JavaScript

In 8 Hours

For Beginners
Learn Coding Fast!

Ray Yao

JavaScript
In 8 Hours

By Ray Yao

For Beginners
Learn Coding Fast

About the Author

Ray Yao:

Certified PHP engineer by Zend, USA

Certified JAVA programmer by Sun, USA

Certified SCWCD developer by Oracle, USA

Certified A+ professional by CompTIA, USA

Certified ASP. NET expert by Microsoft, USA

Certified MCP professional by Microsoft, USA

Certified TECHNOLOGY specialist by Microsoft, USA

Certified NETWORK+ professional by CompTIA, USA

Recommended Books on Amazon:

Advanced C++ In 8 Hours

Advanced Java In 8 Hours

AngularJS In 8 Hours

C# In 8 Hours

C++ In 8 Hours

Chinese Daily Expressions

HTML CSS In 8 Hours

JAVA In 8 Hours

JavaScript 50 Useful Programs

JavaScript In 8 Hours

JQuery In 8 Hours

Linux Command Line

PHP In 8 Hours

Python In 8 Hours

R In 8 Hours

Ruby In 8 Hours

Visual Basic In 8 Hours

Preface

“JavaScript in 8 Hours” covers all essential JavaScript knowledge. You can learn complete primary skills of JavaScript fast and easily.
The book includes more than 80 practical examples for beginners and includes tests & answers for the college exam, the engineer certification, and the job interview.

Source Code for Download

This book provides source code for download; you can download the source code for better study, or copy the source code to your favorite editor to test the programs.

Source code download link:
https://forms.aweber.com/form/72/528312472.htm

Table of Contents

Hour 1

JavaScript Basic

What is JavaScript?

JavaScript is a dynamic computer programming language used to make web pages interactive. Its interpreter is embedded inside the web browser software, such as IE, Chrome, and Firefox etc. It runs on visitor's computer and browser.

The syntax of JavaScript looks like this:

```
<script type = “text/javascript”>
……
</script>
```

Example 1.1

```
<html>
<head>
<title>Hello World</title>
</head>
<body>
<script language="javascript">
alert("Hello World!");
</script>
</body>
</html>
```

Output:

Hello World!

Explanation:

“<script type = “text/javascript”>” is a tag for JavaScript. The JavaScript codes are included within it.

“alert” pops up an alert box displaying a message.

“alert (“Hello World!”)” displays “Hello World!” message.

Each JavaScript command ends with semicolon**;**

JavaScript block can locate anywhere in the HTML file.

Comment

The comment is used to explain the code.

```
// This is a single line comment

/* This is a multi line comment; JavaScript
interpreter will ignore both single line comment and
multi line comment.
*/
```

Example 1.3

```
<script type = “text/javascript”>
alert (“Hello World”);  // “alert” pops up a message.
                        /* “alert” pops up an alert box displaying a message “Hello World”. “alert( )” is usually used if you want to make sure the message comes through to the user.  */
</Script>
```

Explanation:

// “alert” pops up a message is a single line comment.

/ “alert” pops up an alert box displaying a message “Hello World”.*

*…… */* is a multi line comment.

<! - - **- ->** is an html comment.

Run First Program

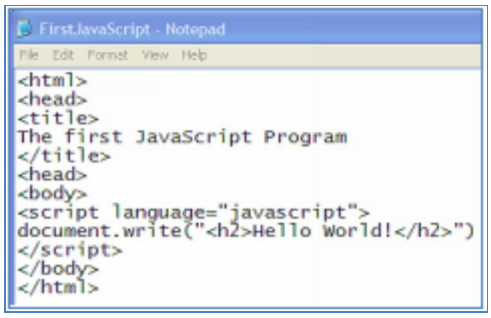
Example 1.2

"Hello World!" program:

Open Notepad, write JavaScript codes as follow:

(Figure 1.1)

```
<Html>        <! - - this is an html file - ->
<head>
<title>
The first JavaScript Program
</title>
<head>
<body>
<script language="javascript">   // start javascript
document.write ("<h2>Hello World! </h2>")  // output
</script>
</body>
</html>
```



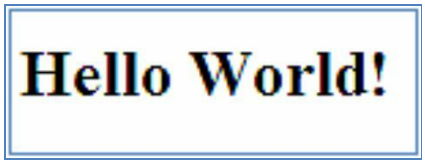
(Figure 1.1 Write JavaScript codes to Notepad)

Please save the file with name "FirstJavaScript.html".

Note: make sure to use **".html"** extension name.

Double click "FirstJavaScript.html" file, the "FirstJavaScript.html" will be run in a browser, and see the output. (Figure 1.2)

Output:



(Figure 1.2)

“document.write ("<h2>Hello World! </h2>")” outputs “Hello World!”

If you want to edit the codes, right click the file “FirstJavaScript.html” > open with > Notepad.

Keywords

Keywords belong to JavaScript itself. These may not be used when choosing identifier names for variables, functions, properties. The following are JavaScript keywords:

break	case	continue	default
delete	do	else	export
false	for	function	if
import	in	new	null
return	switch	this	true
typeof	var	void	while
with	Array	Date	Math
Object	window	location	history
navigator	document	images	links
forms	elements	getElementById	innerHTML

Example 1.4

```
break   // this is a JavaScript keyword.
return    // this is a JavaScript keyword.
```

Explanation:

JavaScript keyword may not be used when choosing identifier names for variables, functions, properties.

Variables

A variable is a symbolic name associated with a value. Variable uses “var” to define.

Example 1.5

```
<script type = “text/javascript”>
var abcde;   // abcde is a variable
var abc888;    // abc888 is a variable
var my_variable;     // my_variable is a variable
</script>
```

Explanation:

abcde, abc888 and my_variable are all variables.

They can store some value. e.g.

var abcde=100;

var abc888=”Hello World”;

var my_variable=true;

Notice that variable naming cannot start with number, cannot have spaces.

e.g. “23var”, “abc de” are invalid variable names.

But “var23”, “abcde” are valid variable names.

Data Types

string – a character or a string of characters

number – an integer or floating point number

boolean – a value with true or false.

function – a user-defined method

object – a built-in or user-defined object

Example 1.6

```
var mystring= “I am a string”;
var myinteger=168;
var myfloat=12.88;
var mybool=true;
```

Explanation:

var mystring=”I am a string”; // mystring data type is a string.

var myinteger=168; // myinteger data type is a number.

var myfloat=12.88; // myfloat data type is a number.

var mybool=true; // mybool data type is boolean.

Note: Double quotes are always used in the string. e.g. “abcde”, “learning”.

Escape Sequences

The “ \ ” backslash character can be used to escape characters.

\n outputs content to the next new line.

\r makes a return

\t makes a tab

\’ outputs a single quotation mark.

\” outputs a double quotation mark.

Example 1.7

```
<html>
<body>
<script language="javascript">
alert("JavaScript says \"Hello World! \"");   /* \” outputs a double
quotation mark. */
</script>
</body>
</html>
```

Output:

JavaScript says “Hello World!”

Explanation:

\” outputs a double quotation mark. Note that “Hello World” has a double quote with it.

Functions

A function is a code block that can repeat to run many times. To define a function, use “function function-name ( ) { }”.

```
function function-name ( ) {……}
```

To call a function, use “function-name ( );”

```
function-name ( );
```

Example 1.8

```
<html>
<head>
<script type="text/JavaScript">
function test() {   // declare a function
  alert("Call a function!");    // output
}
</script>
</head>
<body onload="test()">  <!- - call the function - ->
</body>
</html>
```

Output:

Call a function!

Explanation:

“function test( ) { }” is a declaration of test( ) function.

onload="**test()**" calls function test() after the web has loaded.

“test( )” is a command to call a function named test( ){ }.

When running the function test ( ), it will output “Call a function!”.

<! - - **- ->** is an html comment.

Function with Arguments

A function can have one or more arguments inside the bracket. Arguments are used to pass data to function.

```
function function-name ( var arg ) {……}
```

To call a function, use “function-name (argument);”

```
function-name (argument );
```

Example 1.9

```
<html>
<head>
<title>
JavaScript Test
</title>
<script type="text/JavaScript">
function test(msg) {   // declare a function with arguments
  alert(msg);    // output the value of msg
}
</script>
</head>
<body onload=
"test('Call a function with arguments')">
<!- - call the function and pass arguments- ->
</body>
</html>
```

Output:

Call a function with arguments

Explanation:

onload="**test(“…”)**" calls function test(msg) after the web has loaded. When test(“Call a function with arguments.”) call the function test(msg) {**…**}, it will pass the “Calla function with arguments” to var msg. After var msg has received the data, it will pass the data inside the function. alert (msg) will use the data, and then outputs “Call a function with arguments”.

Return Values

“return” can return a value to the caller.

```
function function-name ( var arg ) {  return value  }
```

To call a function, use “function-name (argument);”

```
function-name (argument );
```

Example 1.10

```
<html>
<head>
<title>
JavaScript Test
</title>
</head>
<body>
<script language="javascript">
function add(num1,num2)
{
return num1+num2;  // pass the result value to caller
}
alert("3 + 5 = " + add( 3, 5 ));  // caller
</script>
</body>
</html>
```

Output:

3 + 5 = 8

Explanation:

“add(3, 5)” means that it calls the function add (num1,num2){ }, and pass the argument “3,5” to var num1 and num2.

“**return** num1+num2” returns the result to caller “add(3,5)”, you can treat it as “add(3,5) = **return** num1+num2”, assigning the value to add(3,5). Namely add (3,5) =8.

“alert("3 + 5 = " + add(3, 5));” will show the result.

Variable Scope

A global variable is declared **outside** the function, it can be used in everywhere; local variable is declared **inside** the function, it is only used inside current function.

Example 1.11

```
<html>
<head>
<title>
Variable Scope
</title>
<script type="text/JavaScript">
var msg=200;  // This num is global variable.
function test(){
var msg=100;   // This num is local variable
}
alert(msg);
</script>
</head>
<body onLoad="test()">
</body>
</html>
```

Output:

200

Explanation:

“alert (num)” outputs 200, which means variable num stores a value of global variable.

Local variables are only visible within the function block where they are declared.

Show the Texts

```
document.write( );
```

“document.write( )” simply prints or display the specified text in the current web page.

Example 1.12

```
<html>
<head>
<title>
Hello World
</title>
</head>
<body>
<script language="javascript">
document.write("Hello World Again!");  // output
</script>
</body>
</html>
```

Output:

Hello World Again!

Explanation:

“document.write (“Hello World Again!”)” displays texts directly into the HTML page.

Note that “alert ( )” pops open a dialog box showing a message.

Undefined Variable

When using a variable that has not defined, JavaScript will return an “undefined” value.

Example 1.13

```
<html>
<head>
<script>
var abcd;    // without initializing
alert( abcd );  // abcd is undefined variable
</script>
</head>
</html>
```

Output:

undefined

Explanation:

“alert(**abcd**);” uses an undefined variable “abcd”. Therefore, the output is “undefined”.

Rule of Variable Naming

1. The first letter must be an ASCII letter, or an underline.
2. The first letter must not be a number.
3. The subsequent letter can be a letter, number or an underline.
4. The variable name cannot use JavaScript keyword or reserved word.

