JavaScript In 8 Hours

For Beginners
Learn JavaScript Fast!

Ray Yao

JavaScript
In 8 Hours

By Ray Yao

For Beginners Learn JavaScript 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

Preface

“JAVASCRIPT in 8 Hours” is a useful book for beginners. You can learn complete primary knowledge of JavaScript fast and easily. The straightforward definitions, the plain and simple examples, the elaborate explanations and the neat and beautiful layout feature this helpful and educative book. You will be impressed by the new distinctive composing style. Reading this book is a great enjoyment! You can master all essential JavaScript skill in 8 hours.

Start Coding Today!

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 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:

```
<script type = “text/javascript”>
alert (“Hello World”);
</Script>
```

(Output: Hello World)

Explanation:

“<script type = “text/javascript”>” is a tags 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.

JavaScript block can located anywhere in HTML.

Comment

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:

```
<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 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.

Note that each JavaScript command ends with semicolon**;**

Keywords

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

The following are part of 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:

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

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

Example:

```
<script type = “text/javascript”>
var abcde;
var abc888;
var my_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 space. *e.g.* “23var”, “abc de” are invalid variable name.

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

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:

```
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 string.

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

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

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

Note: Double quotes are always used in 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:

```
<script type = “text/javascript”>
alert (“JavaScript says \”Hello World! \” ”); </script>
```

(Output: **“**Hello World**”**)

Explanation:

\**”** outputs a double quotation mark. Note that **“**Hello World**”** has a double quotation with it. Another sample: \t makes a tab

alert (“Hello \t\t\t World”);   // Note here has three taps

(Output: Hello   World)

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:

```
<script type = “text/javascript”>
function test( ){
alert (“show a sample”);
                                                   }
test ( );
</script>
(Output:  show a sample)
```

Explanation:

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

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

When running the function test ( ), it will output “show a sample”.

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:

```
<script type = “text/javascript”>
function test( var arg ){
alert ( arg );
                                                }
test(“display a sample”);
</script>
(Output:  display a sample)
```

Explanation:

When test(“display a sample”) call the function test(var arg){**…**}, it will pass the “display a sample” to var arg. After var arg has received the data, it will pass the data inside the function. alert (arg) will use the data, and then outputs “display a sample”.

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:

```
function cube(var num){
return num*num*num
                                        }
alert ( cube(2) ); ( Output:   8 )
```

Explanation:

"cube (2)" means that cube(2) calls the function cube(var num){ }, and pass the argument "2" to var num.

"**return** num*num*num" returns the result to caller "cube(2)", you can treat it as "cube(2) = **return** num*num*num", assigning the value to cube(2).

"alert (cube(2))" will show the result of cube(2);

Variable Scope

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:

```
<script type = “text/javascript”>
var num=200;  // This variable num is global variable.
function test( ){
var num=100;  // This variable num is local variable }
alert (num);
</script>
(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 to the current web page.

Example:

<script type = “text/javascript”>

document.write (“Hello World Again!”); </script>

(Output: Hello World Again!)

Explanation:

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

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

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, remainder is 0.

Example:

var add=100+200;   alert (add); var div=800/2;   alert (div); var mod=10**%**3; alert (mod); var inc=10;   alert (**++**inc); var str=”abc”**+**”de”; alert (str);

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:

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:

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:

```
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

The syntax of conditional operator looks as following:

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

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

Example:

```
<script type = “text/javascript”>
var a=100; var b=200;
var result=(a<b) ? ”apple” : ”boy”; alert ( result );
</script>
```

(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:

```
var a=200;
var b=100;
if (a>b){
alert ( ”a is greater than b.” );
                                              }
```

(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:

```
var a=100; var b=200;
if (a>b){
alert ( ”a is greater than b.”)
                                   }
else {
alert ( “a is less than b” );
                                   }
```

(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 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:

```
var number=20;
switch ( number ) {
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;  }
```

(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 by specified number of times.

Example:

```
<script type = “text/javascript”>
   for (var x = 0; x <= 5; x++) {
   document.write ( "x " );

                                                  }

</script>
```

(Output: 012345)

Explanation:

var x = 0 is an initializer,

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.

Sample:

```
var counter=0;
while (counter < 8){
document.write ( “&” );
counter++;
                                        }
```

(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:

```
var counter=0;
do {
document.write ( “@” );
counter++;
} while (counter<8); ( 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.

