For additional materials, please see http://www.coreservlets.com/. The JavaScript tutorial section contains complete source code for all examples in the entire tutorial series, plus exercises and exercise solutions for each topic.

For customized training related to JavaScript or Java, email hall@coreservlets.com

Marty is also available for consulting and development support

Taught by lead author of Core Servlets & JSP, co-author of Core JSF (4th Ed), and this tutorial.

Available at public venues, or custom versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  - JavaScript, jQuery, Ext JS, JSF 2.3, PrimeFaces, Java 8 programming, Spring Framework, Spring MVC, Android, GWT, custom mix of topics
  - Courses available in any state or country. Maryland/DC companies can also choose afternoon/evening courses.

- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Hadoop, Hibernate/JPA, HTML5, RESTful Web Services

Contact hall@coreservlets.com for details
Topics in This Section

- Conditionals
- Loops
- The Math class
- Embedding JavaScript in HTML
- HTML versions

For additional materials, please see http://www.coreservlets.com/. The JavaScript tutorial section contains complete source code for all examples in the entire tutorial series, plus exercises and exercise solutions for each topic.
If Statements: One or Two Options

- Single option
  ```javascript
  if (condition) {
    statement1;
    ...
    statementN;
  }
  ```

- Two options
  ```javascript
  if (condition) {
    ...
  } else {
    ...
  }
  ```

JavaScript has a liberal definition of what condition is "false" (fails the test)
- "false": false, null, undefined, "" (empty string), 0, NaN
- "true": anything else (including the string "false")

If Statements: More than Two Options

- Multiple options
  ```javascript
  if (condition) {
    ...
  } else if (condition) {
    ...
  } else if (condition) {
    ...
  } else {
    ...
  }
  ```
**Boolean Operators**

- **==, !=**
  - Equality, inequality.

- **===**
  - In addition to comparing primitive types, === tests if two objects are identical (the same object), not just if they appear equal (have the same fields). More details when we introduce objects. Not used frequently.

- `<, <=, >, >=`
  - Numeric less than, less than or equal to, greater than, greater than or equal to.

- **&&, ||**
  - Logical AND, OR. Both use short-circuit evaluation to more efficiently compute the results of complicated expressions.

- **!**
  - Logical negation.

---

**If Statements: Example 1**

```javascript
function flipCoin() {
  if (Math.random() < 0.5) {
    return("Heads");
  } else {
    return("Tails");
  }
}
```

Math.random() returns a number between 0.0 (inclusive) and 1.0 (exclusive). If the random number generator is good, the values should be evenly distributed and unpredictable.
If Statements: Example 2

```javascript
function max(n1, n2) {
    if (n1 >= n2) {
        return(n1);
    } else {
        return(n2);
    }
}
```

Switch Statements

```javascript
function dayName(dayNumber) {
    var dayName;
    switch(dayNumber) {
        case 0: dayName = "Sunday"; break;
        case 1: dayName = "Monday"; break;
        case 2: dayName = "Tuesday"; break;
        case 3: dayName = "Wednesday"; break;
        case 4: dayName = "Thursday"; break;
        case 5: dayName = "Friday"; break;
        case 6: dayName = "Saturday"; break;
        default: dayName = "Invalid Day"; break;
    }
    return(dayName);
}
```
?: Operator (Ternary Operator)

- **Idea**
  - Quick way to return one of two choices based on a value

- **Syntax**
  - booleanTest ? thenValue : elseValue

- **Example: assigning to variable**
  ```javascript
  var coin = (Math.random() > 0.5) ? "Heads" : "Tails";
  coin => "Heads";  // Or possibly "Tails"
  ```

- **Example: returning from function**
  ```javascript
  function flipCoin2() {
    return((Math.random() > 0.5) ? "Heads" : "Tails");
  }
  flipCoin2(); => "Tails"
  flipCoin2(); => "Heads"
  ```

**Loops**

For additional materials, please see http://www.coreservlets.com/. The JavaScript tutorial section contains complete source code for all examples in the entire tutorial series, plus exercises and exercise solutions for each topic.
Looping Constructs: Most Common

- **for**
  ```javascript
  for(init; continueTest; updateOp) {
    body;
  }
  ```

- **while**
  ```javascript
  while(continueTest) {
    body;
  }
  ```

Quick Aside: console.log

- **Idea**
  - Prints output to the Firebug or Chrome console window
  - Used for testing and practice, not for real applications
    - Because end users will not know about the console window

- **Simple values**
  - `console.log("test");`
  - `console.log(1, 2, 3, "test");`

- **Placeholders**
  - `console.log("Sum of %o and %o is %o", x, y, x+y);`
    - You can also use %d and %s for more control of format, see docs below

