

# Tutorial 10 - Programming with JavaScript

HTML - JavaScript, Math 279, Fall 2013

1

## Outlines

- ❑ JavaScript introduction
- ❑ send output to a Web page
- ❑ working with JavaScript
  - variables, data
  - expressions, operators
  - functions
  - *arrays, conditional statements*
  - *program loops*

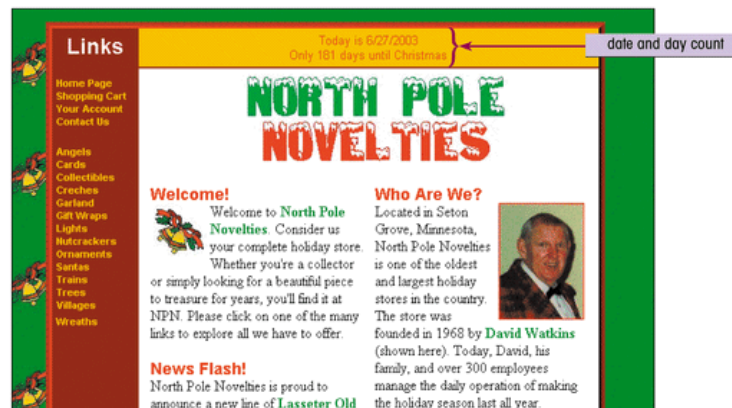
HTML - JavaScript, Math 279,  
Fall 2013

2

## Introduction to JavaScript

- ❑ an interpreted programming or script language
- ❑ used in Web site development to
  - automatically change a formatted date
  - cause a linked-to-page to appear in a popup window
  - cause text or a graphic image to change during a mouse rollover
  - etc.

## Example of Web Site using JavaScript



## Writing a JavaScript program

- ❑ JavaScript programs can either be placed directly into HTML file or saved in external files
  - using an external file hides the program code from users
  - complicated JavaScript program are usually placed in external file
- ❑ JavaScript program can be placed anywhere in a HTML file
  - placing it between `<head>` tags, separate the programming code from the Web page content and layout
  - placing it at the location where the program output is generated and displayed

HTML - JavaScript, Math 279,  
Fall 2013

5

## Using the `<script>` tag

- ❑ use `<script>` tag to distinguish JavaScript code
- ❑ `<script>` tag is a two-sided tag

```
<script src="URL" language="language">  
  Script command and comments  
</script>
```

- `src` attribute is required only if the program is placed in separate file
  - ❑ `URL` is the location of an external document containing the program code
- `language` attribute is the language that the program is written in (usually, JavaScript)

HTML - JavaScript, Math 279,  
Fall 2013

6

## Hiding script from older browsers

- ❑ browsers that do not support JavaScript may display the program code as part of the Web page body.
- ❑ hide the script from old browsers using comment tags

```
<script language="JavaScript">  
  <!-- Hide from non-JavaScript browsers  
    JavaScript commands  
  // Stop hiding from older browsers-->  
</script>
```

- ❑ browsers that doesn't support scripts ignores `<script>` tag

## Sending output to a Web page

- ❑ two methods to display text on a Web page
  - `document.write()` method
  - `document.writeln()` method
- ❑ syntax for these commands is

```
document.write("text");  
document.writeln("text");
```

- **text** is a string of characters for display, e.g.  
`document.write("cadler@mpl.gov");`

- ❑ term "**method**" means an action applied to something existing on a Web page or in the Web browser.

## document.write() & document.writeln()

- ❑ document.writeln() attaches a carriage return to the end of each text string; document.write() does not
- ❑ HTML tags can be used in text strings to format text and to insert images
- ❑ text string specified by document.write() method can be enclosed within either double or single quotation marks

```
document.write("<h3>News Flash!</h3>");  
document.write('<h3>News Flash!</h3>');
```

## JavaScript syntax issues

- ❑ JavaScript commands and names are **case-sensitive**
- ❑ JavaScript command lines end with a semicolon to separate it from the next command line in the program.
  - in some situations, the semicolon is optional
  - semicolons are useful to make your code easier to follow and interpret

## Working with variables and date

- ❑ A **variable** is a named element in a program that stores information
- ❑ The following restrictions apply to variable names:
  - the first character must be either a letter or an underscore character ( \_ )
  - the remaining characters can be letters, numbers, or underscore characters
  - variable names cannot contain spaces
  - you cannot use words that JavaScript has reserved for other purposes
- ❑ Variable names are case-sensitive.