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:

var color = **new 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:

```
<script type = “text/javascript”>
color = new Array(“yellow”, “purple”, “orange”);
document.write ( color[1] );
</script>
```

(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:

```
<script type = “text/javascript”>
var color = new Array(“yellow”, “purple”, “orange”);
var size = color.length; document.write ( size );
</script>
(Output:  3)
```

Explanation:

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

```
var arr = new Array( );
arr[0] = “I”;
arr[1] = “love”;
arr[2] = “JavaScript!”;
document.write( arr.join(“  ”) );  // join by spaces (  Output:  I love JavaScript!  )
```

Explanation:

arr.join(“ ”) can join all elements of the “arr” together by 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:

var arr = new Array(“A”, “B”, “C”);

arr**.reverse( );**   // reverse the element order var rev = arr.join(“,”);

document.write(rev);

(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:

var arr = new Array(“a”, “b”, “c”, “d”, “e”, “f”);

var sli =arr**.slice**(2, 5); document.write(sli);

(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 signify “e”, instead of “f”.

Sort Elements in Order

```
array.sort( )
```

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

Example:

```
var arr = new Array( 2, 5, 3, 1, 4, 6 );
arr.sort ( );
var sor = arr.join(“,”); document.write( sor );
```

(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:

```
var myarray = new Array( );
myarray[0] =”Mon”;
myarray[1] =”Tue”;
myarray[2] =”Wed”;
var threedays = myarray.toString( ); document.write( threedays );
(  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:

```
var myarray = new Array( );
myarray[0] =”Mon”;
myarray[1] =”Tue”;
myarray[2] =”Wed”;
myarray[3] =”Tue”;
myarray.indexOf ( “Tue” );
```

(Returns 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:

```
var myarray = new Array( );
myarray[0] =”Mon”;
myarray[1] =”Tue”;
myarray[2] =”Wed”;
myarray[3] =”Tue”;
myarray.lastIndexOf ( “Tue” );
( Returns 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.

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:

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

Explanation:

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

Ceil ( ) & Floor ( )

```
Math.ceil( );

Math.floor( );
```

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

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

Example:

var num = 9.5;

document.write (“Ceiling number is”+**Math.ceil(num)**); document.write (“Flooring number is”+**Math.floor(num)**); (Output: Ceiling number is 10.0

Flooring number is 9.0)

Explanation:

“Math.ceil(num));” returns an integer that is greater than or equal 9.5, the result is 10.0.

“Math.floor(num);” returns an integer that is less than or equal 9.5, the result is 9.0.

Max ( ) & Min ( )

```
Math.max( );

Math.min( );
```

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

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

Example:

var x = 100;

var y = 200;

document.write (“Greater number is”+**Math.max(x, y)**); document.write (“Less number is”+**Math.min(x, y)**); (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.

Math.PI & random( )

```
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:

var pi = **Math.PI**; var ran = Math.random( );

document.write (“The PI value is” + pi);

document.write (“The random number is” + ran);

```
( Output:  The PI value is 3.141592653589793
           The random number is 0.16869328)
```

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 0.16869328.

Date & Time

JavaScript often use 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:

var DateObject = **new Date( )**; alert (DateObject);

(Output: Sun Apr 12 12:23 UTC 2015)

Explanation:

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

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

Note that: To create an object of date, you can use other variable name, for instance: var **now**= new Date( ); var **TimeObjcet** = new Date( );

Date, Month, Year, Day

The Date Object has following method:

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

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

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

Example:

