JavaScript: Functions, methods and objects

CS 4640 Programming Languages for Web Applications

[Robert W. Sebesta, “Programming the World Wide Web
Jon Duckett, Interactive Frontend Web Development]

Functions

Self-contained bits of JS code that allow us to

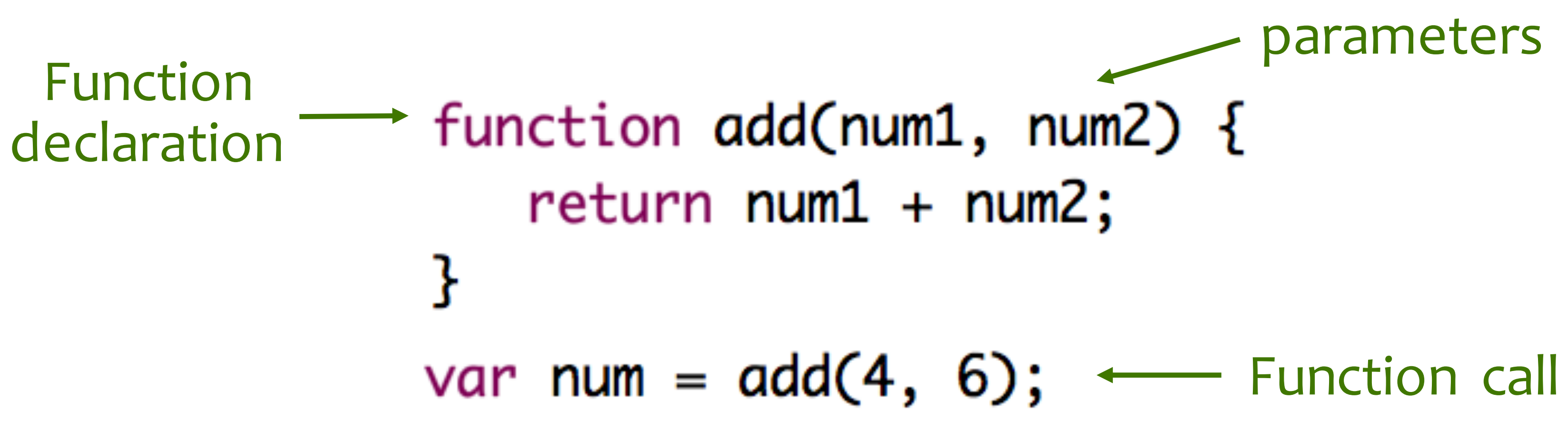
- Organize code
- Reuse the same code any number of times, from different parts of the script

JS supports several types of function. Commonly used types are:

- Named function declaration
- Anonymous functions

Named Functions

- Similar to Java functions but header is somewhat different



```
function add(num1, num2) {
    return num1 + num2;
}
var num = add(4, 6);
```

- Return type not specified (like PHP, since JS has dynamic typing)
- Parameter types also not specified
- Functions execute when they are called, just as in any language

Anonymous Functions and Function Expressions

- Functions can be assigned to variables

```
var magic = function(num1, num2) {
    return num1 + num2;
}
var myNum = magic(4, 6);
```

