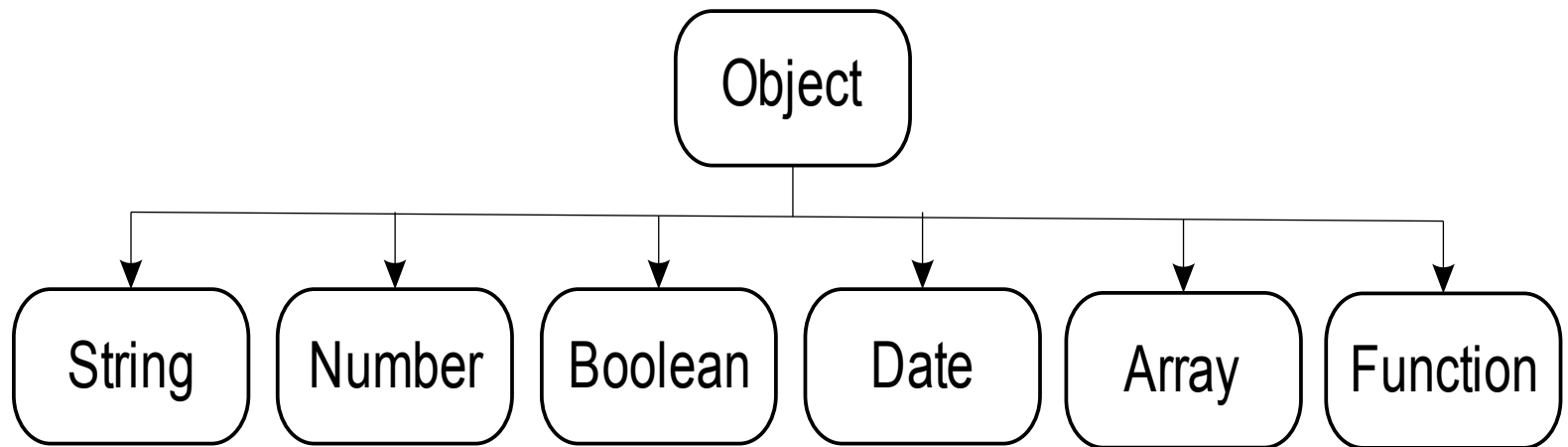


## Chapter 11

# How to create and use objects

## The JavaScript hierarchy of some of the native object types



## The syntax for creating a new object

```
var variable_name = new ObjectType(arguments);
```

## How to create a new object of the Date type

```
var today = new Date();
```

## How to create a new object of the Object type

```
var invoice = new Object();
```

## How to create a new object of the String type

```
var lastName = "Harris"; // = new String("Harris");
```

## How to create a new object of the Number type

```
var taxRate = .0875; // = new Number(.0875);
```

## How to create a new object of the Boolean type

```
var validFlag = true; // = new Boolean(true);
```

## The length property of a String object

```
length = lastName.length;
```

## The toFixed method of a Number object

```
formattedRate = taxRate.toFixed(4);
```

## Two ways to create an object of the Object type

```
var invoice = new Object();    // with the new keyword  
var invoice = {};            // with braces
```

## How to initialize an object with one property

```
var invoice = { taxRate: 0.0875 };
```

## How to initialize an object with one method

```
var invoice = {  
    getSalesTax: function( subtotal ) {  
        return ( subtotal * invoice.taxRate );  
    }  
}
```

# How to initialize an object with properties and methods

```
var invoice = {
  getSalesTax: function( subtotal ) {
    return ( subtotal * invoice.taxRate );
  },
  getTotal: function ( subtotal, salesTax ) {
    return subtotal + salesTax;
  },
  taxRate: 0.0875,
  dueDays: 30
}
```

## How to nest one object within another

```
var invoice = {
  terms: {
    taxRate: 0.0875,
    dueDays: 30
  }
}
```

## How to refer to a nested object

```
alert( invoice.terms.taxRate );           // Displays 0.0875
alert( invoice["terms"]["dueDays"] );    // Displays 30
```

