In this chapter you will learn about the peculiarities of the JavaScript language. These are the details for writing a script that will help you avoid errors while you are creating your own scripts and learning the basics of the JavaScript programming language.

2 Objectives

1. Placing JavaScript in an HTML page
2. Case-sensitivity
3. Semicolons
4. Whitespace
5. Brackets
6. Comments
7. Variable and Function Names
8. Reserved Words

It's better to import Javascript modules or files rather than include the Javascript code directly in the HTML file. This allows you to use syntax coloring in your text ec
building well-structured applications. In modern Javascript, there is often only one file linked from the HTML, and then import (ES6 and beyond) or require statemen
include anything else that is needed. This also allows use of the powerful Node Package Manager (npm) which helps download and assemble powerful Javascript
packages written by others.

Inserting Client Side JavaScript into an HTML Page

JavaScript is added to an HTML page using the SCRIPT tag. The script tags should be placed inside the head tag of the document. If an older browser looks at a page containing script tags it will ignore them, as older browsers are written to ignore tags they can't interpret.

JavaScript code should also be placed inside an HTML Comment tag set.

E.g. <!-- code -->

When used with JavaScripts the ending comment tag will also start with two slashes // which is the JavaScript code for comment. This tells the JavaScript interpreter to ignore that statement.

This is a standard way for adding JavaScript to your HTML pages so that it works properly for browsers that are JavaScript enabled and those that do not support JavaScript.

```
<HTML>
<HEAD>
<TITLE>Web Page containing JavaScript</TITLE>
<SCRIPT LANGUAGE="JAVASCRIPT">

<!-- hide JavaScript code from browsers that are not JavaScript enabled

(JavaScript Statements goes here)

//end hiding of JavaScript code -->

</SCRIPT>
</HEAD>
<BODY>
(HTML document goes here)
</BODY>
</HTML>
```

We may also put in a single line of code attached to an event. Events will be explained later. The general syntax for this structure is:

<HTML_TAG Attibute="option" onEvent="JavaScript code statements go here">stuff in between the opening and closing tag</HTML_TAG>

Syntax and Conventions

Writing in any language follows rules and conventions. For example, the English language requires that each sentence must contain a subject and verb to be legitimate. You must also capitalize the first letter of a sentence, and end each sentence with punctuation such as a period or question mark. This is the syntax, or grammar of the English language. Each programming language has a similar set of rules commonly referred to as the syntax.

JavaScript has several rules and conventions for its syntax:

Case-sensitivity:

JavaScript is a case-sensitive language, meaning that the language will treat these words differently: example, Example, EXAMPLE

Semicolons:

All statements should end in a semicolon. The semicolon separates one statement from another.

I.e. var x = 0; var y = 10;
var x=0, y=10 ;

White Space:

JavaScript, like HTML, ignores spaces, tabs, and newlines that appear in statements. JavaScript does, however recognize spaces, tabs, and newlines that are part of a string. We will talk more about strings later in the course.

var x=0; is the same as var x = 0;

All of these examples will produce the same results. It is a good idea to put some spacing in your code to make it easier to read. It is not good programming practice to stretch statements over several lines.

You do need a space between a programming command and the thing it is working on. For example, you need a space between var and the variable name.

Strings and Quotes:

A string is a sequence of zero or more characters enclosed within single or double quotes ('single',. "double").

The double quotation mark can be found within strings that start, and end with (are delimited by) single quotes ('He said, "JavaScript is an interesting language." ').

The single quotation mark can be used within a string delimited by double quotation marks. This syntax will be used quite often through out the book.

For example:

```
<INPUT TYPE="Button" VALUE="Click Me"

onclick="window.alert('You Clicked me');">
```

In the example above we have line of HTML code that uses double quotes to delimit the tag's attributes. So to create a popup window that displays the string "You Clicked me" we need to enclose the string within single quotes. This is done so that the entire JavaScript statement is interpreted and the HTML syntax also remains valid.

The Backslash (\) and Strings:

The backslash character named as such because it leans backwards, and it should not be confused with the forwardslash character (/) that leans forwards. The backslash has a special purpose in JavaScript strings. It will be followed by another character that represents something in a string that cannot be typed on the keyboard.

