Addison-Wesley's

JavaScript Reference Card

Kathleen M. Goelz and Carol J. Schwartz, Rutgers University

Javascript: A scripting language designed to be integrated into HTML code to produce enhanced, dynamic, interactive web pages.

DATA TYPES

Definition: The classification of values based on the specific categories in which they are stored.

Primitive Types: String, Boolean, Integer, Floating Point, Null, Void

Composite Types: Object, Array, Function. Composite data types are in separate sections of the code.

NUMERIC

Integer: Positive or negative numbers with no fractional parts or decimal places.

Floating Point: Positive or negative numbers that contain a decimal point or exponential notations.

String: A sequence of readable characters or text, surrounded by single or double quotes.

Boolean: The logical values True/False, etc. used to compare data or make decisions.

Null: The variable does not have a value; nothing to report. Null is not the same as zero, which is a numeric value.

Casting: Moving the contents of a variable of one type to a variable of a different type. You don't move the contents to a different variable; it stays in the same variable but the data type is changed or "re-cast".

VARIABLES

Definition: A placeholder for storing data. In JavaScript, a declaration statement consists of the reserved word **var** and the name (identifier) of one or more variables.

Format:

```
var variable_name
```

[**var** command is used to *declare* (create) variables]

Examples:

```
var myHouseColor
var myAddress
var vacation_house, condominium,
    primaryResidence
```

Rules for Naming Variables:

1. Variables cannot be reserved words.
2. Variables must begin with a letter or underscore and cannot begin with symbols, numbers, or arithmetic notations.
3. Spaces cannot be included in a variable name.

Hints:

1. Although variables in JavaScript can be used without being declared, it is good programming practice to declare (initialize), all variables.
2. Variable names are case sensitive; for example *X* does not equal *x*.

PEARSON Addison Wesley

9 780321 320759 90000

ISBN 0-321-32075-1

INITIALIZING VARIABLES

Use the declaration statement to assign a value to the variable. The value is on the right of the equal sign; the variable is on the left.

Format:

```
var  variable_name = value
```

Examples:

```
var  myHouseColor = "yellow"
```
[literal string value yellow assigned to variable `myHouseColor`]

```
var  myAddress = 473
```
[numeric value 473 assigned to variable `myAddress`]

```
var  bookTitle = "Time Capsule", cost =
     28.95, publisher = "Tucker Bay"
```
[multiple variables can be assigned in one statement]

DECISION MAKING AND CONTROL STRUCTURES

Definition: Statements and structures used to change the order in which computer operations will occur.

Types:

Conditional Branching	IF, IF-ELSE, IF-ELSE IF, SWITCH, WHILE, DO, FOR

CONDITIONALS

IF Statement: A conditional branching statement used to determine whether a stated condition is TRUE.

Format:

```
if  (condition)   {
       statements if condition is TRUE
    }
```

Example:

```
if  (score >= 65")  {
       grade = "Pass";
       message = "Congratulations";
}
```

IF-ELSE Statement: A conditional branching statement that includes a path to follow if the condition is TRUE and a path to follow if the condition is FALSE.

Format:

```
if   (condition)   {
   statements if condition is TRUE;
}
else    {
   statements if condition is FALSE;
}
```

Example:

```
if  (score >= 65) {
   grade = "Pass";
   message = "Congratulations";
}
else     {
   grade = "Fail"
   message = "Try again";
}
```

IF-ELSE IF Statement: A conditional branching statement that allows for more than two possible paths. The first time a true condition is encountered, the statement is executed and the remaining conditions will not be tested.

Format:

```
if  (condition) {
   Statements if condition is TRUE;
}
else if (condition)  {
   Statements if condition is TRUE;
}
else  {
   Statements if no prior condition is
     true;
}
```

Example:

```
if       (score>=90)  {
   grade="A";
}
else if  (score>=80)  {
   grade="B";
}
else if  (score>=70)  {
   grade="C";
}
else if  (score>=65)  {
   grade="D";
}
else                  {
   grade="F";
}
```

SWITCH Statement: An alternative to the IF-ELSE IF statement for handling multiple options. Compares the expression to the test values to find a match.

Format:

```
switch (expression or variable name)  {
  case label:
    statements if expression matches
     this label;
    break;
  case label:
    statements if expression matches
     this label;
    break;
  default:
    statements if expression does not
     match any label;
    break;
  }
```

Example:

```
switch (colorchoice)        {
  case "red":
    document.bgColor="red";
    break;
  case "blue":
    document.bgColor="blue";
    break;
  default:
    document.bgColor="white";
    break;
  }
```

LOOPS

Loops cause a segment of code to repeat until a stated condition is met. You can use any loop format for any type of code

FOR LOOP:

Format:

```
For (intialize; conditional test;
     increment/decrement)         {
   Statements to execute;
}
```

Example:

```
For (var i=0; i<=10; i++)         {
   document.write ("This is line " + i);
}
```

DO/WHILE LOOP:

Format:

```
do    {
   Statements to execute;
}
while (condition);
```

Example:

```
var i=0;
do    {
   document.write ("This is line " + i);
   i++;
}
while (i <=10);
```

WHILE LOOP:

Format:

```
while (condition) {
   Statements;
   Increment/decrement;
}
```

Example:

```
var i = 0;
while (i<=10)      {
   document.write ("This is line " + i);
   i++;
}
```

Hint: Watch out for infinite loops, which do not have a stopping condition or have a stopping condition that will never be reached.