## How to add properties and methods to an object

```
// Using the dot operator
invoice.taxRate = 0.0875;           // property
invoice.getSalesTax = function(subtotal) { // method
    return ( subtotal * invoice.taxRate );
};
```

```
// Using brackets
invoice["taxRate"] = 0.0875;       // property
```

## How to modify the properties of an object

```
invoice.taxRate = 0.095;
```

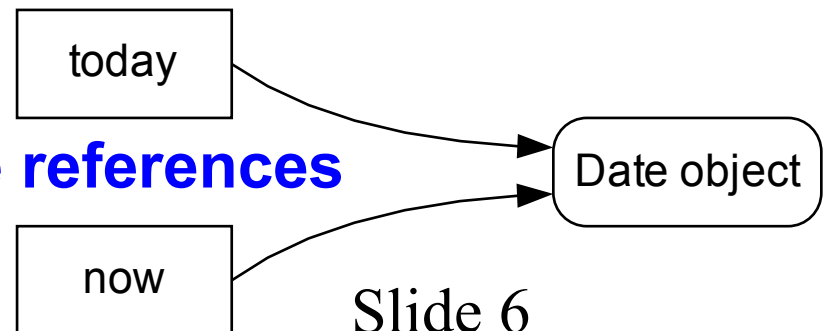
## How to remove a property from an object

```
delete invoice.taxRate;
alert( invoice.taxRate ); // Displays undefined
```

## Two variables that refer to the same object

```
var today = new Date();
var now = today;
```

## A diagram that illustrates these references



Slide 6

## How to delete an object

```
var today = new Date();  
delete today;  
alert(today); // Displays undefined
```

## How to delete a reference to an object

```
var today = new Date();  
var now = today;  
delete today;  
alert( today ); // Displays undefined  
alert( now.getFullYear() ); // Displays the year
```

## How to code constructor functions

```
// A constructor that creates an InvestmentTerms object  
var InvestmentTerms = function (init_rate, init_years) {  
    // Define properties of the object  
    this.rate = init_rate;  
    this.years = init_years;  
}  
  
// A constructor that creates an Invoice object  
var Invoice = function () {  
    this.items = [];  
    this.tax_rate = 0.07;  
}
```

## How to add methods to object types

```
// Add getMonths method to the InvestmentTerms type
InvestmentTerms.prototype.getMonths = function () {
    return this.years * 12;
}

// Add the delete_item method to the Invoice type
Invoice.prototype.delete_item = function(item_code) {
    if ( item_code in this.items ) {
        delete this.items[item_code];
    }
    return this;
}
```

## How to create a new instance of an object type

```
var terms = new InvestmentTerms(0.04, 10);
var invoice = new Invoice();
```

## How to access a new object's properties

```
alert (terms.rate); // Displays 0.04
alert (invoice.tax_rate); // Displays 0.07
```

## How to use a new object's methods

```
var months = terms.getMonths();
invoice.delete_item(item_code);
```

# Terms

- To create your own object type, you code a *constructor function* (or just *constructor*) with the name of the object type.
- Within a constructor, the *this* keyword refers to the object that is created by the constructor.
- In JavaScript, every object type has a *prototype object* that contains the properties and methods of that object type.

## The Vehicle object type

```
var Vehicle = function (make, model) {
  this.make = make;
  this.model = model;
  this.miles = 0;
  this.last_oil_change = 0;
}

Vehicle.prototype.drive = function ( miles ) {
  this.miles += miles;
  return this;
}

Vehicle.prototype.change_oil = function () {
  this.last_oil_change = this.miles;
  return this;
}

Vehicle.prototype.need_oil_change = function () {
  return ( this.miles - this.last_oil_change > 3000 );
}
```

## The Car type that inherits the Vehicle type

```
var Car = function ( make, model, door_count, hatch_back ) {  
    this.make = make;  
    this.model = model;  
    this.door_count = door_count;  
    if ( arguments.length == 4 ) {  
        this.hatch_back = hatch_back;  
    } else {  
        this.hatch_back = false;  
    }  
    this.miles = 0;  
    this.last_oil_change = 0;  
}
```

```
Car.prototype = new Vehicle();
```

## Code that uses the Car object

```
var myCar = new Car("Univeral Imports", "Q", 2);  
  
// Call methods from the Vehicle type  
myCar.drive(2000).change_oil().drive(4000);
```

