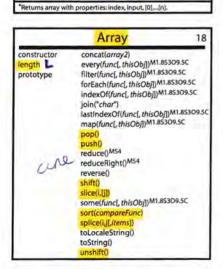
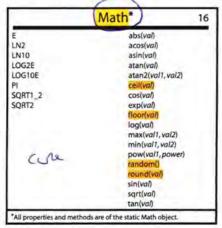


Regi	ular Expressions	45
global ignoreCase inputESC lastIndex multilineEMSC lastMatchEMSC lastParenEMSC	compile(regexp) exec("string")* test("string")	
leftContext prototype rightContext source \$1\$9	che	



Function		23
arguments caller constructor length prototype	apply(this, args Array) call(this[, arg 1 [,arg N] to String() value Of()	(ננ





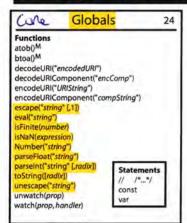
Error		21
prototype constructor description ^E fileName ^M lineNumber ^M message name number ^E	toString()	

statements/frue else { statements for ((init expr); [condition]; [update expr]) { statements for (var in object) { statements	Control Statements	21
statements/frue else statements/fralse result = condition? expr1: expr2 for ([init expr]; [condition]; [update expr]) { statements for (var in object) { statements for each ([var] varName in objectRef) { statements with (objRef) { statements with (condition) while (condition) while (condition) { statements eturn [value] witch (expression) { case (abelN:	f (condition) statements True	
statements/frue else statements/fralse result = condition? expr1: expr2 for ([init expr]; [condition]; [update expr]) { statements for (var in object) { statements for each ([var] varName in objectRef) { statements with (objRef) { statements with (condition) while (condition) while (condition) { statements eturn [value] witch (expression) { case (abelN:		
result = condition? expr1: expr2 for ([init expri]; [condition]; [update expri]) { statements for (var in object) { statements for each ([var] varName in objectRef) { statements	If (condition)	
statementsiffalse result = condition? expr1: expr2 for ([init expri]; [condition]; [update expr]) { statements for (var in object) { statements for each ([var] varName in objectRef) { statements M1.8.1 with (objRef) { statements while (condition) while (condition) { statements case abelN: statements ldefault: statements statements abel: ontinue [label] reak [label] ry { statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not		
for ((init expri); [condition]; [update expri) { statements for (var in object) { statements for each ([var] varName in objectRef) { statements with (objRef) { statements with (objRef) { statements while (condition) while (condition) { statements eturn [value] witch (expression) { case (obelN:		
for ((init expri); [condition]; [update expri) { statements for (var in object) { statements for each ([var] varName in objectRef) { statements with (objRef) { statements with (objRef) { statements while (condition) while (condition) { statements eturn [value] witch (expression) { case (obelN:	1	
for (var in object) { statements for each {[var] varName in objectRef} { statements MI.8.1 with {objRef} { statements with {condition} while {condition} while {condition} { statements light { statements istatements case abelN : statements light statements statements to test statements to test statements if exception occurs in try block statements to run, exception or not	result = condition? expr1:expr2	
for (var in object) { statements for each {[var] varName in objectRef) { statements M1.8.1 with (objRef) { statements io { statements while (condition) while (condition) vhile (condition) { statements eturn [value] witch (expression) { case (abelN:	for ([init expr]; [condition]; [update expr]) \	
statements for each ([var] varName in objectReft) { statements M1.8.1 with (objReft) { statements do { statements while (condition) while (condition) while (condition) { statements eturn [value] witch (expression) { case labelN: statements [break] "" [default: statements] abel: ontinue [label] reak [label] ry { statements to test act (errorInfo) { statements if exception occurs in try block finally { statements to run, exception or not	stalements)	
statements for each ([var] varName in objectReft) { statements M1.8.1 with (objReft) { statements do { statements while (condition) while (condition) while (condition) { statements eturn [value] witch (expression) { case labelN: statements [break] "" [default: statements] abel: ontinue [label] reak [label] ry { statements to test act (errorInfo) { statements if exception occurs in try block finally { statements to run, exception or not	for (var in object) (
for each [[var] varName in objectRef) { statements Mi.8.1 with (objRef) { statements do { statements while (condition) while (condition) { statements eturn [value] witch (expression) { case (obelN:		