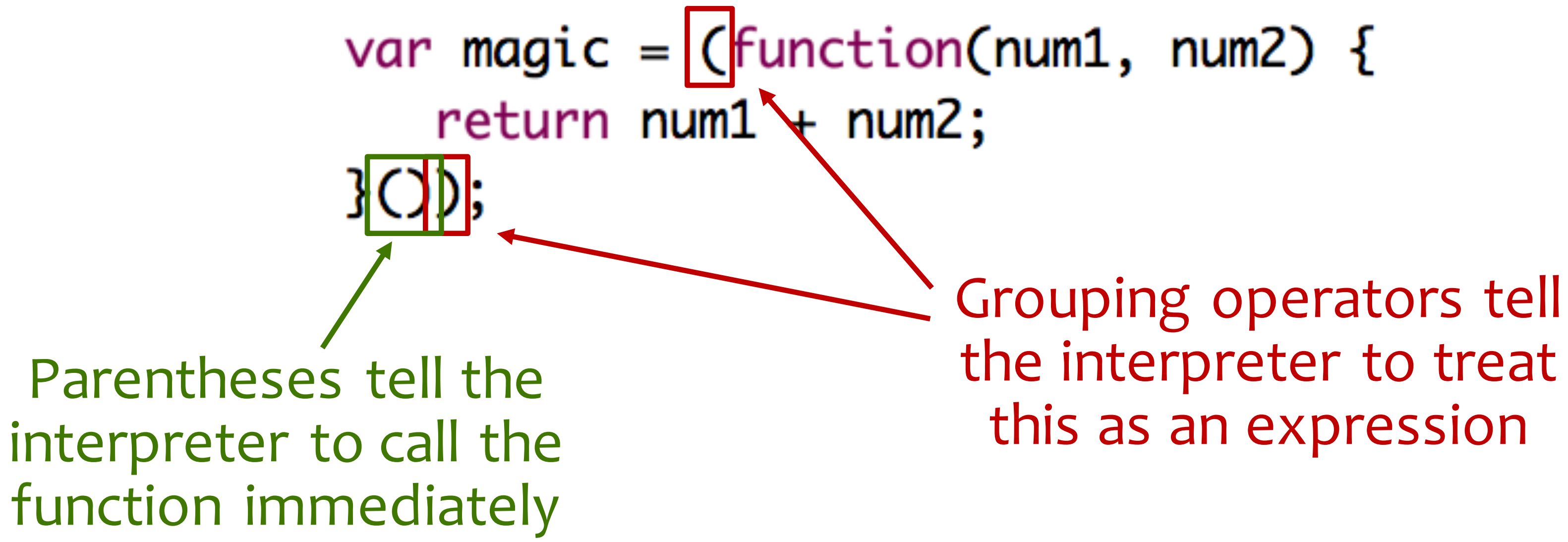
← “Function expression”

- Variables declared in a function are local to the function
- Parameters are all value
  - No parameter type-checking

[see jex3.html]

Immediately Invoked Function Expressions

- Anonymous functions can be executed once as the interpreter comes across them



```
var magic = (function(num1, num2) {
   return num1 + num2;
}());
```

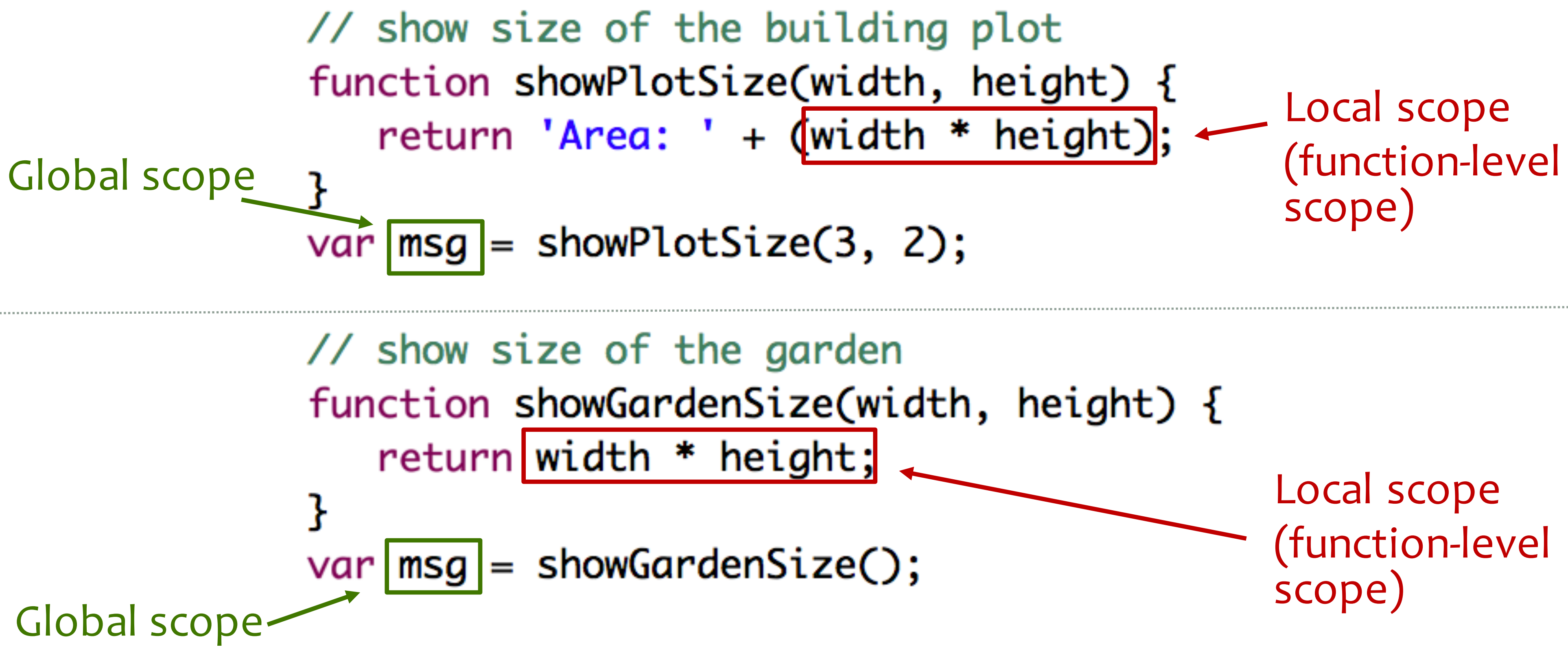
Parentheses tell the interpreter to call the function immediately