Example 1.14

The following are valid variable:

_good

Excellent100

Five_Star

Example 1.15

The following are invalid variable name:

```
200Guys      // cannot begin with number
Andy&Bob     // cannot use & in variable name.
case     // cannot use JavaScript keyword
```

Hands-On Project: A Simple Call

Call a function

Open Notepad, write JavaScript codes:

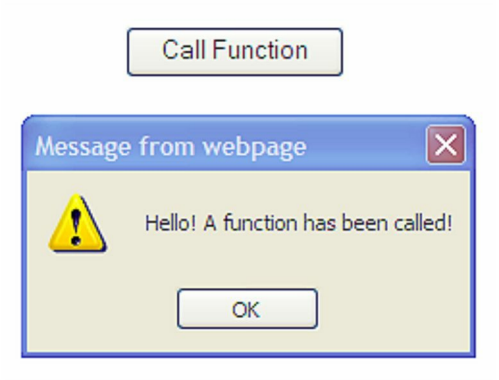
```
<html>
  <head>
     <title>Call a Function</title>
  <head>
  <body>
    <script  language="javascript">
    function  myFunction( )   // declare a function
  {
alert("Hello! A function has been called!")    // output
   }
    </script>
    <form>
      <br><br><br><br><br><br><br>
      <center><input name="button" type="button"
onclick="myFunction( )" value="Call Function"><center>
    </form>        <!- - call the function - ->
</body>
</html>
```

Please save the file with name “CallFunction**.**html”.

Note**:** make sure to use **“.html”** extension name.

Double click “CallFunction**.**html” file, the “CallFunction**.**html” will be run in a browser, please click the button “Call Function”, and see the output.

Output:


Call Function
Message from webpage
Hello! A function has been called!
OK

Explanation:

“function myFunction( ){ } ” defines a function named “myfunction”.

“onclick="myFunction( )"” calls myFunction( ) when clicking the button.

Hour 2

Operators

Arithmetical Operators

Operators	Running
+	add or connect strings
-	subtract
*	multiply
/	divide
%	get modulus
++ or --	increase 1 or decrease 1

% modulus operator divides the first operand by the second operand, returns the remainder. e.g. 4%2, the remainder is 0.

Example 2.1

```
var add=100+200;   alert (add);    // output 300
var div=800/2;   alert ( div );     // output 400
var mod=10%3;   alert ( mod );   // output 1
var inc=10;   alert ( ++inc );   // output 11
var str=”abc”+”de”;  alert ( str );    // output abcde
```

Explanation:

var add=100+200; alert (add) will output 300

var div=800/2; alert (div) will output 400;

var mod=10%3; alert (mod) will output 1;

var inc=10; alert (++inc) will output 11;

var str=”abc”+”de”; alert (str) will output abcde.

Logical Operators

Operators	Equivalent
&&	and
||	or
!	not

After using logical operators, the result will be true or false.

Example 2.2

```
var x=true; var y=false;
var a=x && y;   alert ( a );   // output: false
var b=x || y;   alert ( b );   // output: true
var c=! x;   alert ( c );   // output:  false
```

Explanation:

true && true; returns true;	true && false; returns false;	false &&false; returns false;
true II true; returns true;	true II false; returns true;	false II false; return false;
! false; returns true;	**!** true; returns false;	

Assignment Operators

Operators	Examples:	Equivalent:
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Example 2.3

```
var x=200;   var y=100;
x+=y;  alert ( x );
x/=y;   alert ( x );
x-=y;   alert ( x );
var m=”abc”; var n=”de”;
m+=n   alert ( m );
```

Explanation:

x+=y; // x=x+y; alert (x); outputs 300

x/=y; // x=x/y; alert (x); outputs 2

x-=y; // x=x-y; alert (x); outputs 100

m+=n; // m=m+n; alert (m); outputs abcde

Comparison Operators

Operators	Running
>	greater than
<	less than
>=	greater than or equal
<=	less than or equal
==	equal
!=	not equal

After using comparison operators, the result will be true or false.

Example 2.4

```
var a=100; var b=200;
var result = (a>b); alert ( result );
var result = (a==b); alert ( result );
var result = (a!=b); alert ( result );
```

Explanation:

var result = (a>b); // test 100>200; outputs false.

var result = (a==b); // test 100==200; outputs false.

var result = (a**!**=b); // test 100**!**=200; outputs true.

Conditional Operator

(test-expression) **?** (if-true-do-this) **:** (if-false-do-this);

(test-expression) looks like a<b, x!=y, m==n. etc.

Example 2.5

```
<html>
<head>
<title>Conditional Operator</title>
</head>
<body>
<script>
var a=100; var b=200;
var result=(a<b) ? "apple" : "banana";
// (test-expression) ? (if-true-do-this) : (if-false-do-this);
 alert ( result );
</script>
</body>
</html>
```

Output:

apple

Explanation:

The output is “apple”.

The conditional operator use (a<b) to test the “a” and “b”, because “a” is less than “b”, it is true. Therefore, the output is “apple”.

If Statement

```
if ( test-expression ) {    // if true do this;    }
```

“if statement” executes codes inside { … } only if a specified condition is true, does not execute any codes inside {…} if the condition is false.

Example 2.6

```
<html>
<head>
<title>
JavaScript Test
</title>
</head>
<body>
<script>
var a=200;
var b=100;
if (a>b) {   // if true, do this
alert ( "a is greater than b" );
}
</script>
</body>
</html>
```

Output:

a is greater than b

Explanation:

(a>b) is a test expression, namely (200>100), if returns true, it will execute the codes inside the { }, if returns false, it will not execute the codes inside the { }.

If-else Statement

```
if ( test-expression) {   // if true do this;   }
else  {  // if false do this;   }
```

“if...else statement” runs some code if a condition is true and runs another code if the condition is false

Example 2.7

```
<html>
<head>
<title>
JavaScript Test
</title>
</head>
<body>
<script>
var a=100; var b=200;
if (a>b) {alert ("a is greater than b.")}  // if true do this
else {alert ( "a is less than b" );}   // if false do this
</script>
</body>
</html>
```

Output:

a is less than b

Explanation:

(a>b) is a test expression, namely (100>200), if returns true, it will output ”a is greater than b.” if returns false, it will output “a is less than b”.

Switch Statement

```
switch ( var variable )
{ case 1: if equals this case, do this;  break;
  case 2: if equals this case, do this;  break;
  case 3: if equals this case, do this;  break;
  default : if not equals any case, run default code;
break;
}
```

The value of the variable will compare each case first, if equals one of the “case” value; it will execute that “case” code. “break;” terminates the code running.

Example 2.8

```
<html>
<head>
<title>
JavaScript Test
</title>
</head>
<body>
<script>
var number=20;
switch ( number ) {  // number  value compares each case
case 10 : alert ( "Running case 10" );  break ;
case 20 : alert ( "Running case 20" );  break;
case 30 : alert ( "Running case 30" );  break;
default :  alert ( "Running default code" );  break;  }
</script>
</body>
</html>
```

Output:

Running case 20

Explanation:

The number value is 20; it will match case 20, so it will run the code in case 20.

For Loop

```
for( init, test-expression, increment) { // some code; }
```

“for loop” runs a block of code repeatedly by the specified number of times.

Example 2.9

```
<html>
<head>
<title>
JavaScript Test
</title>
</head>
<body>
<script>
for (var x = 0; x <= 5; x++) {  // repeat 5 times
document.write ( x );
}
</script>
</body>
</html>
```

Output:

012345

Explanation:

var x = 0 is an initializer, initializing variable “x” as 0.

x <= 5 is test-expression, the code will run at most 5 times.

x++ means that x will increase 1each loop.

After 5 times loop, the code will output 012345.

While Loop

```
while ( test-expression ) {  // some js code in here;  }
```

“while loop” loops through a block of code if the specified condition is true.

Example 2.10

```
<html>
<head>
<title>
JavaScript Test
</title>
</head>
<body>
<script>
var counter=0;
while (counter < 8){    // run 8 times
document.write ( "&" );
counter++;
}
</script>
</body>
</html>
```

Output:

&&&&&&&&

Explanation:

“counter< 8” is a test expression, if the condition is true, the code will loop less than 8 times, until the counter is 8, then the condition is false, the code will stop running.

Do-While Loop

```
do{ // some js code in here } while ( test-expression);
```

“do...while” loops through a block of code once, and then repeats the loop if the specified condition is true.

Example 2.11

```
<html>
<head>
<title>
JavaScript Test
</title>
</head>
<body>
<script>
var counter=0;
do {
document.write ( "@" );
counter++;
} while (counter<8);   // run 8 times
</script>
</body>
</html>
```

Output:

@@@@@@@@

Explanation:

“counter< 8” is a test expression, if the condition is true, the code will loop less than 8 times, until the counter is 8, then the condition is false, the code will stop running.

Break Statement

“break” command works inside the loop.

```
break;
```

“break” keyword is used to stop the running of a loop according to the condition.

Example 2.12

```
<html>
<head>
<title>
JavaScript Test
</title>
</head>
<body>
<script>
var num=0;
while (num<10){
if (num==5)  break;   // leave the while loop
num++;
}
document.write( num );
</script>
</body>
</html>
```

Output:

5

Explanation:

“if (num==5) break;” is a break statement. If num is 5, the program will run the “break” command, the break statement will leave the loop, then run “document.write(num)”.

Continue Statement

“continue” command works inside the loop.

```
continue;
```

“continue” keyword is used to stop the current iteration, ignoring the following code, and then continue the next loop.

Example 2.13

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var num=0;
while (num<10){
num++;
if (num==5)  continue;   // go the next while loop
document.write( num );
}
</script>
</body>
</html>
```

Output:

1234678910

Explanation:

Note that the output has no **5**.

“if (num==5) continue;” means: When the num is 5, the program will run “continue” command, skipping the next command “document.write(num)”, and then continue the next while loop.

Hands-On Project: Number One

For Loop Example

Open Notepad, write JavaScript codes:

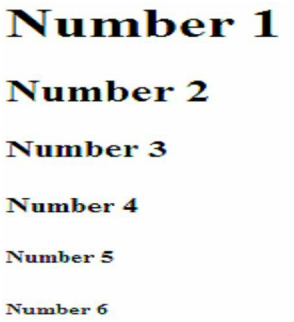
```
<html>
<head>
<title>Number One</title>
</head>
<body>
<script language="javascript">
for(a=1;a<=6;a++){    // repeat 6 times
document.write(       // output
"<h"+a+">"+"Number "+a+"</h"+a+">");
}
</script>
</body>
</html>
```

Please save the file with name “ForLoop**.**html”.

Note**:** make sure to use **“.html”** extension name.

Double click “ForLoop**.**html” file, the “ForLoop**.**html” will be run in a browser, and see the output.

Output:



Explanation:

“for(init, test-expression, increment) { // some code; }” runs a block of code repeatedly by specified number of times.

a = 1 is an initializer, initializing variable “a” as 1.

a <= 6 is test-expression, the code will run at most 6 times.

a++ means that “a” will increase 1each loop.

Hour 3

Array

Create an Array

An array is a particular variable, which can contain one or more value at a time. **new Array( )** creates an array.

The syntax to create an array:

```
var array-name = new Array (“value0”, “value1”, “value2”);
```

Other syntax to create an array:

```
var array-name = new Array ( );
array-name[index0] = “value1”;
array-name[index1] = “value1”;
array-name[index2] = “value2”;
```

Example 3.1

```
var color = new Array ( );    // create an array
color [0] = “red”;
color [1] = “blue”;
color [2] = “green”;
```

Explanation:

Above code creates an array, array name is “color”, and it has three elements: color [0], color [1], color [2]. Its indexes are 0, 1, and 2. Its values are red, blue, and green.

Note that index begins with zero.

Show array element values

To show array element values, use syntax as following:

```
document.write (    array-name[index]    );
```

Example 3.2

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
color = new Array("yellow", "purple", "orange");  document.write (
color[1] );   // show array value
</script>
</body>
</html>
```

Output:

purple

Explanation:

From above, you can know:

The value of color[0] is yellow.

