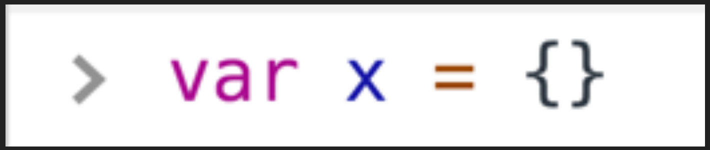
THE INTRODUCTION TO

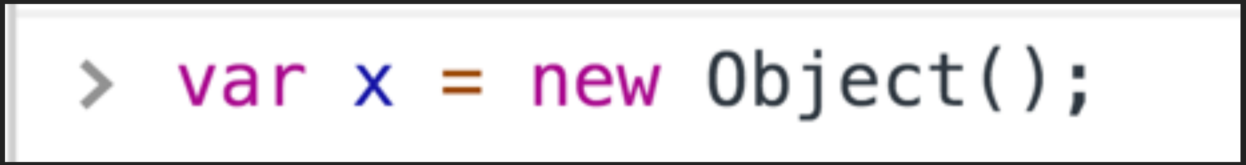
PROTOTYPES AND INHERITANCE

- JavaScript is a prototype-based language, and every object in JavaScript has a internal property called [[Prototype]] that can be use to extend object properties and methods.
- When an object can be cloned and extended - This is know as prototypical inheritance.

- So what does this mean…



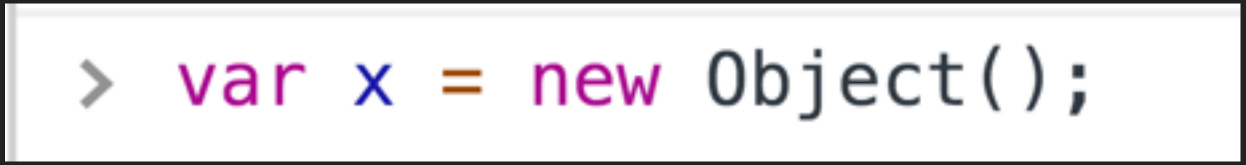
```
> var x = {}
```



```
> var x = new Object();
```

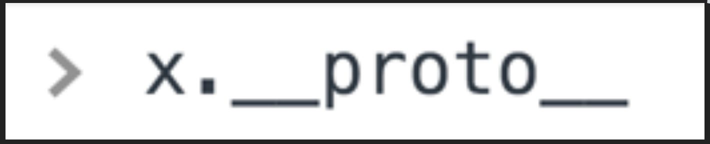
- Both of these accomplish the same thing - they create an object. The example on the right is using the object constructor to create the new object. This creates a new instance of the Object datatype and allows x to inherit all the properties and methods of this data type.

- Lets use the chrome browsers developer tools to inspect this further…
- Open the developer tools and select the console tab.
- First: we want to create a new object.

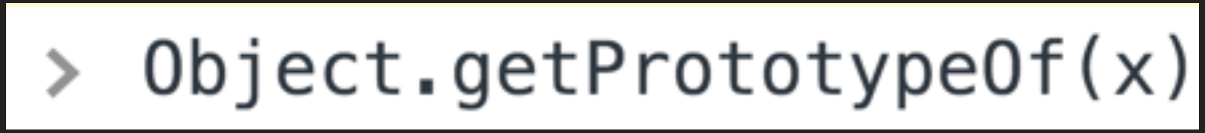


```
> var x = new Object();
```

- Then we want to inspect this object's prototype to see what methods and properties that this object inherited. We can do this by using the __proto__ or getPrototypeOf()

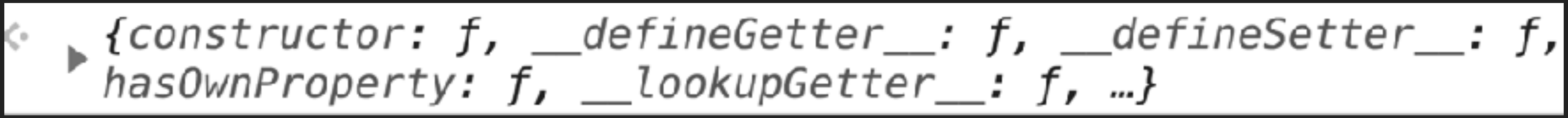


```
> x.__proto__
```



```
> Object.getPrototypeOf(x)
```

- When we execute the second statement. We should see a collapsed list of the prototype of our x object.