OBJECTS

Definition: Objects are a composite data type which contain properties and methods. JavaScript contains built-in objects and allows the user to create custom objects.

Creating Objects: Use the **new** constructor

```
var X = new Array()
```

Examples:

date, time, math, strings, arrays

ARRAY OBJECT

Definition: Array object is a variable that stores multiple values. Each value is given an index number in the array and each value is referred to by the array name and the index number. Arrays, like simple variables, can hold any kind of data. You can leave the size blank when you create an array. The size of the array will be determined by the number of items placed in it.

Format:

```
var arrayname = new Array(size)
```

Hint: When you create an array, you create a new instance of the array object. All properties and methods of the array object are available to your new array.

Example:

```
var days = new Array (7)
```

This creates an array of seven elements using the array constructor.

The first item is `days[0]`, the last item is `days[6]`.

Initializing Arrays:

Array items can be treated as simple variables:

```
days[0] = "Sunday";
days[1] = "Monday";
```

etc.

STRING OBJECT

Definition: String object is created by assigning a string to a variable, or by using the new object constructor.

Example:

```
var name = "Carol";
var name = new String("Carol");
```

Properties:

`Length:`	returns the number of characters in the string
`Prototype:`	allows the user to add methods and properties to the string

Methods:

String formatting methods (similar to HTML formatting tags)

```
String.big
String.blink
String.italics
```

Substring methods (allow user to find, match, or change patterns of characters in the string)

```
indexOf()
charAt()
replace()
```

MATH OBJECT

Definition: Math object allows arithmetic calculations not supported by the basic math operators. Math is a built-in object that the user does not need to define.

Examples:

`Math.abs(number)`	returns absolute value of the numeric argument
`Math.cos(number)`	returns the cosine of the argument, in radians
`Math.round(number)`	rounds number to the nearest integer

DATE/TIME OBJECTS

Date object provides methods for getting or setting information about the date and time.

Note: Dates before January 1, 1970 are not supported.

FUNCTIONS

Definition: A pre-written block of code that performs a specific task. Some functions return values; others perform a task like sorting, but return no value. Function names follow the same rules as variables names. There may or may not be an argument or parameter in the parenthesis, but the parenthesis has to be there.

User-defined Functions:

Example:

`ParseInt()` or `ParseFloat()` convert a string to a number.

To create a function:

Format:

```
function name_of_function (arguments) {
    statements to execute when
     function is called;
}
```

Example:

```
function kilosToPounds (){
    pounds=kilos*2.2046;
}
```

This new function takes the value of the variable kilos, multiplies it by 2.2046, and assigns the result to the variable pounds.

To call a function: Give the name of the function followed by its arguments, if any

`ParseInt(X);` converts the data stored in the variable X into a numeric value.

`kilosToPounds(17);` converts 17 kilos to the same mass in pounds, returning the value 37.4782.

METHODS

Definition: A special kind of function used to describe or instruct the way the object behaves. Each object type in JavaScript has associated methods available.

Examples:

```
array.sort();
document.write();
string.length();
```

Calling: To call or use a method, state the method name followed by its parameters in parentheses.

Example:

```
document.write("Hello, world!");
```

PUTTING IT TOGETHER: JAVASCRIPT AND HTML ON THE WEB

Cookies: Text-file messages stored by the browser on the user's computer

Purpose: To identify the user, store preferences, and present customized information each time the user visits the page

Types:

Temporary (transient, session) — stored in temporary memory and available only during active browser session

Persistent (permanent, stored) — remain on user's computer until deleted or expired

Browser Detection: A script written to determine which browser is running; determine if the browser has the capabilities to load the webpage and support the javascript code; and, if needed, load alternate javascript code to match the browser and platform.

Sniffing: A script written to determine whether a specific browser feature is present; i.e., detecting the presence of Flash before loading a webpage.

Event Handling: Use HTML event attributes (mouseover, mouse click, etc.) and connect event to a JavaScript function called an event handler

OPERATORS

ARITHMETIC

+	addition	adds two numbers
–	subtraction	subtracts one number from another
*	multiplication	multiplies two numbers
/	division	divides one number by another
%	modulus	returns the integer remainder after dividing two numbers
++	increment	adds one to a numeric variable
—	decrement	subtracts one from a numeric variable

STRING

+	concatenation	concatenates or joins two strings or other elements
+=	concatenation/ assignment	concatenates two string variables and assigns the result to the first variable

LOGICAL

&&	logical AND	Compares two operands; returns true if both are true, otherwise returns false
||	logical OR	Compares two operands; returns true if either operand is true, otherwise returns false
!	logical NOT	Returns false if its operand can be converted to true, otherwise returns false

COMPARISON

==	Returns true if the operands are equal
!=	Returns true if the operands are not equal
===	Returns true if the operands are equal and the same data type
!==	Returns true if the operands are not equal and/or not the same data type
>	Returns true if the first operand is greater than the second
>=	Returns true if the first operand is greater than or equal to the second
<	Returns true if the first operand is less than the second
<=	Returns true if the first operand is less than or equal to the second

ASSIGNMENT

=	Assigns the value of the seond operand to the first operand
+=	Adds two numeric operands and assigns the result to the first operand
–=	Subtracts the second operand from the first, and assigns the result to the first
*=	Multiplies two operands, assigns the result to the first
/=	Divides the first operand by the second, assigns the result to the first
%=	Finds the modulus of two numeric operands, and assigns the result to the first

RESERVED WORDS

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	shor	while
do	import	static	with
double	in	super	