Document Object Model and Javascript

Musadaq Mansoor

Lecture 8

- Document Object Model
- DOM Structure
- Accessing Values Using Dot Notation
- Properties of Document Object
- Methods of Document Object
- Forms Collection
- Images Collection
- DOM Objects Classification
- Introduction to JavaScript
- JavaScript Placement in Document
- Variables in JavaScript
- Operators in JavaScript

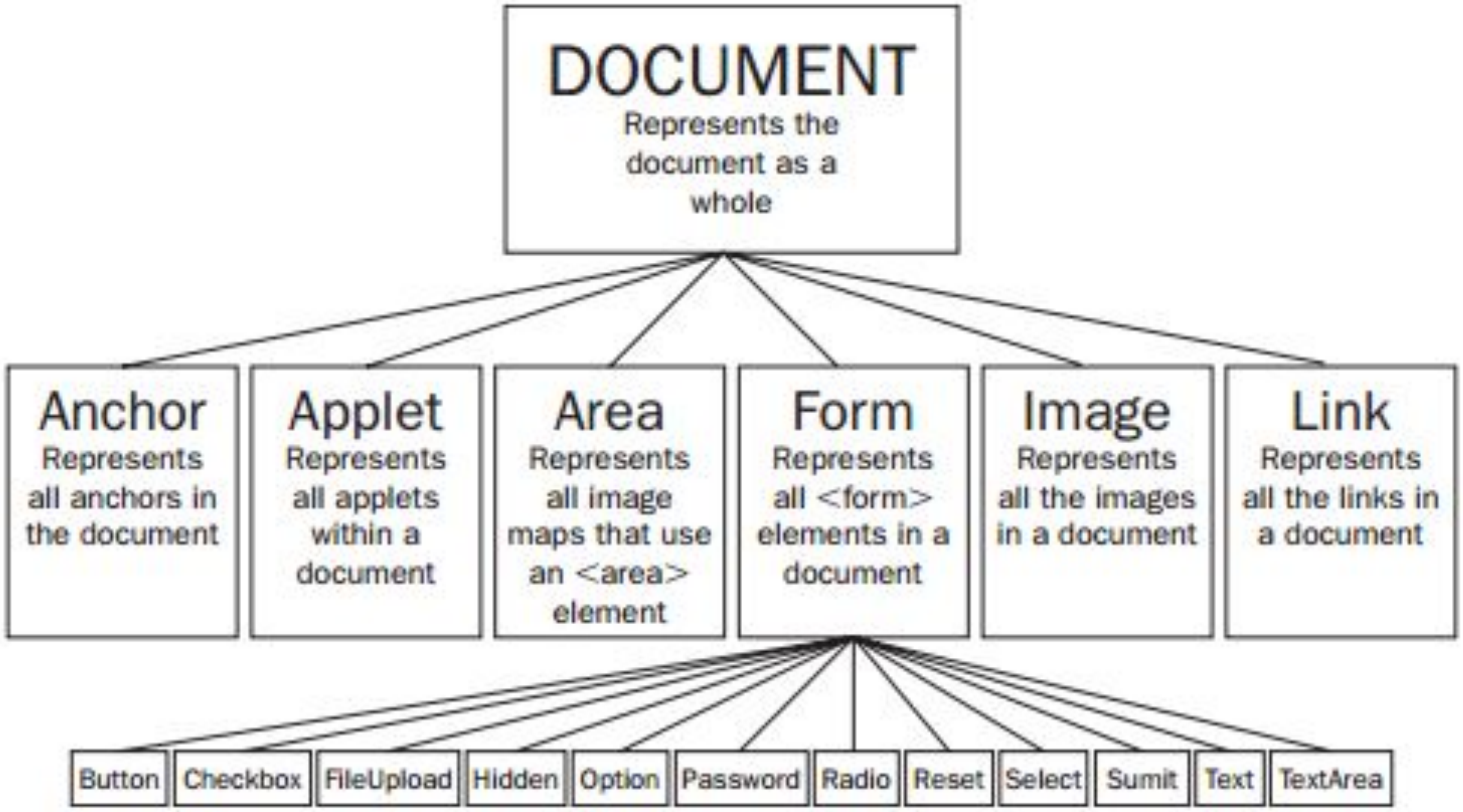
Document Object Model

- When the browser loads a page, it stores it in an electronic form that programmers can then access through something known as an **interface**
- The **interface** is a little like a predefined set of questions and commands. For example, you can ask questions like:
  - What is the title of the page?
  - What is the third item in the bulleted list whose id attribute has a value of ToDoList ?
  - What is the URL of the page in the first link on the page?
- Use commands to tell the browser to change some of these values, or even add new elements into the page.
- The interface that works with web pages is called the Document Object Model
- A document object has properties that describe the background color of the web page or the title of the page.

Document Object Model

- Document Object Model as an interface between the browser and the programming language,
  - you can compare it to a remote control that acts as the interface between your TV and you.
- When working with the DOM, it does not matter what language you program with. As long as you use the right properties and methods, the effect will be the same.