```
var now = new Date ( );
var y = now.getYear( );
var d = now.getDate( );
alert (  y+ “ ” + d );
(  Output:  2015 12 )
```

Explanation:

“now.getYear( )” returns the current year.

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

Hours, Minutes, Seconds

The Date Object has 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:

var now = new Date( );

var h = now.**getHours( );** var m = now.getMinutes( );

var s = now.**getSeconds( )**; alert (“Current time is” + h+ “**:**” + m + “**:**” + s); (
Output: 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:

```
var now = new Date( );
var utc = now.getUTCHours( );
var timezone = now.getTimezoneOffset( );
alert ( utc + timezone );
( Output: 16: 20:38   300 )
```

Explanation:

“getUTCHours( )” returns the UTC time.

“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 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:

```
var now = new Date();
m= now.setMonth( 0 ); // 0 signify January d = now.setDate( 18 );
y = now.setFullYear( 2015 );
document.write ( m + d + y );
( Output:  Jan 18 UTC 2015 )
```

Explanation:

“setMonth(0), setDate(18) and setFullYear(2015) set the month, date and year as Jan 18 2015.

Timer Function

```
window.setTimeout(  )
```

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

Example:

```
function autoTimer( )  // user-defined function
                                                                {
alert ( “Be Careful!” );
window.setTimeout( “autoTmer( )” , 10000 ); }
(  Output:  Be careful! )
```

Explanation:

window.setTimeout(“autoTimer( )” , 10000) has two arguments:

“autoTimer( )” calls the function “autotimer( ){ }”, which is a user-defined function.

“10000” represents 10000 milliseconds, which sets every 10000 milliseconds for autoTimer( ) to call function.

Alert “Be Careful!” displays every other 10 seconds.

Hour 5

String

String length

```
string. length
```

“string.length” can calculate the string length.

Example:

```
<script type = “text/javascript”>
var mystring = “I love JavaScript”;
var stringsize = mystring.length; document.write( “String length is” + stringsize
);
</script>
```

(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:

```
<script type = “text/javascript”>
var string1 = “JavaScript”;
var string2 = “is”;
var string3 = “very easy”;
alert ( string1 + string2 + string3 ); </script>
( Output:  JavaScript is very easy  )
```

Explanation:

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

Search a Character

```
string.charAt( )
```

“string.chart(number)” is used to find a character at a string.

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

Example:

var mystring = “JavaScript is easy”;

var charac = mystring**.charAt (2);** alert (charac);

(Output: v)

Example:

“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:

var mystring = "Hello World!";

var big = mystring**.toUpperCase( );** var small = mystring**.toLowerCase( );** document.write(“Uppercase is” + big + “\n”);

document.write(“Lowercase is” + small);

```
(  Output:  HELLO WORLD!
            hello world!       )
```

Explanation:

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

“\n” 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 to elements of the array by space.

Example:

var mystring = “ JavaScript for Beginners”

var myarray = mystring**.split(“ ”);** document.write(“\n” + myarray[0]);

document.write(“\n” + myarray[1]);

document.write(“\n” + myarray[2]);

```
(  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:

var mystring = “JavaScript is easy.”;

var sub = mystring.**substr(4, 6);** alert (sub);

(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:

var num1 = 12; var num2 = 68;

var str1 = num1.**toString( );** var str2 = num2.**toString( );** var sum = str1 + str2;

document.write(sum);

(Output: 1268)

Explanation:

“num1.toString( )” and “num2.toString( )” change two numbers to string data type. Therefore, the result of str1adding 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:

```
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;
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:

var mystring = "Please find where *find* word occurs!"; var place = mystring.indexOf("find");

alert (place);

(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 string, indexOf( ) returns -1.

Search Specific Text (2)