For example we want the word "You" to appear on one line and the word "Clicked" on a new line and "me" on a third line. The string would look like this:

'You\nClicked\nme'.

The \n represents a carriage return and a line feed. These are the two operations that take place when you use the return on a typewriter. The results would look like this:

You

Clicked

me

These backslash and letter combinations are commonly referred to as escape sequences. Some of the common sequences are:

The ones with * work in EVENT Javascript code.

\b backspace *

\f form feed *

\n new line

\r carriage return (no linefeed)

\t tab

\' single quote (apostrophe)   (work in document.write)

\" double quote * (work in document.write)

The last two are important and can be used like this:

'You didn\'t get that done' or "You didn\'t get that done"

The \ tells the interpreter that in this case it should print the single quote and not interpret it as a delimiter.

Opening and Closing Brackets:

All brackets you open must be closed! This includes ( ), [ ], and { }.

i.e.

```
winpop = window.open('ex1.htm','popup','scrollbars=yes');

if ( x[0] == 10 ) {

x[0] = 0;

x[1] = 0;

}
```

The curly brackets { } are used to contain one or multiple JavaScript statements. In the above example x[0]=0; and x[1]=0; are two different statements.

The square brackets [ ] are part of a special data structure called arrays. Arrays will be covered later in the course.

The curved brackets ( ) are used to contain a function or a method's arguments. Functions and methods will be described shortly. Multiple arguments are separated by commas.

i.e. **('ex1.htm','popup','scrollbars=yes')**.

Comments:

You can create a comment using the double forward slashes, like this:

// this is a comment

Or for multiple line comments you can open the comment with a forward slash and an asterisk "/*", and close it with an asterisk followed by a forward slash "*/" like this:

/* Comments are often used by programmers

to leave notes about their program logic so that when

they return to update it, or someone else needs to edit it,

they can understand what the programmer was doing at the time.

*/

Variable and Function Names

In the next chapter you will be introduced to variables and functions. As the programmer you get to choose and assign the names. The names of the variables and functions must follow these simple rules.

1. The first character must be a letter of the alphabet (lowercase or uppercase), an underscore (_) or a dollar sign ($).
2. You CANNOT use a number as the first character of the name.
3. Names CANNOT contain spaces.
4. Names CANNOT match any of the reserved words.

The following are examples of valid/invalid names?

x
add_two_num
stop
x13
Sum of AB
_whatever
$money_string

We recommend that you use descriptive names for your variables and your functions and that you adopt a standard way of naming things. The two formats that are common are; using the underscore to replace spaces, or capitalizing the first letter of complete words after the first word in the name. For example:

add_two_num

addTwoNumbers

JavaScript tends to use the latter for its naming conventions.

Reserved Words

There are a number of words that make up the components of the JavaScript language. These words cannot be used for variable or function names because the program interpreter would be unable to distinguish between a default JavaScript command and your variable or function name.

abstract	delete	innerWidth	Packages	status
alert	do	instanceof	pageXOffset	statusbar
arguments	document	int	pageYOffset	stop
Array	double	interface	parent	String
blur	else	isFinite	parseFloat	super
boolean	enum	isNaN	parseInt	switch
Boolean	escape	java	personalbar	synchronized
break	eval	length	print	this
byte	export	location	private	throw
callee	extends	locationbar	prompt	throws
caller	final	long	protected	toolbar
captureEvents	finally	Math	prototype	top
case	find	menubar	public	toString
catch	float	moveBy	RegExp	transient
char	focus	moveTo	releaseEvents	try
class	for	name	resizeBy	typeof
clearInterval	frames	NaN	resizeTo	unescape
clearTimeout	Function	native	return	unwatch
close	function	netscape	routeEvent	valueOf
closed	goto	new	scroll	var
confirm	history	null	scrollbars	void
const	home	Number	scrollBy	watch
constructor	if	Object	scrollTo	while
continue	implements	open	self	window
Date	import	opener	setInterval	with
debugger	in	outerHeight	setTimeout	FALSE
default	Infinity	outerWidth	short	TRUE
defaultStatus	innerHeight	package	static	