JavaScript Objects / Events

Devpoint Labs - Jake Sorce / Dave Jungst

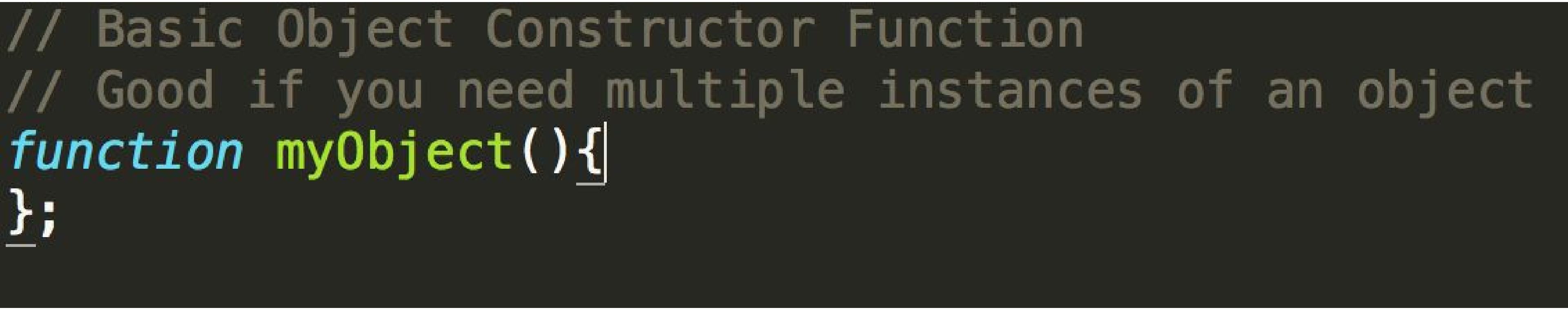
What is a Javascript Object?

An object is a collection of properties, and a property is an association between a name (or key) and a value. A property's value can be a function, in which case the property is known as a method. In addition to objects that are predefined in the browser, you can define your own objects.

Javascript Objects

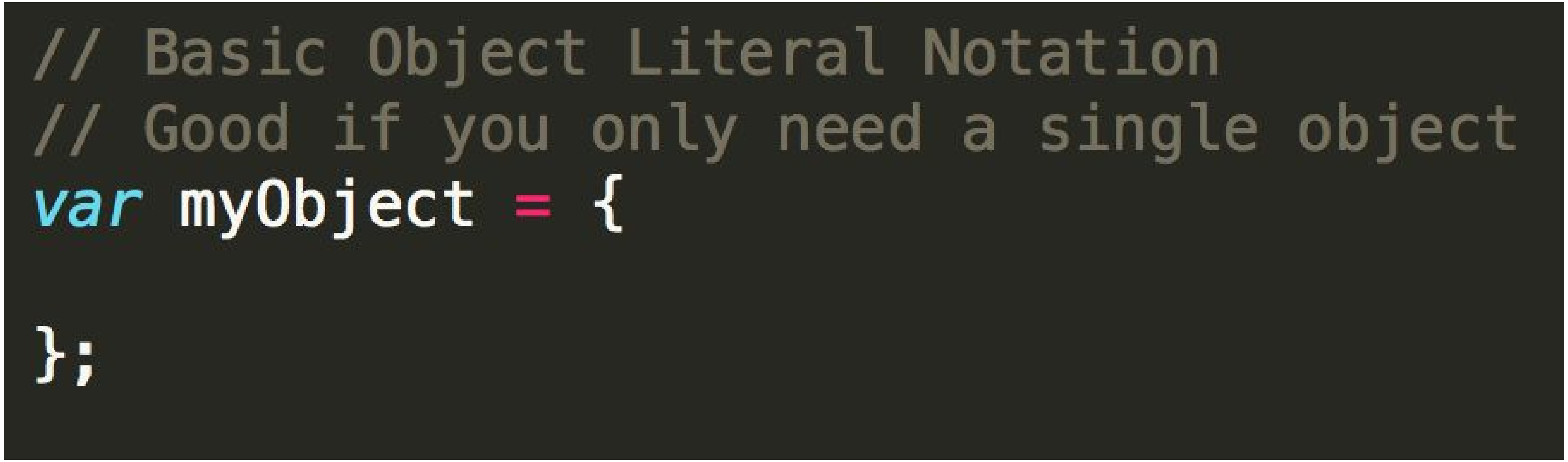
Literal is a preferred option for namespacing so that your JavaScript code doesn't interfere with other scripts running on the page and also if you are using this object as a single object and not requiring more than one instance of the object, whereas Constructor function type notation is preferred if you need to do some initial work before the object is created or require multiple instances of the object where each instance can be changed during the lifetime of the script.

