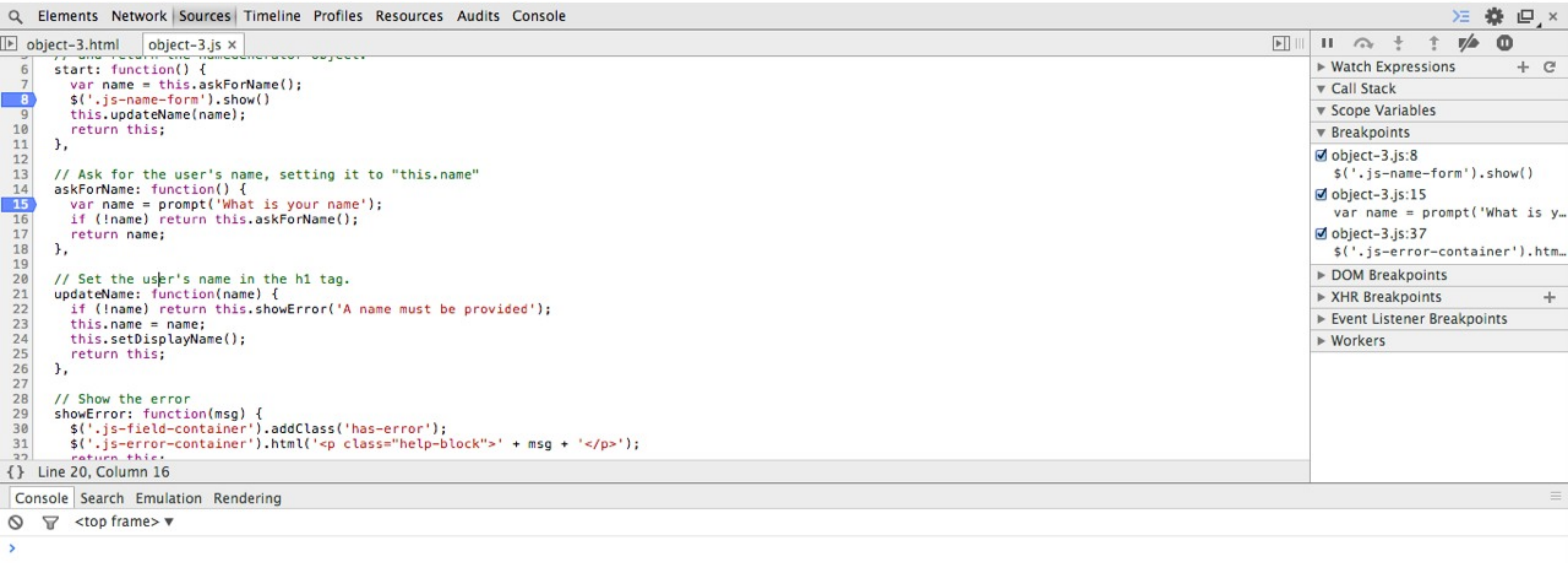
JavaScript: What's different

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



```
  start: function() {
    var name = this.askForName();
    $('.js-name-form').show()
    this.updateName(name);
    return this;
  },

  // Ask for the user's name, setting it to "this.name"
  askForName: function() {
    var name = prompt('What is your name');
    if (!name) return this.askForName();
    return name;
  },

  // Set the user's name in the h1 tag.
  updateName: function(name) {
    if (!name) return this.showError('A name must be provided');
    this.name = name;
    this.setDisplayName();
    return this;
  },

  // Show the error
  showError: function(msg) {
    $('.js-field-container').addClass('has-error');
    $('.js-error-container').html('<p class="help-block">' + msg + '</p>');
    return this;
```

Line 20, Column 16

☑ object-3.js:8
$('.js-name-form').show()
☑ object-3.js:15
var name = prompt('What is y…
☑ object-3.js:37
$('.js-error-container').htm…

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 “_”