The value of color[1] is purple.

The value of color[2] is orange.

Note that the indexes begin counting from zero.

Get the Size of Array

A function can get the size of an array.

```
array.length
```

Example 3.3

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var color = new Array("yellow", "purple", "orange");
var size = color.length;    // get the size of an array
document.write ( size );
</script>
</body>
</html>
```

Output:

3

Explanation:

“var size=color**.**length” can return the size of the array “color”, and assigns the size of the array to “size”.

new Array(“yellow”, “purple”, “orange”) has three elements, so its size is 3.

Join Array Elements

```
array.join(    );

array.join(“  ”);

array.join(“ , ”);
```

array.join( ); can join all array elements without spaces.

array.join(“ “) can join all array elements by spaces.

array.join(“**,**”) can join all array elements by commas.

Example 3.4

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var arr = new Array( );
arr[0] = "I";
arr[1] = "Love";
arr[2] = "JavaScript!";
document.write( arr.join("   ") );  // join elements by spaces
</script>
</body>
</html>
```

Output:

I Love JavaScript!

Explanation:

arr.join(“ ”) can join all elements of the “arr” together with spaces. Of course you can try arr.join(“**,**”) to join elements by commas.

Reverse Element Order

```
array.reverse( );
```

“array.reverse( )” can reverse the element order of an array.

Example 3.5

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var arr = new Array( "A", "B", "C" );
arr.reverse( );   // reverse the elements order
var rev = arr.join(",");  // join all elements by “,”
document.write( rev );
</script>
</body>
</html>
```

Output:

C, B, A

Explanation:

“arr.reverse( )” reverses the element order of “arr” array.

“rev=arr.join(“,”)” joins all elements of “arr” array together and assign the value with new element order to “rev”.

Slice Elements

```
array.slice( start, last-1)
```

“array.slice(start, last-1)” can slice array elements into pieces, extract some elements from start to last-1 index.

“start” means the first element.

“last-1” means the last-1 element.

Example 3.6

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var arr = new Array(  "a", "b", "c", "d", "e", "f" );
var sli = arr.slice( 2, 5 );   /* extract some elements from index 2 to index
5-1  */
document.write( sli );
</script>
</body>
</html>
```

Output:

c, d, e

Explanation:

“arr.slice (2, 5)” extracts elements from “arr” array from 2 to 5-1 index. Note that 5-1 index signifies “e”, instead of “f”.

Sort Elements in Order

```
array.sort( )
```

“array.sort( )” can sort array elements orderly.

Example 3.7

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var arr = new Array( 2, 5, 3, 1, 4, 6 );
arr.sort ( );   // sort all elements in order
var sor = arr.join(",");  // join all elements by “,”
document.write( sor );
</script>
</body>
</html>
```

Output:

1,2,3,4,5,6

Explanation:

“arr**.sort (**)” sort the elements of “arr” sequentially.

“var sor = arr.join(“**,**”)” joins the elements of “arr” by commas; and assign the value to “sor”.

Change Elements to String

```
array.toString();
```

“array.toString()” can change the elements of an array to string.

Example 3.8

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var myarray = new Array( );
myarray[0] ="Mon";
myarray[1] ="Tue";
myarray[2] ="Wed";
var threedays = myarray.toString( );   // conver to string
document.write( threedays );
</script>
</body>
</html>
```

Output:

Mon, Tue, Wed

Explanation:

“myarray.toString( )” converts all elements in myarray to a string whose value is “Mon, Tue, Wed”.

Search Specific Element (1)

indexOf( )

“indexOf( )” can find the first place where a particular element occurs in an array.

Example 3.9

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var myarray = new Array( );
myarray[0] ="Mon";
myarray[1] ="Tue";
myarray[2] ="Wed";
myarray[3] ="Tue";
var num = myarray.indexOf ("Tue");   // get index of “Tue”
alert(num);
</script>
</body>
</html>
```

Output:

1

Explanation:

“myarray.indexOf (“Tue”)” returns the first place where the “Tue” occurs in myarray. The output is 1.

If the element is not found in array, indexOf( ) returns -1.

Search Specific Element (2)

lastIndexOf( )

“lastIndexOf( )” can find the last place where a particular element occurs in an array.

Example 3.10

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var myarray = new Array( );
myarray[0] ="Mon";
myarray[1] ="Tue";
myarray[2] ="Wed";
myarray[3] ="Tue";
var num=myarray.lastIndexOf ( "Tue" );  // get index of “Tue”
alert(num);
</script>
</body>
</html>
```

Output:

3

Explanation:

“myarray.lastIndexOf (“Tue”)” returns the last place where the “Tue” occurs in myarray. The output is 3.

If the element is not found in array, lastIndexOf( ) returns -1.

Add Element to Beginning

unshift( )

Add one or more elements in the beginning of an array.

Example 3.11

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var arr = [4,5,6];
arr.unshift(1,2,3);  //add 1,2,3 to the beginning of “arr”
alert(arr);
</script>
</body>
</html>
```

Output:

1,2,3,4,5,6

Explanation:

unshift() adds one or more elements into the beginning of an array.

“arr.unshift(1,2,3);” adds elements 1,2,3 to the beginning of “arr”.

Remove First Element

shift( )

Remove the first element of an array.

Example 3.12

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var arr = [4,5,6];
arr.shift( );   //remove the first element of the array
alert(arr);
</script>
</body>
</html>
```

Output:

5, 6

Explanation:

shift() removes the first element of an array.

“arr.shift( );” removes the first element of the array “arr”.

Add Element to End

push( )

Add one or more elements to the end of an array.

Example 3.13

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var arr = [1,2,3];
arr.push(4,5,6);  //add 4,5,6 to the end of “arr”
alert(arr);
</script>
</body>
</html>
```

Output:

1,2,3,4,5,6

Explanation:

push() adds one or more elements to the end of an array.

“arr.push(4,5,6);” adds 4,5,6 to the end of “arr”.

Remove Last Element

pop( )

Remove the last element of an array.

Example 3.14

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var arr = [1,2,3,4,5,6];
arr.pop( );  //remove the last element of arr
alert(arr);
</script>
</body>
</html>
```

Output:

1,2,3,4,5

Explanation:

pop() removes the last element of an array.

“arr.pop( );” removes the last element of arr.

Hands-On Project: Reverse Order

Array Example:

Open Notepad, write JavaScript codes:

```
<html>
<head>
<script>
var oldArray = new Array(0,1,2,3,4,5);
document.write(
"Original array is: "+oldArray.join()+"<br>");
var newArray=oldArray.reverse();  //reverse elements order
document.write(
"Reversed array is: "+newArray.join()+"<br>");
</script>
</head>
</html>
```

Please save the file with name “ArrayDemo.html”.

Note: make sure to use **“.html”** extension name.

Double click “ArrayDemo.html” file, the “ArrayDemo.html” will be run in a browser, and see the output.

Output:

Original array is: 0,1,2,3,4,5

Reversed array is: 5,4,3,2,1,0

Explanation:

“array.join()” connects all array elements together.

“array.reverse()” reverses the order of all array elements.

Hour 4

Math, Time

Math Methods

Method	Returns
abs( )	a number’s absolute value
ceil( )	an integer greater than or equal its argument
cos( )	an angle’s trigonometric cosine
exp( )	a number’s Math.E to the power
floor( )	an integer less than or equal its argument.
log( )	a number’s natural logarithm
random( )	a random positive value from 0 to 1
max( )	a greater between two numbers
min( )	a smaller between two numbers
pow( )	the first argument to the power of the second
round( )	an integer
sin( )	an angle’s trigonometric sine
sqrt( )	a number’s square root
tan( )	an angle’s trigonometric tangent

Example 4.1

```
var num = 10.2;
document.write ( Math.round(num) );  // Output:  10
```

Explanation:

Math.round(num) returns an integer of 10.2. Result is 10.

Greater & Less

```
Math.ceil( );
Math.floor( );
```

“Math.ceil( );” returns a closest integer that is greater than or equal to its argument.

“Math.floor( );” returns a closest integer that is less than or equal to its argument.

Example 4.2

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var num = 9.5;
document.write ("Ceiling number is "+Math.ceil( num )+"<br>" );
document.write ("Flooring number is "+Math.floor( num )+"<br>" );
</script>
</body>
</html>
```

Output:

Ceiling number is 10

Flooring number is 9

Explanation:

“Math.ceil(num));” returns a closest integer that is greater than or equal 9.5, the result is 10

“Math.floor(num);” returns a closest integer that is less than or equal 9.5, the result is 9

Maximum & Minimum

```
Math.max( );
Math.min( );
```

“Math.max( )” returns the greater one between two numbers.

“Math.min( )” returns the less one between two numbers.

Example 4.3

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var x = 100;
var y = 200;
document.write ("Greater number is"+Math.max(x, y)+"<br>" );
document.write ("Less number is"+Math.min(x, y)+"<br>");
</script>
</body>
</html>
```

Output:

Greater number is 200

Less number is 100

Explanation:

“Math.max(x, y)” returns the greater number between 100 and 200, the result is 200.

“Math.min(x, y)” returns the less number between 100 and 200, the result is 100.

Power Value

pow( ) returns the value of x to the power of y.(x^y)

Example 4.4

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var num1 = Math.pow(6, 2);
alert( num1);
var num2 = Math.pow(2, 3);
alert( num2 );
</script>
</body>
</html>
```

Output:

36

8

Explanation:

Math.pow(6,2) equals 6^2.

Math.pow(2,3) equals 2^3.

Square Root

sqrt( ) returns the square root of a number.

Example 4.5

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var num1 = Math.sqrt(4);
alert( num1 );
var num2 = Math.sqrt(25);
alert( num2 );
</script>
</body>
</html>
```

Output:

2

5

Explanation:

Math.sqrt(4) returns the square root of the number 4.

Math.sqrt(25) returns the square root of the number 25.

PI & Random Value

Math.PI
Math.random( )

Math.PI is a constant that stores the value of pi.

Math.random( ) generates a number between 0.0 to 1.0.

Example 4.6

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var pi = Math.PI;
var ran = Math.random( );
document.write ("The PI value is " + pi + "<br>");
document.write ("The random number is " + ran);
</script>
</body>
</html>
```

Output:

The PI value is 3.141592653589793

The random number is 0.6315033452119678

Explanation:

The value of Math.PI is 3.141592653589793.

Math.random( ) generates a random number between 0.0 to 1.0, in here the result is 0.6315033452119678

Date & Time

JavaScript often uses date and time to design the web page. When using date and time, you must create an object of date first.

```
var DateObject = new Date(  )
```

“new Date( )” can create an object of date.

Example 4.7

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var DateObject = new Date( );   // create a date object
alert ( DateObject );
</script>
</body>
</html>
```

Output:

Thu Nov 05 2015 21:39:28 GMT-0500 (Eastern Standard Time)

Explanation:

“var DateObject = new Date( )” creates an object of date.

“alert (DateObject)” displays the current date and time.

Note that: To create a date object, you can use other variable name, for instance:

var **now**= new Date( );

var **TimeObjcet** = new Date( );

Date, Month, Year, Day

The Date Object has the following method:

getDate( )	get the date
getMonth( )	get the month
getYear( )	get the year
getDay( )	get the day
getFullYear()	get the full year

Note:

“getDay( )” returns from 0 (Sunday) to 6 (Saturday).

“getMonth( )” returns from 0 (January) to 11 (December).

“getYear( )” returns a year, but need to add 1900.

“getFullYear()” returns a year with the correct format.