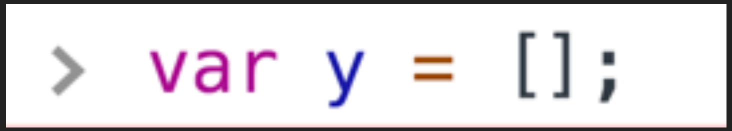


```
{constructor: f, __defineGetter__: f, __defineSetter__: f,
hasOwnProperty: f, __lookupGetter__: f, …}
```

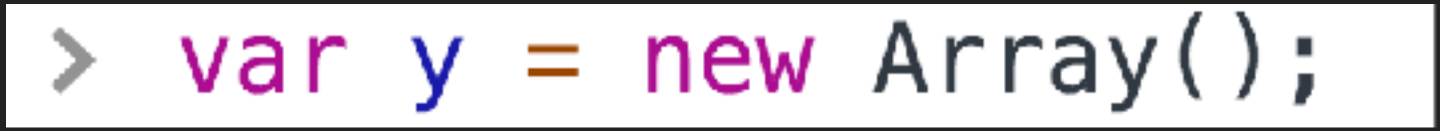
- Expand this list and you should start seeing properties and methods that we've used before.

- If we look at another data type - Arrays for example. We know when we create an array. We can use Array methods such as pop() and push(). The reason you have access to these methods is because any array you create has access to the properties and methods on the Array.prototype

- Lets test this
  First we need to create an array. We can do it both ways.

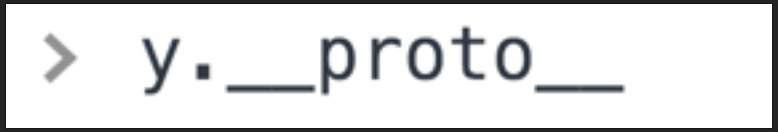


```
> var y = [];
```



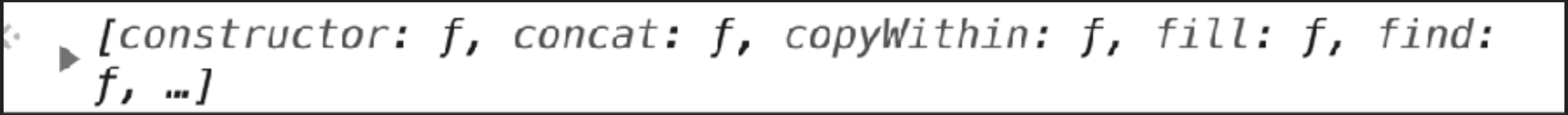
```
> var y = new Array();
```

- Now we would access the prototype of the newly created array in the same manner as before. We would call the name of the array and access its __proto__



```
> y.__proto__
```

- If you execute that statement you'll see the collapsed



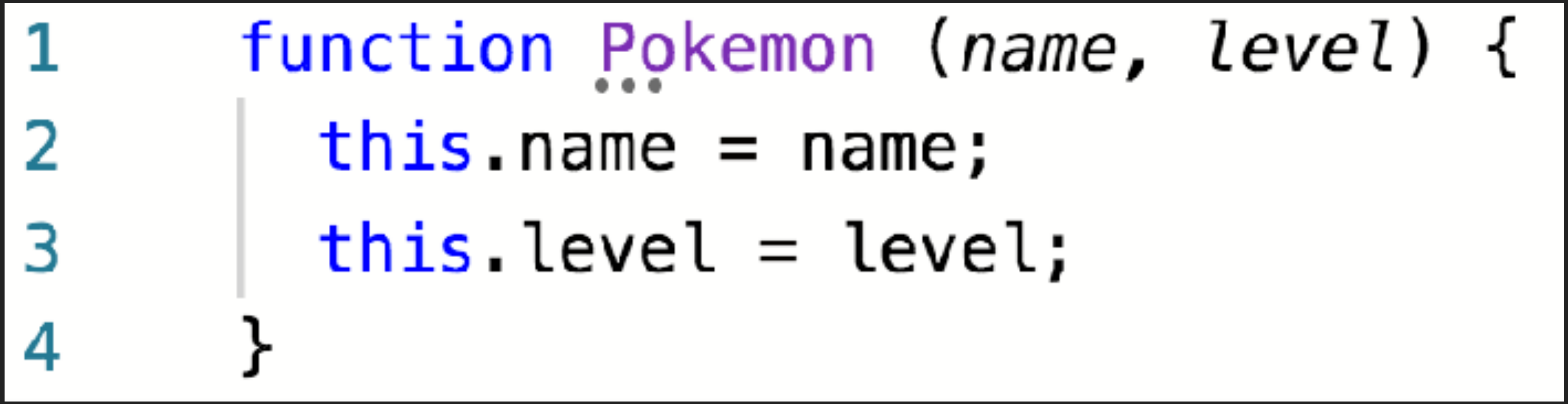
```
< ▶ [constructor: f, concat: f, copyWithin: f, fill: f, find:
    f, …]
```

- When you expand the prototype you'll see all the array methods we have access to – push & pop & shit & unshift ect.

