Creating variables

var *variable-name_1*[= *expression_1*
[, *variable-name_2*[=*expression_2*]...];

Assignment statement

variable = *expression*;
variable+=*expression*; or `-=, *=, /=, %=`
variable++;
variable- -;

Selection statements

if (*test_1*)
statement_1
[else:
statement_2]

switch (*expression*) {
case *constant-expression*: *statements_1*
[case *constant-expression_2*: *statements_2*
:
default: *statements_n*
}

Loops

Pre-test loop
while (*test*)
statement

Pre-test counting loop
for ([*initialization*]; [*test*]; [*incrementation*])
<statement>

Loop through a structure
for (*variable* in *object*)
statement

Post-test loop
do *statement* while (*test*);

Looping-control statements

break Causes loop to terminate immediately.
continue Skip to next iteration immediately.

Empty Placeholder statement

pass

Exception Handling

throw *exception*
try { *dangerous-code* }
catch *exception* { *exception-handling-code* }
finally { *code-executed-exception-or-no-exeption* }

Data Structures

Arrays

```
var a = new Array();          //empty array
var a[ ];                     //empty array
var a = new Array(5)          //array with 5 elements
var a = new Array(2, 4, 8)    //array with elements 2, 4, and 8
var a[ ] = [2, 4, 8]          //array with elements 2, 4, and 8
```

Arrays may contain any variety of elements.
var a[ ] = [3, “Hello”, true, [2, 4, 6]]

Objects: collections of names and corresponding properties.

```
var obj = new Object();    //New object with no properties.
obj.age = 6;               //Assigning properties and values
obj.name = 'Fred';
```

Literal objects:
{age:6, name:'Fred'}
Instantiate object from existing class:
var today = new Date();

Functions, Classes, and Objects

Calling functions

[*variable*=][*obj-name*.]*funct-name*([*par1*, ...])

Defining functions

function *func-name*([*param1*,...]){
statements-using-params
[return *expression*]
}

Creating methods: a method is a function that belongs to an object.

object-name.method-name = function([*param1*,...])
{*function-body*}

The Global Object

Properties

Infinity	Numeric infinity.
NaN	The not-a-number value.
undefined	The undefined value.

Functions

decodeURI( )	Decode escapted URI
encodeURI( )	Escape URI string.
isFinite(*n*)	True if *n* is finite.
isNaN(*x*)	True if *x* is not a number.

Built-in Classes

Generic Classes

Array

Properties Usage: *array.property*
length Integer number of elements in the array.

Methods Usage: *array.method-name*([*param*,...])

concat(*a1*, …) Concatenate arrays.
forEach(*func*, [*obj*]) Apply function to each element.
indexOf(*value* [,*start*]) Search for *value* in array.
lastIndexOf(*value* [,*start*])
Search backwards for *value* in array.
pop() Remove and return last element from a.
push(*el* [,...]) Append *el* to array *a*.
reverse() Return reverse of array *a*.
shift() Remove and return first element from array.
slice(*start*, *end*) Return slice of *a*.
sort(<order-funct>) Sort array. Ascending is default.
//Delete and replace from start with element-1...
splice(*start*,*delete-ct*,*element-1*,...)
toString() Convert *a* to a string.
unshift(*el1*,...) Insert elements at beginning of *a*.

Boolean

Methods
toString() Converts to string.

Date

Methods

Each *get* method has an associated *set* method.
get[UTC]Date( ) Returns local or UTC day of month.
get[UTC]Day( ) Returns day of week
get[UTC]FullYear( ) Returns 4-digit year.
get[UTC]Hours( ) Returns hours field.
get[UTC]Milliseconds( ) Returns millisecond field.
get[UTC]Minutes( ) Returns minutes field.
get[UTC]Month( ) Returns month field.
get[UTC]Seconds( ) Returns seconds field.
toLocaleString( ) Returns local format date.
toLocaleTimeString( ) Returns local timezone time.

Static Methods

Date.now( )
Returns current time in ms since the epoch.
Date.parse(*date*)
converts time-date string to internal format.
Date.UTC(*year, month, day, hours, mins, secs, ms*)
Returns ms rep of date.

Math

Functions

parseFloat(*string*)	Convert *string* to floating pt. no.
parseInt(*string*)	Convert *string* to integer number.