Javascript Objects - Constructor Functions



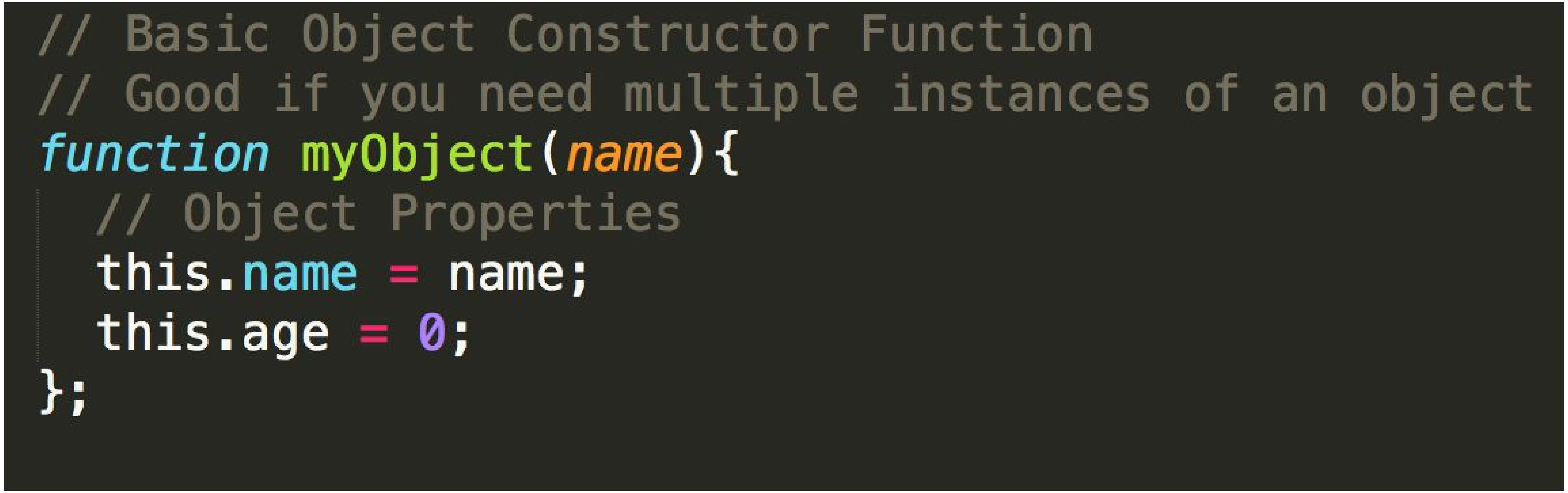
```
// Basic Object Constructor Function
// Good if you need multiple instances of an object
function myObject(){
};
```