## How to add a method to the Date object type

```
Date.prototype.isLeapYear = function () {  
    var year = this.getFullYear();  
    if ( year % 4 == 0 ) {  
        if ( year % 100 == 0 ) {  
            if ( year % 400 == 0 ) {  
                return true;  
            } else {  
                return false;  
            }  
        } else {  
            return true;  
        }  
    } else {  
        return false;  
    }  
}
```

```
var opening = new Date( 2009, 1, 1 );    // Feb 1, 2009  
alert( opening.isLeapYear() );         // Displays false
```

## How to add a method to the String object type

```
String.prototype.reverse = function () {  
    var reverse = "";  
    for (var i = this.length - 1; i >= 0; i-- ) {  
        reverse += this.charAt(i);  
    }  
    return reverse;  
}
```

```
var message = "JavaScript";  
alert( message.reverse() );           // Displays "tpircSavaJ"
```

## How to add a method to the Math object

```
Math.add = function( x, y ) {  
    return parseFloat(x) + parseFloat(y);  
}
```

```
alert( "3" + "5" );                 // Displays 35  
alert( Math.add("3", "5") );        // Displays 8
```

## An add method that is not a cascading method

```
Array.prototype.add_value = function (key, value) {  
    this[key] = value;  
}
```

## Chaining the add method calls won't work

```
var pin_codes = [];  
pin_codes.add_value("Mike", "123").add_value("Ray", "987");
```

## The add methods must be called one at a time

```
var pin_codes = [];  
pin_codes.add_value("Mike", "123");  
pin_codes.add_value("Ray", "987");
```

## Term

- A *cascading method* can be chained with other methods.

## An add method that is a cascading method

```
Array.prototype.add_value = function (key, value) {  
    this[key] = value;  
    return this;  
}
```

## A delete method that is a cascading method

```
Array.prototype.delete_value = function (key) {  
    delete this[key];  
    return this;  
}
```

## Chaining the method calls does work

```
var pin_codes = [];  
pin_codes.add_value("Mike", "123").add_value("Ray", "987");  
pin_codes.add_value("Kelly", "456").delete_value("Ray");
```

## The syntax of the for-in statement

```
for ( var property_name in object ) {  
    // code to execute  
}
```

## The objects used by the for-in loops that follow

```
var application = {  
    rate: 0.04,  
    years: 10  
};
```

```
var invoice = {  
    getSalesTax: function( subtotal ) {  
        return ( subtotal * invoice.taxRate );  
    },  
    getTotal: function ( subtotal, salesTax ) {  
        return subtotal + salesTax;  
    },  
    taxRate: 0.0875,  
    dueDays: 30  
}
```

## A for-in loop with the application object

```
for ( var property in application ) {  
    alert( property + ": " + application[property] );  
}
```

## A for-in loop with the invoice object

```
var propList = "";  
for ( var property in invoice ) {  
    propList += property + ": " + invoice[property] + "\n";  
}  
alert ( propList );
```

## The propertyIsEnumerable method

```
application.propertyIsEnumerable("rate") // true  
application.propertyIsEnumerable("propertyIsEnumerable")  
//false
```

## Term

- A for-in statement loops through all the defined properties of an object that are *enumerable*.

## Objects and variables for the following code

```
var application = {  
    rate: 0.04,  
    years: 10  
};  
  
function Invoice() {  
    this.items = [];  
    this.tax_rate = 0.07;  
}  
var invoice = new Invoice();  
  
var today = new Date();  
var testArray = [];  
var testString = "123";  
var testNumber = 123;
```

## The in operator

```
alert( "rate" in application );           // Displays true
alert( "month" in application );         // Displays false
alert( "items" in invoice );            // Displays true
```

## The instanceof operator

```
alert( today instanceof Object );        // Displays true
alert( today instanceof Date );          // Displays true
alert( application instanceof Object );   // Displays true
alert( application instanceof Date );     // Displays false
alert( invoice instanceof Object );       // Displays true
alert( invoice instanceof Invoice );      // Displays true
```