```
lastIndexOf( )
```

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

Example:

var mystring = "Please find where *find* word occurs!"; var place = mystring.lastIndexOf("find");

alert (place);

(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 string, indexOf( ) returns -1.

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 a object’s property.

Example:

house = new Object ( );

house**.**color = “white”; house**.**size = “big”; document.write(“House color: ” + house**.**color + “\n”); document.write(“House size: ” + house**.**size + “\n”); (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:

windows.history.back( );

(Returns: Previous web page.)

windows.history.forward( );

(Returns: Next web page.)

Explanation:

“back( )” goes to previous page.

“forward( )” goes to next page.

“window.history.method( )” is used to navigate the specified web page. This method is usually called by a button on web page, which will be mentioned in later chapter.

Go to Specified Page

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

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

Example:

window.history.go(1);

(Returns: Next web page.)

window.history.go(-1);

(Returns: Previous web page.)

window.history.go(0);

(Returns: Reload current web page.)

Explanation:

“window.history.method( )” is used to navigate the specified web page. This method is usually called by a button on web page, which will be mentioned in later chapter.

Open Customized Window

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

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

“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.

Example:

window.open(“MyWindow”, “width=300, height=200, status=no, memubar=yes”);

(Returns: A new customized window)

Explanation:

“MyWindow” is a new window name.

“width=300, height=200” specify the size of new window.

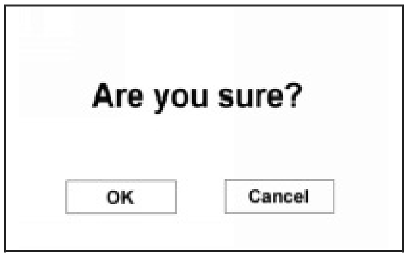
“status= no” means there is no status bar.

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

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:

```
var ask = window.clonfirm( “ Are you sure? ” );
if ( ask == true )
        { alert ( “All Right!” ); }
else {  alert (“No Good!”); }
(  Output:  All Right!  or  No Good!)
```

Explanation:

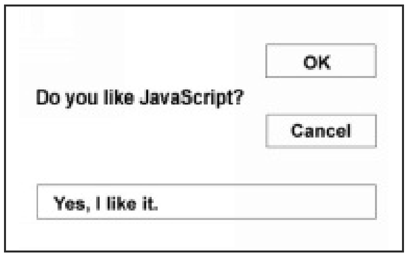
When clicking “OK” button, it outputs “All Right!”.

When clicking “Cancel” button, it outputs “No Good!”.

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 prompt dialog box, for example, enter “Yes, I like it.”, and then click the “OK” or “Cancel” button.

Example:

```
var ask = window.prompt ( “Do you like JavaScript?” );
alert ( ask );
(  Output:  Yes, I like it!  )
```

Explanation:

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 user.

Address Element by ID

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

“getElementById” access an element by its id.

“innerHTML” displays text in HTML document.

Example:

```
<body>
<div id = “position”>
</body>
<script type=”text/javascript”>
document.getElementById(“position”).innerHTML= “OK”;
</script>
```

(Output in “position”: OK)

Explanation:

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

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

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

Address Element by Tag Name

```
document.getElementByITagName(  ).innerHTML
```

“getElementByTagName” access an element by its tag name.

“innerHTML” displays text in HTML document.

Example:

```
<body>
<p> Notice Here </p>
</body>
<script type=”text/javascript”>
document.getElementByTagName(“p”).innerHTML= “OK”;
</script>
```

(Output in “p” tag: OK)

Explanation:

“<p> Notice Here </p>” define a tag name as “p” in HTML document.

“getElementByTagName(“p”)” access “p” tag.

“innerHTML” shows the text “OK” within “p” tag, which replaces the original text.

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:

```
var num=0;
while (num<10){
if (num==5)  break;
num++;
                                    }
document.write( num );
( 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 exit 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 codes, and then continue the next loop.

Example:

```
var num=0;
while (num<10){
num++;
if (num==5)  continue; document.write( num );
                                                }
```

(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 loop.

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:

```
<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:

<form>

<input type=“button” value=”Previous” **onClick=“history.back( )**”> </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.

Previous

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

Onload Event

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

```
onload  = function( );
```

“onload=function( )” means when a web page has loaded, function( ) immediately executes.

Example:

```
<html>
<body onload = alert ( “Welcome to my page!” )>
……
</body>
</html>
```

Explanation:

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

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

“onload=alert( )” means when a web page loaded, alert( ) runs at once, and shows the message “Welcome to my page!”.

Unload Event

When a web page has unloaded, an “unload” event occurs.