Grouping operators tell the interpreter to treat this as an expression

Functions and Default Values (ES6)

```
function add(num1=10, num2=45) {
  return num1 + num2;
}
var r = add();      // 55
r = add(40);        // 85
r = add(2, 6);      // 8
```

Global and Local Scopes



```
// show size of the building plot
function showPlotSize(width, height) {
   return 'Area: ' + (width * height);
}
var msg = showPlotSize(3, 2);
```

```
// show size of the garden
function showGardenSize(width, height) {
   return width * height;
}
var msg = showGardenSize();
```

Global scope
Local scope (function-level scope)
Local scope (function-level scope)
Global scope

Naming collision

- Two JavaScript files, both have a global variable with the same name

It's better to avoid creating too many global variables. Use function parameters if you need to share specific values with a function sit

Objects group variables and functions to create a model representing something you would recognize from the real world

Object type: Hotel

Event	Happens when
Reserve	reservation is made
Cancel	reservation is cancelle

Events are things or interactions that can happen to the objects

Properties	
Name:	Awesome
Rating:	5
Rooms:	70
Bookings:	56
Pool:	true
Gym:	true

Method	What it does
makeReservation()	increases value of *bookings* property
cancelReservation()	decreases value of *bookings* property
checkAvailability()	subtracts value of *bookings* property from value of *rooms* property and returns number of rooms available

Properties tell us the characteristics of the objects

Methods represent tasks that are associated with the objects (or things we can do with the objects)

Car

Accelerate	driver speeds up	changeSpeed()

Properties	
Make:	UVA1
currentSpeed:	30
Color:	yellow
Fuel:	gasoline

Method	What it does
changeSpeed()	increases or decreases value of *currentSpeed* property

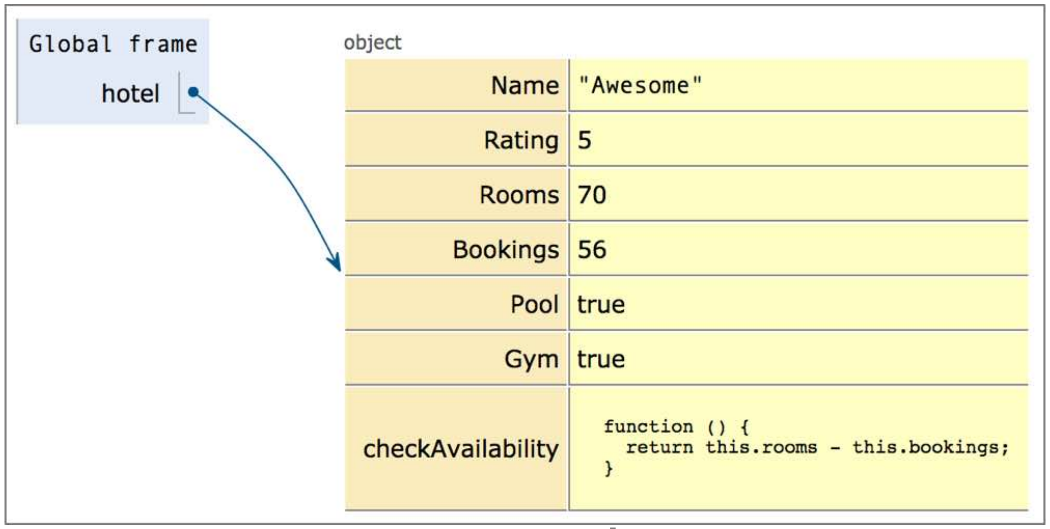
JavaScript Objects

- JavaScript is an object-based language
  - It supports for object-oriented programming but not at the same level as other languages (ES6: introduced `class` – still lacks private property)
- Objects are represented as property-value pair
  - The property values can be data or functions (methods)
- A property is something that can be modified :
  - Data properties : primitive values or references to objects
  - Method properties : can be executed
- Objects can be created and their properties can be changed dynamically
  - JS is not really typed .. If it doesn't care between a number and a string, why care between two kinds of objects?