- Constructor Functions: these are functions that are used to construct objects. To accomplish this we use the new operator. Just like we used when we created a new Array() or new Date().

- But what if we want to create our own? What if we wanted to create a pokemon game where we could call a constructor function to create new pokemon?

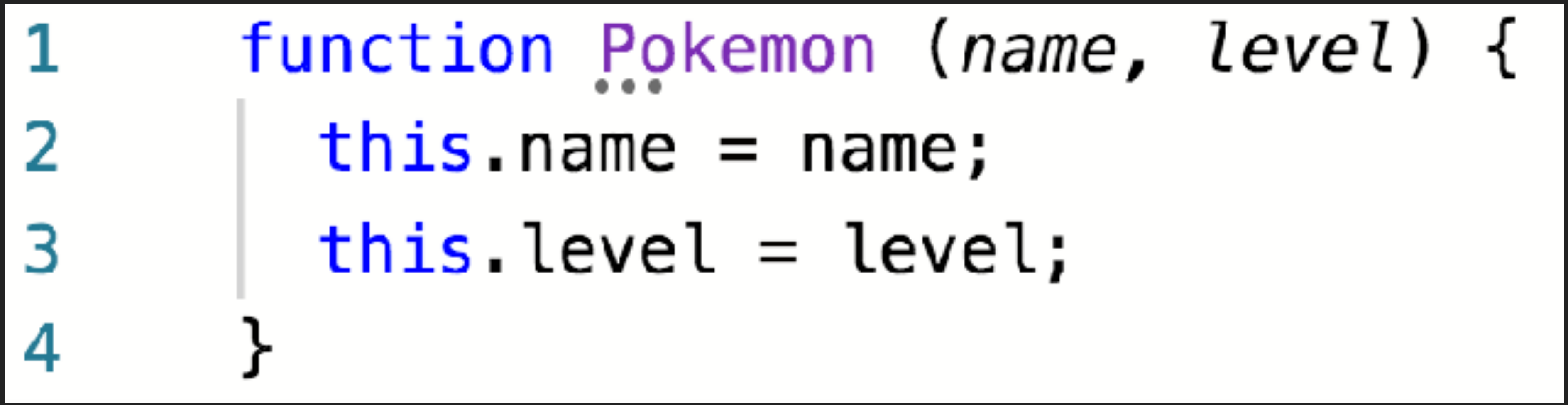
- We can!
- A constructor function is just like a regular function.



```
function Pokemon (name, level) {
  this.name = name;
  this.level = level;
}
```

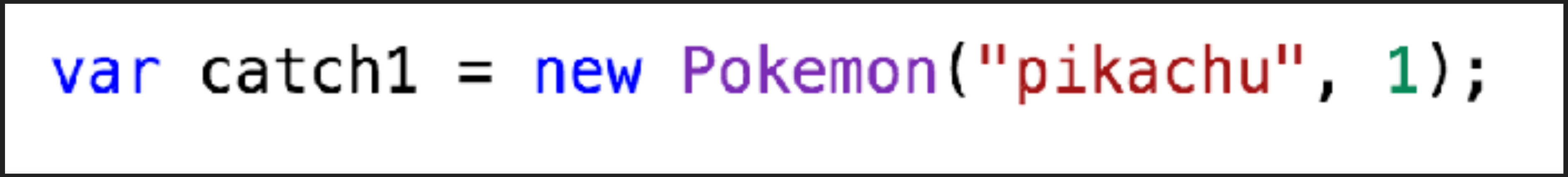
- In JavaScript we capitalize the name of the function name to signify that this function is a constructor function.

- The this keyword will refer to the new instance that is created, so setting this.name to the name parameter ensures the new object will have a name property set.



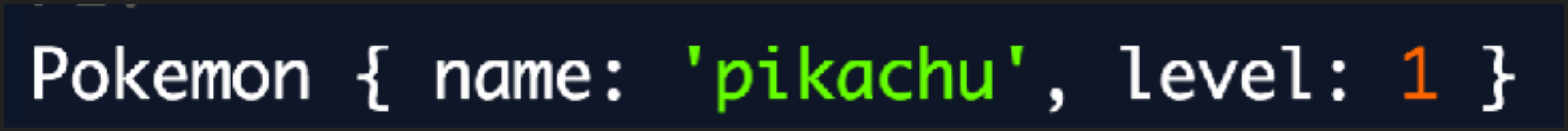
```
function Pokemon (name, level) {
  this.name = name;
  this.level = level;
}
```

- Now lets create a new instance with using the new keyword.



