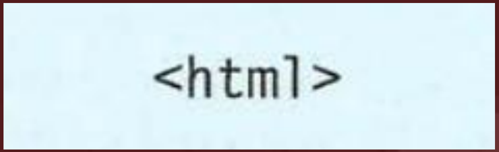
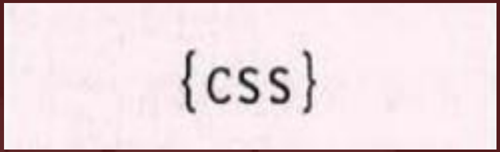
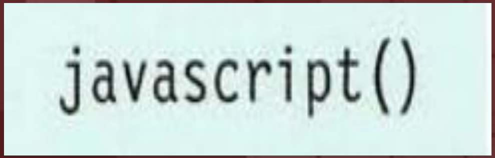
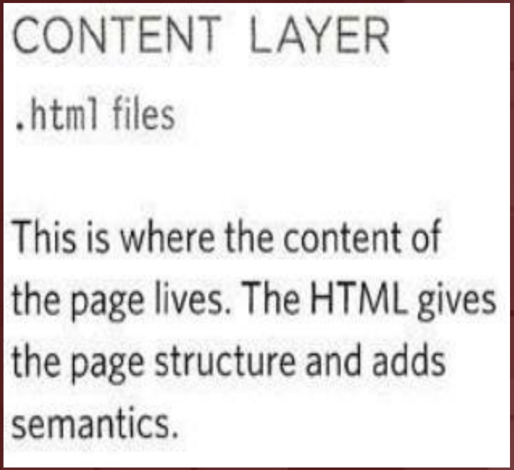
Web Technologies II

Web developers usually talk about three languages that are used to create web pages: HTML, CSS, and JavaScript. Where possible, aim to keep the three languages in separate files, with the HTML page linking to CSS and JavaScript files.


<html>

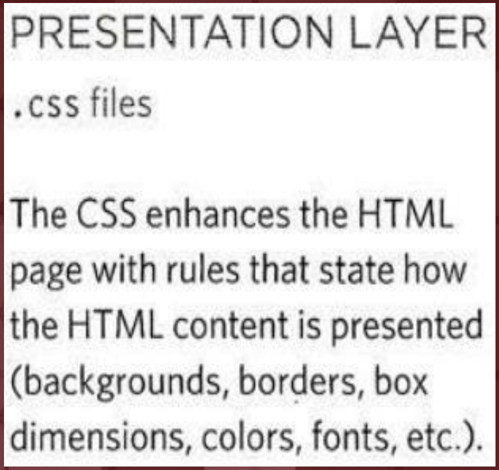

{css}


javascript()


CONTENT LAYER

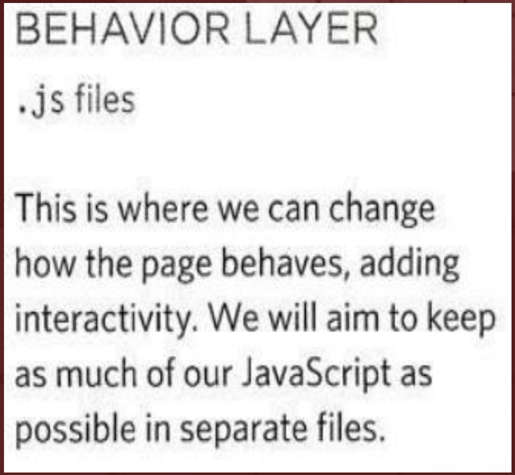
.html files

This is where the content of the page lives. The HTML gives the page structure and adds semantics.


PRESENTATION LAYER

.css files

The CSS enhances the HTML page with rules that state how the HTML content is presented (backgrounds, borders, box dimensions, colors, fonts, etc.).

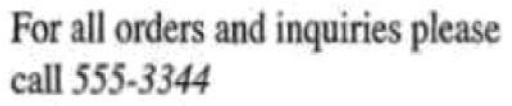

BEHAVIOR LAYER

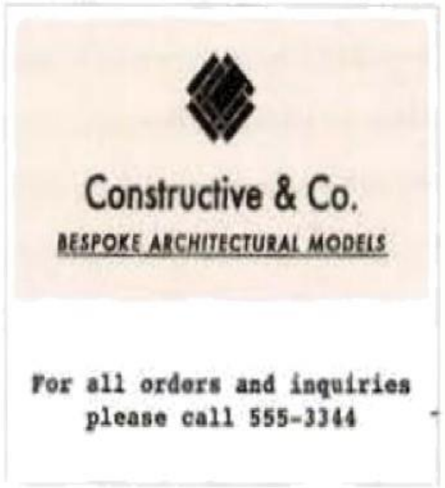
.js files

This is where we can change how the page behaves, adding interactivity. We will aim to keep as much of our JavaScript as possible in separate files.

Each language forms a separate layer with a different purpose. Each layer, from left to right,builds on the previous one.

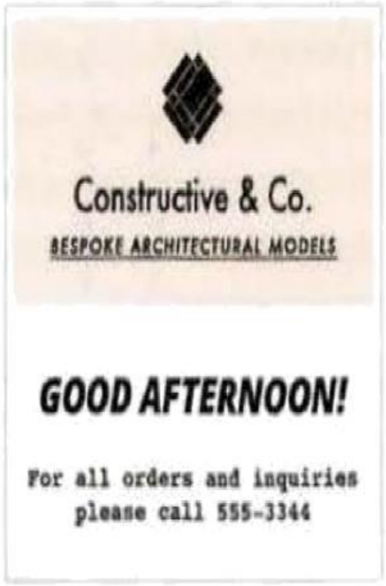
Constructive & Co.