Constants

Math.E	e = 2.7182818284...
Math.PI	π = 3.1415926589...
Math.SQRT2	$\sqrt{2}$
Math.SQRT1_2	$\sqrt{1/2} = {}^{\sqrt{2}}/_{2}$

Methods: Math.func()

abs(x)	|x|
ceil(x)	Round up.
exp(x)	e^x
floor(x)	Round down.
log(x)	ln x
max(x, y)	Larger of two values.
min(x, y)	Smaller of two values.
pow(x, y)	x^y
random()	Pseudorandom x s.t. 0.0 ≤ x < 1.0
round(x)	Round to nearest integer.
sqrt(x)	Square root.

Trig functions:

sin(x)	cos(x)	tan(x)
asin(x)	acos(x)	atan(x)

Random integer in interval [a, b]:
Math.floor(Math.random( ) * (b – a + 1)+a)

Number

Methods

toFixed(*no-of-dec-places*)
String with given no. of decimal places.
toString( ) Returns string equivalent.
toLocaleString( ) Formatted numeric string.

String String indices start at zero.

Property Usage: *string.property*
length

Methods Usage: *string.method-name*([*param*,...])

charAt(*n*) Character at index *n*.
charCodeat(*n*) Code number of character.
concat(*string*,...) Concatenate strings.
indexOf(*sbst*,*start*) Search a string for a substring.
lastIndexOf(*substr*,*start*)
Search string backwards for a substring.
slice(*start*, *end*) Extract a substring.
substring(*from*, *to*) Return a substring of a string.
substr(*start*, *length*) Return a substring of a string.
toLowerCase( ) Convert a string to lower case.
toUpperCase( ) Convert a string to upper case.
String.fromCharCode(*c1, c2*,...)
Create a string from character codes.
match(*regexp*) Test for a regular expression match.
replace(*regexp*, *string*) Replace r.e. match with string.
search(*regexp*) Search for regular expression match.
split(*delimiter*) Make an array of substrings.

Client-Side Classes

Document: an HTML or XML document

Properties

activeElement	Document element with keyboard focus.
body	<body> element.
cookie	Text of a cookie.
domain	Hostname of the document's server
forms	Array of <form> elements.
head	The <head> element
images	Array of <image> elements
links	Array of <a> and <area> elements that have **href** attributes.
readyState	doc load state: *loading* or *complete*

stylesheets Array of CSS stylesheets
title Plain text content of <title> tag.
URL URL from which document was loaded

Methods

createElement(*tagName)* Create new empty element
getElementById(*elementID*) Get handle for an element
getElementsByClassname(*className*)
Array of handles of elements with same *class* value
getElementsByName(*elementName*)
Array of handles of elements with same *name* value
getElementsByTagName(*tagName*) Array of handles of elements with same HTML tag.
write(*text*) Write text to the end of the document.
writeln(*text*) Write text the the end of the document, apending a newline.

Element: the class of all HTML elements.

Properties

attributes[ ] The element's attributes
childElementCount Number of child elements of this element.
children[ ] Direct children of an element
classname Value of the *class* attribute
firstElementChld First child that is an element
id Value of the *id* attribute.
innerHTML Text contained within the element.
lastElementChild Last child that is an element.
outerHTML The HTML of an element.
tagName Tagname of the element
title Value of *title* attribute. (tooltip)

Methods

blur() Transfer focus to *body* element.
focus() Set keyboard focus on this element.
getAttribute(*attrName*) Returns attribute value.
getElementsByClassName(*className*)
getElementsByTagName(*tagName*)
hasAtrtribute(*attrName*) Boolean.
insertAdjacentHTML(*selection, text*) Insert text according to selection:
beforebegin Before opening tag
afterend after closing tag
afterbegin after opening tag
beforeend before closing tag
removeAttribute(*attribute)* Delete attribute.
setAtrtribute(*attr, value*) Set attribute value.

Window

Properties

document Document in this window.
location URL of the current document
name Value of the *name* attributes
parent Parent window
returnValue Return value of modal window.
Becomes the return value of showModalDialog().
top Top-level parent of this window.

