

JavaScript: What's different

functions

```
var x = function() {}; // functions can be values
```

```
_.isObject(x) // functions are object
```

```
x.someProp = val; // possible to set properties on functions
```

```
(new x()) instanceof x) // functions may be class-like (with "new")
```

```
x() // also just plain functions, like you'd expect
```

```
[] instanceof Object
```

Chrome Inspector

The screenshot displays the Chrome DevTools interface with the Source tab active. The main pane shows the source code of a JavaScript file named 'object-3.js'. The code includes several functions: 'start', 'askForName', 'updateName', and 'showError'. Line 15 is highlighted, corresponding to a breakpoint set at 'object-3.js:15'. The right sidebar shows the Breakpoints panel with three active breakpoints: 'object-3.js:8', 'object-3.js:15', and 'object-3.js:37'. The bottom status bar indicates the current cursor position at 'Line 20, Column 16'.

```
6 start: function() {
7   var name = this.askForName();
8   $('.js-name-form').show()
9   this.updateName(name);
10  return this;
11 },
12
13 // Ask for the user's name, setting it to "this.name"
14 askForName: function() {
15  var name = prompt('What is your name');
16  if (!name) return this.askForName();
17  return name;
18 },
19
20 // Set the user's name in the h1 tag.
21 updateName: function(name) {
22  if (!name) return this.showError('A name must be provided');
23  this.name = name;
24  this.setDisplayName();
25  return this;
26 },
27
28 // Show the error
29 showError: function(msg) {
30  $('.js-field-container').addClass('has-error');
31  $('.js-error-container').html('<p class="help-block">' + msg + '</p>');
32  return this;
33 }
34 }
```

Line 20, Column 16

Console Search Emulation Rendering

<top frame>

Prototype-Based Language

prototypes

- Objects which specify the "default values" for an object
- `Object.prototype`, `object.__proto__`
- Powerful / flexible object-oriented programming paradigm
- Optionally allows for inheritance via the prototype chain

jQuery

`$.fn = $.prototype`

"new"

- The "new" operator creates a new *instance* of an object
- When a new instance of an object is created with the "new" operator, "this" is new value of the instance: the prototype chain plus any values set as "this" in the constructor

"this"

in JavaScript the value for "this" is variable, depending on how a function is called

this

- One of the most confusing concepts to those new to JavaScript
- dynamic "hidden" extra value in every function call
- jQuery makes it even more confusing
- Simple rule of thumb "left of the dot"

```
var x = {  
  name: 'Bob',  
  sayName: function() {  
    // this === x (true)  
    alert('Hello ' + this.name);  
  }  
};  
  
x.sayName();
```

```
var x = {  
  name: 'Bob',  
  sayName: function() {  
    alert('Hello ' + this.name);  
  }  
};
```

```
var y = x.sayName;
```

```
y() // "this" is now global
```

jQuery - \$(this)

```
fn.bind(context)
```

```
fn.call(context, [arg1], [arg2]...)
```



```
fn.apply(context, [array / args])
```

```
fn.apply(context, arguments)
```

arguments

- Available in all functions
- Has a “length” property,
- Cannot be used as an array (use `_.toArray(arguments)`)
- Can be used with “_”