- **More details**
  - `http://getfirebug.com/logging`
Example: for Loop

```javascript
function count(n) {
    for(var i=0; i<n; i++) {
        console.log('i is %o', i);
    }
}
```

Example: while Loop

```javascript
function powersOf2(max) {
    var num = 2;
    while(num < max) {
        console.log('num is %o', num);
        num = num * 2;
    }
}
```
Looping Constructs: Less Used

• do
  
do {
    body;
} while (continueTest);

  – Similar to the normal while loop except test is at the end instead of the beginning

Looping Constructs: Less Used

• for/in
  
  for(var property in someObject) {
    body;
  }

  – On surface, looks similar to for/each loop from Java and other languages, but
    • For arrays, values are array indexes, not array values
      – Use this loop for objects (to see property names), not arrays!
        Fails with extended arrays that have additional properties other than the indices.
    • For objects, values are the property names
      var person = { firstName: "Brendan", lastName: "Eich"};
      for(var property in person) {
        doSomethingWith(person[property]);
      }

Code Formatting

Big Ideas

• Indentation matters
  – Developers (including you!) that later modify and update your code will not count the curly braces, they will look at the indentation
  – Code with improper indentation is dramatically harder to understand and thus edit

• Some things are generally agreed upon
  – Indent whenever you start a new block with {
  – Return to previous indentation when you end a block with }

• Some things are open to debate
  – Where you put the {
  – How many space to indent
### Blocks that are Nested More Should be Indented More

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>blah;</td>
<td>blah;</td>
</tr>
<tr>
<td>blah; for(...) {</td>
<td>blah; for(...) {</td>
</tr>
<tr>
<td>blah;</td>
<td>blah;</td>
</tr>
<tr>
<td>blah; for(...) {</td>
<td>blah;</td>
</tr>
<tr>
<td>blah;</td>
<td>blah;</td>
</tr>
<tr>
<td>blah;</td>
<td>blah;</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>

### Blocks that are Nested the Same Should be Indented the Same

<table>
<thead>
<tr>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>blah;</td>
<td>blah;</td>
</tr>
<tr>
<td>blah; for(...) {</td>
<td>blah; for(...) {</td>
</tr>
<tr>
<td>blah;</td>
<td>blah;</td>
</tr>
<tr>
<td>blah; for(...) {</td>
<td>blah;</td>
</tr>
<tr>
<td>blah;</td>
<td>blah;</td>
</tr>
<tr>
<td>blah;</td>
<td>blah;</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>
Number of Spaces and Placement of Braces is Matter of Taste

OK
blah;
blah;
for(...) {
  blah;
  blah;
  for(...) {
    blah;
    blah;
  }
}

OK
blah;
blah;
for(...) {
  blah;
  blah;
  for(...) {
    blah;
    blah;
  }
}

OK
blah;
blah;
for(...) {
  blah;
  blah;
}

For additional materials, please see http://www.coreservlets.com/. The JavaScript tutorial section contains complete source code for all examples in the entire tutorial series, plus exercises and exercise solutions for each topic.
The Math Class

- **Call methods on the class**
  - Math.cos(…), Math.random(…), etc.
    - This syntax is similar to “static methods” from Java and other languages
  - Uses scientific versions: logs are base e, trig functions are in radians (not degrees)

- **Functions**

- **Constants**

Example from Conditionals Section

```javascript
function flipCoin() {
    if (Math.random() < 0.5) {
        return("Heads");
    } else {
        return("Tails");
    }
}
```
Embedding JavaScript in HTML

Loading Scripts: Usual Approach

- **script tag with src (in head section of HTML page)**
  
  `<script src="my-script.js" type="text/javascript"></script>`

- **Purpose**
  - To define functions, objects, and variables
  - Functions will later be triggered by buttons, other user events, inline script tags with body content, etc.

- **HTML5 note**
  - The type attribute is optional and will be omitted in future examples

- **Best practice**
  - Use subfolder for the JavaScript files (just as with images and CSS files)
    
    `<script src="scripts/my-script.js"></script>`
  - Some prefer the equivalent version prefaced with "/" 
    
    `<script src="/scripts/my-script.js"></script>`
Loading Scripts: Alternative Approach

- script tag with body content (in body section of HTML page)

  `<script>JavaScript code that uses document.write(...)</script>`

- Purpose
  - To directly invoke code that will run as page loads
    - E.g., to output HTML content built by JavaScript using `document.write`
  - Don’t use this approach for defining functions
    - Slower (no browser caching) and less reusable