DOM Structure


DOCUMENT
Represents the document as a whole
Anchor
Represents all anchors in the document
Applet
Represents all applets within a document
Area
Represents all image maps that use an <area> element
Form
Represents all <form> elements in a document
Image
Represents all the images in a document
Link
Represents all the links in a document
Button
Checkbox
FileUpload
Hidden
Option
Password
Radio
Reset
Select
Sumit
Text
TextArea

Accessing Values Using Dot Notation

- The DOM would allow a script to access:
  - The content of the form as part of the forms collection
  - The links as part of the links collection
- To access any of the properties, again you use dot notation, so you can access the title of a document like so:

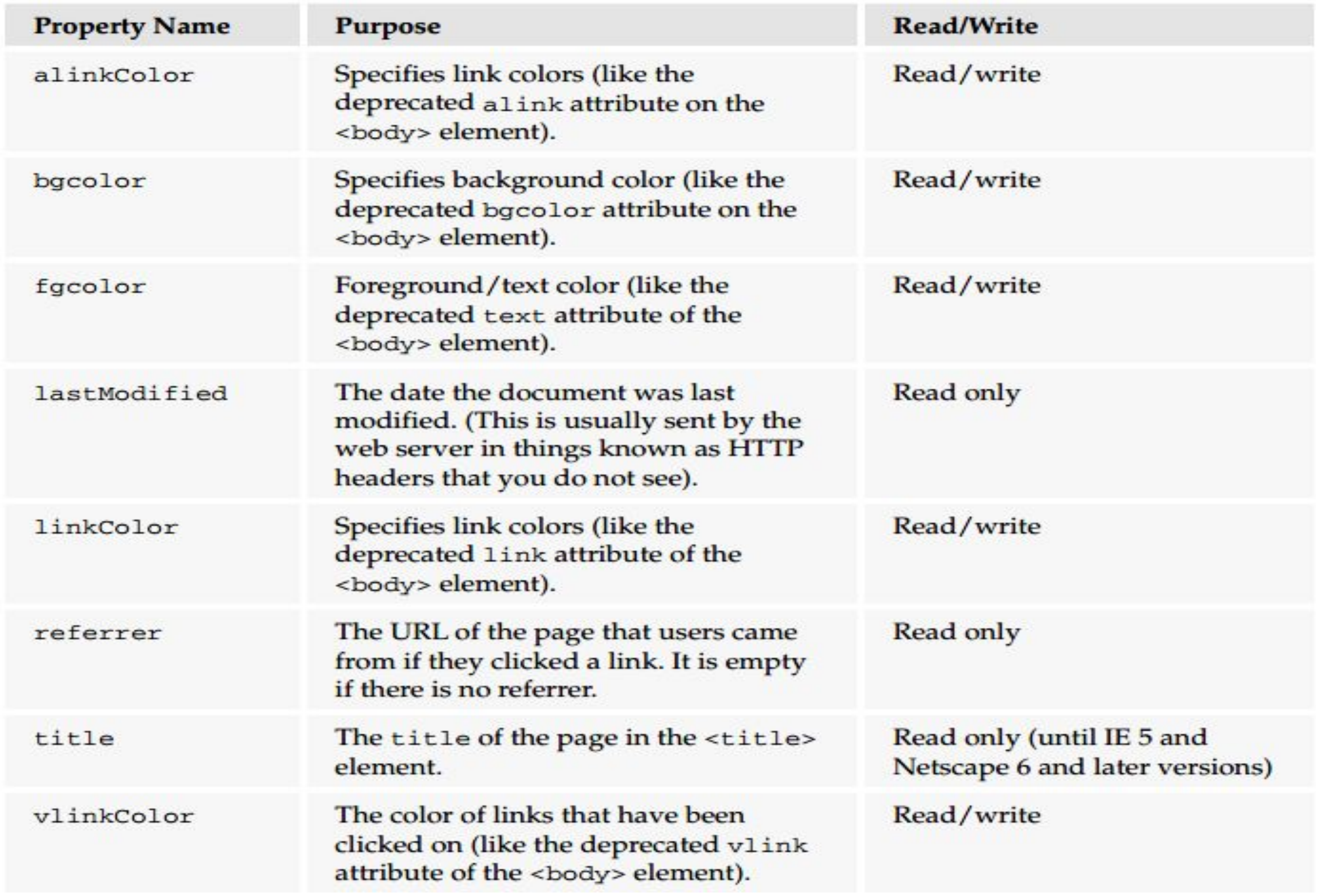
  document.title

Or

- You could find out the date a document was last modified like so:

