EXAMPLE 50: THE `Object` OBJECT AS THE ULTIMATE PROTOTYPE

EXAMPLE 22: ITERATING OVER ALL THE PROPERTIES IN AN OBJECT

RECAP

EXAMPLE 22: ITERATING OVER ALL THE PROPERTIES IN AN OBJECT

SAY YOU HAVE AN OBJECT, AND ARE NOT QUITE SURE WHAT PROPERTIES IT HAS - YOU CAN SIMPLY ITERATE OVER ALL OF ITS PROPERTIES.

RECAP

EXAMPLE 22: ITERATING OVER ALL THE PROPERTIES IN AN OBJECT

```
for (var propertyName in rectangle){
    var propertyValue = rectangle[propertyName];
    console.log("Property name = " + propertyName
    console.log("Property value = " + propertyVal
}
```

THIS IS A SPECIAL KIND OF FOR-LOOP!

RECAP

EXAMPLE 22: ITERATING OVER ALL THE PROPERTIES IN AN OBJECT

```
for (var propertyName in rectangle){
    var propertyValue = rectangle[propertyName];
    console.log("Property name = " + propertyName
    console.log("Property value = " + propertyVal
}
```

THIS IS A SPECIAL KIND OF FOR-LOOP!

ALL OBJECTS CAN BE 'ITERATED OVER' IN THIS WAY

RECAP

EXAMPLE 22: ITERATING OVER ALL THE PROPERTIES IN AN OBJECT

```
for (var propertyName in rectangle){
    var propertyValue = rectangle[propertyName];
    console.log("Property name = " + propertyName
    console.log("Property value = " + propertyVal
}
```

ONCE WE HAVE THE NAME OF THE PROPERTY, GETTING ITS VALUE IS NOT HARD!

THIS IS AN EXCELLENT EXAMPLE OF THE UTILITY OF THE [] WAY OF ACCESSING PROPERTIES - HERE THE . WAY WOULD NOT HAVE

RECAP

EXAMPLE 22: ITERATING OVER ALL THE PROPERTIES IN AN OBJECT

```
for (var propertyName in rectangle){
    var propertyValue = rectangle[propertyName];
    console.log("Property name = " + propertyName
    console.log("Property value = " + propertyVal
}
```

THIS WILL GIVE US THE VALUES OF ALL NON-FUNCTION PROPERTIES. FOR FUNCTION PROPERTIES, IT WILL GIVE US THE UNDERLYING OBJECT.

RECAP

EXAMPLE 22: ITERATING OVER ALL THE PROPERTIES IN AN OBJECT

WE SAID "ALL OBJECTS CAN BE 'ITERATED OVER' IN THIS WAY"

IF YOU ARE WONDERING HOW THAT IS - WELL ITS BECAUSE ALL OBJECTS HAVE A COMMON BASE OBJECT THAT SUPPORTS THIS BEHAVIOUR. MORE LATER :-)

RECAP

EXAMPLE 50: THE `Object` OBJECT AS THE ULTIMATE PROTOTYPE

JAVASCRIPT HAS A SPECIAL BUILT-IN OBJECT, CALLED `Object`

`Object` IS THE PROTOTYPE OF ALL OBJECTS

JAVASCRIPT HAS A SPECIAL BUILT-IN OBJECT, CALLED `Object`

`Object` IS THE PROTOTYPE OF ALL OBJECTS

`Object` HAS A BUNCH OF BUILT-IN PROPERTIES THAT WE USE ALL THE TIME

AND THAT IS WHY WE ARE ABLE TO ITERATE OVER THE PROPERTIES OF ALL OBJECTS :-)

`Object` HAS A BUNCH OF BUILT-IN PROPERTIES THAT WE USE ALL THE TIME

`constructor` IS A PROPERTY THAT WILL RETURN THE CONSTRUCTOR OF A PARTICULAR PROTOTYPE

`Object` HAS A BUNCH OF BUILT-IN PROPERTIES THAT WE USE ALL THE TIME

`toString` IS A PROPERTY THAT RETURNS A NICE, READABLE VERSION OF AN OBJECT

`Object` HAS A BUNCH OF BUILT-IN PROPERTIES THAT WE USE ALL THE TIME

`propertyIsEnumerable` IS A PROPERTY THAT ALLOWS US TO ITERATE OVER ALL PROPERTIES OF AN OBJECT

`Object` HAS A BUNCH OF BUILT-IN PROPERTIES THAT WE USE ALL THE TIME

`hasOwnProperty` IS A PROPERTY TELLS WHETHER A PARTICULAR PROPERTY IS INSIDE AN OBJECT, OR INHERITED VIA A PROTOTYPE

`Object` HAS A BUNCH OF BUILT-IN PROPERTIES THAT WE USE ALL THE TIME

`isPrototypeOf` IS A PROPERTY TELLS WHETHER A PARTICULAR OBJECT IS PROTOTYPE OF ANOTHER

`Object` HAS A BUNCH OF BUILT-IN PROPERTIES THAT WE USE ALL THE TIME

`valueOf` IS A PROPERTY RETURNS A SPECIFIC VALUE OF AN OBJECT (BY DEFAULT THE OBJECT ITSELF, BUT CAN BE OVERRIDDEN)