statements M1.8.1 with (objRef) { statements io { statements while (condition) while (condition) vhile (condition) { statements eturn [value] witch (expression) { case (abelN:	1	
statements M1.8.1 with (objRef) { statements io { statements while (condition) while (condition) vhile (condition) { statements eturn [value] witch (expression) { case (abelN:	for each ([var] varName in objectRef) (
with (objRef) { statements do { statements while (condition) while (condition) { statements eturn [value] witch (expression) { case labelN: statements [break] statements statements statements statements statements statements statements statements statements statements statements statements statements statements statements to test statements to test statements to run, exception or not	statements	
statements do statements while (condition) while (condition) { statements eturn [value] witch (expression) { case (abelN : statements loreak	F	
statements while (condition) while (condition) { statements eturn [value] witch (expression) { case (abelN:	with (objRef) {	
statements while (condition) while (condition) { statements eturn [value] witch (expression) { case labelN: statements [break] cut (default: statements] abel: ontinue [label] reak [label] ry { statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not	statements	
statements while (condition) while (condition) { statements eturn [value] witch (expression) { case (abelN: statements [break] [default: statements] abel: ontinue [label] preak [label] ry { statements to test atch (errotinfo) { statements if exception occurs in try block finally { statements to run, exception or not		
while (condition) while (condition) { statements etum [value] witch (expression) { case labelN: statements [break] [ldefault: statements] abel: ontinue [label] reak [label] ry { statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not	do (
eturn [value] witch (expression) (case (abelN:) while (condition)	
eturn [value] witch (expression) (case (abelN:	while (condition) [
witch (expression) (case lobelN: statements [break] [default: statements] abel: ontinue [label] preak [label] ry { statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not		
witch (expression) (case lobelN: statements [break] [default: statements] abel: ontinue [label] preak [label] ry { statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not		
case labelN: statements [break] [default: statements] abel: ontinue [label] streak [label] ry { statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not	return [value]	
case labelN: statements [break] [default: statements] abel: ontinue [label] streak [label] ry { statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not	switch (expression) (
[break] [default: statements] abel: ontinue [label] reak [label] ry { statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not	case labelN:	
[default: statements] abel: ontinue [label] oreak [label] ry { statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not		
statements] abel: ontinue [label] reak [label] ry { statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not	-	
abel: ontinue [label] reak [label] ry { statements to test atch (errorInfo) { statements if exception occurs in try block finally { statements to run, exception or not		
ontinue [label] ry { statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not)	
ontinue [label] ry { statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not	Ishali	
reak [label] ry { statements to test atch (errorInfo) { statements if exception occurs in try block finally { statements to run, exception or not	continue [label]	
statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not	break [label]	
statements to test atch (errorinfo) { statements if exception occurs in try block finally { statements to run, exception or not	try {	
statements if exception occurs in try block finally { statements to run, exception or not		
statements if exception occurs in try block finally { statements to run, exception or not) catch (errorinfo) {	
statements to run, exception or not		
statements to run, exception or not) finally	
	n	
brow value	throw value	

Cine

	lumber	16	CUR
constructor MAX_VALUE MIN_VALUE	toExponential(n) toFixed(n) toLocaleString()		
NaN NEGATIVE_INFINITY POSITIVE_INFINITY prototype	toString([radix]) toPrecision(n) valueOf()		

	Boolean	16
constructor	toString()	
prototype	valueOf()	



Appendix A JavaScript Bible, 7th Edition by Danny Goodman

How to Use This Quick Reference

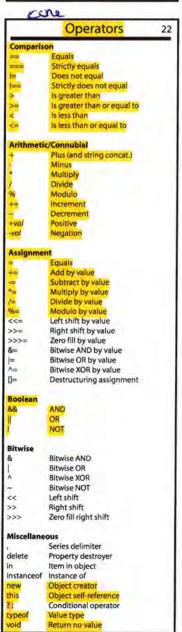
This guide contains quick reference info for the core JavaScript language and browser object models starting with IE 5.5, Mozilla, and Safari.