Example 4.8

```
<html>
<body>
<script>
var now = new Date ( );   // create a date object
var m = now.getMonth( ) + 1;  // note it need add 1
var d = now.getDate( );
var y = now.getYear( ) + 1900;  // note it need add 1900
var fy = now.getFullYear();
alert (  m + " - " + d + " - " + y + "\n\n"
+ "This year is  " + fy);
</script>
</body>
</html>
```

Output:

11- 6 - 2015

This year is 2015

Explanation:

“now.getMonth( )” returns a month, but need to add 1.

“now.getDate( )” returns a current date.

“now.getYear( )” returns a year, but need to add 1900.

“getFullYear()” returns a year with the correct format.

Hours, Minutes, Seconds

The Date Object has the following method:

getHours( )	get the hours
getMinutes( )	get the minutes
getSeconds( )	get the seconds
getTime( )	get the time

Note that getTime( ) returns a number from January 1,1970 to current time.

Example 4.9

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var now = new Date( );    // create a date object
var h = now.getHours( );
var m = now.getMinutes( );
var s = now.getSeconds( );
alert ( "Current time is  " + h + ":" + m + ":" + s );
</script>
</body>
</html>
```

Output:

The current time is 19:30:28

Explanation:

“getHours” returns the current hours.

“getMinutes” returns the current minutes.

“getSeconds” returns the current seconds.

Different Time

```
getUTCHours( )
getTimezoneOffset( )
```

“getUTCHours( )” returns the Greenwich Mean Time. (UTC)

“getTimezoneOffset( )” returns the time difference between UTC time and local time. For example, if your time zone is GMT+2, -120 will be returned.

Example 4.10

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var now = new Date( );    // create a date object
var utc = now.getUTCHours( );  //get Greenwich Mean Time
var timezone = now.getTimezoneOffset( );   //get time zone
alert ( utc )
alert ( timezone );
</script>
</body>
</html>
```

Output:

13

300

Explanation:

“getUTCHours( )” returns the UTC hour.

“getTimezoneOffeset” returns a number. For instance,

300 means Eastern Time. 360 means Central Time. 420 means Mountain Time. 480 means Pacific Time.

Set Date & Time

The Date Object has the following method:

setFullYear( )	set the year
setMonth( )	set the month
setDate( )	set the date
setHours( )	set the hours
setMinutes( )	set the minutes
setSeconds( )	set the seconds

Note: Please use setFullYear( ) instead of setYear( ).

Example 4.11

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var now = new Date();   // create a date object
now.setMonth(0);
now.setDate(18);
now.setFullYear(2016);
var today = now;
document.write(today);
</script>
</body>
</html>
```

Output:

Mon Jan 18 2016 09:38:41 GMT-0500 (Eastern Standard Time)

Explanation:

“setMonth(0), setDate(18) and setFullYear(2016) set the month, date and year as Jan 18 2016.

“0” represents January in setMonth(0).

Timer Function

window.setTimeout( )

“window.setTimeout( )” can set the interval time to perform an action repeatedly.

Example 4.12

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body onload="autoTimer()">
<script>
var count=0;
function autoTimer( )  // user-defined function
{
count++;
document.write( count );   // show the increasing numbers
window.setTimeout( "autoTimer()" , 5000 );   /* call autoTime() every other 5 seconds */
}
</script>
</body>
</html>
```

Output:

12345678……

Explanation:

“window.setTimeout( )” can set the interval time to perform an action repeatedly.

“autoTimer( );” calls the function “autotimer( ){ }” every other 5000 milliseconds, which is a user-defined function.

“5000” represents 5000 milliseconds, which sets every 5000 milliseconds for autoTimer( ) to call the function.

The increasing number displays every other 5 seconds.

Hands-On Project: This Month…

Show Current Month

Open Notepad, write JavaScript codes:

```
<html>
<head>
<script>
var dateObj=new Date();   // create a date object
document.write(
"The current month is: "+ (dateObj.getMonth()+1));
</script>
</head>
</html>
// Note: the index of month starts with 0.
// “0” represents January. “1” represents February……
```

Please save the file with name “ShowMonth.html”.

Note: make sure to use **“.html”** extension name.

Double click “ShowMonth.html” file, the “ShowMonth.html” will be run in a browser, and see the output.

Output:

The current month is: 11

Explanation:

“var dateObj=new Date();” creates a Date Object.

Date Object has many methods that are used to set or get the date information.

“getMonth( )” returns a number from 0 (January) to 11 (December).

Hour 5

String

String length

```
string. length
```

“string.length” can calculate the string length, including spaces.

Example 5.1

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var mystring = "I love JavaScript";
var stringsize = mystring.length;   // get the string length
document.write( "String length is " + stringsize );
</script>
</body>
</html>
```

Output:

String length is 17

Explanation:

“mystring.length” returns the length of mystring, including spaces.

Join Strings

```
string1 + string2 + string3 + string4 + string 5……
```

The “+” operator is used to join strings together.

Example 5.2

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var string1 = "JavaScript ";
var string2 = "is ";
var string3 = "very easy! ";
alert ( string1 + string2 + string3 );  // join strings
</script>
</body>
</html>
```

Output:

JavaScript is very easy!

Explanation:

“string1 + string2 + string3” uses “+” to concatenate strings together.

Search a Character

string.charAt( )

“string.charAt(number)” is used to find a character in a string.

Argument “number” is an index of the string, beginning with 0.

Example 5.3

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var mystring = "JavaScript is easy";
var charac = mystring.charAt ( 2 );  /* search a character at index 2 */
alert ( charac );
</script>
</body>
</html>
```

Output:

v

Explanation:

“mystring.charAt(2)” find a character in mystring with index 2. The result is “v”.

Note that index of string begins with zero.

Convert Character Case

```
string.toUpperCase( )
string.toLowerCase( )
```

“toUpperCase( )” and “toLowerCase( )” can change word’s case.

Example 5.4

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var mystring = "Hello World!";
var big = mystring.toUpperCase( );
var small = mystring.toLowerCase( );
document.write( "Uppercase is " + big + "<br>" );
document.write( "Lowercase is " + small);
</script>
</body>
</html>
```

Output:

Uppercase is HELLO WORLD!

Lowercase is hello world!

Explanation:

“mystring.toUpperCase( )” and “str.toLowerCase( )” change the case of mystring.

“<br>” changes to new line.

Change String to Array

```
string.split( “  ” );
```

“string.split(“ ”);” can change a string to an array.

Argument (“ ”) is a separator which separates the string into elements of the array by space.

Example 5.5

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var mystring = "JavaScript for Beginners";
var myarray = mystring.split( " " );  /* convert a string to an array,
separate all elements by spaces  */
document.write( "<br>" + myarray[0]);
document.write( "<br>" + myarray[1]);
document.write( "<br>" + myarray[2]);
</script>
</body>
</html>
```

Output:

JavaScript
for
Beginners

Explanation:

“mystring.split(“ ”)” separates the string to elements by space separator, which makes three elements: myarray[0], myarray[1], myarray[2].

Extract Substring

```
string.substr( start, length );
```

“substr(start, length)” can extract a substring from a string.

“start” argument specify the start position to extract.

“length” argument specify the substring length.

Example 5.6

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var mystring = "JavaScript is easy.";
var sub = mystring.substr( 4, 6 );  /* extract a sub-string from index 4,
and length 6.  */
alert  ( sub );
</script>
</body>
</html>
```

Output:

Script

Explanation:

“mystring.substr(4, 6)” extract a substring from mystring.

“(4, 6)” specify substring’s start position is 4, length is 6.

Note that string index begins with zero.

Convert a Number to String

```
number.toString( );
```

To meet the requirement of JS programming, sometimes a number needs to convert to a string data type.

“number.toString( )” can change a number to a string.

Example 5.7

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var num1 = 12; var num2 = 68;
var str1 = num1.toString( );
var str2 = num2.toString( );
var sum = str1 + str2;  // connecting instead of adding
document.write( sum );
</script>
</body>
</html>
```

Output:

1268

Explanation:

“num1.toString( )” and “num2.toString( )” change two numbers to string data type. Therefore, the result of str1 adding str2 is 1268, instead of 80.

Convert a String to a Number

```
parseInt(string);
parseFloat( string);
```

“parseInt(string)” and parseFloat(string) can change a string to integer or floating point number.

Example 5.8

```
var str1 = “12”;  var str2 = “68”;
var sum = str1 + str2;
alert ( sum );
(  Output: 1268  )
var num1 = parseInt( str1 );
var num2 = parseInt( str2 );
var addition = num1 + num2;   // add instead of join
alert ( addition );
```

Output: 80

Explanation:

“parseInt(str1)” and “parseInt(str2)” convert two strings to number data type. Therefore, the result of num1adding num2 is 80, instead of 1268.

Search Specific Text (1)

indexOf( )

“indexOf( )” can find the first place where a particular text occurs in a string.

Example 5.9

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var mystring = "Please find where find word occurs!";
var place = mystring.indexOf("find");  // return index
alert ( place );
</script>
</body>
</html>
```

Output:

7

Explanation:

“mystring.indexOf (“find”)” returns the first place where the “find” occurs in mystring. The output is 7.

Character index begins with zero.

If the text is not found in the string, indexOf () returns -1.

Search Specific Text (2)

lastIndexOf( )

“lastIndexOf( )” can find the last place where a particular text occurs in a string.

Example 5.10

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var mystring = "Please find where find word occurs!";
var place = mystring.lastIndexOf("find");  // return index
alert ( place );
</script>
</body>
</html>
```

Output:

18

Explanation:

“mystring.lastIndexOf (“find”)” returns the last place where the “find” occurs in mystring. The output is 18.

Character index begins with zero.

If the text is not found in the string, indexOf () returns -1.

Unicode

```
strObject.charCodeAt(index)
```

“strObject.charCodeAt(index)” returns Unicode code at the specified position.

Example 5.11

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var myString = "Hello world!";
var result = myString.charCodeAt(4); // return unicode
alert("The Unicode of the 5th character is " + result );
</script>
</body>
</html>
```

Output:

The Unicode of the 5th character is 111.

Explanation:

“charCodeAt(4)” returns the Unicode at the 5th character.

Note: Index starts with 0.

The 5th character is “o”, its Unicode is 111.

Add a Link for Text

strObject.link(linkString)

“strObject.link(linkString)” adds a link to a string.

Example 5.12

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var myString = "Please click me!";
var result = myString.link("http://www.xxxxxx.xxx");
// add a hyperlink to the string
document.write("Welcome to our website:  " + result );
</script>
</body>
</html>
```

Output:

Welcome to our website: Please click me!

Explanation:

“myString.link(“http://www.xxxxxx.xxx”)” adds a link of “www.xxxxxx.xxx” to myString.

When you click “Please click me!”; it redirects to specified website.

Hands-On Project: UpperCase

String Operation

Open Notepad, write JavaScript codes:

```
<html>
<head>
<script>
var oldText="Java in 8 Hours"
var newText= oldText.toUpperCase();
document.write("Original: Java in 8 Hours<br>");
document.write("Uppercase: "+ newText);
</script>
</head>
</html>
```

Please save the file with name “ToUppercase.html”.

Note: make sure to use **“.html”** extension name.

Double click “ToUppercase.html” file, the “ToUppercase.html” will be run in a browser, and see the output.

Output:

Original: Java in Hours

Uppercase: JAVA IN 8 HOURS

Explanation:

“string**.**toUpperCase()” changes the string letters to upper case.

“string**.**toLowerCase()” changes the string letters to lower case.

Hour 6

Object

Object Declaration

Object refers to a concrete something. For example: a car, a book, a dog, a house, etc…

```
obj = new Object( );
obj.property;
```

“obj = new Object( )” creates a new object named “obj”.

“obj.property” means an object’s property.

Example 6.1

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
house = new Object ( );  // create an object “house”
house.color = "white";   // an object references a variable
house.size = "big";  // an object references a variable
document.write( "House color: " + house.color + "<br>");
document.write( "House size: " + house.size + "<br>" );
</script>
</body>
</html>
```

Output:

House color: white

House size: big

Explanation:

“house=new Object( )” creates an object name “house”.