For all orders and inquiries please call 555-3344


Constructive & Co.

BESPOKE ARCHITECTURAL MODELS

For all orders and inquiries please call 555-3344


Constructive & Co.

BESPOKE ARCHITECTURAL MODELS

GOOD AFTERNOON!

For all orders and inquiries please call 555-3344

HTML ONLY

Starting wik the HTMLlayer allows you tofocusonthe most tmportant thing about your site: its content.

HTNL*CSS

Adding the CtS rules in a separate file keeps rules regarding how the page looks away from the content itself.

HTML+CSS+JAVASCRIPT

The JavaScript is added last ardenhancestheusa§ility0f th‹page0rth#experieñC90f interacting with the site.

CREATING A BASIC JAVASCRIPT

A JavaScript file is just a text file (like HTML and CSS files are) but it has a **.js** file extension, so save this file with the name add-content . Js

```
var today= new Date();
var hourNow =
today.getHours(); var
greeting ;
if (hourNow > 18) {
greeting= 'Good
evening!'; else if
(hourNow > 12) {
greeting = ' Good
afternoon!'; else if
(hourNow > 0) {
greeting = 'Good morni
ng!'; else {
greeting = 'Welcome! ' ;
```

```
document. write( ' <h3>' +greeting + ' </ h3 > ');
```

When you want to use JavaScript with a web page, you use the HTML<script> element to tell the browser it is coming across a script.It **src** attribute tells people where the JavaScript file is stored.

```
<!DOCTYPE html>
<html>
<head>
<title> Constructive &amp; Co. </ title>
<link rel =" stylesheet " href = "css / c01.css" />
</ head>
<body>
<hl> Constructive &amp ; Co. </ hl>
<script src = "js / add-content.js">< / script >
<p> For all orders and inquiries please call
<em> SSS -33 44 </ em>< / p>
</ body>
</html>
```

When the browser comes across a <script> element, it stops to load the script and then checks to see if it needs to do anything.

Basics of JavaScript:

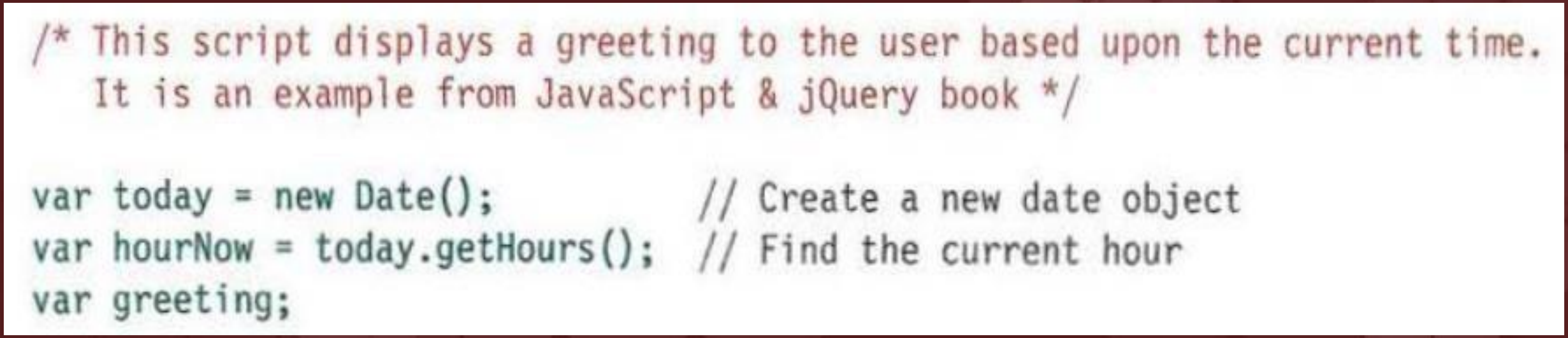
A script is a series of instructions that a computer can follow one-by-one. Each individual instruction or step is known as a statement. Statements should end with a semicolon.

```
var today= new Date{);
var hourNow = today.getHours{() ;
var greeting;
```

JAVASCRIPT IS CASE SENSITIVE!

Some statements are surrounded by curly braces;these are known as code blocks. The closing curly brace is not followed by a semicolon.

You should write comments to explain what your code does. They help make your code easier to read and understand. This can help you and others who read your code.



```
/* This script displays a greeting to the user based upon the current time.
   It is an example from JavaScript & jQuery book */

var today = new Date();           // Create a new date object
var hourNow = today.getHours();   // Find the current hour
var greeting;
```

To write a comment that stretches over more than one line, you use a multi - line comment, starting with the /* characters and ending with the */ characters.

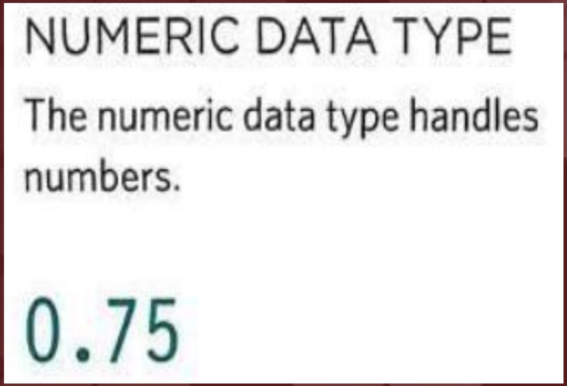
In a single-line comment, anything that follows the two forward slash characters // on that line will not be processed by the JavaScript interpreter. Single-line comments are often used for short descriptions of what the code is doing.