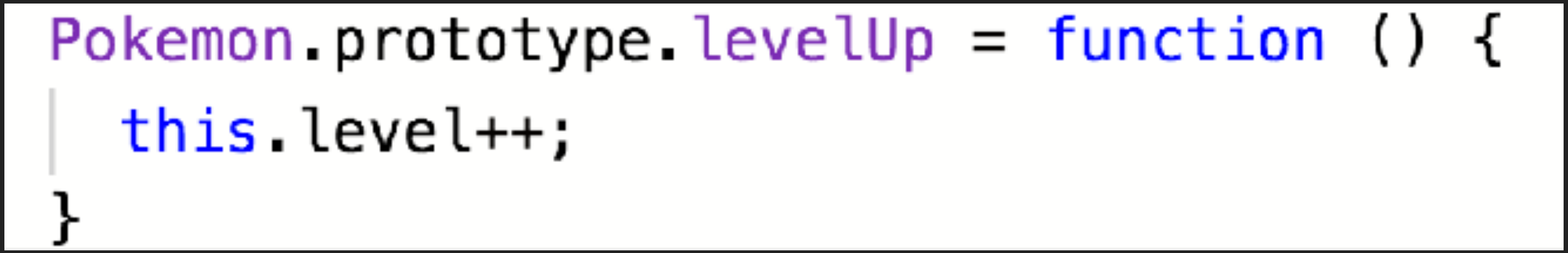
```
var catch1 = new Pokemon("pikachu", 1);
```

- Now lets inspect this catch1 variable to see what is assigned.



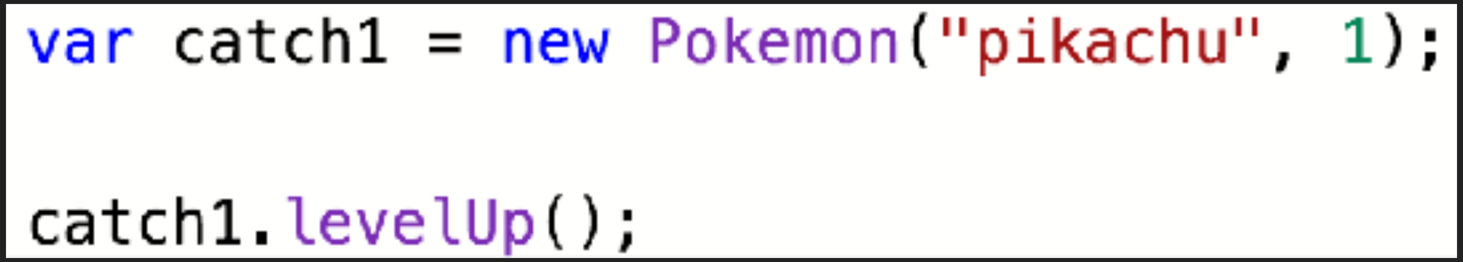
```
Pokemon { name: 'pikachu', level: 1 }
```

- Now that we can create new Pokemon we also want the ability to add methods. We can do this by assigning them right on the prototype.



```
Pokemon.prototype.levelUp = function () {
  this.level++;
}
```

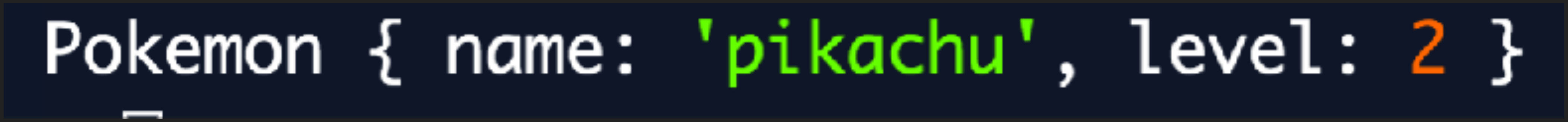
- So now that we created a method on the Pokemon prototype. After we created out new catch1 variable. We can now call that method on the catch1 variable to make the pokemon level up.



```
var catch1 = new Pokemon("pikachu", 1);

catch1.levelUp();
```

- Now when we inspect catch1 - we should see.



```
Pokemon { name: 'pikachu', level: 2 }
```

- Sources:
  MDN
  Digital Ocean
  Youtube