“house.color” means house’s color.

“house.size” means house’s size.

Navigate Web Page

```
windows.history.back( );
windows.history.forward( );
```

“back( )” is used to navigate the previous web page.

“forward( )” is used to navigate the next web page.

Example 6.2

windows.history.back( ); // to last page

(Returns: Previous web page.)

windows.history.forward( ); // to next page

(Returns: Next web page.)

Explanation:

“back( )” goes to the previous page.

“forward( )” goes to the next page.

“window.history.method( )” is used to navigate to the specified web page. This method is usually called by a button on a web page.

Go to Specified Page

```
window.history.go( number );
```

“window.history.go(number)” can turn to the specified web page.

Example 6.3

window.history.go(1); // to next page

(Returns: Next web page.)

window.history.go(-1); // to last page

(Returns: Previous web page.)

window.history.go(0); // refrash the current page

(Returns: Reload current web page.)

Explanation:

“window.history.method( )” is used to navigate to the specified web page. This method is usually called by a button on a web page.

Open Customized Window

```
window.open( url, name, width, height, status, menubar )
```

“window.open” can open a new customized window.

“url” argument means new window url.

“name” argument means new window name.

“width” argument means new window’s width.

“height” argument means new window’s height.

“status” argument defines window’s status bar.

“menubar” argument defines window’s menu bar.

“toolbar” argument defines the window’s toolbar.

Example 6.4

```
<html>
<script>
function openWindow(){
windowObj=window.open(" ","MyWindow",    "height=200, width=350,
toolbar=yes, menubar=yes, status=no, top=200, left=200");   // open a
customized window & create an object
windowObj.document.write("This is a new window.");
}
</script>
<form><center><input type="button" onClick="openWindow()"
value="Open Window"></center></form>
</html>
```

Output:

This is a new window.

Explanation:

“MyWindow” is a new window name.

“height=200,width=350” specify the size of the new window.

“toolbar=yes” means there is a toolbar.

“memubar=yes” means there is a menu bar.

“status= no” means there is no status bar.

“top=200,left=200” specifies the location of the new window.

Close Current Window

```
window.close( );
```

“window.close( );” can close the current window.

For example:

```
<html>
<script>
function closeWindow(){
window.close( );   // close the current window
}
</script>
<form>
<center><br><br>
<input type = "button"
onclick = "closeWindow()"   // click and call the function
value = "Close Window">
</center>
</form>
</html>
```

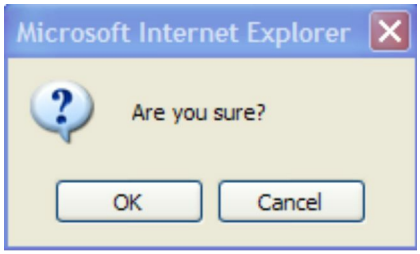
Explanation:

“window.close( )” is used to close the current window.

Confirmation

```
window.confirm( );
```

“window.confirm( )” requires user to confirm. Confirm dialog box looks like this:



When clicking “OK”, it returns true in scripting.

When clicking “Cancel”, it returns false in scripting.

Example 6.5

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var ask = window.confirm("Are you sure?");   /* confirm() will return
true or false  */
if ( ask == true )
{ alert ( "All Right!" ); }
else
{  alert ("Cancel!"); }
</script>
</body>
</html>
```

Output:

All Right!

Cancel!

Explanation:

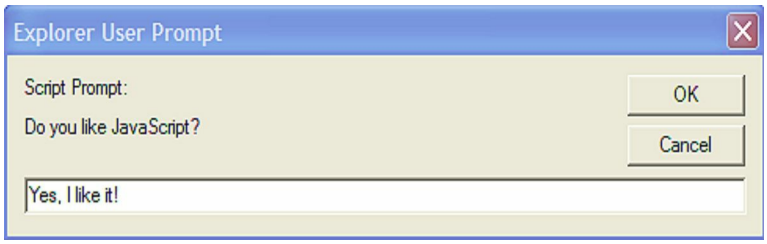
When clicking “OK” button, it outputs “All Right!”.

When clicking “Cancel” button, it outputs “Cancel!”.

Prompt to Input

```
window.prompt( );
```

“window.prompt( )” prompt user to input something to prompt dialog box, which looks like this:



The user should type some words into the prompt dialog box, for example, enter “Yes, I like it.”, and then click the “OK” or “Cancel” button.

Example 6.6

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var ask=window.prompt ("Do you like JavaScript?");
// prompt() will return the data inputted by user
alert ( ask );
</script>
</body>
</html>
```

Output:

Explanation:

The user enters a sentence “Yes, I like it!”, and then click “OK” button, the value is assigned to “ask”.

“alert (ask)” displays the contents that is inputted by the user.

Address Element by ID

```
document.getElementById(  ).innerHTML
```

“getElementById” access an element by its id.

“innerHTML” displays text in HTML document.

Example 6.7

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<div id = "position"></div>
<script>
document.getElementById("position").innerHTML= "OK";
// show “OK” at the specificed id “position”
</script>
</body>
</html>
```

Output:

OK

Explanation:

“id = position” define an id as “position” in an HTML document.

“getElementById(“position”)” access “position” element.

“innerHTML” shows the text “OK” in “position”.

Get Elements by Tag Name

```
document.getElementsByTagName(  ).innerHTML
```

“getElementsByTagName” access elements by their tag name.

“innerHTML” displays or get value in HTML document.

Example 6.8

```
<html>
<body>
<ol>
  <li>Red</li>
  <li>Yellow</li>
  <li>Green</li>
</ol>
<button onclick="myFunction()">Click</button>
<p id="show"></p>
<script>
function myFunction() {
var col = document.getElementsByTagName("li");
// get all values by tag “li”, such as Red, Yellow, Green
document.getElementById("show").innerHTML = col[2].innerHTML;  /*
display the third element to id “show”, col[2] means the third element
“Green”.  */
}
</script>
</body>
</html>
```

Output:

Green

Explanation:

“getElemenstByTagName(“li”)” access “li” tag and get their values.

“document.getElementById("show").innerHTML” shows the value of the element in tag “show”.

“col[2].innerHTML;” means the value of the col[2], the third element’s value.

Connect two Strings

```
string1.concat(string2);
```

“string1.concat(string2);” can connect two strings together.

Example 6.9

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var txt1 = "Hello! "
var txt2 = "My friend.";
var result = txt1.concat ( txt2 );  // connect two strings
document.write( result );
</script>
</body>
</html>
```

Output:

Hello! My friend.

Explanation:

txt1.concat (txt2); connects two strings together.

Convert Number to String

```
Number.toFixed(decimalPlace);
```

“Number.toFixed(decimalPlace);” converts a number into a string with the specified decimal place.

Example 6.10

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var num = 10.28;
var str = num.toFixed(1);   // 1 is decimal place
document.write(str);
</script>
</body>
</html>
```

Output:

10.3

Explanation:

Number.toFixed(decimalPlace) converts a number into a string with the specified decimals place.

“var str = **num.toFixed(1)**;” converts the decimal number to string with 1 decimal place.

From Jan, 1, 1970

```
Date.parse(dateValue);
```

“Date.parse(dateValue)” returns the number of milliseconds between January 1, 1970 and specified date.

Example 6.11

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var d = new Date (2010, 10, 1);   // specify a date
var result = Date.parse(d);  /* get the milliseconds from  January 1, 1970 and specified date  */
alert( result );
</script>
</body>
</html>
```

Output:

1288587600000

Explanation:

“Date.parse(dateValue)” returns the number of milliseconds between January 1, 1970 and specified date.

Absolute Value

Math.abs( ) returns an absolute value of a number.

Example 6.12

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var num1 = Math.abs(100);
document.write( num1 + "<br>");
var num2 = Math.abs( -100 );
document.write( num2 + "<br>");
</script>
</body>
</html>
```

Output:

100

100

Explanation:

Math.abs( ) returns an absolute value of a number.

Print Current Window

window.print( ) prints the contents of the current window.

Example 6.13

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
function printWindow( ){
this.print( );     //"this" represents the current window
}
</script>
<form>
<input type="button" onclick="printWindow( )" value="Print">
</form>
</body>
</html>
```

Output:

(Display the "Print Window")

Explanation:

window.print( ) prints the contents of the current window.

onclick=”printWindow( ) calls the function “printWinow( )” when clicking the button.

Check Java Enabled

navigator.javaEnabled( ) checks the browser whether it supports Java.

Example 6.14

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
if(navigator.javaEnabled( ) == true)     /* checks the browser whether it supports Java. */
```

```
document.write("The browser supports the Java.");
else
document.write("The browser doesn’t support the Java.");
</script>
</body>
</html>
```

Output:

The browser supports the Java.

Or:

The browser doesn’t support the Java.

Explanation:

navigator.javaEnabled( ) checks the browser whether it supports Java.

Screen’s Width & Height

Screen.width returns the width of the screen.

Screen.height returns the height of the screen.

Example 6.15

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body>
<script>
var w = screen.width;
document.write(w + "<br>");
var h = screen.height;
document.write(h + "<br>");
</script>
</body>
</html>
```

Output:

1024
768

Explanation:

Screen.width returns the width of the screen.

Screen.height returns the height of the screen.

Hands-On Project: Max & Min

Math Object

Open Notepad, write JavaScript codes:

```
<html>
<head>
<script>
var number1 = 20;
var number2 = 50;
var number3 = 80;
document.write(
"The maximum number among 20, 50, 80 is:  "+
Math.max(number1,number2,number3)+"<br><br>");
// return maximum among three numbers
document.write(
"The minimum number among 20, 50, 80 is:  "+
Math.min(number1,number2,number3)+"<br><br>");
// return minimum among three numbers
</script>
</head>
</html>
```

Please save the file with name “MathObject**.**html”.

Note**:** make sure to use **“.html”** extension name.

Double click “MathObject**.**html” file, the “MathObject**.**html” will be run in a browser, and see the output.

Output:

The maximum number among 20, 50, 80 is: 80

The minimum number among 20, 50, 80 is: 20

Explanation:

“Math.max( )” gets the maximum number.

“Math.min( )” gets the minimum number.

Hour 7

Event

HTML Basic

When we check the source code of a web page, we will find some HTML elements or tags, which look like this:

Example 7.1

```
<html>
<head>
<title>…</title>
</head>
<body>
<form>…</form>
…..
</body>
</html>
```

Explanation:

<html>…</html> define the document of a web page.

<head>…</head> define the head of a web page.

<title>…</title> define the title of a web page.

<body>…</body> define the body of a web page.

<form>…</form> define a form, which accepts the requests or input from a user.

Click Event

When a user clicks an element on the web page, a “click” event occurs.

```
onClick = function( );
```

“onClick = function( );” means when clicking an element, a function immediately executes.

Example 7.2

<form>

<input type="button" value="Previous" **onClick="history.back( )"**> <!-- click event - ->

</form> (Returns: Previous web page.)

Explanation:

“<from>” and “</form>” is HTML tags, which is in web page, “form” is used to accept the request from users.

“<input type=“button” value=”Previous”…>” defines a button in the form, the button has “Previous” text on it.

“onClick=“history.back( )”” means when clicking the button, the “history.back( )” runs at once, and then go back to the previous web page.

Load Event

When a web page has completely loaded, an “Load” event occurs.

onload = “function( )”

onload=”function( )” executes the function when a web page has loaded.

