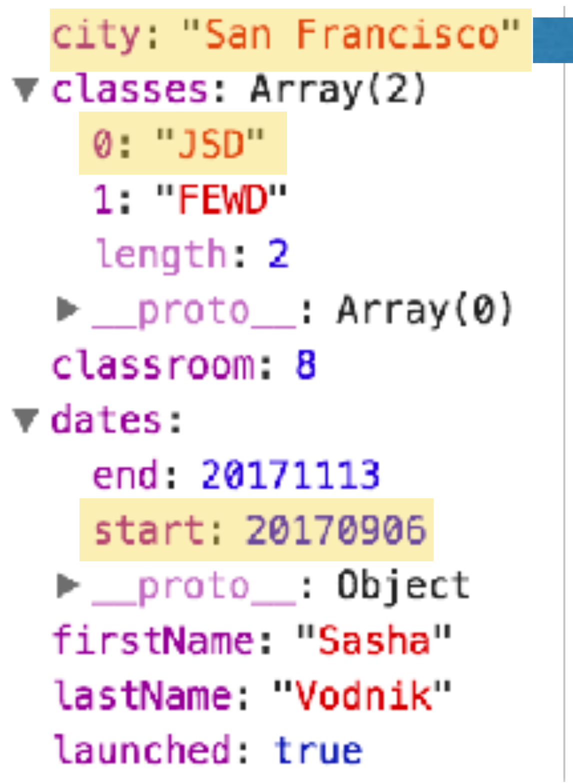
3. LOCATE THE DATA YOU WANT TO REFERENCE



```
 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



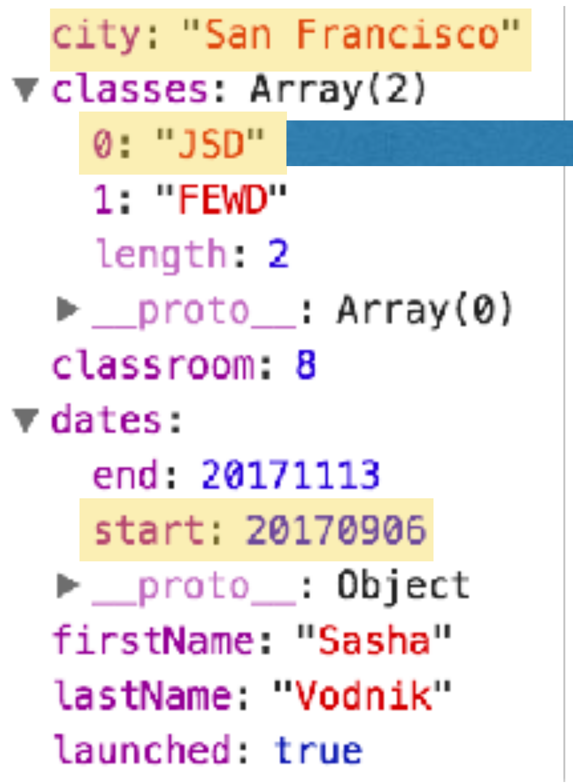
```
  city: "San Francisco"
▼ classes: Array(2)
    0: "JSD"
    1: "FEWD"
    length: 2
  ▶ __proto__: Array(0)
  classroom: 8
▼ dates:
    end: 20171113
    start: 20170906
  ▶ __proto__: Object
  firstName: "Sasha"
  lastName: "Vodnik"
  launched: true
```

direct property:

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

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
```

