This is a review of what we covered in this tutorial on objects.

We have many types of values that we can store in JavaScript values, and sometimes we want to store a bunch of related values together: that's where ***objects*** come in!

An ***object*** is a data type that let us store a collection of properties in a single variable. To create an object, we declare a variable like we normally would, and then we use curly braces to surround key-value property pairs:

```
var objectName = {
  propertyName: propertyValue,
  propertyName: propertyValue,
  ...
};
```

Here's an object that describes Winston - this object has two properties, "hometown" and"hair", and each of the property values are strings:

```
var aboutWinston = {
  "hometown": "Mountain View, CA",
  "hair": "no"
};
```

Here's a more complex object that describes a cat, with four properties "age","furColor", "likes", and "birthday".

```
var lizzieTheCat = {
  "age": 18,
  "furColor": "grey",
  "likes": ["catnip", "milk"],
  "birthday": {"month": 7, "day": 17, year: 1994}
};
```

Notice how each property stores a different data type- "age" stores a number, "furColor" stores a string, "likes" stores an array, and "birthday" stores another object. That's the cool thing about objects (well, one of the cool things) - they can store other objects inside them! So you can have deeply nested objects to describe complex data.

But an object is not useful unless we can look inside it and grab the values of the different properties. We can do that two ways - first, using what we call "dot notation", where we write the name of the variable, followed by a ".", and then the property name:

```
var aboutWinston = {
  "hometown": "Mountain View, CA",
  "hair": "no"
};

text("Winston is from " + aboutWinston.hometown, 100, 100);
text("Winston has " + aboutWinston.hair + " hair", 100, 150);
```

We can also use "bracket notation", which looks similar to how we access array elements, where we write the variable name, then square brackets, with the property name inside in quotes:

```
var hisHometown = aboutWinston["hometown"];
var hisHair = aboutWinston["hair"];
```

If we try to access a property that doesn't exist, we'd see "undefined":

```
text("Winston's life goal is " + aboutWinston.lifeGoal, 100, 150);
```

Just like when we store other data types in variables, we can change the values of the object properties at any time during a program, using the dot or bracket notation:

```
aboutWinston.hair = "curly"; // Winston gets a wig!
```

We can also add entirely new properties!

```
aboutWinston.lifeGoal = "teach JavaScript";
```

If we're done with a property, we can delete it (but most of the time we'll probably just change it value):

```
delete aboutWinston.hair;
```

Now that you know both arrays and objects, you can combine them to make arrays of objects, which are actually really useful ways of storing data in programs. For example, an array of cats:

```
var myCats = [
  {name: "Lizzie",
   age: 18},
  {name: "Daemon",
   age: 1}
];

for (var i = 0; i < myCats.length; i++) {
  var myCat = myCats[i];
  println(myCat.name + ' is ' + myCat.age + ' years old.');
}
```

Notice that we iterate through an array of objects the same way that we iterate through an array of numbers or strings, using a for loop. Inside each iteration, we access the current array element with bracket notation, and then access the properties of that array element (an object) with dot notation.

Here's another practical example that you might use in your programs, an array of coordinate positions:

```
var positions = [
  {"x": 200, "y": 100},
  {"x": 200, "y": 200},
  {"x": 200, "y": 300}
];

for (var i = 0; i < positions.length; i++) {
  var position = positions[i];
  ellipse(position.x, position.y, 100, 100);
}
```

Pretty neat, aye? Objects can be confusing at first, but keep using them, and eventually you'll be addicted and turn everything into an object!