Javascript Objects - Literal Notation



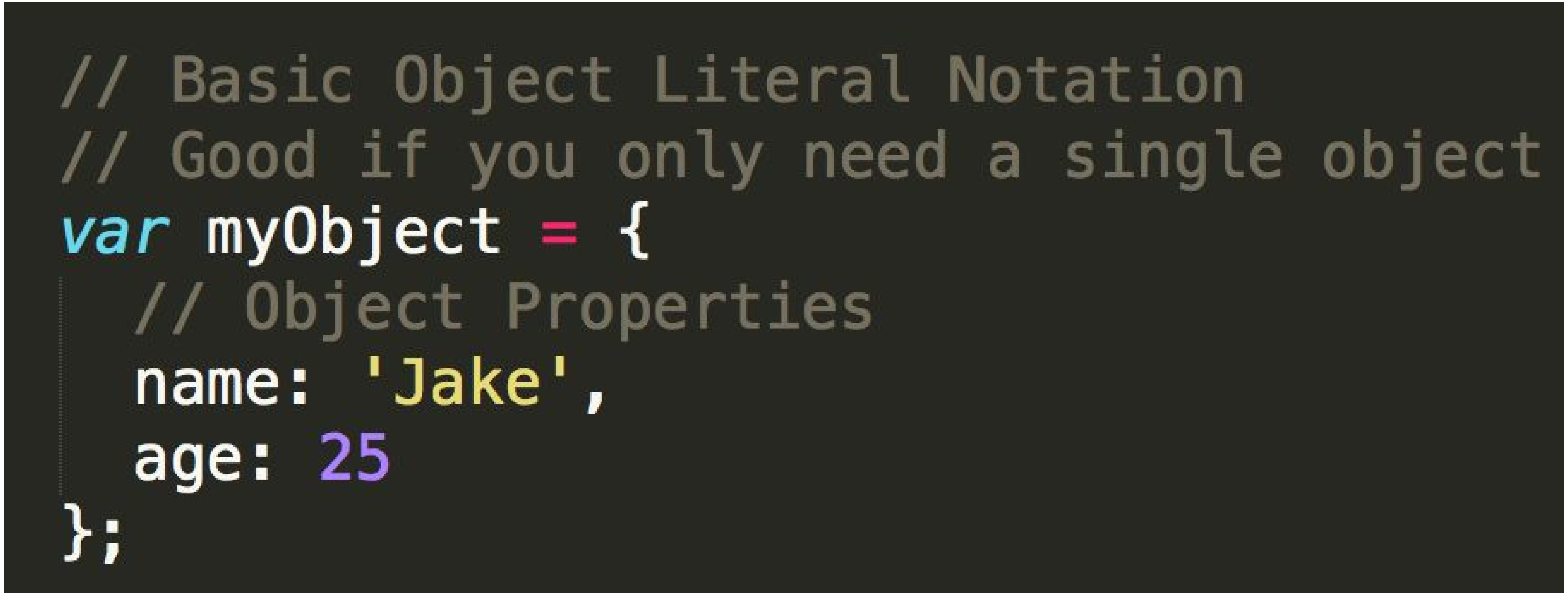
```
// Basic Object Literal Notation
// Good if you only need a single object
var myObject = {

};
```