Variables:

A script will have to temporarily store the bits of information it needs to do it s job. It can store this data in variables Before you can use a variable you have to announce that you want to use it. This involves creating the variable and giving it a name. Programmers say that you declare a variable. Once you have created a variable you can tell what information you would like it to store for you. Programmers say that you assign a value to a variable

Data Types

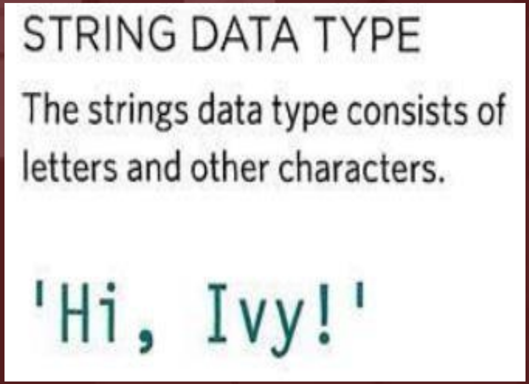
JavaScript distinguishes between numbers, strings and true or false values known as Booleans.



NUMERIC DATA TYPE

The numeric data type handles numbers.

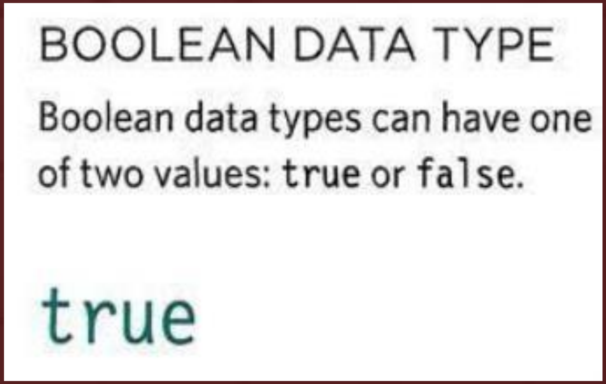
```
0.75
```



STRING DATA TYPE

The strings data type consists of letters and other characters.

```
'Hi, Ivy!'
```



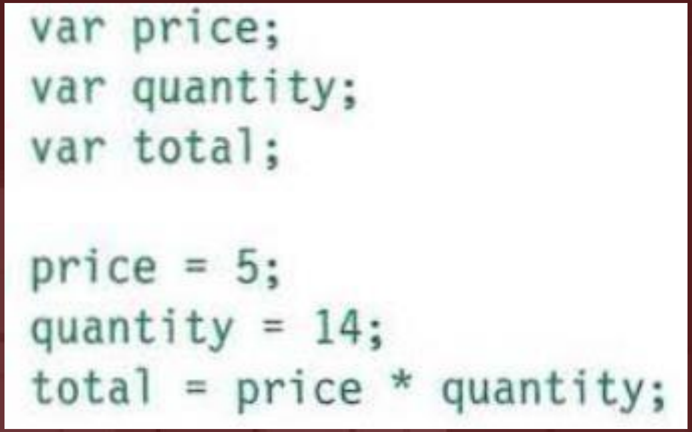
BOOLEAN DATA TYPE

Boolean data types can have one of two values: `true` or `false`.

```
true
```

JavaScript distinguishes between numbers, strings and true or false values known as Booleans.

USING A VARIABLE TO STORE A NUMBER



```
var price;
var quantity;
var total;

price = 5;
quantity = 14;
total = price * quantity;
```

Here, three variables are created and values are assigned to them.

- price holds the price of an individual tile
- quantity holds the number of tiles a customer wants
- total holds the total cost of the tiles

Note that the numbers are not written inside quotation marks.

USING A VARIABLE TO STORE A STRING

```
var username;
var message;
username = 'Molly';
message = 'See our upcoming range';
```

Two variables are declared (username and message), and they are used to hold strings. Note how the string is placed inside quote marks. The quotes can be single or double quotes, but they must match

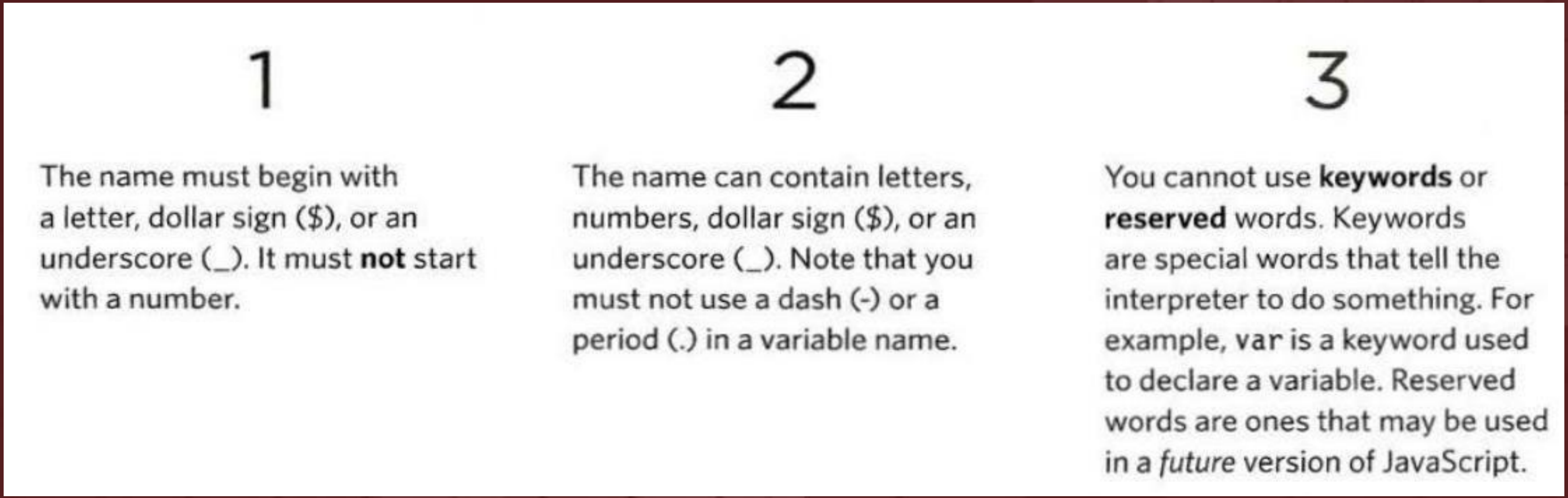
USING A VARIABLE TO STORE A BOOLEAN

```
var inStock;
var shipping;
inStock = true;
shipping = false;
```