  document.lastModified

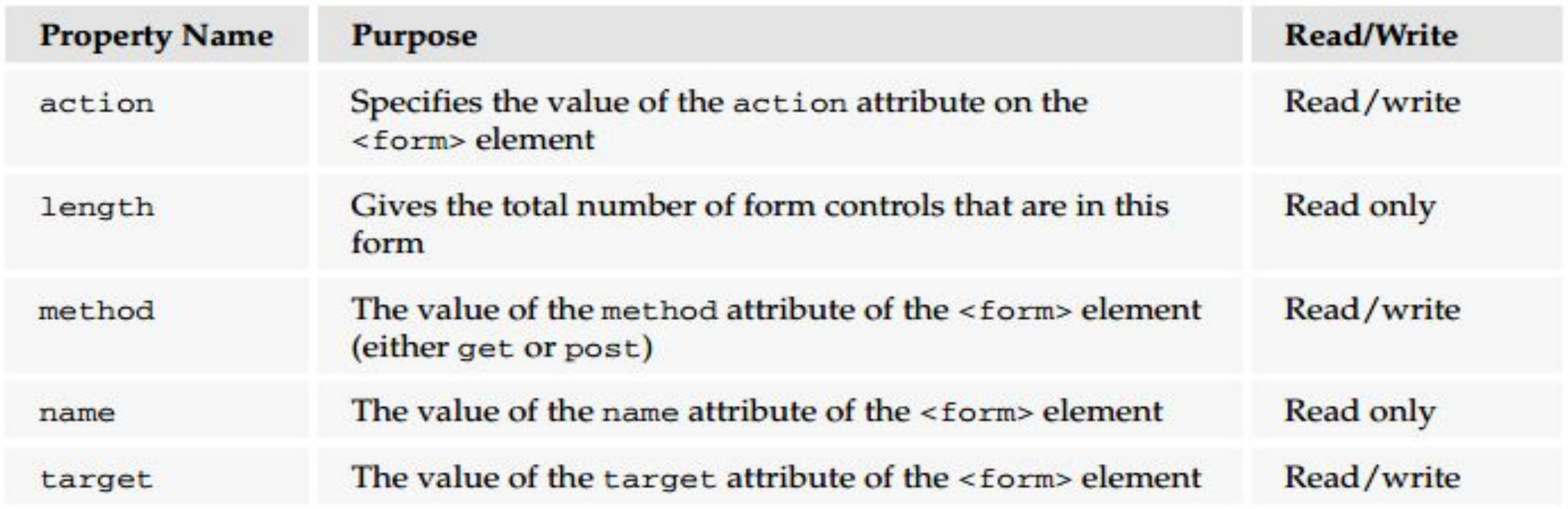
Properties of Document Object



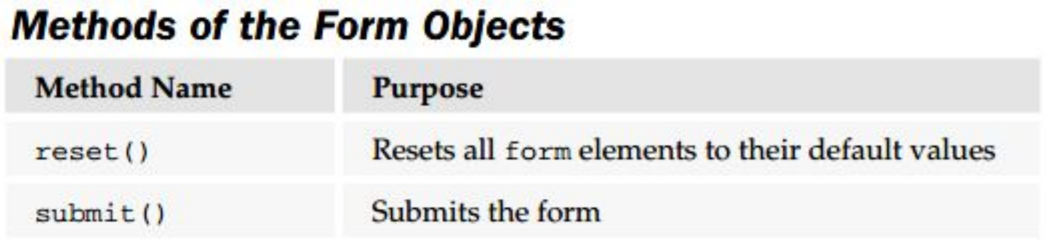
Property Name	Purpose	Read/Write
`alinkColor`	Specifies link colors (like the deprecated `alink` attribute on the `<body>` element).	Read/write
`bgcolor`	Specifies background color (like the deprecated `bgcolor` attribute on the `<body>` element).	Read/write
`fgcolor`	Foreground/text color (like the deprecated `text` attribute of the `<body>` element).	Read/write
`lastModified`	The date the document was last modified. (This is usually sent by the web server in things known as HTTP headers that you do not see).	Read only
`linkColor`	Specifies link colors (like the deprecated `link` attribute of the `<body>` element).	Read/write
`referrer`	The URL of the page that users came from if they clicked a link. It is empty if there is no referrer.	Read only
`title`	The `title` of the page in the `<title>` element.	Read only (until IE 5 and Netscape 6 and later versions)
`vlinkColor`	The color of links that have been clicked on (like the deprecated `vlink` attribute of the `<body>` element).	Read/write

Forms Collection

- Holds references corresponding to each of the < form > elements in the page
- DOM deals with this by having a separate form object to represent each of the individual forms
  - document.forms[0].action
  - document.frmLogin.action
- **Properties of the Form Objects**



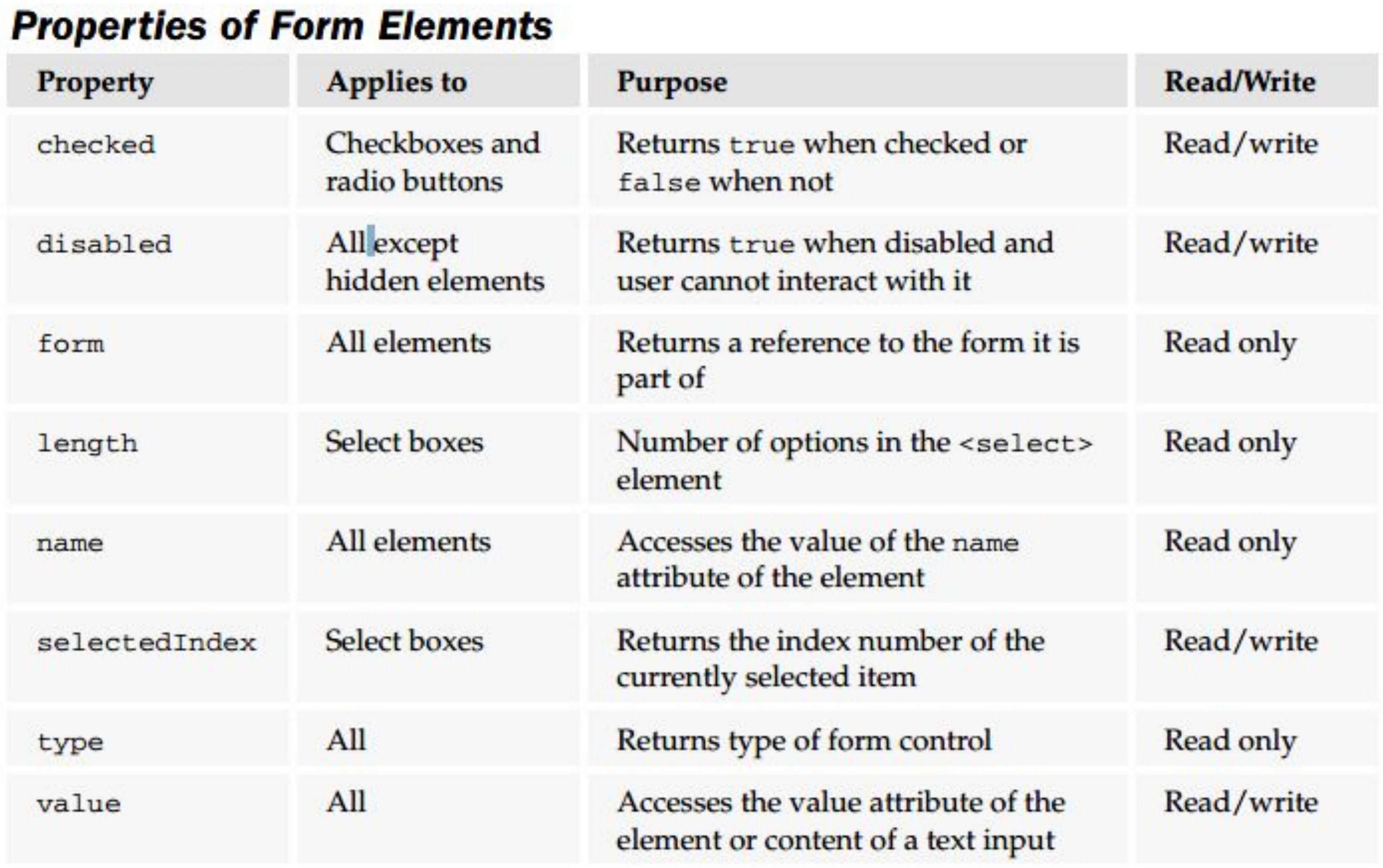
Property Name	Purpose	Read/Write
`action`	Specifies the value of the `action` attribute on the `<form>` element	Read/write
`length`	Gives the total number of form controls that are in this form	Read only
`method`	The value of the `method` attribute of the `<form>` element (either `get` or `post`)	Read/write
`name`	The value of the `name` attribute of the `<form>` element	Read only
`target`	The value of the `target` attribute of the `<form>` element	Read/write