Javascript Object Properties - Constructor



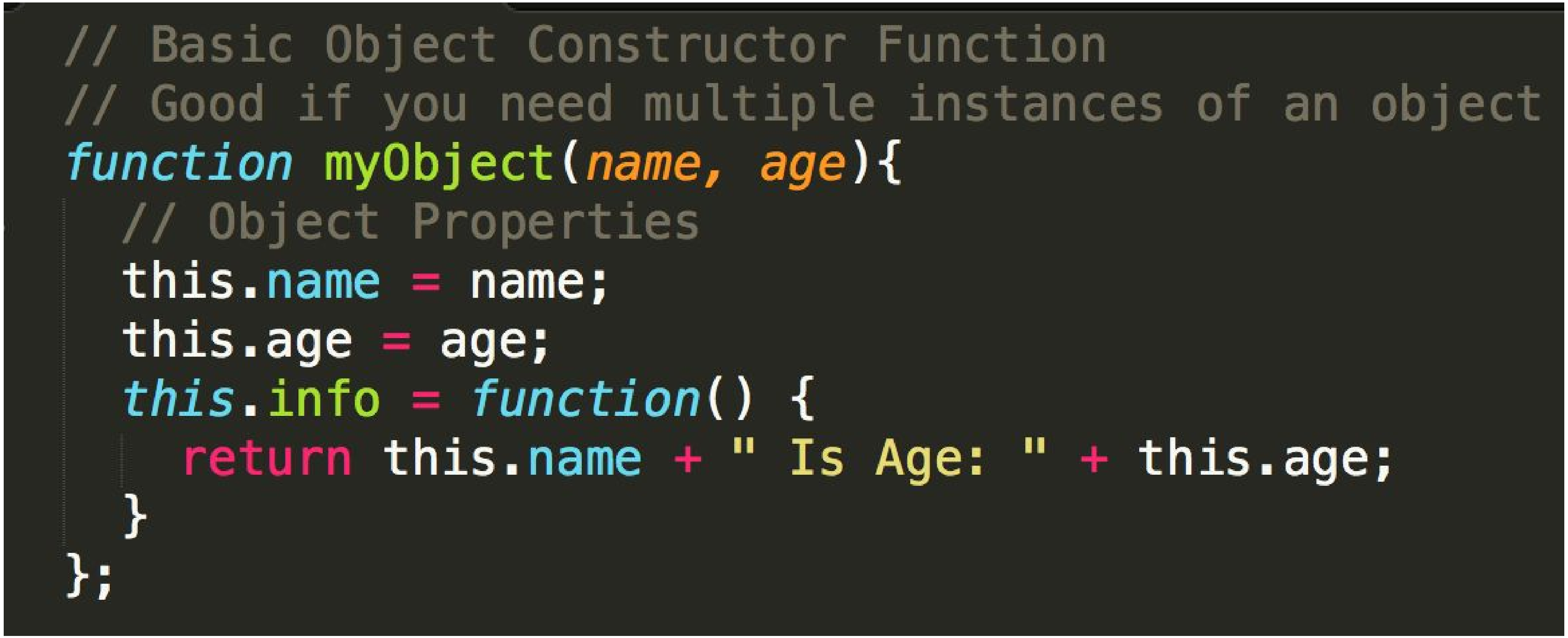
```javascript
// Basic Object Constructor Function
// Good if you need multiple instances of an object
function myObject(name){
  // Object Properties
  this.name = name;
  this.age = 0;
};
```

Javascript Object Properties - Literal



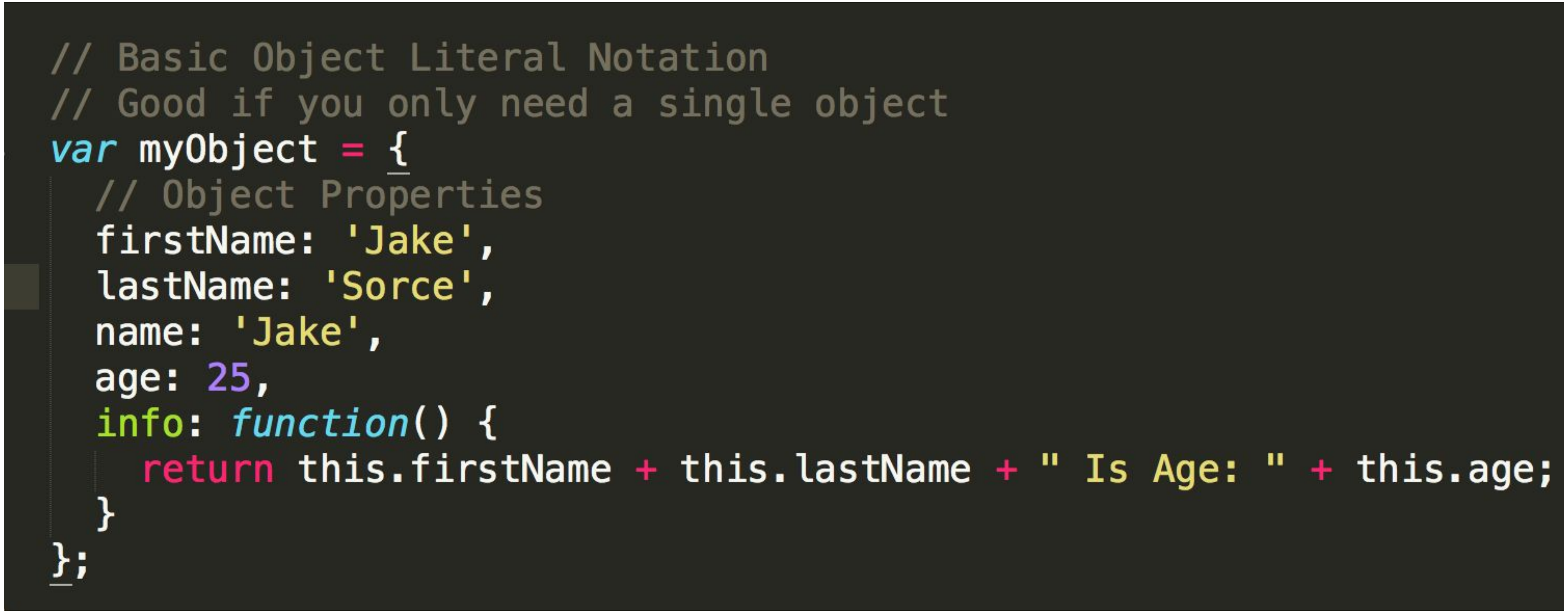
```
// Basic Object Literal Notation
// Good if you only need a single object
var myObject = {
  // Object Properties
  name: 'Jake',
  age: 25
};
```