Example 7.3

```
<html>
<head>
<title>
JavaScript Codes
</title>
</head>
<body onload = "test( )">   <!- - load event - ->
<script>
function test(){
alert ("Welcome to my website!");
}
</script>
</body>
</html>
```

Output:

Welcome to my website!

Explanation:

<html> and </html> is html tags, which defines a web page.

<body> and </body> define a web page body.

onload= “test( )” executes test() immediately when web page has been loaded.

KeyPress Event

When a key is pressed, a “KeyPress” event occurs.

```
onKeyPress  = function( );
```

“onKeyPress=function( )” executes the function when a key is pressed.

Example 7.4

```
<html>
<head>
<title>JavaScript Codes</title>
<script type="text/JavaScript">
function myFunction(msg) {
  alert(msg);
}
</script>
</head>    <!- - keypress event - ->
<body onkeypress="myFunction('Hello! My Friend.')">
Please press a key.
</body>
</html>
```

Output:

Hello! My Friend.

Explanation:

<html> and </html> is html tags, which defines a web page.

<body> and </body> define a web page body.

“onKeyPress=function( )” executes the function when a key is pressed.

Mouseover Event

When a mouse pointer moves over an element, a “mouseover” event occurs.

```
onMouseOver = function( );
```

“onMouseOver = function( )” executes the function immediately when the mouse moves over an element.

Example 7.5

```
<html>
<head>
<title>
JavaScript Codes
</title>
<script type="text/JavaScript">
function myFunction(msg) {
  alert(msg);
}
</script>
</head>
<body>
<br>
<center>      <!- - mouseover event - ->
<textarea onMouseOver="myFunction('MouseOver event occurs!')">
Please put the mouse on me!</textarea>
</center>
</body>
</html>
```

Output:

MouseOver event occurs!

Explanation:

“onMouseOver="myFunction();” executes myFunction() immediately when the mouse moves over an element.

MouseOut Event

When a mouse pointer moves out of an element, a “mouseout” event occurs.

```
onMouseOut = function( );
```

“onMouseout = function( )” means when mouse pointer moves out of an element, function( ) executes immediately.

Example 7.6

```
<html>
<head>
<title>
JavaScript Codes
</title>
<script type="text/JavaScript">
function myFunction(msg) {
  alert(msg);
}
</script>
</head>
<body>
<br>
<center>     <!- - mouseout event - ->
<textarea onMouseOut="myFunction('See You!')"> Please move the mouse away.</textarea>
</center>
</body>
</html>
```

Output:

See You!

Explanation:

"onMouseout=myFunction( );" executes myFunction() when mouse leaves the textarea.

Keyup Event

When a key is released, a "Keyup" event occurs.

```
onkeyup = function( );
```

"onkeyup = function( )" means when a key is released, a function( ) executes immediately.

Example 7.7

```
<html>
<head>
<title>Untitled Document</title>
<script type="text/JavaScript">
function myFunction(msg) {
  alert(msg);
}
</script>
</head>      <!- - keyup event - ->
<body onkeyup="myFunction('Keyup event occurs!')">
Please press a key, and release.
</body>
</html>
```

Output:

Keyup event occurs!

Explanation:

“onkeyup="myfunction( )"” means when a key is released, myfunction( ) runs at once.

Focus Event

When the user moves the cursor onto an element in the web page, a “focus” event occurs.

```
onfocus = function ( );
```

“onfocus = function ( )” means when the user moves the cursor onto an element of the web page, a function( ) immediately executes.

Example 7.8

```
<html>
<head>
<title>JavaScript Codes</title>
</head>
<body>
<script>
function myfunction( )
{ confirm ("Are you ready to input data?");}
</script>
<center>      <!- - focus event - ->
<input type="text" onfocus="myfunction( )">
</center>
</body>
</html>
```

Output:

Are you ready to input data?

Explanation:

“onfocus="myfunction( )"” means when the user moves the cursor into input field in the web page, myfunction( ) immediately executes.

Blur Event

When the user moves the cursor out of an element in the web page, a “blur” event occurs.

```
onblur = function ( );
```

“onblur = function ( )” means when the user moves the cursor out of an element of the web page, a function( ) immediately executes.

Example 7.9

```
<html>
<head>
<title>JavaScript Codes</title>
</head>
<body>
<script>
function myfunction( )
{ confirm ("Are you sure to leave?");}
</script>
<center>     <!- - blur event - ->
<input type="text" onBlur="myfunction( )">
</center>
</body>
</html>
```

Output:

Are you sure to leave?

Explanation:

“onblur="myfunction( )"” means when the user moves the cursor out of input field on the web page, myfunction( ) immediately executes.

Reset Event

onreset = “ScriptCode( )”

onreset = “ScriptCode( )” executes the “ScriptCode( )” when the form is reset.

Example 7.10

```
<html>
<head>
<title>JavaScript Codes</title>
</head>
<body>
<br>
<br>
<br>
<center>     <!- - reset event - ->
<form onreset = "alert('The form has been reset!')">
<input type = "text" id = "txt">
<input type = "reset" value = "RESET" >
</form>
</center>
</body>
</html>
```

Output:

The form has been reset!

Explanation:

onreset = “alert(‘The form has been reset!’)” runs alert( ) when clicking the RESET button.

Submit Event

onsubmit = “ScriptCode( )”

onsubmit = “ScriptCode( )” executes the “ScriptCode( )” when the form is submitted.

Example 7.11

```
<html>
<head>
<title>JavaScript Codes</title>
</head>
<body>
<br>
<br>
<br>
<center>      <!- - submit event - ->
<form onsubmit = "alert('The form has been submitted!')">
<input type = "text" id = "txt">
<input type = "submit" value = "submit" >
</form>
</center>
</body>
</html>
```

Output:

The form has been submitted!

Explanation:

onsubmit = “alert(‘The form has been submitted!’)” runs alert( ) when clicking the SUBMIT button.

Hands-On Project: Mouse Out

Event Example

Open Notepad, write JavaScript codes:

```
<html>
<head>
<script>
function  show1(){
   var y=document.getElementById("textID");
y.value ="Welcome!";
}
function  show2(){
   var y=document.getElementById("textID");
y.value ="Good Bye!";
}
</script>
</head>
<body><br><br>
<center>      <!- - mouseover & mouseout event - ->
<input type="text" id="textID"
onMouseOver=" show1()" onMouseOut=" show2()"/>
</center>
</body>
</html>
```

Please save the file with name “EventDemo**.**html”.

Note**:** make sure to use **“.html”** extension name.

Double click “EventDemo**.**html” file, please put the mouse in or out the text field, and see the output.

Output:

Explanation:

“**onMouseOver=" show1()"**” executes function show1() when mouse moves over the text field.

“**onMouseOut=" show2()**” executes function show2() when mouse moves out the text field.

Hour 8

Form & Dom

Form Basic

A “form” in html is a very important element, which is used to accept the requests or inputs from the user.

Example 8.1

```
<form name=”myForm” method=”get” action=”f.php”>
<input name=”txt” type=”text”>
<input name=”pwd” type=”password”>
<input name=”rd” type=”radio” value=”val2”>
<input name=”chkb” type=”checkbox” value=”val3”>
<input name=”sbmt” type=”submit” value=”Submit”>
<input name=”rst” type=”reset” value=”Cancel”>
</form>
```

Explanation:

“name” means a name of the element.

“method=”get/post”” means the data send by “get” or “post”.

“get” method sends data openly, small quantity.

“post” method sends data secretly, more quantity.

“action=f.php” means the data will be processed by f.php

“type” specify the type of element.

The Element of the form

```
“document.formName.elementName.type”     // returns the element type. such as text, radio, checkbox…
“document.formName.elements["Name"].nodeName”
        // returns the element type. such as text, radio, checkbox…
```

Example 8.2

```
<html>
<body>
<form name="myForm">
    1.<input type="text" name="txt" /><br>
    2.<input type="radio" name="rd" /><br>
    3.<input type="checkBox" name="chkb" /><br>
    4.<input type="password" name="pwd" /><br>
    5.<select name="select"></select><br>
    6.<textarea name="area"></textarea><br>
    7.<input type="submit" name="submit" /><br>
</form>
<script type="text/javascript">
document.write("The 1st element is:  " +document.myForm.txt.type+"
<br>");
document.write("The 2nd element is:  " +document.myForm.rd.type+"
<br>");
document.write("The 3rd element is:  " +document.myForm.chkb.type+"
<br>");
document.write("The 4th element is:  " +document.myForm.pwd.type+"
<br>");
document.write("The 5th element is:  "
+document.myForm.elements["select"].nodeName+"<br>");
document.write("The 6th element is:  "
+document.myForm.elements["area"].nodeName+"<br>");
document.write("The 7th element is:  "
+document.myForm.elements["submit"].type+"<br>");
</script>
</body>
```

</html>

Output:

The 1st element is: text.

The 2nd element is: radio.

The 3rd element is: checkbox.

The 4th element is: password.

The 5th element is: SELECT.

The 6th element is: TEXTAREA.

The 7th element is: submit.

Explanation:

“document.myForm.txt.type” returns the element type.

“document.myForm.elements["select"].nodeName” returns the element type.

Access the Form

document.formName.name document.forms[Index].name

“document.formName.name” can get the form’s name.

“document.forms[Index].name” returns the form’s name.

Example 8.3

```
<html>
<head>
<title>JavaScript Code</title>
<script type="text/javascript">
</script>
</head>
<body>
<form name="form1"></form>
<form name="form2"></form>
<script type="text/javascript">
document.write(document.form1.name + "<br>");
document.write(document.form2.name + "<br>");
document.write(document.forms[0].name + "<br>");
// forms[0] means the first form
document.write(document.forms[1].name + "<br>");
// forms[1] means the second form
</script>
</body>
</html>
```

Output:

form1

form2

form1

form2

Explanation:

“document.form1.name” returns form1’s name.

“document.forms[0].name” returns form[0]’s name.

Set ID for a Form

```
<form id="xxx"  action=" "   method=” ”>
```

“id=”xxx”” sets an ID for the form.

Example 8.4

```
<html>
<body>
<br><br>
<center>
<form id="myForm" action=""> <!- -set id for a form- ->
Username: <input type="text" />
<br><br>
Password: <input type="password">
</form>
<script>
document.write("Note: ID of the Form is: "
+ document.getElementById("myForm").id);    /*  access an id named
myForm   */
</script>
</center>
</body>
</html>
```

Output:

Note: ID of the Form is: myForm.

Explanation:

<form id="myForm" action=""> sets ID as “myForm”.

“document.getElementById("myForm").id” gets the ID of “myForm”.

Show the Value of an Element

```
document.write(object.value);
alert(object.value);
```

“document.write(object.value);” and “alert(object.value);” can show the value of an element.

Example 8.5

```
<html>
<head>
<title>JavaScript Codes</title>
</head>
<body>
<br><br>
<center>
<form>
<button id="myButton" value="12345678">Button</button>
</form>
</center>
<script type="text/javascript">
var button = document.getElementById('myButton');
button.onclick = function(){
document.write ( button.value );   // show button value
alert ( button.value );     // show button value
}
</script>
</body>
</html>
```

Output:

12345678

12345678

Explanation:

“button id="myButton" value="12345678"” sets the value of the button as “12345678”.

“document.write(button.value);” and “alert(button.value);” show the button’s value.

Input Data to Form

```
<input type="text" name="myText" id="myText" />
```

“<input type=”text”>” accepts the data inputted by user.