## The typeof operator

```
alert( typeof application );             // Displays object
alert( typeof testArray );               // Displays object
alert( typeof testString );              // Displays string
alert( typeof testNumber );              // Displays number
alert( typeof application.rate );        // Displays number
alert( typeof Invoice );                  // Displays function
```

# The user interface for the Invoice application

## Invoice Manager

Item Code:

Item Name:

Item Cost:

Quantity:

---

Item Code:

Current Invoice

Item Code	Item Name	Qty	Item Cost	Line Cost
CS08	C# 2008	1	\$ 52.50	\$ 52.50
JSP2	Java Servlets and JSP	2	\$ 52.50	\$ 105.00
VB08	Visual Basic 2008	1	\$ 52.50	\$ 52.50

Subtotal:

Sales Tax:

Total:

## The XHTML file (excerpt)

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Invoice Manager</title>
  <link rel="stylesheet" type="text/css"
        href="invoice.css" />
  <script type="text/javascript"
          src="invoice_library.js"></script>
  <script type="text/javascript"
          src="invoice.js"></script>
</head>
```

## The XHTML file (continued)

```
<body>
<div id="content">
  <h1>Invoice Manager</h1>
  <div class="formLayout">
    <label for="item_code">Item Code:</label>
    <input type="text" name="item_code"
      id="item_code" /><br />
    <label for="item_name">Item Name:</label>
    <input type="text" name="item_name"
      id="item_name" /><br />
    <label for="item_cost">Item Cost:</label>
    <input type="text" name="item_cost"
      id="item_cost" /><br />
    <label for="item_qty">Quantity:</label>
    <input type="text" name="item_qty"
      id="item_qty" value="1" /><br />
    <label>&nbsp;</label>
    <input type="button" id="item_add"
      value="Add/Update Item" /><br />
  </div>
</div>
```

## The XHTML file (continued)

```
<div class="formLayout deleteForm">
  <label for="item_delete_code">Item Code:</label>
  <input type="text" name="item_delete_code"
    id="item_delete_code" /><br />
  <label>&nbsp;</label>
  <input type="button" id="item_delete"
    value="Delete Item" /><br />
</div>
<p class="startInvoice">Current Invoice</p>
<p><textarea id="item_list"
  rows="5" cols="80"></textarea></p>
```

## The XHTML file (continued)

```
<div class="formLayout">
  <label for="subtotal">Subtotal:</label>
  <input type="text" name="subtotal" id="subtotal"
    class="disabled" disabled="disabled" /><br />
  <label for="sales_tax">Sales Tax:</label>
  <input type="text" name="sales_tax" id="sales_tax"
    class="disabled" disabled="disabled" /><br />
  <label for="total">Total:</label>
  <input type="text" name="total" id="total"
    class="disabled" disabled="disabled" /><br />
</div>
```

```
</body>
</html>
```

## The invoice\_library.js file

```
String.prototype.pad_left = function() {
    if ( arguments.length < 1 || arguments.length > 2 ) {
        return this;
    }
    var width = parseInt(arguments[0]);
    var pad = " ";
    if ( arguments.length == 2 ) pad = arguments[1];
    var result = this;
    while ( result.length < width ) {
        result = pad + result;
    }
    return result;
}
```

## The invoice\_library.js file (continued)

```
String.prototype.pad_right = function() {
    if ( arguments.length < 1 || arguments.length > 2 ) {
        return this;
    }
    var width = parseInt(arguments[0]);
    var pad = " ";
    if ( arguments.length == 2 ) pad = arguments[1];
    var result = this;
    while ( result.length < width ) {
        result = result + pad;
    }
    return result;
}

var Item_Info = function ( item_name, item_cost, item_qty )
{
    this.item_name = item_name;
    this.item_cost = parseFloat(item_cost);
    this.item_qty  = parseInt(item_qty);
}
```

## The invoice\_library.js file (continued)

```
var Invoice = function () {
    this.items = [];
    this.tax_rate = 0.07;
}

Invoice.prototype.add_item = function(item_code, item_info)
{
    if ( ! item_info instanceof Item_Info ) return this;
    if ( isNaN(item_info.item_cost) ) return this;
    if ( isNaN(item_info.item_qty) ) return this;
    if ( item_info.item_name == "" ) return this;
    if ( item_code == "" ) return this;

    item_code = item_code.toUpperCase();
    if ( item_code in this.items ) {
        delete this.items[item_code];
    }
    this.items[item_code] = item_info;
    this.sort_by_code();
    return this;
}
```

## The invoice\_library.js file (continued)