Creating Objects

Create an object and assign variables and functions directly by using { } syntax

```
var hotel = {
   name: "Awesome",
   rating: 5,
   rooms: 70,
   bookings: 56,
   pool: true,
   gym: true,
   checkAvailability: function() {
      return this.rooms - this.bookings;
   }
};
```



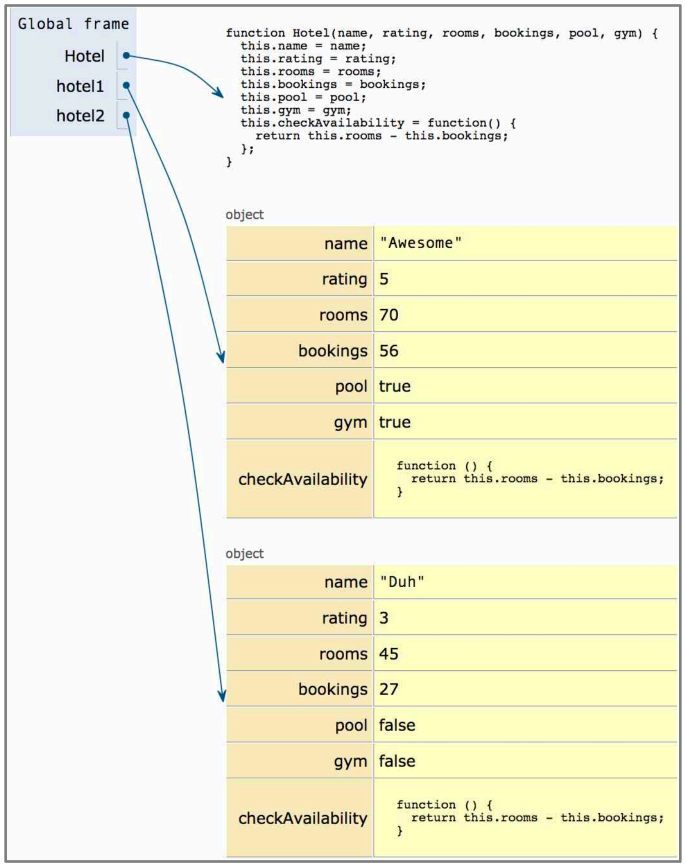
Name	"Awesome"
Rating	5
Rooms	70
Bookings	56
Pool	true
Gym	true
checkAvailability	`function () { return this.rooms - this.bookings; }`

Creating Objects with Constructors

Create an instance of the object using the constructor function and the new keyword

```
function Hotel(name, rating, rooms, bookings, pool, gym) {
   this.name = name;
   this.rating = rating;
   this.rooms = rooms;
   this.bookings = bookings;
   this.pool = pool;
   this.gym = gym;
   this.checkAvailability = function() {
      return this.rooms - this.bookings;
   };
}

var hotel1 = new Hotel('Awesome', 5, 70, 56, true, true);
var hotel2 = new Hotel('Duh', 3, 45, 27, false, false);
```



```
function Hotel(name, rating, rooms, bookings, pool, gym) {
  this.name = name;
  this.rating = rating;
  this.rooms = rooms;
  this.bookings = bookings;
  this.pool = pool;
  this.gym = gym;
  this.checkAvailability = function() {
    return this.rooms - this.bookings;
  };
}
```

object

name	"Awesome"
rating	5
rooms	70
bookings	56
pool	true
gym	true
checkAvailability	function () { return this.rooms - this.bookings; }

object

name	"Duh"
rating	3
rooms	45
bookings	27
pool	false
gym	false
checkAvailability	function () { return this.rooms - this.bookings; }