Example 8.6

```
<html >
<body>
<center>
<br><br>
<br>Please enter data, and Enter.
<br><br>
<form>
<input type="text" name="myText" id="myText" />
<!- - “<input type=”text”>” accepts the data input- ->
</form>
</center>
<script type="text/javascript">
document.getElementById("myText").onchange = function(){
// change event
alert("You have entered: " + this.value);  // “this” represents myText
}
</script>
</body>
</html>
```

Output:

(Show what user has inputted.)

Explanation:

“<input type=”text”>” accepts the data inputted by the user.

“document.getElementById("myText")” accesses an element “myText” in the current document.

“this.value” means the value of “myText”.

“.onchange = function()” executes the function when the value of the element changed.

Show What Inputted

document.formName.elementName.value

“document.formname.elementName.value” can get or set a value for the specified element.

Example 8.7

```
<html>
<head>
<title>JavaScript Codes</title>
</head>
<body>
<br><br>
<center>
<form name="myForm">
<input type="text" name="myText" value="" />
<input type="submit" onClick="show()">
</form>
</center>
<script type="text/javascript">
function show(){
alert("You have inputted: " + document.myForm.myText.value);   /*
gets the value of data which the user has inputted  */
}
</script>
</body>
</html>
```

Output:

(Show what user has inputted.)

Explanation:

“document.myForm.myText.value” gets the value of the text field, to which user has inputted data.

Reset Method

```
reset( )
```

“reset( )” is used to clear the form data.

Example 8.8

```
<html>
<head>
<title>JavaScript Codes</title>
</head>
<body>
<br><br>
<center>
<form id="myform" >
<textarea></textarea>
<br><br>
<input type="button" onclick="myfunction()" value="Clear" >
</form>
</center>
<script type="text/javascript">
function myfunction() {
document.getElementById("myform").reset();
// “reset( )” clears “myform” data
}
</script>
</body>
</html>
```

Output:

(The form data are completely cleared.)

Explanation:

“onclick="myfunction()” executes the myfunction( ) immediately when clicking the button.

“getElementById("myform")” access the “myform” by its id.

“reset( )” clears “myform” data.

Submit Method

submit( )

“submit( )” is used to submit the form data to the server.

Example 8.9

```
<html>
<body>
<br><br>
<center>
<form id="myform" action="">
<textarea></textarea>
<br><br>
<input type="button" onclick="myfunction()" value="submit" >
</form>
</center>
<script type="text/javascript">
function myfunction() {
document.getElementById("myform").submit();
// “submit( )” submits “myform” data to the server.
alert("All data have been submitted successfully!");
}
</script>
</body>
</html>
```

Output:

All data have been submitted successfully!

Explanation:

“onclick="myfunction()” executes the myfunction( ) immediately when clicking the button.

“getElementById("myform")” access the “myform” by its id.

“submit( )” submits “myform” data to the server.

“Select” Selection

```
<select id="myID " onchange="function ( )">
```

When a “select” value in the form is changed, “onChange” event occurs and runs the “function ( )” at once.

Red	v

Example 8.10

```
 <html>
<head>
<script language="javascript">
function choose(color){
alert("The traffic light is  "+ color);
}
</script>
</head>
<body>
<br><br>
<center>
<form id="myForm">
<select name="trafficLight" onChange="choose(this.value)">     <!- -
runs the function “choose()” when selecting an option - ->
<option value="Red">Red</option>
<option value="Yellow">Yellow</option>
<option value="Green">Green</option>
</select>
</form>
</center>
</body>
</html>
```

Output:

The traffic light is Green.

Explanation:

<select name="trafficLight" onChange="choose(this.value)"> runs the function “choose()” when selecting an option, and pass the parameter “this value” to alert( ).

“Radio” Selection

<input type=”radio” id= “myID” value= “myValue”>

A group of radio buttons allows one circle to be checked at any time and submit the selected value to handle.

Example 8.11 (Radio Button: ○ Red ○ Yellow ○ Green)

```
<html>
<body>
<br><br>
<center>
<form name="myform">
<input type= "radio"  id="color" value= "Red" onclick=
"myfunct(this.value )">Red</input>   <!- - runs myfunct() when
clicking the radio- ->
<input type= "radio"  id="color" value= "Yellow" onclick=
"myfunct(this.value )">Yellow</input>
<input type= "radio"  id="color" value= "Green" onclick=
"myfunct(this.value )">Green</input>
</form>
<script>
function myfunct( v) {
```

```
alert ( “You select:  ” + v );
}
</script>
</center>
</body>
</html>
```

Output:

You select: Red

Explanation:

<input type= "radio" id="color" value= "Red" > defines a radio button's value as Red.

"onclick= "myfunct(this.value)" runs myfunct() when clicking.

"CheckBox" Selection

<input type="checkbox" id= "ID value= "myValue">

A group of checkboxes allows multiple boxes to be checked at any time and submit the selected value to handle.

Example 8.12 (CheckBox: ☐ red ☐ blue ☐ pink)

```
<html>
</head>
<body>
<br><br>
<center>
<form name="myform">
<input type= "checkbox"  id="color" value= "Red" onclick= "myfunct(this.value )">Red</input>      <!- - runs myfunct() when clicking the checkbox - ->
<input type= "checkbox"  id="color" value= "Yellow" onclick= "myfunct(this.value )">Yellow</input>
<input type= "checkbox"  id="color" value= "Green" onclick= "myfunct(this.value )">Green</input>
</form>
<script>
```

```
function myfunct( v) {
alert ( "You select:  " + v );
}
</script>
</center>
</body>
</html>
```

Output:

You select: Red

Explanation:

<input type= "checkbox" id="color" value= "Red" > defines a checkbox's value as Red.

"onclick= "myfunct(this.value)" runs myfunct() when clicking.

Submit the Form

"<input type="submit" value="Submit">" submits the data in the form to server.

Example 8.13

```
<html>
<script language="javascript">
function submitForm(){
answer=confirm("Are you sure to submit the form?")
if (answer){
document.myForm.submit()
alert("The form has been submitted successfully!")
}
else{
alert("Cancel to submit the form!")
}}
</script>
<body>
<br><br>
<center>
<form name="myForm" >
User Name:    <input type="text" name="user"><br><br>
Information:  <textarea rows="1" name="message" cols="30">
</textarea><br><br><br>
<input type="submit" value="Submit" onClick="submitForm()">
<!- - run submitForm() when clicking the submit button - ->
</form>
</center>
</body>
</html>
```

Output:

Are you sure to submit the form?

Explanation:

“<input type="submit" value="Submit">” is a command to submit the data in the form.

DOM: CreateElement( )

DOM is a W3C standard. Dom treats button, radio, checkbox, tag as “**element**”**;** and treats element, attribute, comment, text as “**node**”. (“element” is one kind of “node”.)

```
window.document.createElement( );
appendClild( );
```

“window.document.createElement( )” creates an element node with the specified name.

“appendClild( )” appends a node as the last child node.

Example 8.14

```
<html>
<body>
<br><br>
<center>
<script>
var myLine = document.createElement("hr");
// creates an element node “hr”, a horizontal ruled line tag
document.body.appendChild(myLine);
// appends “myLine” element to the HTML document body.
</script>
</center>
</body>
</html>
```

Output:

Explanation:

“myLine = document.createElement("hr")” creates an element node named “myLine”.

“hr” is a horizontal ruled line tag.

“document.body.appendChild(myLine)” appends “myLine” element in the HTML document body.

DOM: CreateTextNode( )

```
window.document.createTextNode( );
appendChild( );
```

“window.document.createTextNode( )” creates a text node with the specified text.

“appendClild( )” appends a node as the last child node.

Example 8.15

```
<html>
<body>
<br><br>
<center>
<script>
var myText = document.createTextNode("Hello Dom!");
// creates a text node named “myText”.
document.body.appendChild(myText);
// appends “myText” text node in the HTML document body.
</script>
</center>
</body>
</html>
```

Output:

Hello Dom!

Explanation:

“myText = document.createTextNode("Hello Dom!")” creates a text node named “myText”.

“Hello Dom!” is a specified string.

“document.body.appendChild(myText)” appends “myText” text node in the HTML document body.

DOM: set/getAttribute( )

```
setAttribute( attributeName, value);
getAttribute( attributeName );
```

“setAttribute(attributeName, value)” adds the specified attribute and specified value to an element.

“getAttribute(attributeName)” returns the value of a specified attribute on an element.

Example 8.16

```
<html>
<head>
<title>
Dom
</title>
</head>
<body>
<br>
<p>Hello Dom!</p><br>
<center>
<script>
var s = document.getElementsByTagName("p")[0];
s.setAttribute("align", "center");   // set align as center
</script>
</center>
</body>
</html>
```

Output:

Hello Dom!

Explanation:

“align” is an attribute name, “center” is an attribute value.

“s.setAttribute("align", "center")” sets the attribute of “s” object.

Example 8.17

```
<html>
<head>
<style>
.setColor {
color: green;
}
</style>
</head>
<body>
<br>
<center>
<p class="setColor">Hello Dom!</p><br>
<!- - attribute name is “class”, attribute value is “setColor”- ->
<script>
var g = document.getElementsByTagName("p")[0];  // first p
var attr = g.getAttribute( "class" );  // get class value
document.write( "The attribute is:  " + attr );
</script>
</center>
</body>
</html>
```

Output:

Hello Dom!

The attribute is: setColor

Explanation:

“class” is an attribute name, “setColor” is an attribute value.

“g.getAttribute("class")” gets the attribute value of “g” object.

DOM: hasChildNodes( )

hasChildNodes ( )

“hasChildNodes( )” returns true if the specified node has any child nodes, otherwise returns false.

Example 8.18

```
<html>
<head>
<title>
Dom
</title>
</head>
<body>
<br><br>
Color
<ul id="color">
<li>Green</li>
<li>Yellow</li>
<li>Red</li>
</ul>
<script>
var child = document.getElementById("color").hasChildNodes( );
// check if the “color” id has nested child elements
alert ("Color has some child nodes?   " + child );
</script>
</body>
</html>
```

Output:

Color has some child nodes? true

Explanation:

“hasChildNodes( )” tests to see if the “color” id has nested child elements. Because “color” id contains three child elements “Green”, “Yellow” and “Red”, it returns true.

Hands-On Project: Clear Anything

Form Reset

Open Notepad, write JavaScript codes:

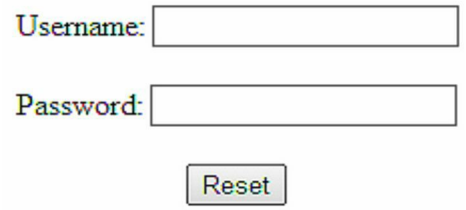
```
<html>
<body>
<center>
<form id="myForm"  method="get"   action="xxx.php">
<br><br>
Username: <input type="text",id="txt1" /><br><br>
Password: <input type="text",id="txt2" /><br><br>
<input type="button" value="Reset" onClick="doReset()"/>
<script>
function doReset(){
  var myForm= document.getElementById("myForm");
  myForm.reset();    // clear any input data
}
</script>
</center>
</body>
</html>
```

Please save the file with name “FormReset.html”.

Note: make sure to use **“.html”** extension name.

Double click “FormReset.html” file, please enter a username and password to the text field, click the “Reset” button, and see the output.

Output:



Explanation:

“**onClick="doReset()**” executes the function “doDoset( )” when clicking the button.

“**myForm.reset();**” clears all data in myForm, and restore the original values.

Appendix
JavaScript Tests & Answers

Tests

Please choose the correct answer.