A Boolean variable can only have a value of true or false, but this data type is very helpful.

RULES FOR NAMING VARIABLES

Here are six rules you must always follow when giving a variable a name:



1

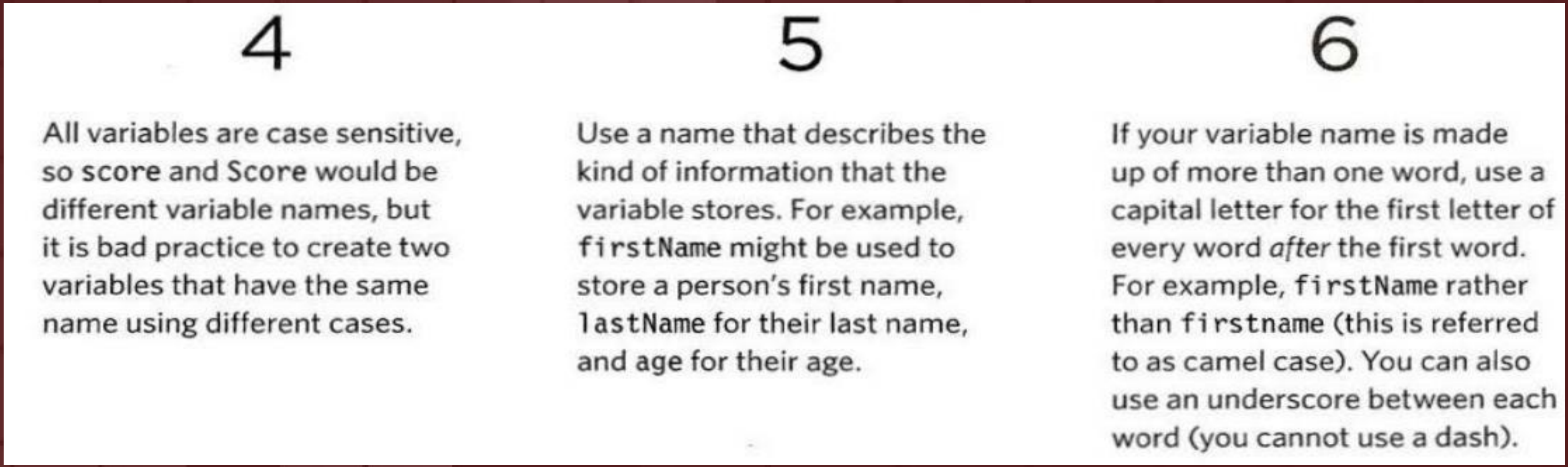
The name must begin with a letter, dollar sign ($), or an underscore (_). It must **not** start with a number.

2

The name can contain letters, numbers, dollar sign ($), or an underscore (_). Note that you must not use a dash (-) or a period (.) in a variable name.

3

You cannot use **keywords** or **reserved** words. Keywords are special words that tell the interpreter to do something. For example, `var` is a keyword used to declare a variable. Reserved words are ones that may be used in a *future* version of JavaScript.



4

All variables are case sensitive, so `score` and `Score` would be different variable names, but it is bad practice to create two variables that have the same name using different cases.

5

Use a name that describes the kind of information that the variable stores. For example, `firstName` might be used to store a person's first name, `lastName` for their last name, and `age` for their age.

6

If your variable name is made up of more than one word, use a capital letter for the first letter of every word *after* the first word. For example, `firstName` rather than `firstname` (this is referred to as camel case). You can also use an underscore between each word (you cannot use a dash).

Arrays:

An array is a special type of variable. It doesn't just store one value; it stores a list of values.

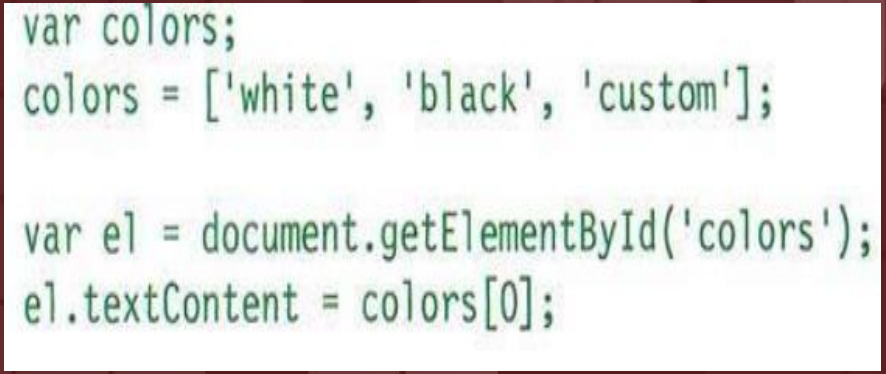
You should consider using an array whenever you are working with a list or a set of values that are related to each other.