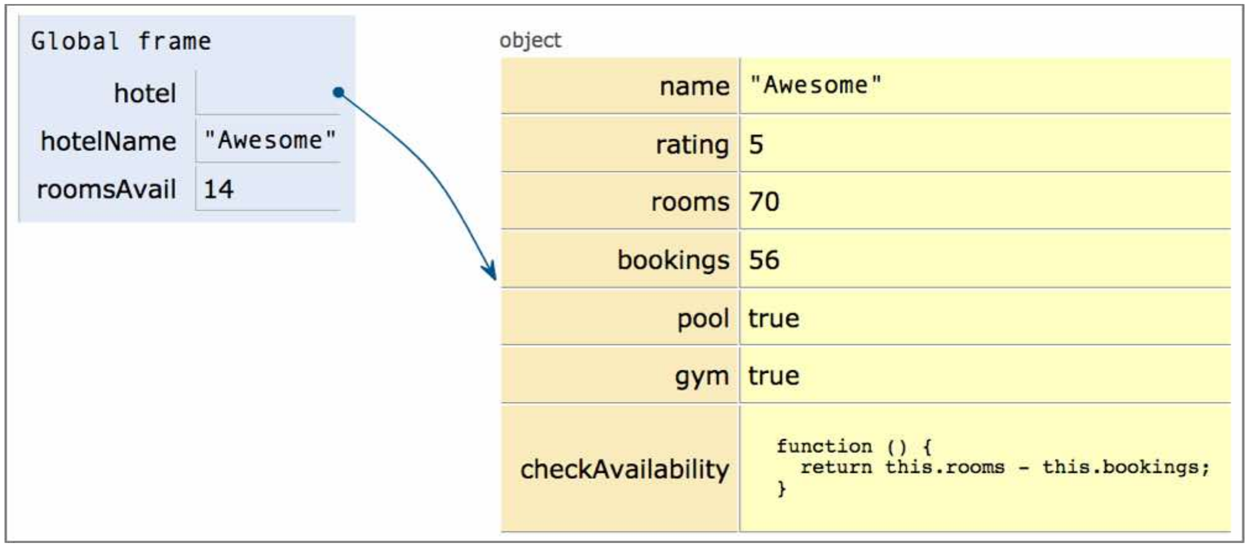
Accessing Objects

- Access properties or methods of an object using dot notation

```
var hotelName = hotel.name;
var roomsAvail = hotel.checkAvailability();
```

- Access properties or methods using square brackets

```
var hotelName = hotel['name'];
var roomsAvail = hotel['checkAvailability']();
```



hotel	
hotelName	"Awesome"
roomsAvail	14

name	"Awesome"
rating	5
rooms	70
bookings	56
pool	true
gym	true
checkAvailability	`function () { return this.rooms - this.bookings; }`

