"For...In" Statements

I've shown you some ways that you can iterate over the elements in an array, but you can also iterate over the elements inside of an object, using the For...In statement.

Iterating Over Object Properties

If you want to iterate over all of the properties of an object using the methods we've shown so far, probably the first thing you would need is an array of keys showing each property in the object. And then you would use a `for` loop to iterate through it.

```
var toy = {"color":"red", "size":"large", "shape":"round"};
var toyProperties = Object.keys(toy); // get an array of keys
var counter;
var style;

console.log(toyProperties);
// ["color", "size", "shape"]

for (counter = 0; counter < toyProperties.length; counter++) {
  style = toyProperties[counter];
  console.log("The " + style + " is " + toy[style]);
  // "The color is red" (first time)
  // "The size is large" (second time)
  // "The shape is round" (third time)
}
```

"For...In" Syntax for Object Properties

JavaScript provides a For...In syntax specifically for iterating through the elements of an object. The For...In iterator loops through each property in an object, assigning the property name to a given variable.

```
var toy = {"color":"red", "size":"large", "shape":"round"};
var style;

for (style in toy) // iteration for each property
  { // block of code
    console.log("The " + style + " is " + toy[style]);
    // "The color is red" (first time)
    // "The size is large" (second time)
    // "The shape is round" (third time)
  }
```

This is a much cleaner and simpler way to iterate through all of the items in an object.

Don't Use Dot Syntax in Loops

One important thing to note is that you must not use the dot syntax when you're iterating through objects in a loop like this.

```
var toy = {"color":"red", "size":"large", "shape":"round"};
var style;

for (style in toy) {
  console.log("The " + style + " is " + toy.style);
  // "The color is undefined" (first time)
  // "The size is undefined" (second time)
  // "The shape is undefined" (third time)
}
```

The dot syntax will recognize the variable that you're passing in as if it were a property on the object. And it will look for a named property based on the name of that variable.

So when looping through an object, you cannot use the Dot Syntax to access the sub-elements.

Order is Not Guaranteed

Another interesting gotcha about objects is that the order is not guaranteed. Unlike an array, the properties of an object are not stored in any particular order, so they can't be accessed in any particular order. You can't rely on them being in the same order that you define them. So when you're writing your code to iterate through the elements of an object, you should be prepared for the results to come back in any order. Make sure you write your code to keep that in mind.

```
var toy = {"color":"red", "size":"large", "shape":"round"};
var style;

for (style in toy) {
  console.log("The " + style + " is " + toy.style);
}
// "The color is red"
// "The size is large"
// "The shape is round"
// or maybe
// "The size is large"
// "The color is red"
// "The shape is round"
// or...
```

You may always see them coming back in the same order in your local environment, but that doesn't necessarily mean every environment where JavaScript is run will return them in the same order, so be prepared for that.