Methods of the Form Objects

Method Name	Purpose
`reset()`	Resets all `form` elements to their default values
`submit()`	Submits the form

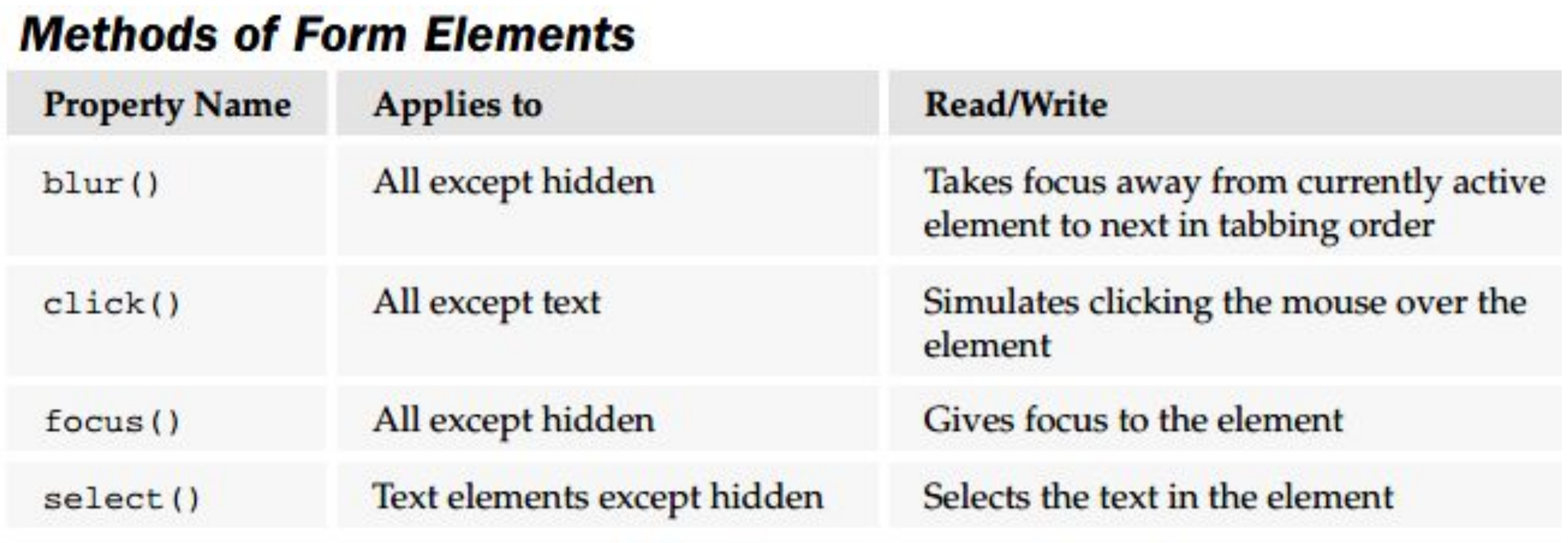
Forms Collection



Properties of Form Elements

Property	Applies to	Purpose	Read/Write
`checked`	Checkboxes and radio buttons	Returns `true` when checked or `false` when not	Read/write
`disabled`	All except hidden elements	Returns `true` when disabled and user cannot interact with it	Read/write
`form`	All elements	Returns a reference to the form it is part of	Read only
`length`	Select boxes	Number of options in the `<select>` element	Read only
`name`	All elements	Accesses the value of the `name` attribute of the element	Read only
`selectedIndex`	Select boxes	Returns the index number of the currently selected item	Read/write
`type`	All	Returns type of form control	Read only
`value`	All	Accesses the value attribute of the element or content of a text input	Read/write