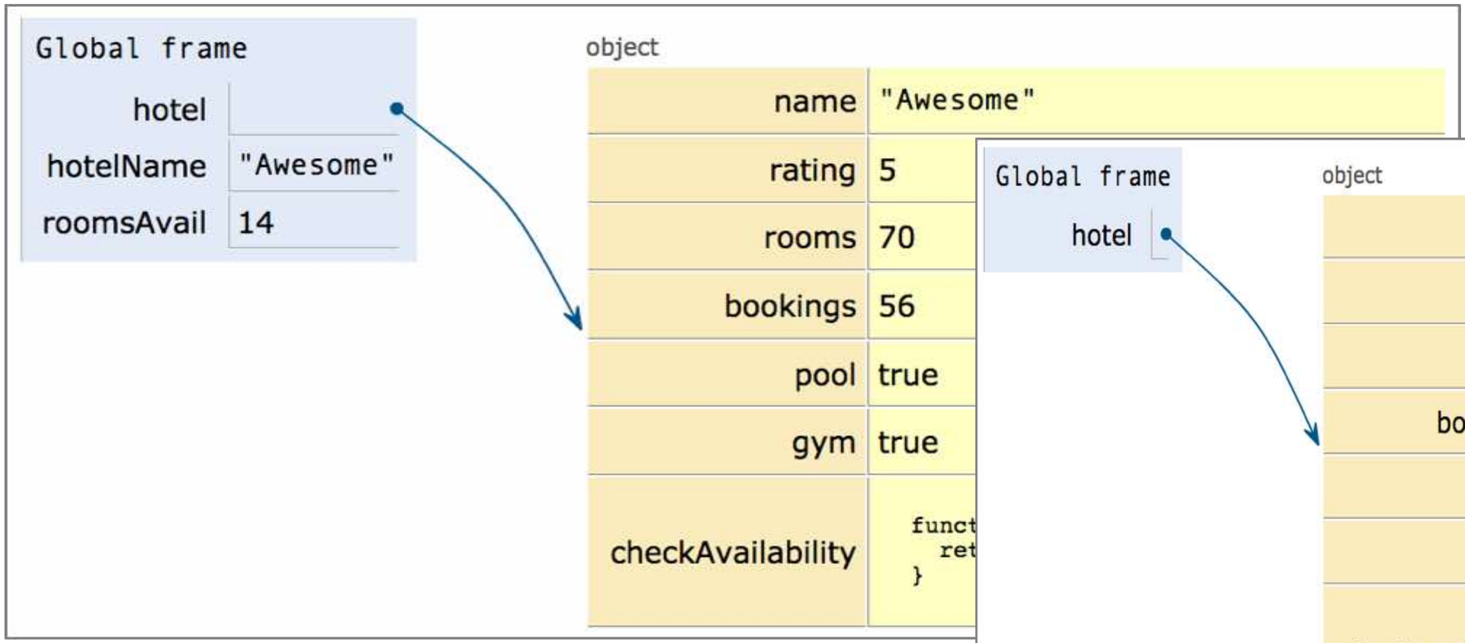
Updating Properties

- Update properties using dot notation

```
hotel.name = 'VeryAwesome';
```

- Update properties using square brackets

```
hotel['name'] = 'VeryAwesome';
```

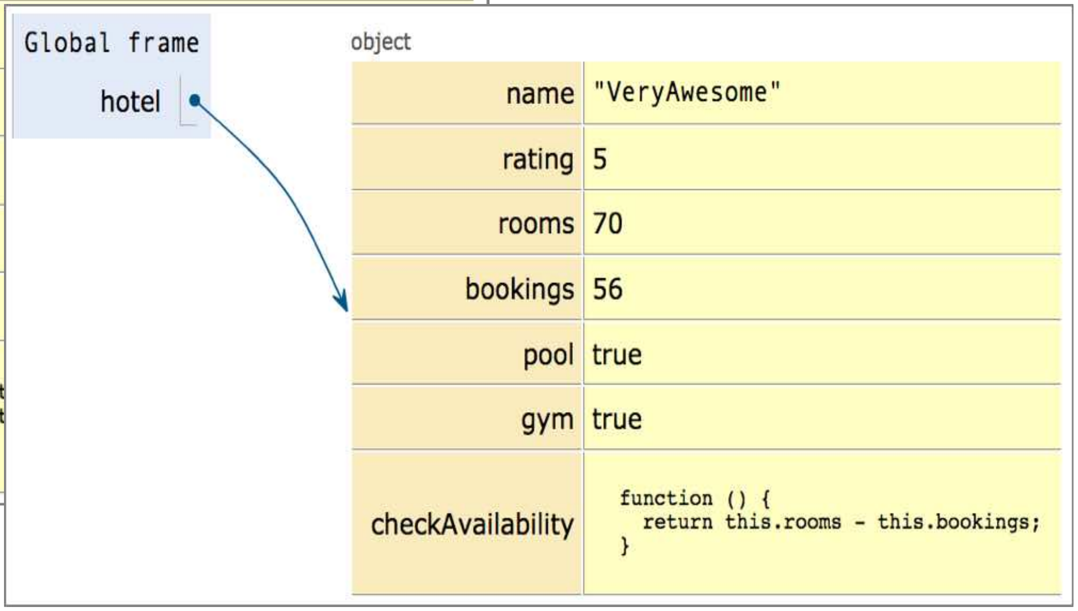


Global frame

hotel	
hotelName	"Awesome"
roomsAvail	14

object

name	"Awesome"
rating	5
rooms	70
bookings	56
pool	true
gym	true
checkAvailability	funct ret }