## Example of a variable

- ❑ “x” is a variable to store the value of current year,  
**var x=2013;**  
**document.write(x);**
- ❑ The above code displays value 2013 on the Web page
- ❑ text can be combined with the variable value by using a plus symbol (+)  
**document.write(“It is year ” + x + “ now.”);**
  - this command displays the text “It is year 2013 now.” on the Web page

## Types of Variables

- ❑ JavaScript supports four different types of variables:
  - **numeric variables** can be a number, such as 13, 22.5, or -3.14159
  - **string variables** is any group of characters, such as "Hello" or "Happy Holidays!"
  - **Boolean variables** are variables that accept one of two values, either true or false
  - **null variables** is a variable that has no value at all

## Declaring a variable

- ❑ before using a variable, you need to create it; also known as **declaring a variable**
- ❑ to declare a variable in JavaScript, use **var** command or by assigning the variable a value
- ❑ any of the following commands is a legitimate way of creating a variable named "Month":
  - `var Month;`
  - `var Month = "April";`
  - `Month = "April";`
  - the first command creates the variable without assigning it a value, while the second and third commands both create the variable and assign it a value

## Declaring a variable (cont.)

- ❑ some tips:
  - it's good to include the `var` command whenever you create a variable.
  - it's good to place all of variable declarations at the beginning of a program along with comments describing the purpose of each variable
- ❑ the following are some sample JavaScript variables:
  - **Today** - containing information about the current date and time
  - **ThisDay** - storing the current day of the month
  - **ThisMonth** - storing a number indicating the current month
  - **ThisYear** - storing a number indicating the current year
  - **DaysLeft** - storing number of days until a selected date

## Working with dates

- ❑ JavaScript allows you to create a **date object**, which is an object containing date information.
- ❑ two ways to create a date object:

```
variable_name = new Date("month, day, year,  
hours:minutes:seconds");
```

or

```
variable_name = new Date("month, day, year,  
minutes, seconds");
```

  - **variable\_name** is the name of the variable that contains the date information
  - **month, day, year, hours, minutes, and seconds** indicate the date and time
  - Example

```
var Today = new Date("November, 19, 2013, 20:50:30");
```



## Working with dates (cont.)

- ❑ variable **Today** is a *date object*, has all the date and time information (November, 19, 2013, 20:50:30)
- ❑ For each part of a date object, a *method* is used to retrieve its value
  - to get the day of the month, use **getDate()** method:  
**var        ThisDay = Today.getDate();**

## Working with dates (cont.)

- to get the current month, use **getMonth()** method  
**var        ThisMonth = Today.getMonth();**
- JavaScript starts counting months with 0 for January, 1 for February, ... need increasing by 1 for correct month value  
**var        ThisMonth = Today.getMonth()+1;**
- to get the year value, use **getFullYear()** method  
**var        ThisYear = Today.getFullYear();**

## Methods of *date* object

METHOD	DESCRIPTION	VALUE
In the following examples, assume that the variable <code>Today</code> stores the date object: <code>Date("April, 8, 2004, 12:25:28")</code>		
<code>Today.getSeconds()</code>	Retrieves the seconds from the date	28
<code>Today.getMinutes()</code>	Retrieves the minutes from the date	25
<code>Today.getHours()</code>	Retrieves the hour from the date	12
<code>Today.getDate()</code>	Retrieves the day of the month from the date	8
<code>Today.getDay()</code>	Retrieves the day of the week from the date (0=Sunday, 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6=Saturday)	4
<code>Today.getMonth()</code>	Retrieves the month from the date (0=January, 1=February, ...)	3
<code>Today.getFullYear()</code>	Retrieves the four digit year number from the date	2004
<code>Today.getTime()</code>	Retrieves the time value, as expressed in milliseconds since December 31, 1969, 6 P.M.	1,081,445,128,000

## Exercise

Modify `mpl.htm` to write the following two sentences on the bottom of the Web page

**Today is November, 29th, 2013  
Only 27 days until Christmas**

where, the month (November), the day of month (29), the year (2013), and the days left (27) are integrated by using variables, **ThisMonth**, **ThisDay**, **ThisYear**, **DaysLeft**

## Working with expressions and operators

- ❑ **Expressions** are JavaScript commands that assign values to variables  
`var DaysLeft=999` assigns value 999 to the variable **DaysLeft**
- ❑ Expressions are created using variables, values, and **operators** (elements that perform actions within the expression)  
`var ThisMonth = Today.getMonth()+1;`  
↑  
operator
- ❑ arithmetic operator perform simple mathematical calculations