Javascript Object Functions - Constructor



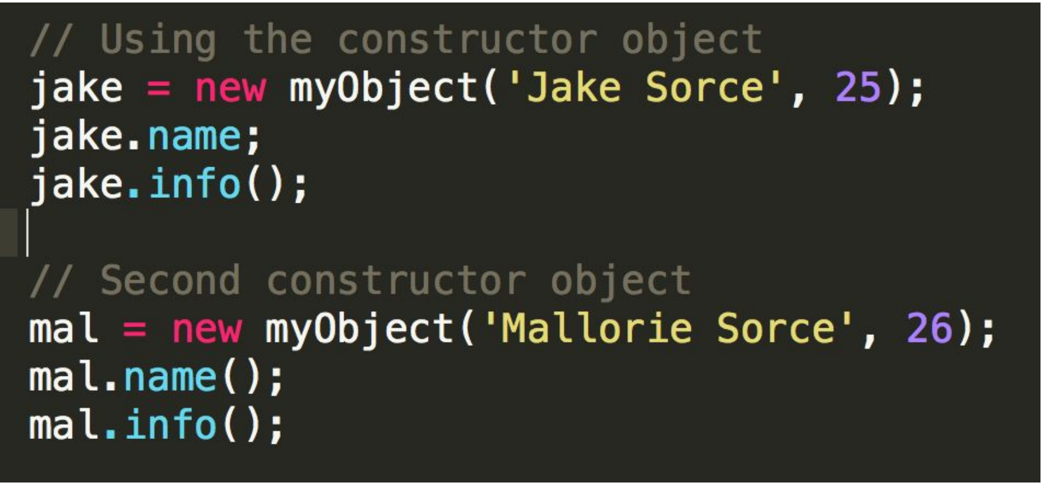
```
// Basic Object Constructor Function
// Good if you need multiple instances of an object
function myObject(name, age){
  // Object Properties
  this.name = name;
  this.age = age;
  this.info = function() {
    return this.name + " Is Age: " + this.age;
  }
};
```

Javascript Object Functions - Literal



```
// Basic Object Literal Notation
// Good if you only need a single object
var myObject = {
  // Object Properties
  firstName: 'Jake',
  lastName: 'Sorce',
  name: 'Jake',
  age: 25,
  info: function() {
    return this.firstName + this.lastName + " Is Age: " + this.age;
  }
};
```

Using Constructor Objects



```
// Using the constructor object
jake = new myObject('Jake Sorce', 25);
jake.name;
jake.info();

// Second constructor object
mal = new myObject('Mallorie Sorce', 26);
mal.name();
mal.info();
```