```
unload  = function( );
```

“unload=function( )” means when a web page has unloaded, function( ) immediately executes.

Example:

```
<html>
<body unload = confirm ( “Are you sure?” )>
……
</body>
</html>
```

Explanation:

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

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

“unload=confirm( )” means when a web page unloaded, confirm( ) runs at once, and shows the message “Are you sure?”.

Mouseover Event

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

```
mouseover = function( );
```

“mouseover = function( )” means when mouse pointer moves over an element, function( ) executes immediately.

Example:

```
<body>

<img src = “myphoto.jpg” mouseover = alert (“Please view the photo.”) ;>
</body>
```

Explanation:

<img> is a tag for defining an image.

“src=”myphoto.jpg”” specify an image “myphoto.jpg” showing in web page.

“mouseover=alert( );” means when mouse pointer moves over the image, alert( ) runs immediately, and displays the message “Please view the photo”.

Mouseout Event

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

```
mouseout = function( );
```

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

Example:

```
<body>
<img src = “myphoto.jpg” mouseout = alert (“See you!”) ;>
</body>
```

Explanation:

<img> is a tag for defining an image.

“src=”myphoto.jpg”” specify an image “myphoto.jpg” showing in web page.

“mouseout=alert( );” means when mouse pointer moves out of the image, alert( ) runs immediately, and displays the message “See you!”.

Onkeydown Event

When a key is pressed down, an “onkeydown” event occurs.

```
onkeydown = function( );
```

“onkeydown = function( )” means when a key is pressed down, a function( ) executes immediately.

Example:

```
<body>
<input type="text" onkeydown="myfunction( )">
</body>
<script>
function myfunction( )
{  alert ("You pressed a key."); }
</script>
```

Explanation:

<input type=”text”…> means there is an input field in web page for user entering data. Input field looks like this:

“onkeydown="myfunction( )"” means when a key is pressed down, myfunction( ) runs at once.

Onkeyup Event

When a key is released, an “onkeyup” event occurs.

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

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

Example:

```
<body>
<input type="text" onkeyup="myfunction( )">
</body>
<script>
function myfunction( )
{  alert ("You released a key."); }
</script>
```

Explanation:

<input type=”text”…> means there is an input field in web page for user entering data. Input field looks like this:

“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:

```
<body>
<input type="text" onfocus="myfunction( )">
</body>
<script>
function myfunction( )
{  confirm ("Are you ready to input?."); }
</script>
```

Explanation:

“onfocus="myfunction( )"” means when the user moves the cursor onto 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:

```
<body>
<input type="text" onblur="myfunction( )"> </body>
<script>
function myfunction( )
{  confirm ("Are you sure to leave?."); }
</script>
```

Explanation:

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

Hour 8

Work with Form

Form Basic

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

Example:

```
<form id=”myform” method=”get” action=”f.php”>
<input id=”txt” type=”text”>
<input id=”pas” type=”password”>
<input id=”rad” type=”radio” value=”val2”>
<input id=”chb” type=”checkbox” value=”val3”>
<input id=”sub” type=”submit” value=”OK”>
<input id=”rst” type=”reset” value=”Cancel”>
</form>
```

Explanation:

“id” means a name of the element.

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

“get” method sends link data openly, small quantity.

“post” method sends link data secretly, more quantity.

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

“type” specify the type of element.

Access Form Attributes

“window.document.forms” object can access the form attributes.

```
document.forms. id. attribute
```

“document.forms. id. attribute” can access the attribute in a form.

Example: (Take example of previous page's form.)