Example (phish.js)

```
function getMessage() {
    var amount = Math.round(Math.random() * 100000);
    var message =
        "You won $" + amount + "!\n"
        "To collect your winnings, send your credit card\n"
        "and bank details to oil-minister@phisher.com.";
    return(message);
}

function showWinnings1() {
    alert(getMessage());
}

function showWinnings2() {
    document.write("<h1>" + getMessage() + "</h1>"进修);
}
```

"alert" pops up dialog box
"document.write" inserts text into page at location that it is called
Example (phish.html)

```html
<!DOCTYPE html><html>
<head><title>Loading Scripts</title>
...
<script src="scripts/phish.js"></script>
</head>
<body>
...
<input type="button" value="How Much Did You Win?"
onclick='showWinnings1()' />
...
<script>showWinnings2()</script>
...
</body></html>
```

**Example (Results)**

When page first comes up

![Image of a webpage showing a message: "You won $79868! To collect your winnings, send your credit card details to oil-minister@phisher.com." and an option to show winnings1()]

After pressing “How Much Did You Win” push button

![Image of a webpage showing a dialog box with the message: "You won $4067. To collect your winnings, send your credit card and bank details to oil-minister@phisher.com." and an option to show winnings2()]

Since page uses HTML5 syntax, the “type” attribute is omitted from both script tags.
Loading Scripts: Special Cases

- **Internet Explorer bug**
  - In some IE versions, scripts with src fail to load if you use `<script.../>`.
  - You must use `<script src="..." ..."></script>`

- **XHTML: script tag with body content**
  - It is an error if body of the script contains special XML characters such as & or <
  - E.g.
    `<script...>if (a<b) { this(); } else { that(); }</script>`
  - So, use CDATA section unless body content is simple and clearly has no special characters
    `<script type="text/javascript"><![CDATA[
    JavaScript Code
    ]]></script>`

For additional materials, please see http://www.coreservlets.com/. The JavaScript tutorial section contains complete source code for all examples in the entire tutorial series, plus exercises and exercise solutions for each topic.
**Summary**

- **XHTML**
  - Previously the most common HTML version used with Ajax apps or Dynamic HTML apps (JavaScript apps that manipulate the DOM)
  - Follows XML syntax, lowercase tags
- **HTML 5 (sort of)**
  - Growing in popularity for Ajax or DHTML apps.
  - Version most commonly used now is basically XHTML but with a simpler DOCTYPE and `<html>` start tag
    - Doesn't matter if browser fully supports HTML 5
- **HTML 4**
  - Still somewhat common in non-JavaScript apps
  - Not recommended for Ajax apps

---

### XHTML

**Summary**

- Follows XML syntax. Lowercase tags, end tags required, quotes around attribute values.

**Basic structure**

```xml
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>...</title></head>
<body>...</body></html>
```
### XHTML

**Pros**
- Code corresponds very directly to internal (DOM) representation by the browser

**Cons**
- DOCTYPE and `<html>` start tag are long and tedious

### Pseudo-HTML 5

**Summary**
- Follows XML syntax. Real HTML5 lets you violate rules of XML, but idea is to use XHTML (transitional) syntax but with simpler DOCTYPE, `<html>` tag, `<script>` tag, and `<link>` (stylesheet) tag

**Basic structure**

```html
<!DOCTYPE html>
<html>
<head><title>...</title></head>
<body>...</body>
</html>
```
### Pseudo-HTML 5

**Pros**
- Code corresponds very directly to internal (DOM) representation by the browser
- Simple DOCTYPE and `<html>` start tag
- Can omit “type” attribute when using script tags

**Cons**
- Your code might not always be strictly compliant with spec. May get warnings from formal validators, especially with non-CSS formatting
  - E.g., `<table border="1"` is not strictly legal HTML 5, but widely used because it is hard to reproduce this effect across browsers in a consistent manner with only CSS

### HTML 4

**Summary**
- Does not follow XML syntax. Tags not case sensitive. End tags and quotes on attribute values sometimes optional.

**Basic structure**
```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head><title>…</title></head>
<body> … </body></html>
```
**HTML 4**

- **Pros**
  - Simple code
  - Still somewhat used in non-Ajax apps

- **Cons**
  - Source code and internal browser representation can be substantially different, requiring mental translation when thinking of how to manipulate DOM from JavaScript

---

**Wrap-up**
Summary

• **Basic JavaScript syntax**
  – Most common conditional
    
    ```javascript
    if (someCondition) {
        doThis();
    } else {
        doThat();
    }
    ```
  – Most common loop
    
    ```javascript
    for(var i=0; i<someCutoff; i++) { doSomeStuff(); }
    ```
  – See methods in Math class. Use Math.random() for practice.

• **Embedding in browser**
  
  ```html
  <script src="scripts/some-script.js"></script>
  ```
  
  – Use pseudo-HTML 5 or possibly XHTML for the HTML