Arrays are especially helpful when you do not know how many items a list will contain because, when you create the array, you do not need to specify how many values it will hold.

Creating an Array

You create an array and give it a name just like you would any other variable (using the var keyword followed by the name of the array).

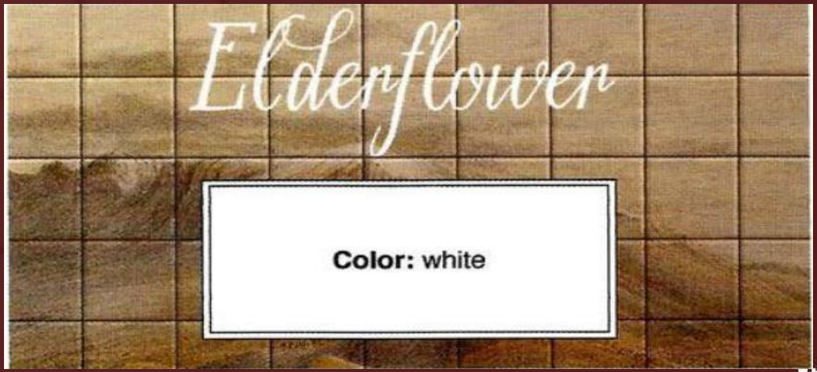
JavaScript :



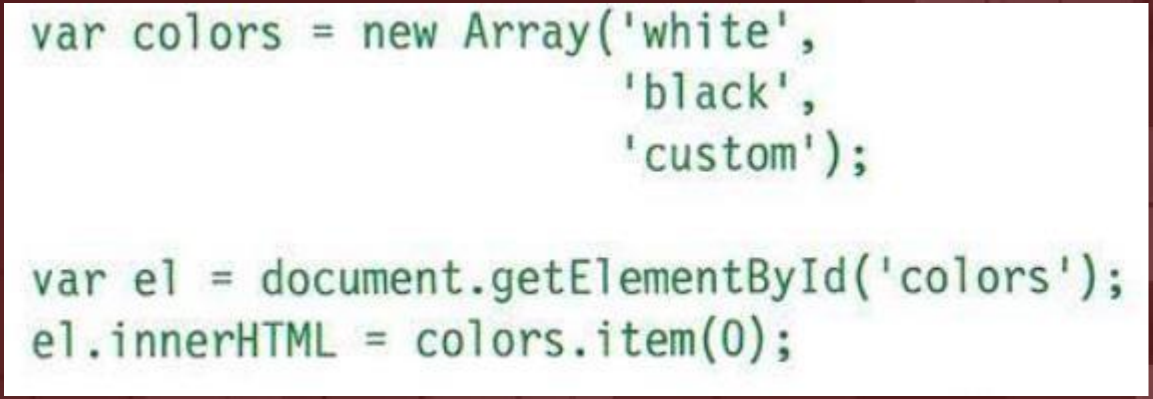
```
var colors;
colors = ['white', 'black', 'custom'];

var el = document.getElementById('colors');
el.textContent = colors[0];
```

output:



The values are assigned to the array inside a pair of square brackets, and each value is separated by a comma. This technique for creating an array is known as an array literal . It is usually the preferred method for creating an array.



```
var colors = new Array('white',
                       'black',
                       'custom');

var el = document.getElementById('colors');
el.innerHTML = colors.item(0);
```

On the left, you can see an array created using a different technique called an array constructor. This uses the new keyword followed by Array();The values are then specified in parentheses (not square brackets), and each value is separated by a comma. You can also use a method called item() to retrieve data from the array.

Values in an array are accessed as if they are in a numbered list. It is important to know that the numbering of this list starts at zero (not one).

Each item in an array is automatically given a number called an index. Consider the following array which holds three color s:

```
var colors;
colors = ['white',
          'black',
          'custom'];
```

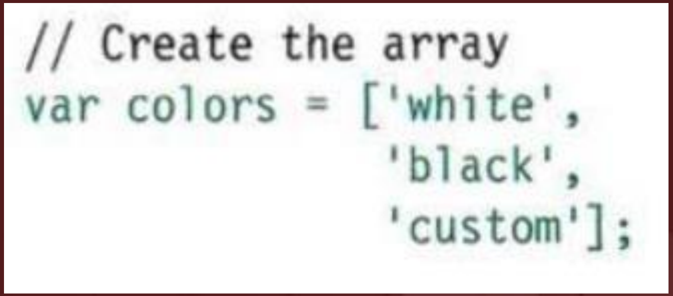
Confusingly, index values start at 0 (not 1). To retrieve the third item on the list, the array name is specified along with the index number in square brackets.

```
var itemThree;
itemThree = colors[2];
```

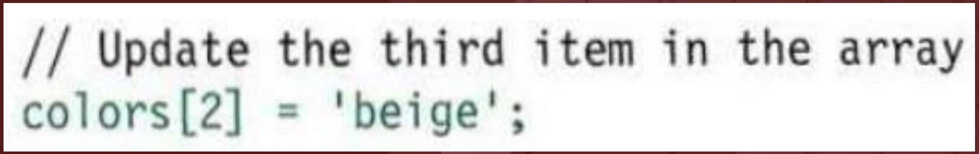
Each array has a property called **length**, which holds the number of items in the array. Below you can see that a variable called numColors is declared. Its value is set to be the number of the items in the array. The name of the array is followed by a period symbol (or full stop) which is then followed by the length keyword.

```
var numColors;
numColors = colors.length;
```