Numbers in the upper right corners of object squares are chapter numbers in which the object is covered in detail.

Each term is supported by all baseline browsers unless noted with a superscript symbol indicating browser brand

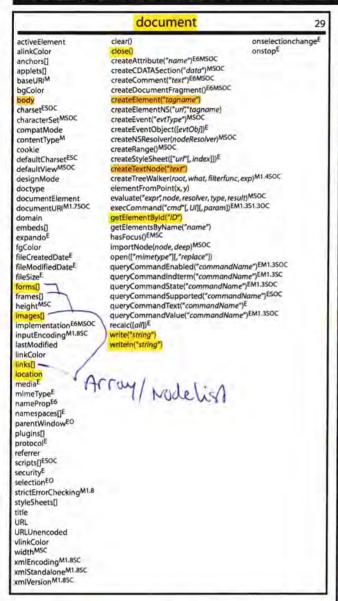
E-Internet Explorer M-Mozilla S-Safari

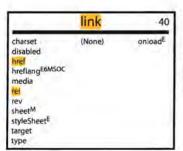
O—Opera C—Google Chrome
For example, M1.4 means the term is supported only by Mozilla 1.4 or later; E means the terms is supported only by Internet Explorer.

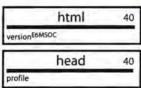




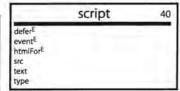








	meta	40
charsetE		
content		
httpEquiv		
name		
urlE		



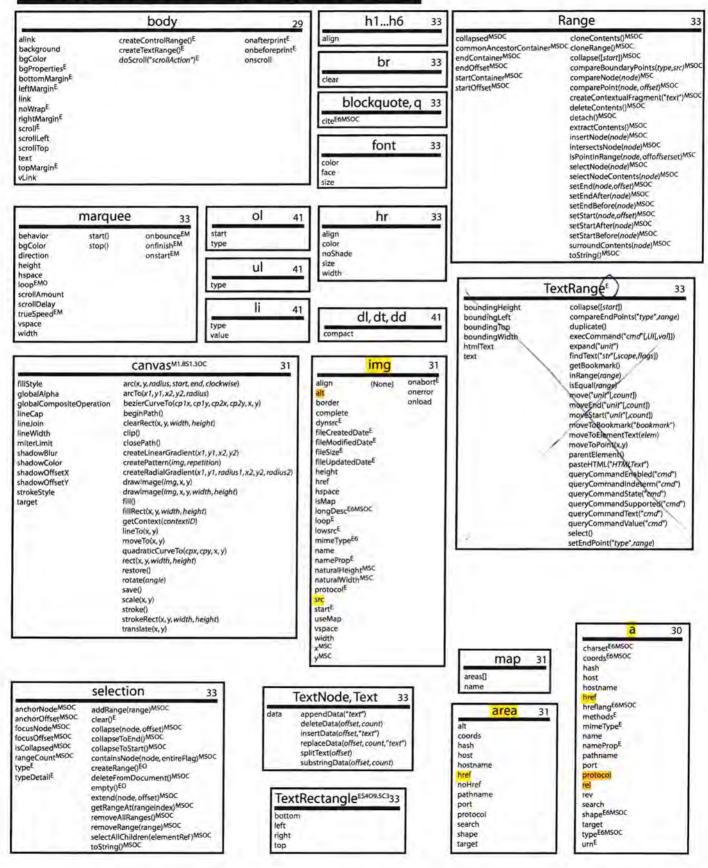


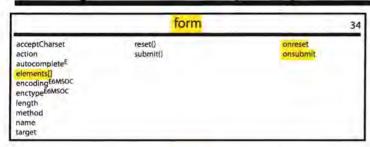
	title	40
text		

base	40
href	
target	

Appendix A: JavaScript and Browser Objects Quick Reference

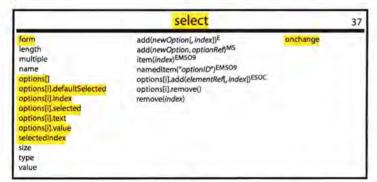
JavaScript and Browser Objects Quick Reference

