Objects and Arrays

So far we've talked about having one variable at a time: one first name, one last name, one price, etc. What if we have a collection of data? It'd be nice to group together like data. Good news! You can!



1

const person = {

2

 name: "Brian Holt",

3

 city: "Seattle",

4

 state: "WA",

5

 favoriteFood: "🌮",

6

 wantsTacosRightNow: true,

7

 numberOfTacosWanted: 100

8

};

9

console.log(person);

10

console.log(person.name);

11

console.log(person["name"]); // same as the line above; definitely prefer using the other one



1

Object {

2

 "city": "Seattle",

3

 "favoriteFood": "🌮",

4

 "name": "Brian Holt",

5

 "numberOfTacosWanted": 100,

6

 "state": "WA",

7

 "wantsTacosRightNow": true,

8

}

9

"Brian Holt"

10

"Brian Holt"

11

undefined

This is called an object. They're extremely useful in JavaScript; they're how you'll group together like-information so that they can be used together. They contain a bunch of **keys** and **values**. The keys are on the left side of the : and represent how you get that piece data of out of the object. name is one such key, and the way I get the name of the

Used in conjunction with functions they're very powerful. Take this example:



1

const person1 = {

2

 name: "Brian",

3

 ageRange: "25-35"

4

};

5

const person2 = {

6

 name: "Jack",

7

 ageRange: "65-75"

8

};

9

​

10

function suggestMusic(person) {

11

 if (person.ageRange === "25-35") {

12

   console.log("We think you'll like Daft Punk you crazy millenial.");

13

} else if (person.ageRange === "65-75") {

14

   console.log(

15

     "You're obviously going to like Johnny Cash. He walks the line."

16

  );

17

} else {

18

   console.log(

19

     "Uh, maybe try David Bowie? Everyone likes David Bowie, right?"

20

  );

21

}

22

}

23

​

24

suggestMusic(person1);

25

suggestMusic(person2);



1

"We think you\'ll like Daft Punk you crazy millenial."

2

"You\'re obviously going to like Johnny Cash. He walks the line."

3

undefined

Now we're able to pass all this information as one package which makes it easy to keep track of since we're just passing one variable. You'll see this become even more useful as we start integrating with servers and APIs.

Objects can even have their functions! Let's see that.



1

const dog = {

2

 name: "dog",

3

 speak() {

4

   console.log("woof woof");

5

}

6

};

7

​

8

dog.speak();



1

"woof woof"

2

undefined

Objects can as well have nested objects inside of them.



1

const me = {

2

 name: {

3

   first: "Brian",

4

   last: "Holt"

5

},

6

 location: {

7

   city: "Seattle",

8

   state: "WA",

9

   country: "USA"

10

}

11

};

12

​

13

console.log(me);



1

Object {

2

 "location": Object {

3

   "city": "Seattle",

4

   "country": "USA",

5

   "state": "WA",

6

},

7

 "name": Object {

8

   "first": "Brian",

9

   "last": "Holt",

10

},

11

}

12

undefined

Context

Using the above object, wouldn't it be nice if we could use a function to nicely print where that person was from?



1

const me = {

2

 name: {

3

   first: "Brian",

4

   last: "Holt"

5

},

6

 location: {

7

   streetNumber: 500,

8

   street: "Fakestreet",

9

   city: "Seattle",

10

   state: "WA",

11

   zipCode: 55555,

12

   country: "USA"

13

},

14

 getAddress() {

15

   return `${this.name.first} ${this.name.last}

16

${this.location.streetNumber} ${this.location.street}

17

${this.location.city}, ${this.location.state} ${this.location.zipCode}

18

${this.location.country}`;

19

}

20

};

21

​

22

console.log(me.getAddress());



1

"Brian Holt

2

500 Fakestreet

3

Seattle, WA 55555

4

USA"

5

undefined

This is our first time seeing the weird this keyword. This is a strange, complicated, and difficult concept in JavaScript known as context and trips up all sorts of people, new and old to the language. If you decide to pursue a career as a developer, interviewers will often ask questions about context in JavaScript. It's worth investment to understand how it works.

In the simplest form, anywhere you are in JavaScript you have a context you are in. You can reference that context by using this. If I just reference this from the outtermost layer, it'll be the global object, which in the browser is something called window. window already has a bunch of stuff on it. For example:



1

console.log(this === window);

2

console.log(this.scrollY);

3

console.log(window.scrollY);



1

true

2

0

3

0

4

undefined

As you can see from the first line, you can see that in this context, window is the this at that time. However, in the example above when we're doing the address, the this is the object since when I **call the function**, it's created inside of an object. That object then becomes this when getAddress is called. As soon as the function completes, the context is destroyed and the context goes back to being what it was before, in this case window.

A good rule of thumb (that is unfortunately not always true) is that if you're inside an object of some sort, the this will be that object. If not, it'll be the global object, window. There are crazy exceptions to this and you can even manipulate it yourself. For now, operate with that definition. It's a deep-and-dark rabbit hole to go down so let's continue and you can take [Kyle Simpson's course](https://frontendmasters.com/courses/javascript-foundations/this/) later where he goes in depth on it.

Arrays

Objects are un-ordered collections of datas using keys and values. Arrays, in contrast, are **ordered collections of data**. If you put something in an array, it has an order. For example, you might a list of the days of the week.