You can change the value of an item an array by selecting it and as signing it a new value just as you would any other variable (using the equals sign and the new value for that item)



```
// Create the array
var colors = ['white',
              'black',
              'custom'];
```



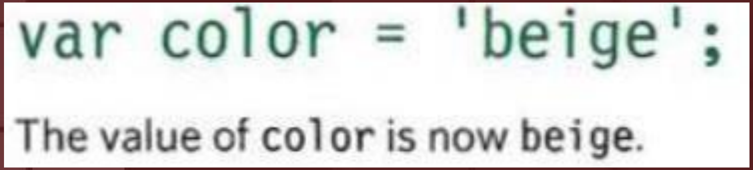
```
// Update the third item in the array
colors[2] = 'beige';
```

Expression:

An expression evaluates into (results in) a single value. Broadly speaking there are two types of expressions.

1. EXPRESSIONS THAT JUST ASSIGN A VALUE TO A VARIABLE

In order for a variable to be useful, it needs to be given a value. As you have seen, this is done using the assignment operator (the equals sign).

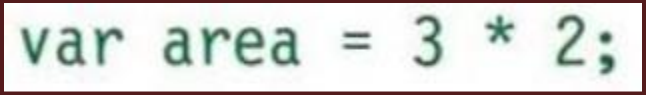


```
var color = 'beige';
```

The value of color is now beige.

EXPRESSIONS THAT USE TWO OR MORE VALUES TO RETURN A SINGLE

You can perform operations on any number of individual values to determine a single value. For example:



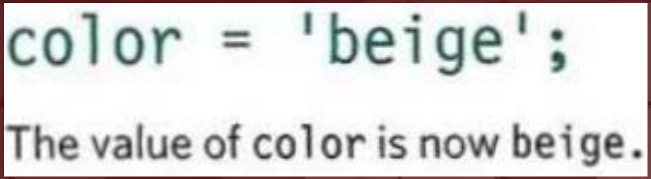
```
var area = 3 * 2;
```

Operators:

Expressions rely on things called operators; they allow program - -mmers to create a single value from one or more values.

ASSIGNMENT OPERATORS

Assign a value to avariable

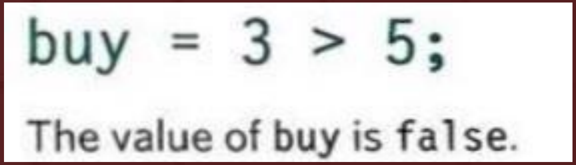


```
color = 'beige';
```

The value of `color` is now `beige`.

COMPARISON OPERATORS

Compare two values andreturn true or false

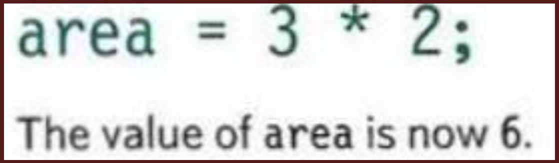


```
buy = 3 > 5;
```

The value of buy is `false`.

ARITHMETIC OPERATORS

Perform basic math

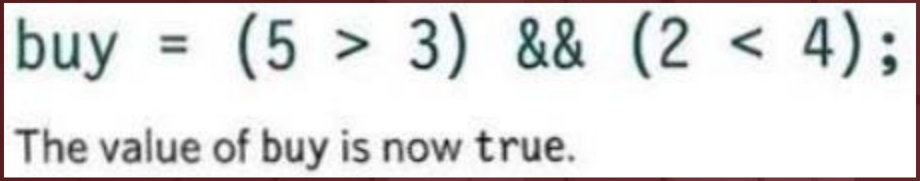


```
area = 3 * 2;
```

The value of area is now 6.

LOGICAL OPERATORS

Combine expressions and return true or false

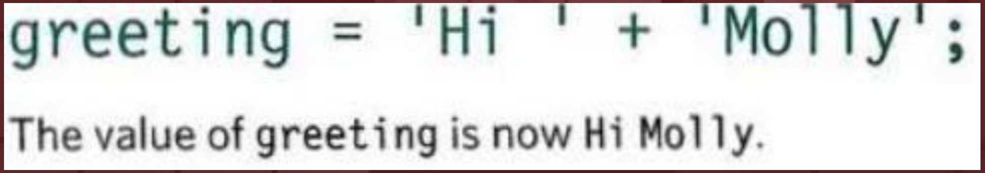


```
buy = (5 > 3) && (2 < 4);
```

The value of buy is now true.