```
var message1 = document.forms.myform.method;
var message2 = document.forms.myform.action;
document.write ( message1);
document.write ( message2);
(Output:  get   f.php  )
```

Explanation:

“document.forms.myform.method” can access the “method” in “myform”, the result is “get”.

“document.forms.myform.action” can access the “action” in “myform”, the result is “f.php”.

Assign Text Value

```
document.forms.id.element.value = “newvalue”;
```

“document.forms.id.element.value = “newvalue”;” can assign a new value to an element.

Example: (Take example of previous page's form.)

document.forms.myform.txt.**value** = "Hello World!";
document.forms.myform.txt.**style.fontFamily** = "Arial";
document.forms.myform.txt.**style.fontSize** = "16pt";
document.forms.myform.txt.**style.color** = "blue";
document.forms.myform.txt.**style.background** = "yellow"; (Output: Hello World!   )

Explanation:

“myform” is the form id.

“txt” is the “text field” id.

value = “Hello World!” assign a new value to text field.

style.fontFamily = “Arial” sets the font family.

style.fontSize = “16pt” sets the font size.

style.color = “blue” sets the font color.

style.background = “yellow” sets the background color.

Reset Method

```
reset( )
```

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

Example:

```
<form id=”myform” method=”get” action=”f.php”>
……
<input type="button" onclick="myfunction()" value="Clear” > </form">
<script>
function myfunction() {
document.getElementById("myform").reset(); }
</script>
```

(Result: The form data are completely cleared.)

Explanation:

“onclick="myfunction()” means when click the button, myfunction( ) immediately executes.

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

“reset( )” clears “myform” data.

Submit Method

```
submit( )
```

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

Example:

```
<form id=”myform” method=”get” action=”f.php”>
……
<input type="button" onclick="myfunction()" value="Submit” > </form">
<script>
function myfunction() {
document.getElementById("myform").submit(); }
</script>
```

(Result: The form data are submitted to the server.)

Explanation:

“onclick="myfunction()” means when click the button, myfunction( ) immediately executes.

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

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

“Select” Selection

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

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

Red v

Example:

```
<select id="mySelect" onchange="myfunction()"> <option value="Red">Red
  <option value="Blue">Blue
  <option value="Pink">Pink
</select>
<script type=”text/javascript”>
function myfunction( ) {
var v=document.getElementById("mySelect").value;
alert ("You selected: " + v);  }    // e.g. select Pink
</script>
(Output:  You selected: Pink  )
```

Explanation:

“getElementById("mySelect")” access “mySelect” by its id.

“getElementById("mySelect")**.**value” means to get a value.

“Radio” Selection

```
<input type=”radio” id= “myid” value= “myvalue”>
```

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

Example: (Display: ○red ○blue ○pink) <input **type= “radio**” id= “color” value= “Red”>Red <input **type= “radio”** id= “color” value= “Blue”>Blue <input **type= “radio”** id= “color” value= “Pink”>Pink <input type= “submit” value= “submit” onclick= “myfunct( )”>

```
<script>
function myfunct( ) {
var v=document.getElementById("color").value;
alert ( v );   // e.g. select Blue
}
</script>
```

(Output: Blue)

Explanation:

“getElementById("color")” access “color” by its id.

“getElementById("color")**.**value” means to get a value.

“CheckBox” Selection

```
<input type=”checkbox” id= “myid” value=
“myvalue”>
```

A group of check buttons allows multiple boxes to be check at any time and submit the selected value to process.

Example: (Display: □red □blue □pink) <input **type= “checkbox**” id= “color” value= “Red”>Red <input **type= “checkbox”** id= “color” value= “Blue”>Blue <input **type= “checkbox”** id= “color” value= “Pink”>Pink <input type= “submit” value= “submit” onclick= “myfunct( )”>

<script>

function myfunct( ) {

var v=document.getElementById("color").value;

alert (v); // *e.g.* select Blue Pink

}

</script>

(Output: Blue Pink)

Explanation:

“getElementById("color")” access “color” by its id.

“getElementById("color")**.**value” means to get a value.

JavaScript: Function ( ) (1)

The “href” attribute in a hyperlink usually includes code of “JavaScript: function( )”.