## The Math object

- ❑ JavaScript provides a **Math** object that supplies methods to perform specific calculations
- ❑ the syntax for applying a Math method is
  - `x = Math.method_name(variable);`
    - *method\_name* is the **method** that applies to a variable
    - *x* is a variable that has the resulting value from the Math method
    - example  
`x = Math.abs(-10);`      ➔ `x=10;`  
here, `abs` is a Math method

## A table of Math methods

MATH METHOD	DESCRIPTION
<code>Math.abs(<i>number</i>)</code>	Returns the absolute value of <i>number</i>
<code>Math.sin(<i>number</i>)</code>	Calculates the sine of <i>number</i> , where <i>number</i> is an angle expressed in radians
<code>Math.cos(<i>number</i>)</code>	Calculates the cosine of <i>number</i> , where <i>number</i> is an angle expressed in radians
<code>Math.round(<i>number</i>)</code>	Rounds <i>number</i> to the closet integer
<code>Math.ceil(<i>number</i>)</code>	Rounds <i>number</i> up to the next highest integer
<code>Math.floor(<i>number</i>)</code>	Rounds <i>number</i> down to the next lowest integer
<code>Math.random()</code>	Returns a random number between 0 and 1

HTML - JavaScript, Math 279,  
Fall 2013

23

## Creating JavaScript functions

- ❑ a **function** is a series of commands that performs an action or calculates a value
- ❑ a function consists
  - a **function name**, which identifies it
  - **parameters**, which are values used by the function
  - a set of **commands** that are processed when the function is used
- ❑ not all functions require parameters.
- ❑ the general syntax of a JavaScript function is:

```
function function_name(parameters) {  
    JavaScript commands  
}
```

HTML - JavaScript, Math 279,  
Fall 2013

24

## Creating JavaScript functions (cont.)

- ❑ function names are **case-sensitive**
- ❑ a function name must begin with a letter or underscore ( \_ ) and cannot contain any spaces
- ❑ the parameters must be placed within parentheses, following the function name, and the parameters must be separated by commas

## Performing an action with a function

- ❑ the following function displays a message with the current date:

```
function ShowDate(date) {  
    document.write("Today is " + date + "<br>");  
}
```

- function name is **ShowDate**, and it has one parameter, **date**
- there is one line in the function's command block, which displays the current date along with a text string

## Run a function

- ❑ To run a function, insert a JavaScript command containing the function name and any parameters it requires, this process is known as **calling** a function
- ❑ To call the ShowDate function, enter the following commands:

```
var Today = "12/01/2013";  
ShowDate(Today);
```

  - the first command creates a variable named **"Today"** and assigns it with the text string, **"12/01/2013"**
  - the second command runs the ShowDate function, using the value of the Today variable as a parameter
  - result should be **"Today is 12/01/2013"**

## Returning a value from a function

- ❑ use **return** command along with a variable or value to return a value from a function
- ❑ example using the Area function:

```
function Area(Width, Length) {  
    var Size = Width*Length;  
    return Size;  
}
```

  - the **Area** function calculates the area of a rectangular region and places the value in a variable named **"Size"**
  - the value of the Size variable is returned by the function

## Calling the Area function

- ❑ A simple JavaScript program is:

```
var x = 8;  
var y = 6;  
var z = Area(x,y);
```

- the first two commands assign the values 8 and 6 to the x and y variables
- the values of both of these variables are then sent to the Area function, corresponding to the Width and Length parameters
- the **Area** function uses these values to calculate the area, assigning the value to the z variable
- result is "48", which is assigned to the value of the z variable

HTML - JavaScript, Math 279,  
Fall 2013

29

## Placing a function in an HTML file

- ❑ function definition must be placed **before** the command that calls the function
- ❑ one programming convention is to place all of the function definitions between the <head> and </head> tags
- ❑ a function is executed only when called by a JavaScript command

HTML - JavaScript, Math 279,  
Fall 2013

30

## Placing a function in an HTML file

- ❑ To use a function on several Web pages, place the function in a separate file and access the function from each Web page
- ❑ To access the externally located function, use  
`<script src="URL" language="JavaScript">`  
`</script>`
  - where, *URL* is the filename and location of the external file containing the functions

## Exercise: build function XmasDays()

- ❑ in *mpl.htm* build a function that calculates the number of days from the current day to Christmas
- ❑ name this function XmasDays()



## XmasDays()

```
function XmasDays(currentDay) {  
    var XYear=currentDay.getFullYear();  
    var XDay=new Date("December, 25, 2013");  
    XDay.setFullYear(XYear);  
    var DayCount=(XDay-currentDay)/(1000*60*60*24);  
    DayCount=Math.floor(DayCount);  
    return DayCount;  
}  
  
var DaysLeft=XmasDays(Today);
```

HTML - JavaScript, Math 279,  
Fall 2013

33

## Setting Date Values

This figure shows additional JavaScript functions that allow you to set or change the values of date objects.