Forms Collection



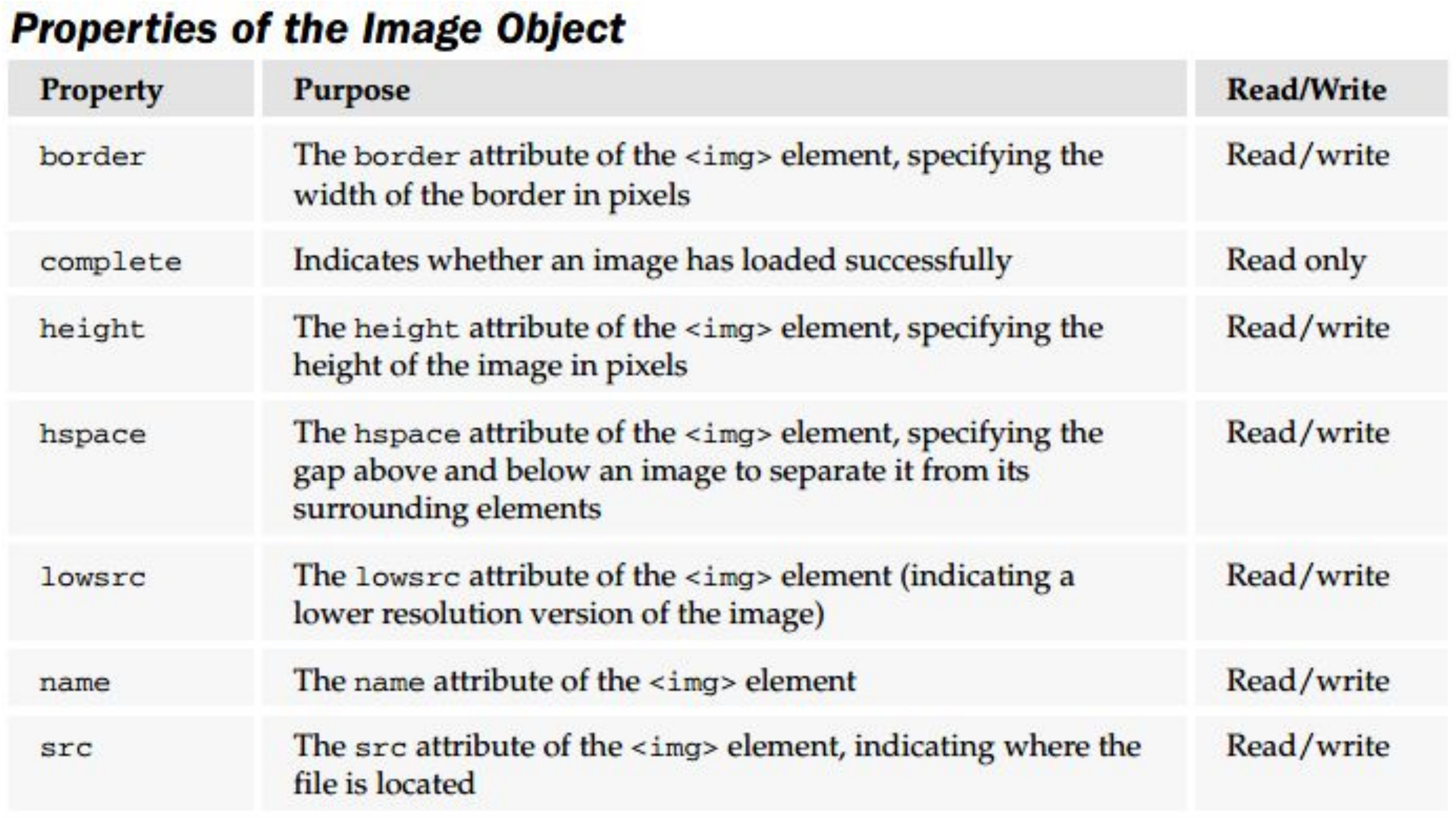
Methods of Form Elements

Property Name	Applies to	Read/Write
`blur()`	All except hidden	Takes focus away from currently active element to next in tabbing order
`click()`	All except text	Simulates clicking the mouse over the element
`focus()`	All except hidden	Gives focus to the element
`select()`	Text elements except hidden	Selects the text in the element

Images Collection

- Provides references to image objects, one representing each image in a document.
- The src attribute of the first image can be found using the index number like so:

  document.images[0].src



Properties of the Image Object

Property	Purpose	Read/Write
`border`	The `border` attribute of the `<img>` element, specifying the width of the border in pixels	Read/write
`complete`	Indicates whether an image has loaded successfully	Read only
`height`	The `height` attribute of the `<img>` element, specifying the height of the image in pixels	Read/write
`hspace`	The `hspace` attribute of the `<img>` element, specifying the gap above and below an image to separate it from its surrounding elements	Read/write
`lowsrc`	The `lowsrc` attribute of the `<img>` element (indicating a lower resolution version of the image)	Read/write
`name`	The `name` attribute of the `<img>` element	Read/write
`src`	The `src` attribute of the `<img>` element, indicating where the file is located	Read/write

Introduction to Javascript

- JavaScript gives web developers a programming language to use in web pages that allows them to perform tasks such as the following:
  - Read elements from documents and write new elements and text into documents
  - Manipulate or move text
  - Perform mathematical calculations on data
  - React to events, such as a user clicking a button
  - Retrieve the current date and time from a user ’ s computer or the last time a document was modified
  - Determine the user ’ s screen size, browser version, or screen resolution
  - Perform actions based on conditions such as alerting users if they enter the wrong information into a form
- JavaScript is not the same as Java, which is a different programming language (although there are some similarities).