```
< a href= “javascript: function( )”> text </a>
```

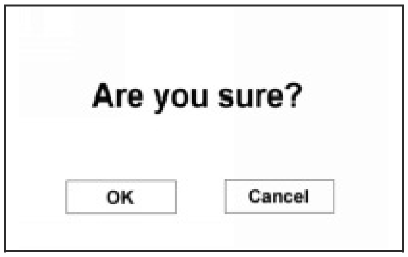
<a href> defines a hyperlink.

Example:

<p>

<a href = “**javascript:confirm** (‘Are you sure?’)”> Learn JavaScript! </a> </p>

(Output: Learn JavaScript!)


Are you sure?

Explanation:

When you clicks the hyperlink “Learn JavaScript!”, confirm dialog box displays “Are you sure?” immediately, you can click “OK” or “Cancel”.

JavaScript: Function ( ) (2)

The “href” attribute in a hyperlink usually includes code of “JavaScript: function( )”.

```
< a href= “javascript: function( )”> text </a>
```

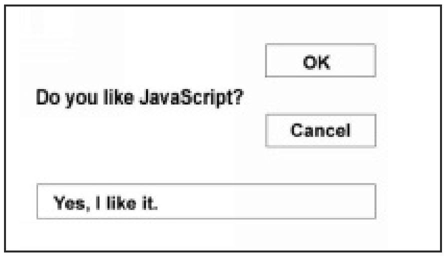
<a href> defines a hyperlink.

Example:

<p>

<a href = "**javascript:prompt** ('Do you like JavaScript?')"> JavaScript in 8 Hours! </a> </p>

(Output: JavaScript in 8 Hours!



Do you like JavaScript?

Yes, I like it.

)

Explanation:

When you clicks the hyperlink “JavaScript in 8 Hours!”, Prompt dialog box displays “Do you like JavaScript?” immediately, you can enter “Yes, I like it.”, then “OK”!

Appendix

HTML & CSS

HTML Review

Tag	**Description**
<!-- -->	html comments
<a>	for a hyperlink
<align>	align to left, right centered or justified
<b>	make the text bold
<big>	make font larger than regular font
<body>	define the body of document
<br>	a break of line
<button>	create a push button
<caption>	define caption in the table
<center>	make center of the contents
<col>	define column widths
<div>	divide body into blocks for css style
<form>	forms are used to collect user input.
<frame>	specify the location of each frame
<frameset>	define the frame width
<h1>	the first level heading
<h7>	the 7th level heading
<head>	define the document’s header
<hr>	draw a horizontal ruled line
<html>	define the html document
<i>	display italic text
<img>	add images to the content
<input>	defines an input element to a form
<li>	enclose the list items
<link>	link external resources
<map>	define image map hyperlinks
<meta>	declare data in its attributes
<object>	embed all kinds of content
<ol>	contains the items in an ordered list
<option>	submit the name/value pair in <select>

<p>	a new paragraph
<param>	add parameters
<pre>	show text including tabs, spaces, breaks
<s>	show text with a delete line
<script>	enclose JavaScript codes
<select>	create a selection menu
<small>	make font smaller than regular font
<span>	set css style for characters
<strong>	display text in a bold font
<style>	enclose CSS codes
<table>	a chart contains columns and rows
<td>	define table 's cell
<textarea>	create multiline text input fields
<title>	the html document's title
<tr>	define the row of table's cell
<u>	add an underline for text
<ul>	contains the items in an unordered list

HTML Symbol

Symbols	Source Codes
<	&lt;
>	&gt;
&	&amp;
“”	&quot; &quot
©	&copy;
®	&reg;
space	&nbsp;

CSS Review

CSS has 4 kinds of coding:

(1)

<style type = “text/css”>

tag{attribute**:** value}

tag**.**class{attribute**:** value}

tag #id{attribute**:** value}

.class{attribute**:** value}

#id{attribute**:** value}

</style>

(2)

<tag style = “attribute**:** value”>**……**</tag> **(3)**

<link rel = “stylesheet” title = “My Style”

type = “test/css” href = “myfile.css”>

(4)

@import url (“external/myfile.css”)

How time flies! It is time to say good-bye.

Conclusion

My friend,

I will appreciate so much if you write a positive review for this book. Thank you very much for your support!

Sincerely

Best Regards

Ray

My friend, See you!