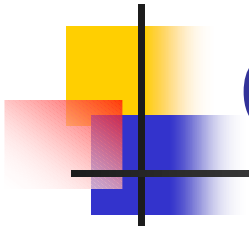
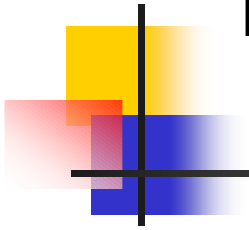


CSC309 Tutorial - JavaScript

Lei Jiang

Feb. 10, 2003

Embedding JavaScript



- To insert a script in an HTML document, use the <script> tag.
- Use the type attribute to define the scripting language.

```
<html>
<head>
</head>
<body>
    <script type="text/javascript">
        document.write("Hello World!")
    </script>
</body>
</html>
```

Embedding JavaScript (cont'd)

- Use HTML comment tag to provide compatibility with older browsers that don't support <script> tag

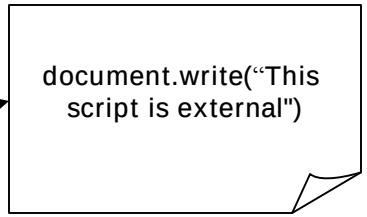
```
<script type="text/javascript">
<!--
    some statements
//-->
</script>
```

- The two forward slashes (//) are a JavaScript comment symbol
  - prevent the JavaScript from trying to compile the line

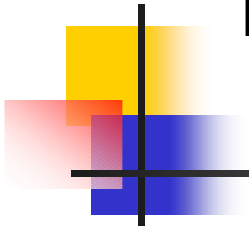
Embedding JavaScript (cont'd)

- **External JavaScript**
  - Write a script in an external file, and save with .js file extension.

```
<html>
<head>
</head>
<body>
<script src="xxx.js"></script>
</body>
</html>
```

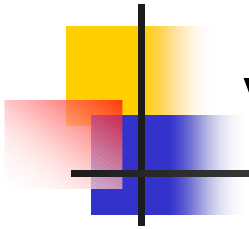


  - **The external script cannot contain the <script> tag**



Embedding JavaScript (cont'd)

- Can place >1 scripts in the document
  - in the body section and/or the head section
- Head section contains scripts
  - to be executed when the scripts are called,
  - or when an event is triggered
  - Used to ensure that the script is loaded before anyone uses it
- Body section contains scripts
  - to be executed when the body section is loaded
  - Used to generates the content of the page.



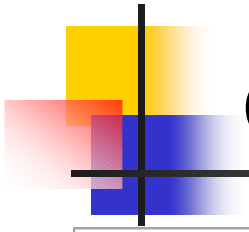
Variables

- **Variables can be declared with/without the var statement**

  var strname = *some value*

  strname = some value

- **Variables declared within a function is called *local***
  - **Is accessible only within that function**
  - **Is destroyed when the function exits**
- **Variables declared outside a function is called *globe***
  - **Is accessible anywhere**
  - **Is destroyed when the page is closed**



Operators

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Operators (cont'd)

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	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

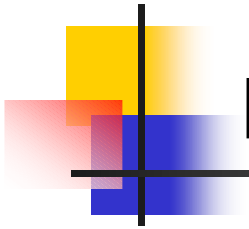
Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Operators (cont'd)

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 x != y returns true



Functions

- **Usually, functions are defined at the beginning of a file (in the head section) and are called in the document.**
- **Syntax:**

```
function myfunction(argument1,argument2,etc)
{
    some statements
}
```

- **Use return statement to return a value to the calling expression.**

```
function result(a,b)
{
    c=a+b; return c
}
```

- **Semicolon between statements are optional, except that there are multiple statements in a line.**

Conditional Statements

- **If and If...else Statement**
  - **Syntax:**

```
if (condition)
{
   code to be executed if condition is true
}
```

  - **Syntax:**

```
if (condition)
{
   code to be executed if condition is true
} else {
   code to be executed if condition is false
}
```

Conditional Statements (cont'd)

- **Example:**

```
<script type="text/javascript">
//If the time on your browser is less than 10,
//you will get a "Good morning" greeting.
var d=new Date()
var time=d.getHours()
    if (time<10)
    {
      document.write("<b>Good morning</b>")
    } else {
      document.write("Good day!")
    }
</script>
```

Conditional Statements (cont'd)