Global frame

hotel	

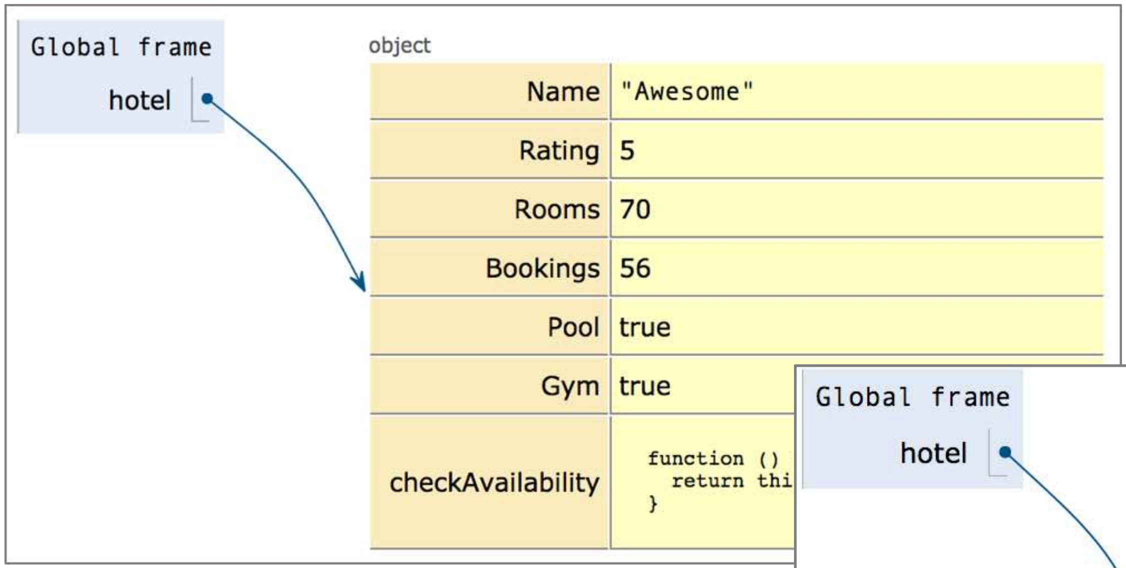
object

name	"VeryAwesome"
rating	5
rooms	70
bookings	56
pool	true
gym	true
checkAvailability	function () { return this.rooms - this.bookings; }