(1)

```
<html>
  <body>
    <script  language="javascript">
    fill in here  myFunction( )   // declare a function
  {
alert("Hello! A function has been called!")
   }
    </script>
    <form>
       <br><br><br><br><br><br><br>
       <center><input name="button" type="button"
onclick="myFunction( )" value="Call Function"><center>
    </form>
</body>
</html>
```

1. var B. void C. declare D. function

(2)

```
<html>
<body>
<script language="javascript">
```

fill in here (a=1;a<=6;a++) // loop statement

```
document.write(
"<h"+a+">For Loop Example!<br></h"+a+">");
</script>
</body>
</html>
```

1. while B. do C. for D. switch

(3)

```
<html>
<body>
<script language="javascript">
function add(num1,num2)
{
```

fill in here num1+num2; // send back a result to caller

```
}
alert("3 + 5 = " + add( 3, 5 ));
</script>
</body>
</html>
```

1. return B. submit C. send D. back

(4)

```
<html>
```

```
<body>
<script>
var number=20;
fill in here ( number ) {      // evaluate the variable
case 10 : alert ( "Running case 10" );  break ;
case 20 : alert ( "Running case 20" );  break;
case 30 : alert ( "Running case 30" );  break;
default :  alert ( "Running default code" );  break;  }
</script>
</body>
</html>
```

1. for B. while C. switch D. evaluate

(5)

```
<html>
<body>
<script>
var a=100; var b=200;
var result=(a<b) fill in here "apple" : "banana";  /* conditional operator */
alert ( result );
</script>
</body>
</html>
```

1. == B. ? C. != D. &&

(6)

```
<html>
<head>
<script>
var oldArray = fill in here Array(0,1,2,3,4,5);  // create an array
document.write(
"Original array is: "+oldArray.join()+"<br>");
var newArray=oldArray.reverse();
document.write(
"Reversed array is: "+newArray.join()+"<br>");
</script>
</head>
</html>
```

1. array B. set C. create D. new

(7)

```
<html>
<head>
<script>
var dateObj=new fill in here;    // create a Date object
document.write(
"The current month is: "+ (dateObj.getMonth()+1));
</script>
</head>
</html>
```

1. object B. Date() C. dateObj D. Day()

(8)

```
<html>
<head>
<script>
var oldText="Java in 8 Hours"
var newText= oldText. fill in here;    // change to uppercase
document.write("Original: Java in 8 Hours<br>");
document.write("Uppercase is: "+ newText);
</script >
</head>
</html>
```

1. UpperCase() B. upper C. toUpperCase() D. capital

(9)

```
<html>
<head>
<script>
var number1 = 20;
var number2 = 50;
var number3 = 80;

document.write(
```

```
"The maximum number among 20, 50, 80 is:  "+
fill in here.max(number1,number2,number3)+"<br><br>");
// Math method
document.write(
"The minimum number among 20, 50, 80 is:  "+
fill in here.min(number1,number2,number3)+"<br><br>");
// Math method
</script>
</head>
</html>
```

1. Obj B. Mathematics C. Set D. Math

(10)

```
<html>
<head>
<script>
function  show1(){
   var y= fill in here.getElementById("textID");  // DOM
y.value ="Welcome!";
}
function  show2(){
   var y= fill in here.getElementById("textID");   // DOM
y.value ="Good Bye!";
}
</script>
</head>
```

```
<body><br><br>
<center>
<input type="text" id="textID"
onMouseOver=" show1()" onMouseOut=" show2()"/>
</center>
</body>
</html>
```

1. document B. dom C. object D. window

(11)

```
<html>
<body>
<center>
<form id="myForm"  fill in here ="get"   action="xxx.php">
<!-- the method to send data to server  -->
Username: <input type="text",id="txt1" /><br><br>
Password: <input type="text",id="txt2" /><br><br>
<input type="button" value="Reset" onClick="doReset()"/>
<script>
function doReset(){
  var myForm= document.getElementById("myForm");
  myForm.reset();
}
</script>
</center>
</body>
```

```
</html>
```

1. param B. method C. send D. data

(12)

```
<html>
<body>
<script>
var arr = new Array( 2, 5, 3, 1, 4, 6 );
arr. fill in here;   // sort the array elements
var sor = arr.join(",");
document.write( sor );
</script>
</body>
</html>
```

1. classify() B. arrange() C. set() D. sort()

(13)

```
<html>
<body onload="autoTimer()">
<script>
var count=0;
function autoTimer( )  // user-defined function
{
count++;
```

```
document.write( count );
window. fill in here ( "autoTimer()" , 5000 );   // set interval time to perform an action repeatedly.
}
</script>
</body>
</html>
```

1. set() B. setTime() C. setTimeOut() D. timer()

(14)

```
<html>
<body>
<script>
var ask = window.fill in here ("Do you like JavaScript?");
// prompt to input data
alert ( ask );
</script>
</body>
</html>
```

1. prompt B. data C. input D. alert

(15)

```
<html>
<body fill in here = "test( )">  <! - -load event - ->
```

```
<script>
function test(){
alert ("Welcome to my website!");
}
</script>
</body>
</html>
```

1. load B. onload C. event D. method

(16)

```
<html>
<body>
<br><br>
<center>
<form fill in here ="myForm" action="">   <!- - set id for a form- ->
Username: <input type="text" />
<br><br>
Password: <input type="password">
</form>
<script>
document.write("Note: ID of the Form is: "
+ document.getElementById("myForm").id);
</script>
</center>
</body>
</html>
```

1. setID B. set C. name D id

(17)

```
<html>
<body>
<script>
var mystring = "JavaScript for Beginners";
var myarray = mystring. fill in here ( " " );
// convert string to array elements
document.write( "<br>" + myarray[0]);
document.write( "<br>" + myarray[1]);
document.write( "<br>" + myarray[2]);
</script>
</body>
</html>
```

A. convert B. split C. toArray D. change

(18)

```
<html>
<body>
<script>
var ask = window. fill in here ("Are you sure?");
// confirm your input
if ( ask == true )
```

```
{ alert ( "All Right!" ); }
else
{  alert ("Cancel!"); }
</script>
</body>
</html>
```

1. confirm B. validate C. verify D. proof

(19)

```
<html>
<form>
<input type="button" value="Previous" fill in here ="history.back( )">
// click the button, the web will go back to previous page
</form>
</html>
```

1. click B. link C. onClick D. url

(20)

```
<html>
<body>
<ol>
  <li>Red</li>
  <li>Yellow</li>
  <li>Green</li>
```

```
</ol>

<button onclick="myFunction()">Click</button>
<p id="show"></p>
<script>
function myFunction() {
var col = document. fill in here ("li");
// get an element by its tag name
document.getElementById("show").innerHTML = col[2].innerHTML;
}
</script>
</body>
</html>
```

1. get B. tag C. name D. getElementsByTagName

(21)

Which line is **not** correct?

```
<html>
<head>        // line1
<script>
var abcd;
alert( abcd );  // line2
</script>        // line3
</head>        // line4
</html>
```

A. line1 B. line2 C. line3 D. line4

(22)

```
<html>
<body>
<script>
var num=0;
while (num<10){
num++;
if (num==5)  fill in here;   // go to the next while loop
document.write( num );
}
</script>
</body>
</html>
```

A. break B. exit C. continue D. next

(23)

```
<html>
<body>
<script>
var myarray = new Array( );
myarray[0] ="Mon";
myarray[1] ="Tue";
myarray[2] ="Wed";
```

```
myarray[3] ="Tue";
var num = myarray. fill in here("Tue");   // get index of “Tue”
alert(num);
</script>
</body>
</html>
```

A. indexOf B. getIndex C.index D. key

(24)

```
<html>
<script>
var now = new Date( );    // create a date object
var utc = now.fill in here( );  // get Greenwich Mean Time
var timezone = now.getTimezoneOffset( );   // get time zone
alert ( utc )
alert ( timezone );
</script>
</html>
```

A. UTC B. getUTC C. getHours D. getUTCHours

(25)

```
<html>
<body>
<script>
```

```
var myString = "Please click me!";
var result = myString.fill in here("http://www.xxxxxx.xxx");
// add a hyperlink for the string
document.write("Welcome to our website:  " + result );
</script>
</body>
</html>
```

A. hyperlink B. link C. href D. url

(26)

```
<html>
<body>
<script>
var txt1 = "Hello! "
var txt2 = "My friend.";
var result = txt1.fill in here( txt2 );   // connect two strings
document.write( result );
</script>
</body>
</html>
```

A. connect B. join C. concat D. &

(27)

```
<html>
```

```
<head>
<title>JavaScript Codes</title>
<script type="text/JavaScript">
function myFunction(msg) {
  alert(msg);
}
</script>
</head>            <!- - keypress event - ->
<body fill in here ="myFunction('Hello! My Friend.')">
Please press a key.
</body>
</html>
```

A. keypress B. onkeypress C. keyEvent D.pressKey

(28)

```
<html >
<body>
<form>
<input type="text" name="myText" id="myText" />
<!- - “<input type=”text”>” accepts the data input- ->
</form>
</center>
<script type="text/javascript">
document.getElementById("myText").onchange = function(){
// change event
alert("You have entered: " + fill in here.value);   /* an object represents
```

```
myText */
}
</script>
</body>
</html>
```

A. this B. object C. input D. text

(29)

```
<html>
<body>
<script>
var myString = "Hello world!";
var result = myString.fill in here(4);     // return unicode
alert("The Unicode of the 5th character is " + result );
</script>
</body>
</html>
```

A. getUnicode B. Unicode C. charCode D. charCodeAt

(30)

```
<html>
<body>
<script>
var num = 10.28;
```

```
var str = num.fill in here(1);   /* converts a number into a string with 1 decimal place */
document.write(str);
</script>
</body>
</html>
```

A. toString B. toFixed C. toStr D. toCharacter

(31)

```
<html>
<body>
<!- - submit event - ->
<form fill in here = "alert('The form has been submitted!')">
<input type = "text" id = "txt">
<input type = "submit" value = "submit" >
</form>
</center>
</body>
</html>
```

A. submit B. submitEvent C. onsubmit D. clicksubmit

(32)

```
<html>
<body>
```

```
<script>
var myLine = document.fill in here("hr");
// creates an element “hr”, a horizontal ruled line tag
document.body.appendChild(myLine);
// appends “myLine” element to the HTML document body.
</script>
</center>
</body>
</html>
```

A. node B. element C. createNote D. createElement

Answers

01	D	17	B
02	C	18	A
03	A	19	C
04	C	20	D
05	B	21	B
06	D	22	C
07	B	23	A
08	C	24	D
09	D	25	B
10	A	26	C
11	B	27	B
12	D	28	A
13	C	29	D
14	A	30	B
15	B	31	C
16	D	32	D

Review My Book

My friends:

(1) If you found any problem in this book, could you please send an email to me instead of posting your review? Kindly please give me an opportunity to correct the error. Thank you very much!

Email:

yaowining@yahoo.com

I will be very grateful for your email. Thank you very much!

(2) If you found this book helpful, and would like to help other people, would you be kind enough to leave a positive review for this book? It'd be greatly appreciated! Thank you very much!

Review Link:

https://www.amazon.com/Ray-Yao

Thank you very much!

Source code download link:

https://forms.aweber.com/form/72/528312472.htm

Recommended Books

Advanced C++ In 8 Hours

Advanced Java In 8 Hours

AngularJS In 8 Hours

C# In 8 Hours

C++ In 8 Hours

Chinese Daily Expressions

HTML CSS In 8 Hours

JAVA In 8 Hours

JavaScript 50 Useful Programs

JavaScript In 8 Hours

JQuery In 8 Hours

Linux Command Line

PHP In 8 Hours

Python In 8 Hours

R In 8 Hours

Ruby In 8 Hours

Visual Basic In 8 Hours

My friends, See You!