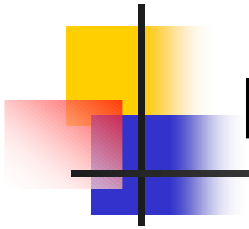
- **Switch Statement**
  - **Syntax:**

```
switch (expression)
{
case label1:
   code to be executed if expression = label1
   break
case label2:
   code to be executed if expression = label2
   break
default:
   code to be executed if expression is different from both label1
   and label2
}
```

Conditional Statements (cont'd)

- **Example:**

```
<script type="text/javascript">
//You will receive a different greeting based on what day it is.
// Note that  Sunday=0, Monday=1, Tuesday=2, etc.
var d=new Date()
theDay=d.getDay()
switch (theDay)
{
case 5: // Friday
    document.write("Finally Friday"); break
case 6: // Saturday
    document.write("Super Saturday"); break
case 0: // Sunday
    document.write("Sleepy Sunday"); break
default: //other days
    document.write("I'm looking forward to this weekend!")
}
</script>
```

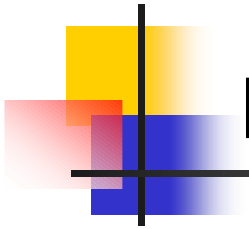


Looping

- **while**
  - **Syntax:**

    while (*condition*)
    {
        code to be executed
    }
  - **The block of code will be executed until the condition is false**
    - **May be executed 0 time**
- **do…while**
  - **Syntax:**

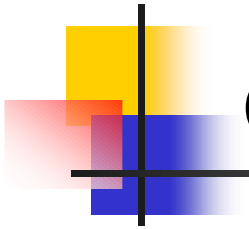
    do
    {
        code to be executed
    } while (*condition)*
  - **The block of code will be executed until the condition is false**
    - **Will be executed at least once**



Looping (cont'd)

- **for**
  - **Syntax:**

```
for (initialization; condition; increment)
{
   code to be executed
}
```

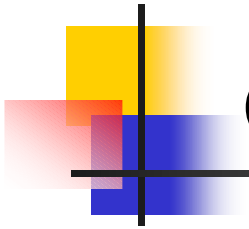


Objects

- **Creating objects: new operator**
  - **Syntax:**

    variable = new objectType(parameters)
  - **Example:**

```
// creates an instance of Date with
// the current date and time
currentDate = new Date()

//creates an instance of Date for January 1, 2005
currentDate = new Date(2005,1,1)
```



Objects (cont'd)

- **String Objects**
  - **Allow strings to be accessed as objects.**
  - **The `length` property identifies the string's length in characters.**
  - **Any variable containing a string value is able to use following methods:**

Objects (cont'd)

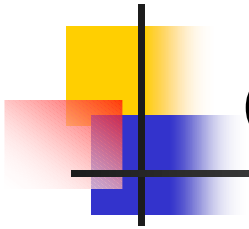
Method	Description
`charAt(index)`	Returns a string that consists of the character at the index specified.
`concat(s1, …, sn)`	Concatenates the specified strings to the string on which the method is invoked and returns the new string.
`substring(startIndex)`	Returns the substring of a string beginning at `startIndex`.
`substring(startIndex, endIndex)`	Returns the substring of a string beginning at `startIndex` and ending at `endIndex`.
`toLowerCase()`	Returns a copy of the string converted to lowercase.
`toString()`	Returns the string value of the object.
`toUpperCase()`	Returns a copy of the string converted to uppercase.
`split(separator, limit)`	Separates a string into an array of substrings based on the separator. If limit is specified, then the array is limited to the number of elements given by limit.
`valueOf()`	Returns the string value of the object.

Objects (cont'd)

- **Example:**

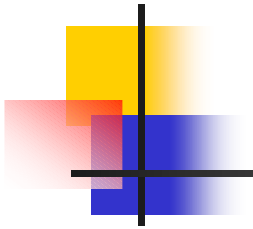
```
<HTML>
<HEAD>
<TITLE>Using the String Object Type</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!--
    function displayLine(text) {
    document.write(text+"<BR>")
    }

    s = new String("This is a test of the JavaScript String methods.")
    displayLine('s = '+s)
    displayLine('s.charAt(1) = '+s.charAt(1))
    displayLine('s.substring(22,32) = '+s.substring(22,32))
    displayLine('s.toLowerCase() = '+s.toLowerCase())
    displayLine('s.toUpperCase() = '+s.toUpperCase())
    split = s.split(" ")
    for(i=0; i<split.length; ++i)
    displayLine('split['+i+'] = '+split[i])
// -->
</SCRIPT>
</BODY>
</HTML>
```

```
s = This is a test of the JavaScript String methods.
s.charAt(1) = h
s.substring(22,32) = JavaScript
s.toLowerCase() = this is a test of the javascript string methods.
s.toUpperCase() = THIS IS A TEST OF THE JAVASCRIPT STRING METHODS.
split[0] = This
split[1] = is
split[2] = a
split[3] = test
split[4] = of
split[5] = the
split[6] = JavaScript
split[7] = String
split[8] = methods.
```