Adding Properties

- Add a property using a dot notation

```
hotel.shuttle = true;
```



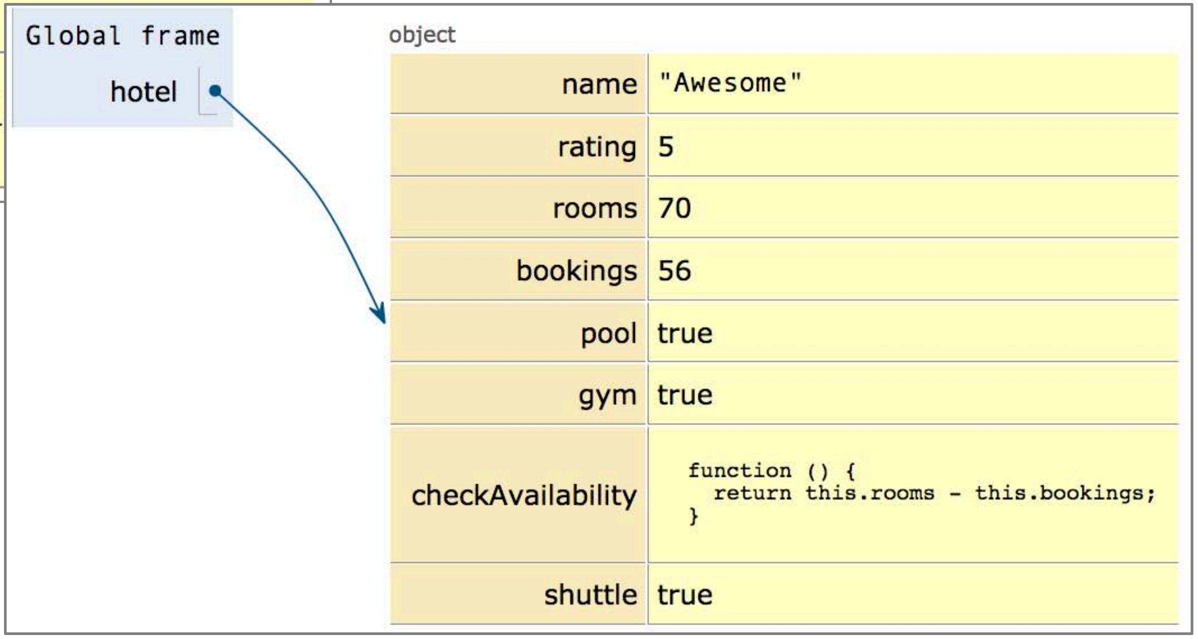
Global frame

hotel

object

Name	"Awesome"
Rating	5
Rooms	70
Bookings	56
Pool	true
Gym	true

checkAvailability	function ()
	  return thi
	}



Global frame

hotel

object

name	"Awesome"
rating	5
rooms	70
bookings	56
pool	true
gym	true

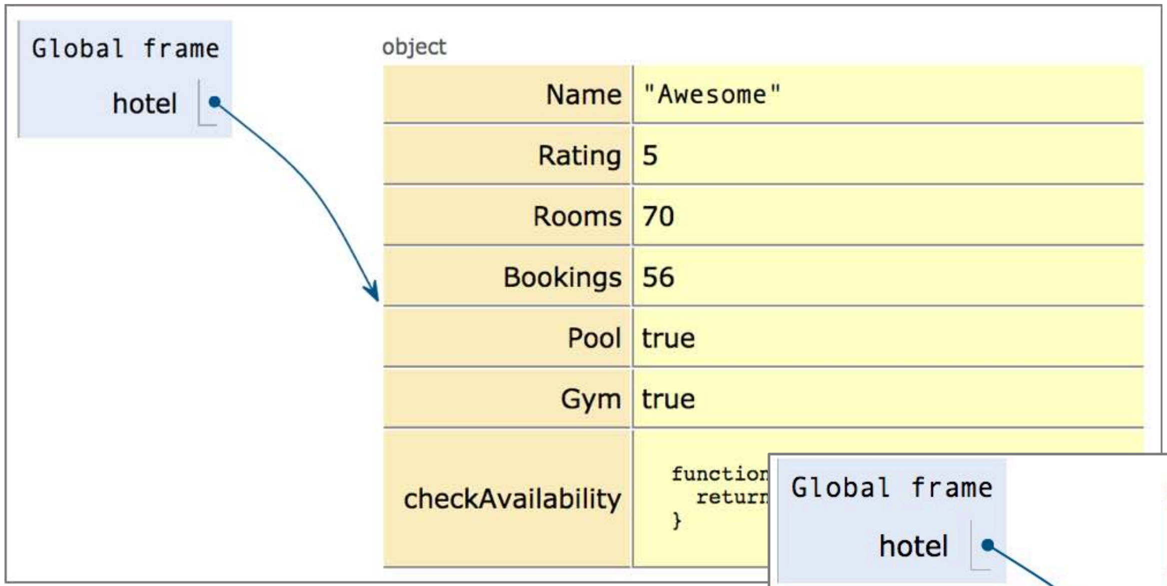
checkAvailability	function () {
	  return this.rooms - this.bookings;
	}

shuttle	true

Deleting Properties

- Delete a property using the delete keyword

```
delete hotel.rating;
```



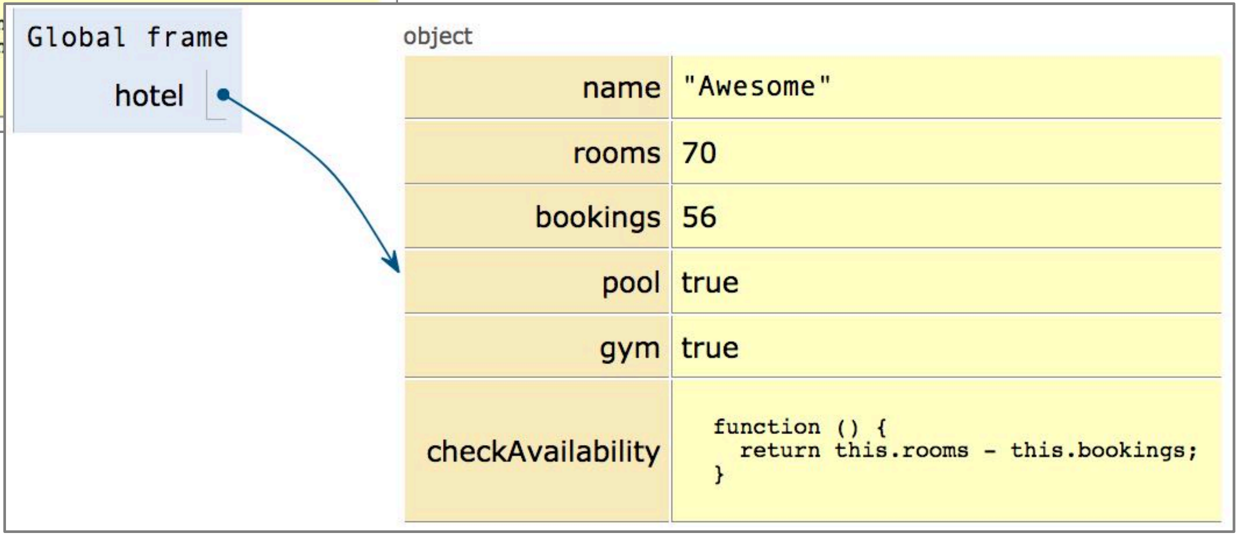
Global frame

hotel

object

Name	"Awesome"
Rating	5
Rooms	70
Bookings	56
Pool	true
Gym	true

checkAvailability	function
	return
	}



Global frame

hotel

object

name	"Awesome"
rooms	70
bookings	56
pool	true
gym	true

checkAvailability	function () {
	  return this.rooms - this.bookings;
	}