STRING OPERATORS

Combine two strings

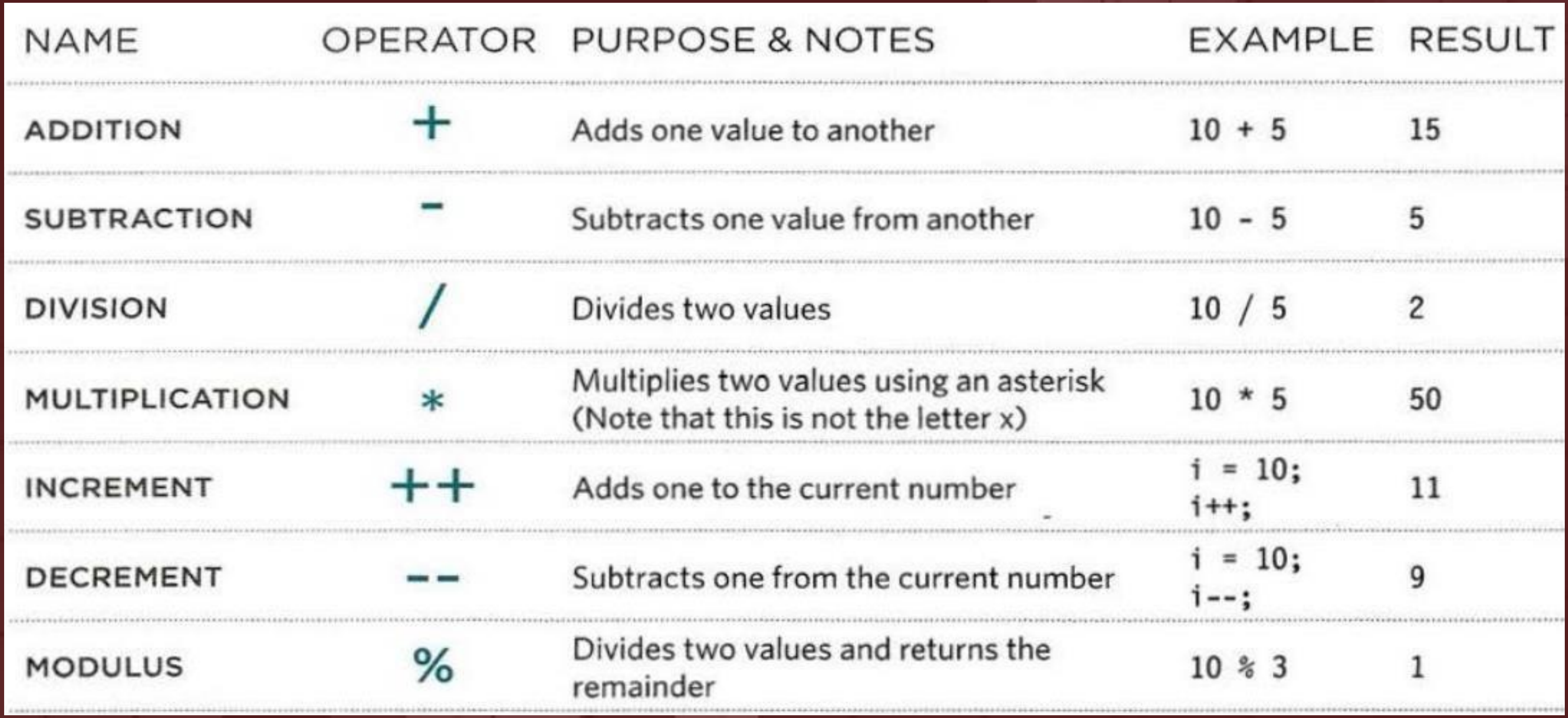


```
greeting = 'Hi ' + 'Molly';
```

The value of greeting is now Hi Molly.

Mathematical Expression:

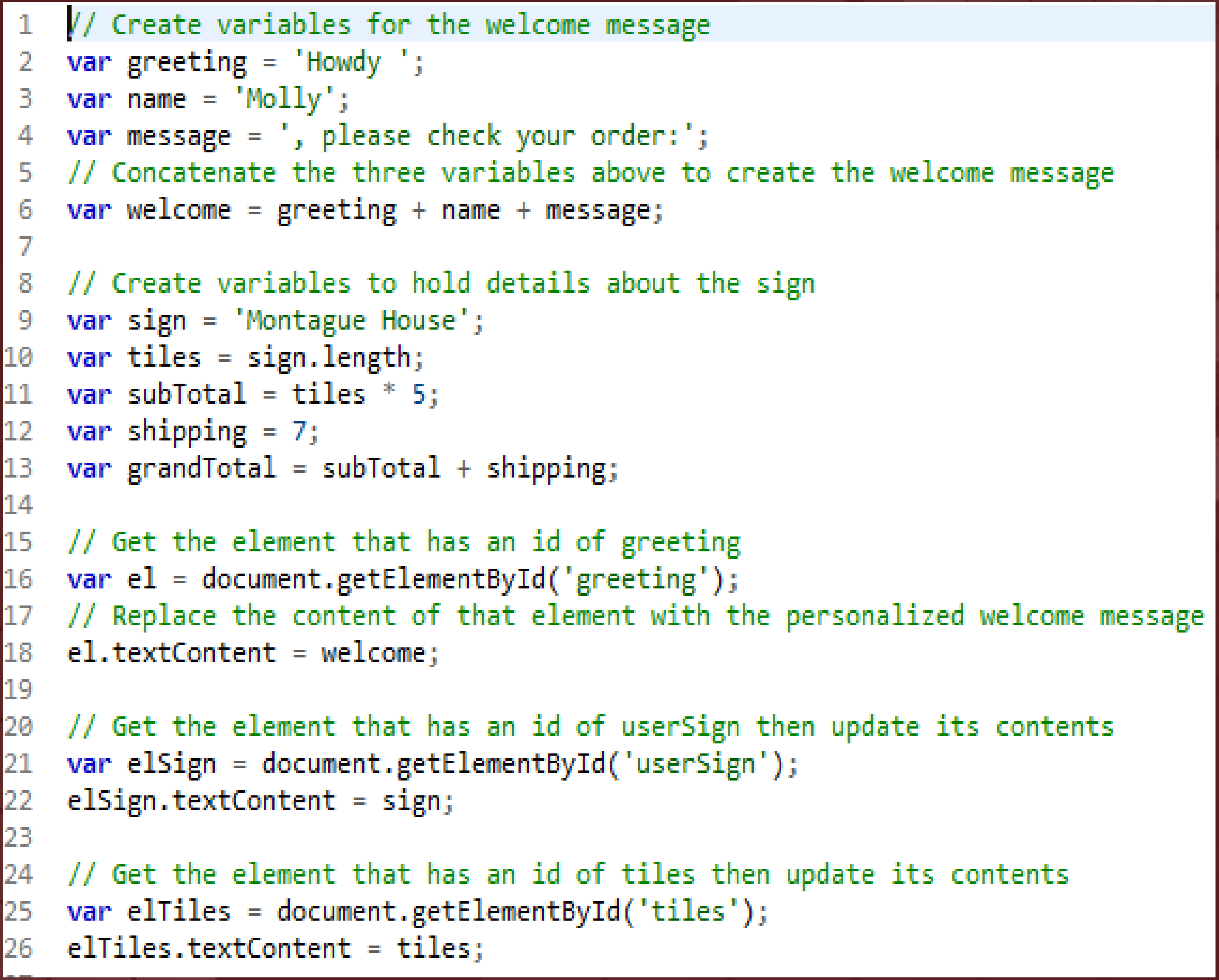
JavaScript contains the following mathematical operators, which you can use with numbers. You may remember some from math class.