Objects (cont'd)

- **Array Objects**
  - **Array objects can store values with different types.**
  - **The index number starts at zero**
  - **Methods include:**
    - **`toString()` Returns a string version of an array. Array elements are separated by commas.**
    - **`concat(item1, ..., itemn)` Appends the items to the array and returns the modified array.**
    - **`pop()` Removes the last element from the array and returns it.**
    - **`push(item1, ..., itemn)` Appends the items to the array. If an item is an array, then the elements of the item array are appended.**
    - **`sort(comparisonFunction)` Sorts the elements of an array according to a comparison function.**
      - **If no comparison function is specified, the array elements are sorted in dictionary order.**
      - **If a comparison function is specified, it should take two parameters, p1 and p2, and return**
        - **a negative integer if p1 < p2,**
        - **zero if p1 = p2, and,**
        - **a positive integer if p1 > p2.**



Objects (cont'd)

- **Date Objects**
  - **Used to work with dates and times.**

Constructor	Description
`Date()`	Creates a `Date` instance with the current date and time.
`Date(dateString)`	Creates a `Date` instance with the date specified in the `dateString` parameter. `dateString`: "`month day, year hours:minutes:seconds`".
`Date(milliseconds)`	Creates a `Date` instance with the specified number of *milliseconds* since midnight January 1, 1970.
`Date(year, month, day, hours, minutes, seconds, milliseconds)`	Creates a `Date` instance with the date specified by the year, month, day, hours, minutes, seconds, and milliseconds integers. The year and month parameters must be supplied. If other parameters are included, then all preceding parameters must be supplied.

Objects (cont'd)

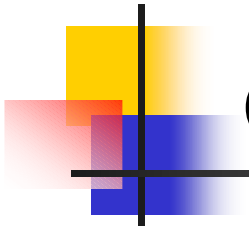
Method	Description
`getDate()`	Returns or sets the day of the month of the `Date` object.
`getDay()`	Returns the day of the week of the `Date` object.
`getHours()` `setHours(hours [, min, sec, ms])`	Returns or sets the hour of the `Date` object.
`getMinutes()` `setMinutes(min [, sec, ms])`	Returns or sets the minutes of the `Date` object.
`getMonth()` `setMonth(month [, date])`	Returns or sets the month of the `Date` object.
`getSeconds()` `setSeconds(sec [, ms])`	Returns or sets the seconds of the `Date` object.
`getTime()` `setTime(time)`	Returns or sets the time of the `Date` object.

`getYear()`
`getFullYear()`
`setYear(year)`
`setFullYear(year [,month, date])`
	Returns or sets the year of the `Date` object. The full year methods use four-digit year values.

`toGMTString()`	Converts a date to a string in Internet GMT (Greenwich Mean Time) format.

`toString()`
`toDateString()`
`toTimeString()`
	Returns a string value of a `Date` object.

`valueOf()`	Returns the number of milliseconds since midnight on January 1, 1970.

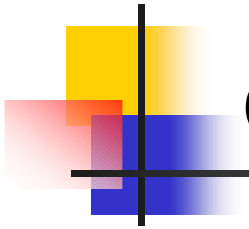


Objects (cont'd)

- **Example:**

```
<HTML>
<HEAD>
<TITLE>Using the Date Object Type</TITLE>
</HEAD>
<BODY>
<H1>Using the Date Object Type</H1>
<SCRIPT LANGUAGE="JavaScript"><!--
currentDate = new Date()
with (currentDate)
{
    document.write("Date: "+(getMonth()+1)+"/"+getDate()+"/"+ getFullYear() +"<BR>")
    document.write("Time: "+getHours()+":"+getMinutes()+":+ getSeconds())
}
// --></SCRIPT>
</BODY>
</HTML>
```