1

const daysOfTheWeek = [

2

 "Monday",

3

 "Tuesday",

4

 "Wednesday",

5

 "Thursday",

6

 "Friday",

7

 "Saturday",

8

 "Sunday"

9

];

10

console.log(daysOfTheWeek);

11

console.log(daysOfTheWeek[0]);

12

console.log(daysOfTheWeek[1]);

13

console.log(daysOfTheWeek[6]);



1

Array [

2

 "Monday",

3

 "Tuesday",

4

 "Wednesday",

5

 "Thursday",

6

 "Friday",

7

 "Saturday",

8

 "Sunday",

9

]

10

"Monday"

11

"Tuesday"

12

"Sunday"

13

undefined

You first can see how we declare an array, using [ ]. Inside of an array, you can store anything you can store in a variable. You can have an array of numbers, an arry of strings, an array of objects, an array of arrays, an array of arrays of arrays, etc.

You can also see above how we access individual elements in an array: we use square brackets again and then we reference the number that we want to access. Again, remember, the numbering starts at 0. So the first element is index 0.

Arrays also have many methods (another word for functions that live on an object) and properties (another word for key/value pairs) that live on them. Let's see some of those:



1

const primeNumbers = [1, 2, 3, 5, 7, 11, 13, 17];

2

console.log(primeNumbers.length);

3

console.log(primeNumbers.join(" | "));



1

8

2

"1 | 2 | 3 | 5 | 7 | 11 | 13 | 17"

3

undefined

primeNumbers.length gives you back an number that is how long the array is. In this case there are eight elements in the array so it gives us back 8. primeNumbers.join(" | ")) takes your whole array and makes it into one string. THe " | " paramenter I'm passing is what I want put between each element, so you end up with the string "1 | 2 | 3 | 5 | 7 | 11 | 13 | 17".

So what if I want to add an element to the array after I've created. Use push!



1

const courses = [

2

{ teacher: "Kyle Simpson", course: "JS Function Lite" },

3

{ teacher: "Sarah Drasner", course: "Intro to Vue" },

4

{ teacher: "Brian Holt", course: "Complete Intro to React v3" },

5

{ teacher: "Steve Kinney", course: "State Management" }

6

];

7

​

8

courses.push({ teacher: "Sean Larkin", course: "Webpack" });

9

​

10

console.log(courses);

11

​

12

courses[2] = { teacher: "Brian Holt", course: "Complete Intro to React v4" };

13

​

14

console.log(courses);



1

Array [

2

 Object {

3

   "course": "JS Function Lite",

4

   "teacher": "Kyle Simpson",

5

},

6

 Object {

7

   "course": "Intro to Vue",

8

   "teacher": "Sarah Drasner",

9

},

10

 Object {

11

   "course": "Complete Intro to React v3",

12

   "teacher": "Brian Holt",

13

},

14

 Object {

15

   "course": "State Management",

16

   "teacher": "Steve Kinney",

17

},

18

 Object {

19

   "course": "Webpack",

20

   "teacher": "Sean Larkin",

21

},

22

]

23

Array [

24

 Object {

25

   "course": "JS Function Lite",

26

   "teacher": "Kyle Simpson",

27

},

28

 Object {

29

   "course": "Intro to Vue",

30

   "teacher": "Sarah Drasner",

31

},

32

 Object {

33

   "course": "Complete Intro to React v4",

34

   "teacher": "Brian Holt",

35

},

36

 Object {

37

   "course": "State Management",

38

   "teacher": "Steve Kinney",

39

},

40

 Object {

41

   "course": "Webpack",

42

   "teacher": "Sean Larkin",

43

},

44

]

45

undefined

The first thing we do is add an element to the end using the push function that arrays have. It "pushes" the element on the end.

Below that, we're overrinding index 2 with a new course. This will throw away what was there before and set it to be what we've set it to be.

Okay, now, given that, what if we wanted to console.log everything in the array? You already have all the tools to do that? Let's see to do it.



1

const cities = [

2

 "Seattle",

3

 "San Francisco",

4

 "Salt Lake City",

5

 "Amsterdam",

6

 "Hong Kong"

7

];

8

​

9

// method 1

10

for (let i = 0; i < cities.length; i++) {

11

 console.log(cities[i]);

12

}

13

​

14

// method 2

15

cities.forEach(function(city) {

16

 console.log(city);

17

});



1

"Seattle"

2

"San Francisco"

3

"Salt Lake City"

4

"Amsterdam"

5

"Hong Kong"

6

"Seattle"

7

"San Francisco"

8

"Salt Lake City"

9

"Amsterdam"

10

"Hong Kong"

11

undefined

The first way, using a for loop, we're using that i control variable which gets incremented each loop. We use that i to access each item in the array on each iteration of the loop. We have the loop stop when i gets equal to the length of cities. Very useful pattern. You'll see it a lot.

The second way is using a function that arrays have called forEach. This forEach method takes in a function and that function will be called once on each item of the array. It will pass that item into the function, which is what city is in this situation. Both are useful patterns to know. You'll use both frequently. While you're getting started, just use the one you feel comfortable with. They have different things that make them preferable in different situations but usually you can use either. Method 2 may be a bit more advance but I don't think you should be scared of it.