Methods

alert(*text*) Displays the text in a dialog box.
blur() Remove keyboard focus from window.
clearInterval(*id*) Stop setInterval() code execution.
clearTimeout(i*d*) Cancel deferred execution.
close() Close browser window.
confirm(*question*) Launch OK-or-cancel dialog box.
prompt(*msg, defaultt*) Prompt with a message. Return the user's entry or default.
scrollTo(a, b) Scroll the document.
setInterval(*code, delay*) Execute *code* at intervals of *delay* ms. Returns an id.
setInterval(*func, delay, args...*)
setTimeout(*code, delay*) Defer code execution. Returns an id.
showModalDialog(*url, args...*)

Regular Expressions

Most characters match themselves.
Some, e.g., `.?*+^$\()`, have special meaning.
Delineate regular expressions with forward slashes, e.g., /**[**0-9**]**+$/.
Evaluating a r.e.: /<reg-exp>/.test(<text>)

ch Backslash escapes the special meaning of character *ch*.
^ $ Beginning/ending of line/string
[…] Match any one character listed inside brackets.
[^...] Match any one character not listed inside brackets.
. Match any single character.
\b Match word boundary.
\B Match non-boundary of word.
\s Match whitespace (spaces, tabs)
\S Match non-whitespace
\w Match any word.
\W Match any non-word.
\d Match any digit.
\D Match any non-digit.
? Match previous item zero times or one time.
* Match previous item zero or more times.
+ Match previous item one or more times.
{*n*} Match previous item *n* times.
{*n*,} Match previous item *n* or more times.
{*n*,*m*} Match previous item *n* to *m* times.
a | b Match expression a or expression b.
(<re>) Match expression and remember the match.
n Match parenthesized expression #*n* again.
$*n* In replacement strings, characters remembered in parenthesized expression #*n*.

Events

Event Handler	Supported by Object...
onAbort	image
onBlur	text elements, window, form elements
onFocus	
onChange	select, text input elements
onClick	button elements, link. Return `false` to cancel default action.
onDblClick	document, link, image, button elements
onError	image, window
onKeyDown	document, image, link, text elements. Return `false` to cancel.
onKeyPress	
onKeyUp	
onLoad	window, image
onUnload	
onMouseDown	document, link, image, button elements. Return `false` to cancel.
onMouseUp	
onMouseOver	link, image, layer. Return `true` to prevent URL display.
onMouseOut	
onReset	form. Return `false` to prevent reset or submission.
onSubmit	

JavaScript Quick Reference Card

Michael P. Conlon, Ph.D.
Slippery Rock University
Computer Science Department
michael.conlon@sru.edu
September 18, 2015

Italics mean replace the italicized text with an actual instance.
Brackets ([ ]), unless **bold**, mean an optional part of the statement.

Simple Data Types

Numbers: 64-bit floating point. e.g., 1024, -18, 3.14159, 1.657e-12, 5.65E+72

Booleans: true, false

Strings: Strings are immutable.
'This is a string.' "This is a string"

Escape Sequences

\b	backspace	\r	carriage return	\'	apostrophe
\f	formfeed	\t	tab	\ \	backslash
\n	newline			\"	quote

nnn character encoded with octal digits *nnn*
\x*nn* character encoded with hex digits *nn*
\u*nnnn* Unicode character encoded with hex digits *nnnn*

Reserved Words

Keywords

break	delete	false	import	return	true	void
case	do	for	in	switch	typeof	while
continue	else	function	new	this	var	with
default	export	if	null			

Other Reserved Words

catch	const	enum	finally	throw
class	debugger	extends	super	try

Identifiers

Identifiers name variables, functions, classes, etc. Identifiers start with an underscore or letter, followed by any letters, digits, dollar-signs, underscores. JavaScript is case-sensitive.

Operators

Arithmetic operators **String concatenation**
\+ - * / % ** +

Assignment with arithmetic operation
+= -= *= /= %= ++ - -

Relational operators
== != > >= < <=

Logical operators **Identity operators** **Object access**
&& | | ! === !== .

Grouping operators **Array element access**
( ) **[ ]**

Comments

//This is a comment.
/*This is a
multiline comment.*/

Statements

Statements end with a semicolon.
Semicolon is optional if the statement ends with a newline.

Structured statement, or block:

{
statement_1;
:
statement_n;
}