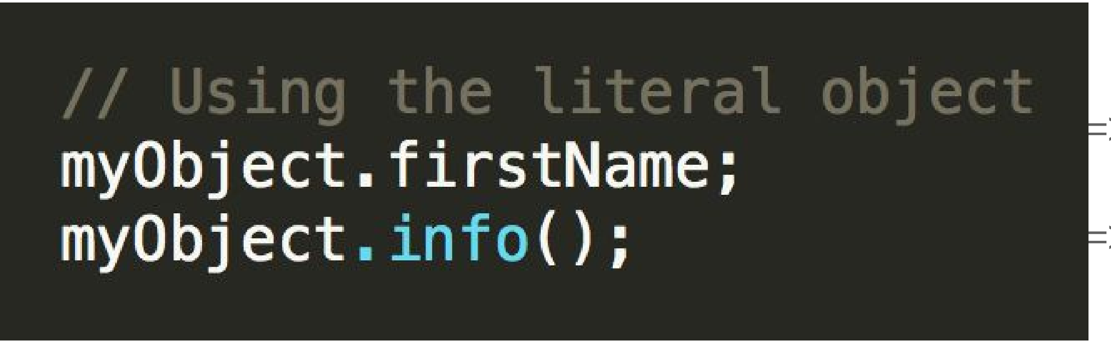
=> Jake Sorce

=> Jake Sorce Is Age: 25

=> Mallorie Sorce

=> Mallorie Sorce Is Age: 26

Using Object Literals



```
// Using the literal object
myObject.firstName;
myObject.info();
```

=> Jake

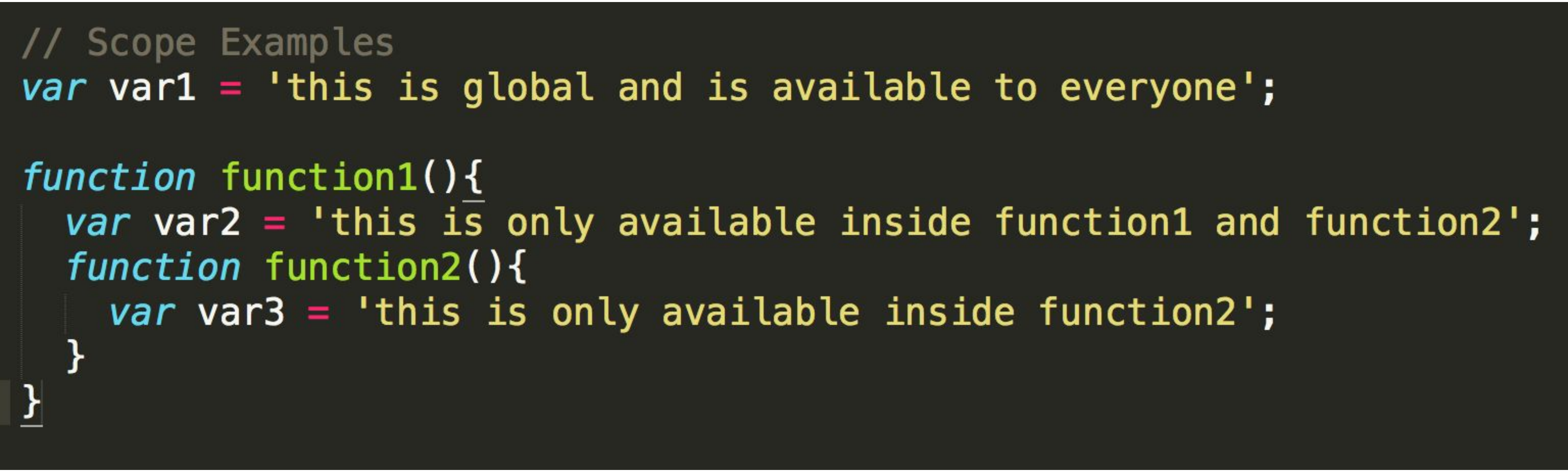
=> Jake Sorce Is Age: 25

Javascript Scope

- Scope in JavaScript is function/object based, so that means if you're outside of a function, you can't use a variable that is defined inside a function.

- There is however a scope chain, which means that a function inside another function can access a variable defined in its parent function.