Javascript Placement

- JavaScript can either be embedded in a page or placed in an external script file
- To work in the browser, the browser must have JavaScript enabled
- You add scripts to your page inside the < script > element.
  - The type attribute on the opening < script > tag indicates what scripting language will be found inside the element
- There are several other scripting languages that do a very similar job to JavaScript (such as VBScript or Perl), but JavaScript is the main programming language used in web browsers.
- If you put script in the body of a page — as in below example — then it will run (or execute) as the page loads.
- The following code using the write() method to add a new line of text into the web page

```
< html > < body >
    < p >
< script type=”text/javascript” > document.write(“My first JavaScript”)
< /script >
< /p > < /body > < /html >
```

Javascript Placement

- Placing javascript in external files have the following advantages:
  - If your script is used by more than one page you do not need to repeat the script in each page that uses it.
  - If you want to update your script you need only change it in one place.
  - It makes the XHTML page cleaner and easier to read.
- You need to use the src attribute on the < script > element; the value of the src attribute should be an absolute or relative URL pointing to the file containing the JavaScript. For example:

  < script type=”JavaScript” src=”scripts/validation.js” > < /script >
- There are three places where you can put your JavaScripts:
  - **In the < head > of a page**: These scripts will be called when an event triggers them.
  - **In the < body > of a page**: These scripts will run as the page loads.
  - **In an external file**:
    - If the link is placed inside the < head > element, the script is treated the same as when the script lives inside the head of the document waiting for an event to trigger it,
    - if it is placed in the < body > element it will act like a script in the body section and execute as the page loads.

Comments in JavaScript

- Add comments to your JavaScript code in two ways:
  - **Single Line**: comment out anything on that line after the comment marks (//).
  - **Multiple Lines**: comment out multiple lines using the syntax, holding the comment between the opening characters /* and closing characters */

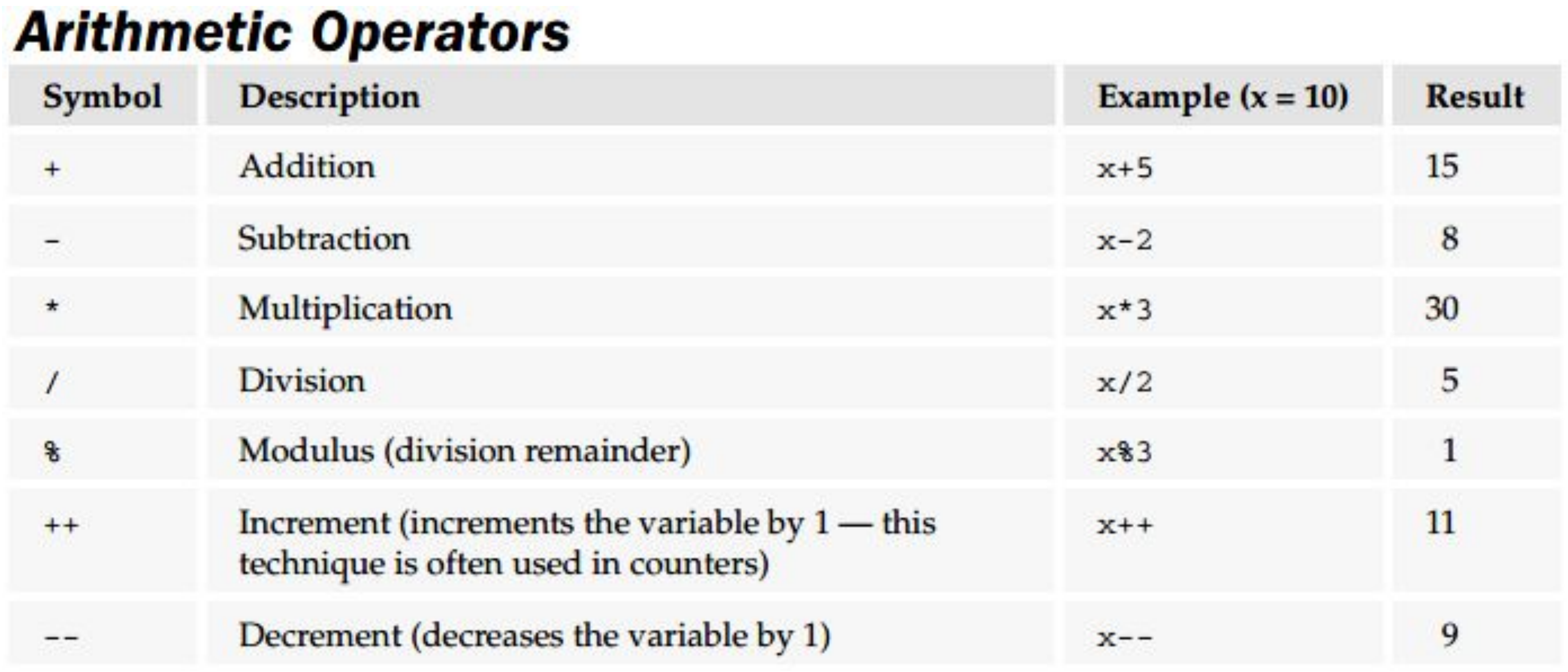
- **The < noscript >Element**
  - The < noscript > element offers alternative content for users who have disabled JavaScript.
  - It can contain any XHTML content that the author wants to be seen in the browser if the user does not have JavaScript enabled.