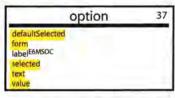




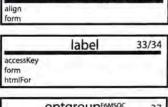
	input	35/36/37
checked(#) completeE(image) defaultChecked(†) defaultValue(###) form maxLength(###) name readOnly(####) size(####) size(####) size(mage) type value	blur()(+++) click()(+++) focus()(+++) select()(++++) † checkbox, radio ++ button, checkbox, radio ++++ text, password, hidden +++++ text, password, hidden	onafterupdateE(fff) onbeforeupdateE(fff) onblur(fff) onclock(fff) onerforupdateE(fff) onerforupdateE(fff) onerousdown(button) onmouseup(button) onselect(fff))

	textarea	36
cols form name readOnly rows type	createTextRange() ^E select()	onafterupdate ^E onbeforeupdate ^E onchange onerrorupdate ^E
value wrap ^E		

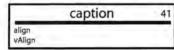




fieldset, legend 33/34



	optgroup ^{E6M5OC}	37
form		
label		

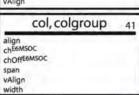


screen	42
availHeight	
availLeftMSC	
availTopMSC	
availWidth	
bufferDepth ^E	
colorDepth	
fontSmoothingEnabled ^E	
height	
pixelDepth	
updateInterval ^E	
width	

	table	41
align background ^E bgColor border borderColorDark ^E borderColorDark ^E borderColorDark ^E caption cellPadding cells ^E datePageSize ^E frame height rows summaryE6MSOC tBodies tFoot tHead width	createCaption() createTFead() deleteCaption() deleteRow(i) deleteTFoot() deleteTFoot() deleteTHead() firstPage() ^E insertRow(i) lastPage() ^E moveRow(srcIndex, destIndex) ^E nextPage() ^E previousPage() ^E refresh() ^E	

tbody, tfoot, thead		41
align bgColor chE6MSOC chOffE6MSOC rows vAlign	deleteRow(i) insertRow(i) moveRow(srcIndex, destIndex) ^E	

t	r 41
align bgColor borderColorE borderColorDarkE borderColorLightE cells cells chE6MSOC chOffE6MSOC heightE0 rowIndex sectionRowIndex vAlign	deleteCell(i) insertCell(i)
col, col	group 41



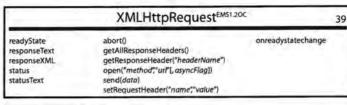






	object (object)	23
constructor prototype	hasOwnProperty("propName") isPrototypeOf(objRef) propertylsEnumerable("propName") toSource(M toString() unwatch("propName") ^M valueOf() watch("propName") ^M	

Notes



	xml ^E	39
src XMLDocument		

a	pplet	44
align altE6MSOC altHTMLE archiveE6MSOC code codeBase height hspace name objectE vspace width (Applet variables)	(Applet r	nethods)

	embed	44
alignMSOC height hidden ^E name pluginspage ^E src units ^E width (Object variables	(Object methods)	onload() onscroll(

mimeType ^{MSOC}	42
description enabledPlugin	7
type suffixes	- 4

plugin ^{MSOC}		42
name filename description length	refresh()	

	object (element)	44
alignESOC altE6 altHTMLE archiveE6MSOC baseHrefE baseURJMSOC borderE6MSOC classidE code codeBase codeBase codeType contentDocumentMSOC data declareE6MSOC form height hspace name objectE standbyE6MSOC type useMapE6MSOC vspace width (Object variables)	(Object methods)	oncellchange ondatasvallable ondatasetchanged ondatasetcomplete onload onrowenter onrowexit onrowsdelete onrowsinserted onscroll

L/m ES 50	react, vue :Eto+
	Kredo -