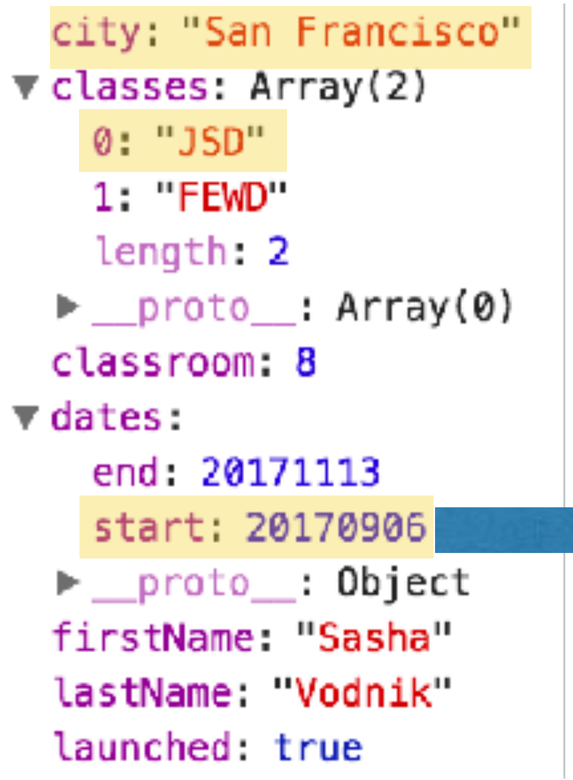
direct property > array element

```
console.log(personObject.classes);
> [“JSD”,”FEWD”]

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



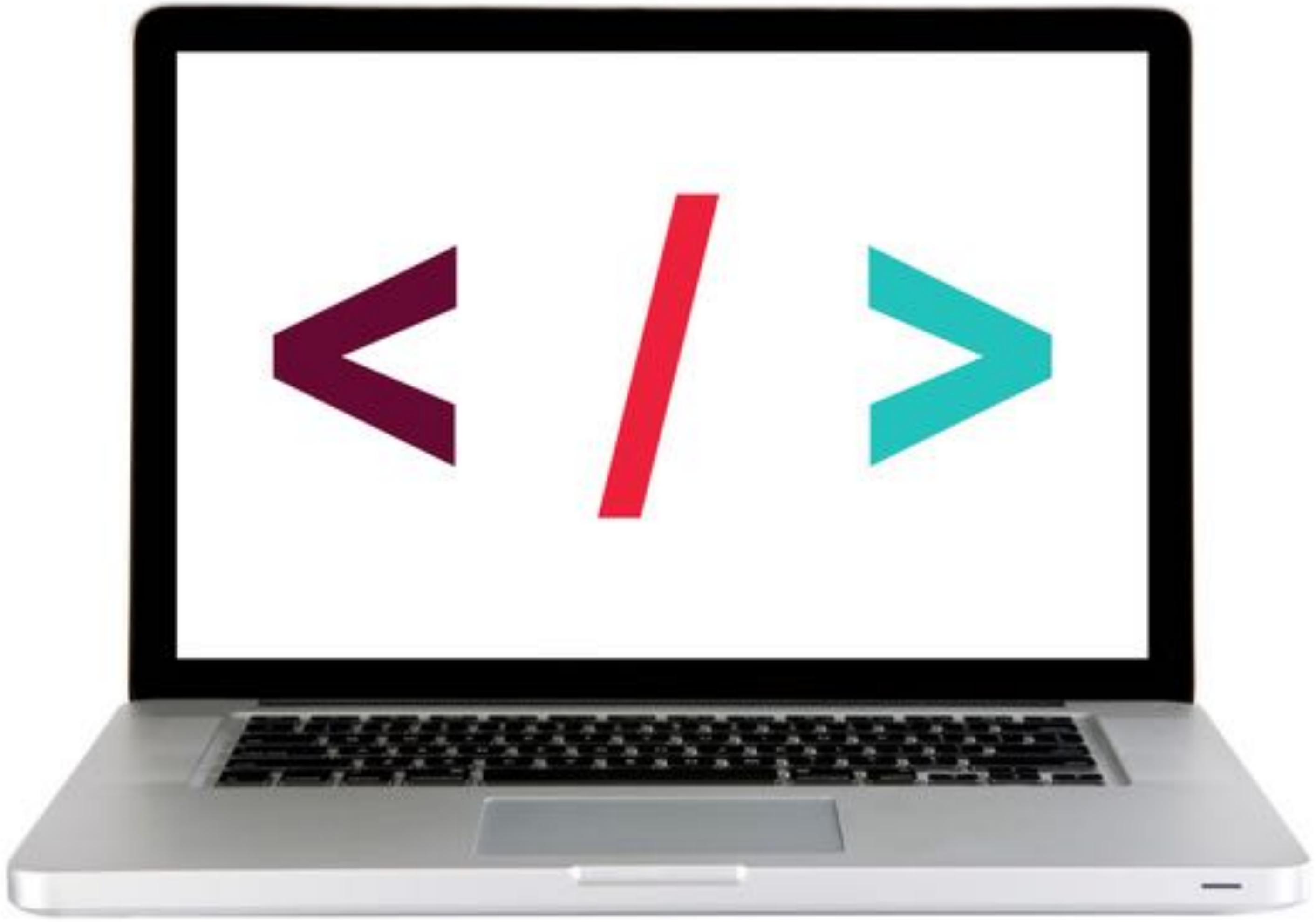
```
  city: "San Francisco"
▼ classes: Array(2)
    0: "JSD"
    1: "FEWD"
    length: 2
  ▶ __proto__: Array(0)
  classroom: 8
▼ dates:
    end: 20171113
    start: 20170906
  ▶ __proto__: Object
  firstName: "Sasha"
  lastName: "Vodnik"
  launched: true
```

direct property > nested object property

```
console.log(personObject.dates);
> {end:20171113,start:20170906}

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

LET'S TAKE A LOOK



LAB — JSON

LAB

KEY OBJECTIVE

- Implement and interface with JSON data

TYPE OF EXERCISE

- Individual or pair

TIMING

10 min

1. Open `starter-code > 4-data-structure—exercise > app.js` in your editor.
2. Follow the instructions to write code that produces the stated output.

Exit Tickets!

(Class #6)

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 & jQuery

- Describe the difference between the DOM and HTML.
- Select DOM elements and properties using jQuery.
- Manipulate the DOM by using jQuery selectors and functions.
- Create DOM event handlers using jQuery.

Q&A