Variables in Javascript

- Variables are used to store data.
- When you declare a variable in a function, it can be accessed only in that function called **local variables**
- Give the variable a name and put an equal sign between it and the value you want it to have.
  - example: var userName = “Bob Stewart”
- There are a few rules you must remember about variables in JavaScript:
  - They must begin with a letter or the underscore character.
  - Variable names are case - sensitive.
  - Avoid giving two variables the same name within the same document as one might override the value of the other, creating an error.
  - Do not call two variables the same name, but use different cases to distinguish them (e.g., username and UserName) as this is a common source of confusion later.
  - Try to use descriptive names for your variables. This makes your code easier to understand (and will help you debug your code if there is a problem with it).

Operators

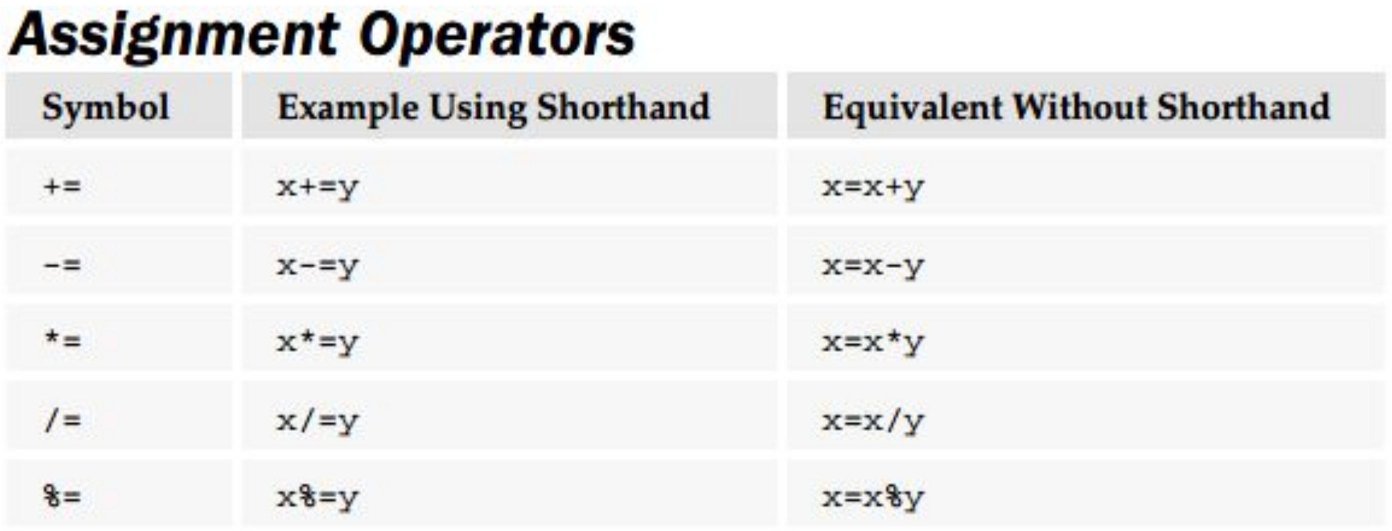
- The different types of operators you will see in this section are:
  - Arithmetic operators
  - Assignment operators
  - Comparison operators
  - Logical operators
  - String operators



Arithmetic Operators

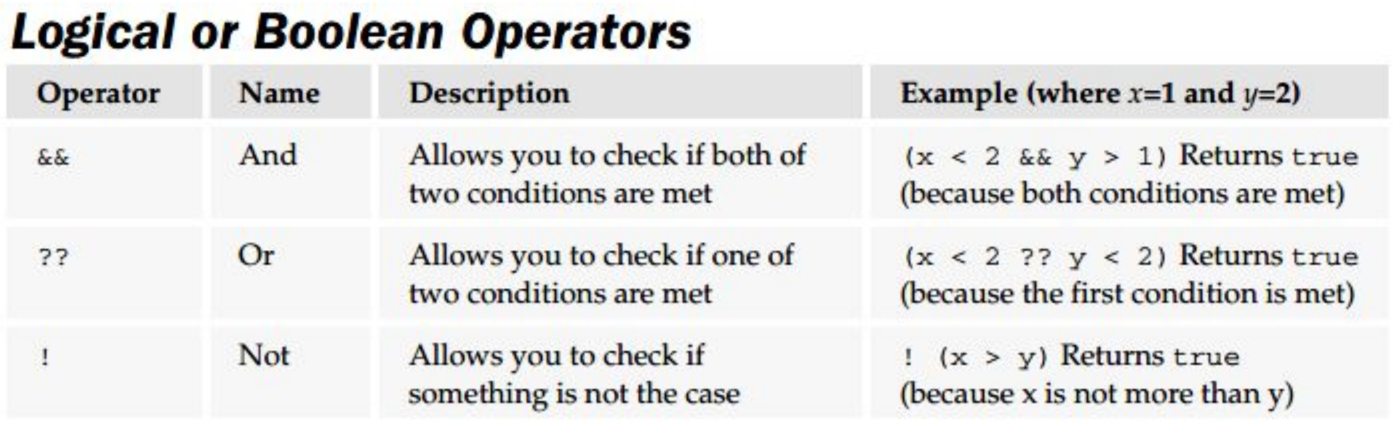
Symbol	Description	Example (x = 10)	Result
+	Addition	x+5	15
-	Subtraction	x-2	8
*	Multiplication	x*3	30
/	Division	x/2	5
%	Modulus (division remainder)	x%3	1
++	Increment (increments the variable by 1 — this technique is often used in counters)	x++	11
--	Decrement (decreases the variable by 1)	x--	9