Javascript Scope - Examples



```
// Scope Examples
var var1 = 'this is global and is available to everyone';

function function1(){
  var var2 = 'this is only available inside function1 and function2';
  function function2(){
    var var3 = 'this is only available inside function2';
  }
}
```

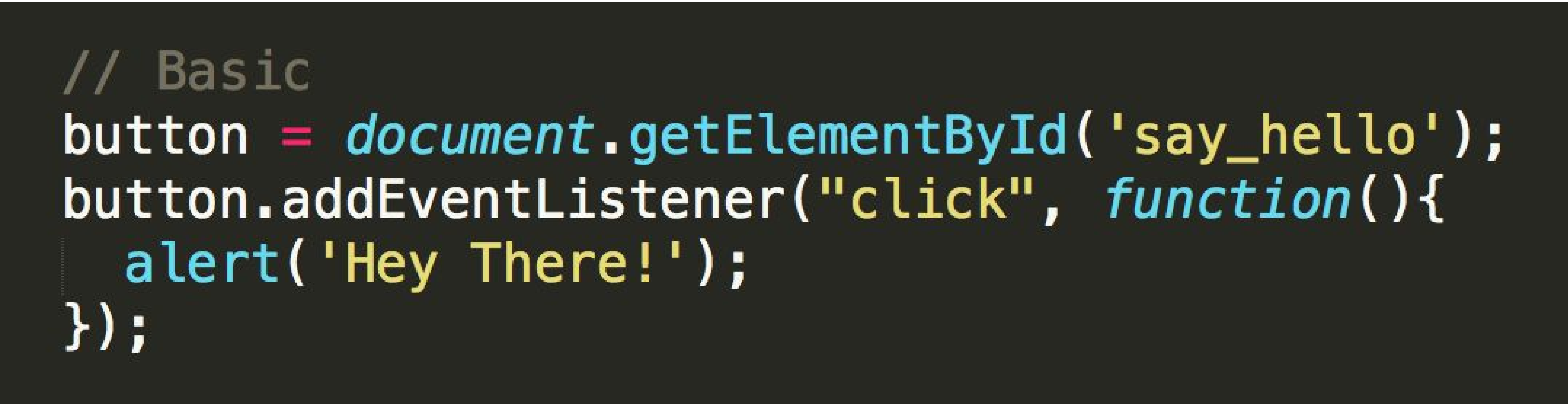
JavaScript Events

Events in Javascript are functions that get bound to different element events.

Some of the most common element events are:

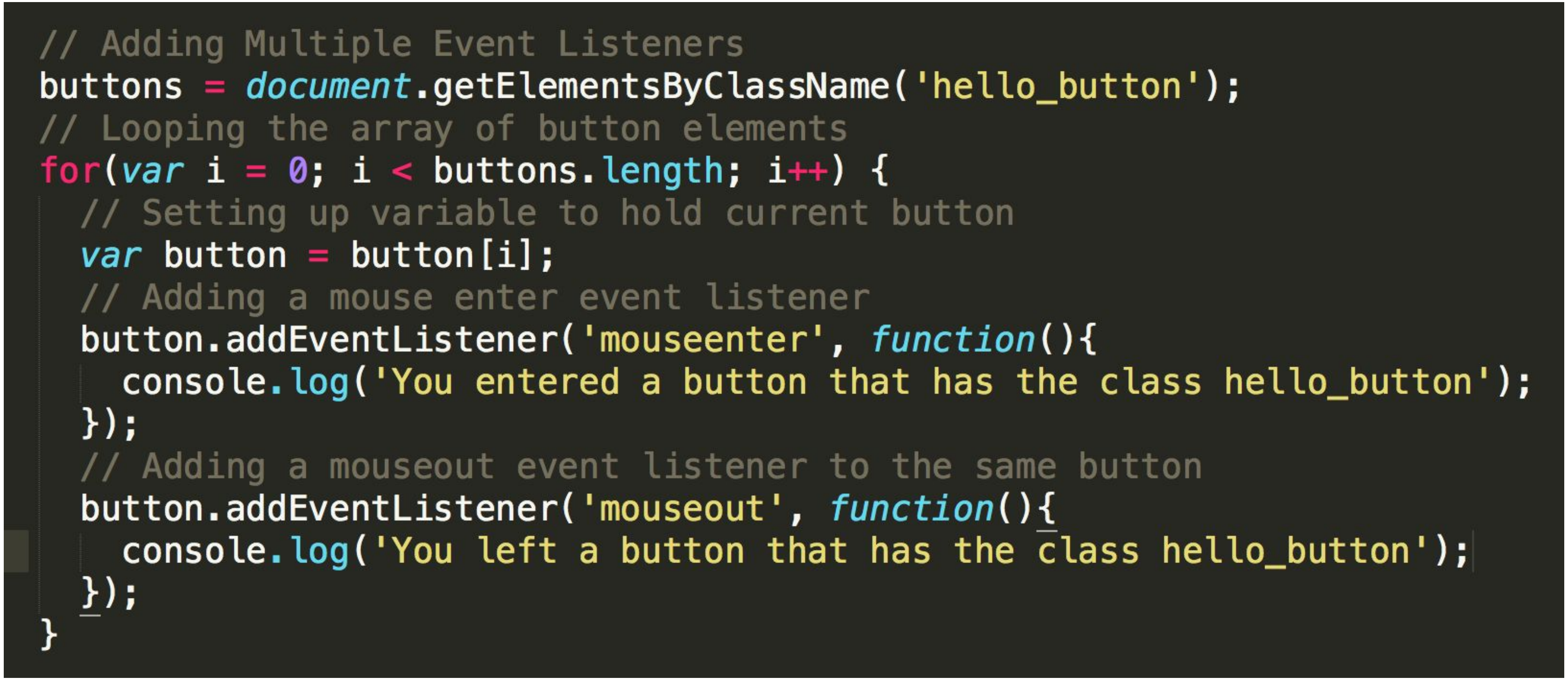
- onclick
  - this event fires when an element is clicked
- onfocus
  - this event fires when an element is focused
- onmouseout
  - this event fires when the mouse leaves the element
- onmouseenter
  - this event fires when the mouse enters the element
- onkeydown
  - this event fires when a key is pressed
- onkeyup
  - this event fires when a key is released

Javascript Events - Examples



```
// Basic
button = document.getElementById('say_hello');
button.addEventListener("click", function(){
  alert('Hey There!');
});
```

Javascript Events - Examples Cont...



```
// Adding Multiple Event Listeners
buttons = document.getElementsByClassName('hello_button');
// Looping the array of button elements
for(var i = 0; i < buttons.length; i++) {
  // Setting up variable to hold current button
  var button = button[i];
  // Adding a mouse enter event listener
  button.addEventListener('mouseenter', function(){
    console.log('You entered a button that has the class hello_button');
  });
  // Adding a mouseout event listener to the same button
  button.addEventListener('mouseout', function(){
    console.log('You left a button that has the class hello_button');
  });
}
```

Mini Project - Personal Budget App

Basic Goals:

- create a new directory
- create a index.html file in that directory
- create a budget.js file in that directory
- create a custom.css file in that directory
- link them together inside index.html (eg. <script> <link>)
- create an text input for monthly income
- create a bunch of inputs to add bill amounts to
- create a button to click on to calculate total left over after bills are paid
- display the total left over to the user

Mini Project - Personal Budget App Bonus

Bonus Goals:

- make adding and removing bill inputs dynamic (hint: search appendChild javascript)
- add a print button to print the page (hint: use materialize icon and add event listener)
- hide the unneeded items in the print view (hint: @media print)
- input validation for empty inputs, 0 or no values, ect...(hint: check their value)
- display the total left over in a way that you can: (hint: hidden header element use materialize helper class to hide an element and remove that class with javascript hint: (element.className = ‘’))
  - make the text red if the total left over is negative (hint: use materialize text color class)
  - make the text green if the total left over is positive (hint: use materialize text color class)
- use a frontend css / js framework (eg. bootstrap, materialize, foundation)