```
Invoice.prototype.delete_item = function(item_code) {
    item_code = item_code.toUpperCase();
    if ( item_code in this.items ) {
        delete this.items[item_code];
    }
    return this;
}
```

```
Invoice.prototype.get_item_list = function() {
    var item_list, line_cost, item_count = 0;
    item_list = "Item Code".pad_right(10) + " ";
    item_list += "Item Name".pad_right(40) + " ";
    item_list += "Qty ";
    item_list += "Item Cost ";
    item_list += "Line Cost\n";
    item_list += "".pad_right(10, "-") + " ";
    item_list += "".pad_right(40, "-") + " ";
    item_list += "--- ";
    item_list += "".pad_right(9, "-") + " ";
    item_list += "".pad_right(9, "-") + "\n";
    for ( var code in this.items ) {
```

## The invoice\_library.js file (continued)

```
    line_cost =
        this.items[code].item_qty *
        this.items[code].item_cost;
    item_list +=
        code.pad_right(10) + " ";
    item_list +=
        this.items[code].item_name.pad_right(40) + " ";
    item_list +=
        this.items[code].item_qty.toString().pad_left(3)
        + " ";
    item_list +=
        "$" +
        this.items[code].item_cost.toFixed(2).pad_left(8)
        + " ";
    item_list += "$" + line_cost.toFixed(2).pad_left(8)
        + "\n";
    item_count++;
}
return (item_count == 0) ? "" : item_list;
}
```

## The invoice\_library.js file (continued)

```
Invoice.prototype.get_subtotal = function () {
    var subtotal = 0, line_cost;
    for ( var code in this.items ) {
        line_cost =
            this.items[code].item_qty *
            this.items[code].item_cost;
        subtotal += parseFloat( line_cost.toFixed(2) );
    }
    return subtotal;
}
```

```
Invoice.prototype.get_sales_tax = function () {
    var subtotal = this.get_subtotal();
    var sales_tax = subtotal * this.tax_rate;
    return parseFloat( sales_tax.toFixed(2) );
}
```

```
Invoice.prototype.get_total = function () {
    var total = this.get_subtotal() + this.get_sales_tax();
    return parseFloat( total.toFixed(2) );
}
```

## The invoice\_library.js file (continued)

```
Invoice.prototype.sort_by_code = function () {  
    var code, codes = [], sorted_list = [];  
    for ( code in this.items ) codes.push(code);  
    codes.sort();  
    for ( code in codes ) {  
        sorted_list[ codes[code] ] =  
            this.items[ codes[code] ];  
    }  
    this.items = sorted_list;  
    return this;  
}
```

## The invoice.js file

```
var invoice = new Invoice();

var $ = function(id) { return document.getElementById(id); }

var update_invoice = function () {
    $("#item_list").value = invoice.get_item_list();
    $("#subtotal").value = invoice.get_subtotal().toFixed(2);
    $("#sales_tax").value =
        invoice.get_sales_tax().toFixed(2);
    $("#total").value = invoice.get_total().toFixed(2);

    $("#item_code").value = "";
    $("#item_name").value = "";
    $("#item_cost").value = "";
    $("#item_qty").value = "1";
    $("#item_delete_code").value = "";

    $("#item_code").focus();
}
```

## The invoice.js file (continued)

```
var item_add_click = function() {
    var item_code = $("item_code").value;
    var item_name = $("item_name").value;
    var item_cost = $("item_cost").value;
    var item_qty = $("item_qty").value;
    var item_info =
        new Item_Info(item_name, item_cost, item_qty);
    invoice.add_item(item_code, item_info);
    update_invoice();
}
var item_delete_click = function() {
    var item_code = $("item_delete_code").value;
    invoice.delete_item(item_code);
    update_invoice();
}
window.onload = function () {
    $("item_add").onclick = item_add_click;
    $("item_delete").onclick = item_delete_click;
    $("item_code").focus();
}
```