Operators



Assignment Operators

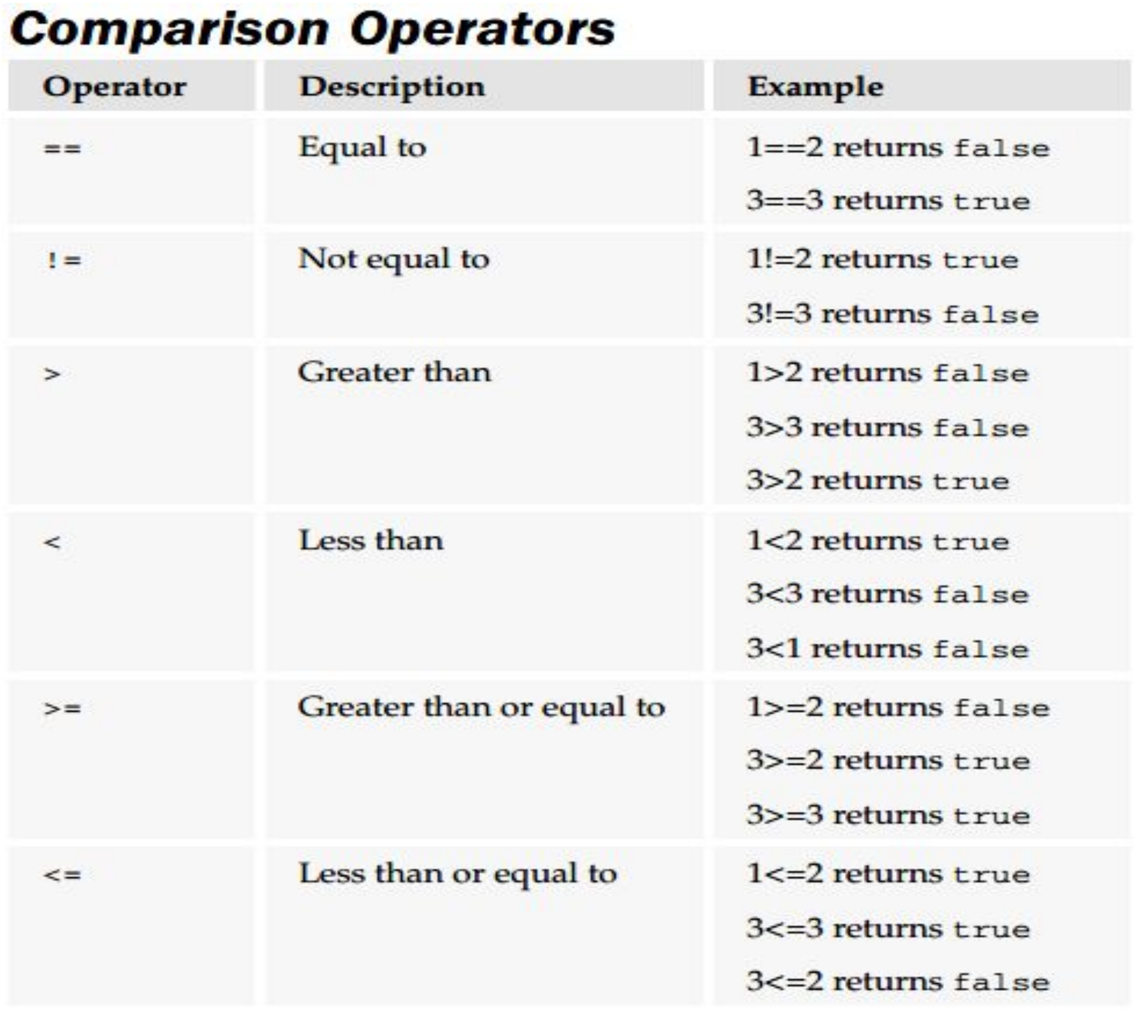
Symbol	Example Using Shorthand	Equivalent Without Shorthand
`+=`	`x+=y`	`x=x+y`
`-=`	`x-=y`	`x=x-y`
`*=`	`x*=y`	`x=x*y`
`/=`	`x/=y`	`x=x/y`
`%=`	`x%=y`	`x=x%y`



Logical or Boolean Operators

Operator	Name	Description	Example (where *x*=1 and *y*=2)
`&&`	And	Allows you to check if both of two conditions are met	`(x < 2 && y > 1)` Returns `true` (because both conditions are met)
`??`	Or	Allows you to check if one of two conditions are met	`(x < 2 ?? y < 2)` Returns `true` (because the first condition is met)
`!`	Not	Allows you to check if something is not the case	`! (x > y)` Returns `true` (because x is not more than y)

Operators



Comparison Operators

Operator	Description	Example
==	Equal to	1==2 returns false 3==3 returns true
!=	Not equal to	1!=2 returns true 3!=3 returns false
>	Greater than	1>2 returns false 3>3 returns false 3>2 returns true
<	Less than	1<2 returns true 3<3 returns false 3<1 returns false
>=	Greater than or equal to	1>=2 returns false 3>=2 returns true 3>=3 returns true
<=	Less than or equal to	1<=2 returns true 3<=3 returns true 3<=2 returns false

String Operators

You can also add text to strings using the + operator.
For example, here the + operator is being used to add two variables that are strings together:

firstName = “Bob”
lastName = “Stewart”
name = firstName + lastName

THANKS. ANY QUESTIONS?