NAME	OPERATOR	PURPOSE & NOTES	EXAMPLE	RESULT
ADDITION	+	Adds one value to another	`10 + 5`	15
SUBTRACTION	-	Subtracts one value from another	`10 - 5`	5
DIVISION	/	Divides two values	`10 / 5`	2
MULTIPLICATION	*	Multiplies two values using an asterisk (Note that this is not the letter x)	`10 * 5`	50
INCREMENT	++	Adds one to the current number	`i = 10;` `i++;`	11
DECREMENT	--	Subtracts one from the current number	`i = 10;` `i--;`	9
MODULUS	%	Divides two values and returns the remainder	`10 % 3`	1

Several arithmetic operations can be performed in one expression, but it is important to understand how the result will be calculated. Multiplication and division are performed before addition or subtraction. To change the order in which operations are performed, place the calculation you want done first inside parentheses.

Following is the example.js (JavaScript file) combines many techniques we have discussed.



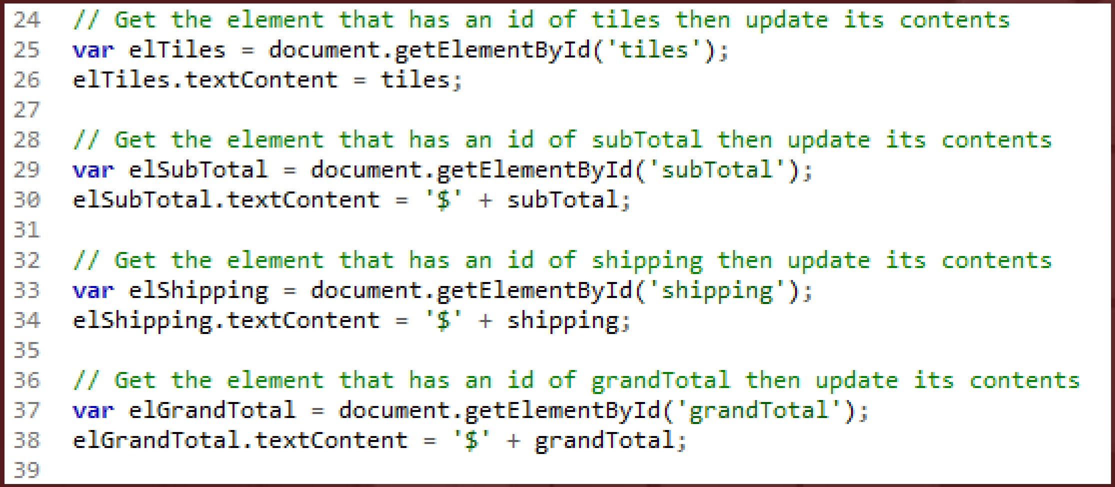
```javascript
// Create variables for the welcome message
var greeting = 'Howdy ';
var name = 'Molly';
var message = ', please check your order:';
// Concatenate the three variables above to create the welcome message
var welcome = greeting + name + message;

// Create variables to hold details about the sign
var sign = 'Montague House';
var tiles = sign.length;
var subTotal = tiles * 5;
var shipping = 7;
var grandTotal = subTotal + shipping;

// Get the element that has an id of greeting
var el = document.getElementById('greeting');
// Replace the content of that element with the personalized welcome message
el.textContent = welcome;

// Get the element that has an id of userSign then update its contents
var elSign = document.getElementById('userSign');
elSign.textContent = sign;

// Get the element that has an id of tiles then update its contents
var elTiles = document.getElementById('tiles');
elTiles.textContent = tiles;
```



```javascript
// Get the element that has an id of tiles then update its contents
var elTiles = document.getElementById('tiles');
elTiles.textContent = tiles;

// Get the element that has an id of subTotal then update its contents
var elSubTotal = document.getElementById('subTotal');
elSubTotal.textContent = '$' + subTotal;

// Get the element that has an id of shipping then update its contents
var elShipping = document.getElementById('shipping');
elShipping.textContent = '$' + shipping;

// Get the element that has an id of grandTotal then update its contents
var elGrandTotal = document.getElementById('grandTotal');
elGrandTotal.textContent = '$' + grandTotal;

```

We have covered the basics of JavaScript's. Now Perform the labs I have uploaded to Moodle

Questions ?