- **Uses the methods of the Date object type to write the current date and time to the current document object.**



Objects (cont'd)

- **Math object**
  - **provides a standard library of mathematical constants and functions.**
  - **Math is a built-in object and not an object type**

Property	Description
E	Euler's constant
LN2	The natural logarithm of 2
LN10	The natural logarithm of 10
LOG2E	The base 2 logarithm of e
LOG10E	The base 10 logarithm of e
PI	The constant π
SQRT1_2	The square root of _
SQRT2	The square root of 2

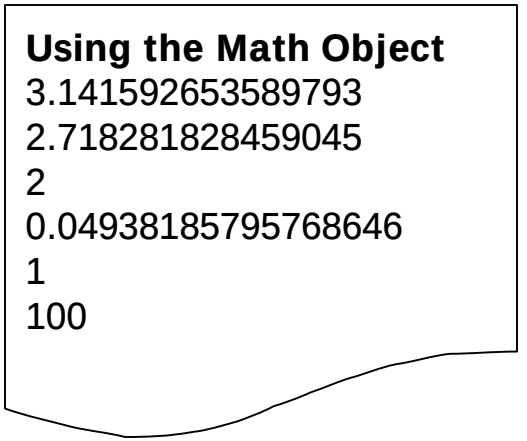
Objects (cont'd)

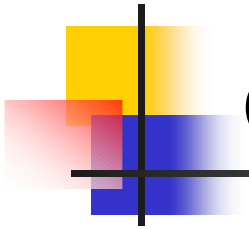
Method	Description
`abs(x)`	Returns the absolute value of *x*.
`acos(x)`	Returns the arc cosine of *x* in radians.
`asin(x)`	Returns the arc sine of *x* in radians.
`atan(x)`	Returns the arc tangent of *x* in radians.
`atan2(x,y)`	Returns the angle of the polar coordinate corresponding to (*x*,*y*).
`ceil(x)`	Returns the least integer that is greater than or equal to *x*.
`cos(x)`	Returns the cosine of *x*.
`exp(x)`	Returns e^x.
`floor(x)`	Returns the greatest integer that is less than or equal to *x*.
`log(x)`	Returns the natural logarithm of *x*.
`max(x,y)`	Returns the greater of *x* and *y*. If more than two arguments are supplied, the method returns the greatest of all the arguments.
`min(x,y)`	Returns the lesser of *x* and *y*. If more than two arguments are supplied, the method returns the least of all the arguments.
`pow(x,y)`	Returns «Xtags error: No such font: tag f»*xy*.
`random()`	Returns a random number between 0 and 1.
`round(x)`	Returns *x* rounded to the closest integer.
`sin(x)`	Returns the sine of *x*.
`sqrt(x)`	Returns the square root of *x*.
`tan(x)`	Returns the tangent of *x*.

Objects (cont'd)

- **Example:**

```
<HTML>
<HEAD>
<TITLE>Using the Math Object</TITLE>
</HEAD>
<BODY>
<H1>Using the Math Object</H1>
<SCRIPT LANGUAGE="JavaScript">
<!--
        document.write(Math.PI+"<BR>")
        document.write(Math.E+"<BR>")
        document.write(Math.ceil(1.234)+"<BR>")
        document.write(Math.random()+"<BR>")
        document.write(Math.sin(Math.PI/2)+"<BR>")
        document.write(Math.min(100,1000)+"<BR>")
// -->
</SCRIPT>
</BODY>
</HTML>
```





Objects (cont'd)

- **Example:**

```
<html>
<body>

<script type="text/javascript">
    no=Math.random()*10
    document.write(Math.floor(no))
</script>

</body>
</html>
```

Guidelines

- **JavaScript is Case Sensitive**
- **Open symbols, like ({ [" ', must have a matching closing symbol, like ' "] }).**
- **Extra spaces are ignored in JavaScript.**
  - **You can add white space to your script to make it more readable. (**name="Hege" v.s. name = "Hege"**)**
- **Use backslash (\) to break up a code line within a text.**

```
document.write("Hello \
World!")
```

```
document.write \
("Hello World!")
```

- **Insert special characters (like " ' ; &) with the backslash**

```
document.write ("You \& I sing \"Happy Birthday\".")
```

- **add a comment using two forward slashes "//"**