METHOD	DESCRIPTION
<i>DateObject.setSeconds(seconds)</i>	Set the seconds value of the <i>DateObject</i> to <i>seconds</i>
<i>DateObject.setMinutes(minutes)</i>	Set the minutes value of the <i>DateObject</i> to <i>minutes</i>
<i>DateObject.setHours(hours)</i>	Set the hours value of the <i>DateObject</i> to <i>hours</i>
<i>DateObject.setDate(date)</i>	Set the day of the month value of the <i>DateObject</i> to <i>date</i>
<i>DateObject.setMonth(month)</i>	Set the month value of the <i>DateObject</i> to <i>month</i>
<i>DateObject.setFullYear(year)</i>	Set the full year (four digit) value of the <i>DateObject</i> to <i>year</i>
<i>DateObject.setTime(time)</i>	Set the time of the <i>DateObject</i> to <i>time</i> , which is the number of milliseconds since December 31, 1969 at 6 P.M.

HTML - Javascript, Math 279,  
Fall 2013

34

## Working with conditional statement

- ❑ A **conditional statement** runs only when specific conditions are met, e.g., **if** statement.
- ❑ syntax

```
if (condition) {  
    JavaScript Commands  
}
```
- **condition** is an expression that is either true or false
  - *Boolean Expression*
  - ❑ if the condition is true, the JavaScript Commands in the command block are executed
  - ❑ if the condition is not true, then no action is taken

HTML - JavaScript, Math 279,  
Fall 2013

35

## Using comparison or logical operators

- ❑ to create a conditional statement, use
  - **comparison operator** (e.g., **==**, **!=**, **>**, **<**, etc): creates a Boolean expression that returns a value of either true or false
  - **logical operator** (e.g., **&&**, **||**, etc): connects two or more Boolean expressions and returns a value of either true or false

HTML - JavaScript, Math 279,  
Fall 2013

36

## Examples on boolean expressions

### □ two examples of Boolean expressions:

**x < 100;**

- if **x** is less than 100, this expression returns the value true; however, if **x** is 100 or greater, the expression is false

**y == 20;**

- If **y** variable have an exact value of 20, the expression is true
- Note: a double equal sign (==) rather than a single one is used!

## Comparison operators

- **==**, returns true if values are equal (**x == y**)
- **!=**, returns true if values are NOT equal (**x != y**)
- **>**, returns true if the variable on the left is greater than the variable on the right (**x > y**)
- **<**, returns true if the variable on the left is smaller than the variable on the right (**x < y**)
- **>=**, returns true if the variable on the left is greater or equal to the variable on the right (**x >= y**)
- **<=**, returns true if the variable on the left is smaller or equal to the variable on the right (**x <= y**)

## Example on logical operator

- ❑ Example of a logical operator
  - logical operator **&&** returns true only if all of the Boolean expressions are true  
`(x < 100) && (y == 20) ;`  
is true only if x is less than 100 AND y is equal to 20

## Logical operators

- ❑ **&&**, returns true if both expressions are true
  - e.g., `(x==20) && (y==25)`
- ❑ **||**, returns true if either expressions is true
  - e.g., `(x==20) || (y==25)`
- ❑ **!**, returns true if the expression is false;  
returns false if the expression is true
  - e.g., `!(x==20)`

## Using if...else statement

- general syntax

```
if (condition) {  
    JavaScript Commands if condition is  
    true  
} else {  
    JavaScript Commands if condition is  
    false  
}
```

- **condition** is an expression that is either true or false. One set of commands is run if the condition is true, and another set of commands is run if the condition is false

## Exercise

- In *mpl.htm*, REPLACE the second line of “document.write()” with the following code

```
if (DaysLeft>0) {  
    document.write(“Only ”+ DaysLeft+ “ days until Christmas.”);  
} else {  
    document.write( “Happy Holiday!” );  
}
```

## Using arrays

- ❑ An **array** is an ordered collection of values referenced by a single variable name.

- ❑ syntax

```
var variable = new Array(size);
```

- *variable*, name of the array variable
- *size*, number of elements in the array (optional)

- ❑ once an array is created, a value is created for each individual element in the array

## Creating DayTxt() function

```
function DayTxt(DayNumber) {  
    var Day = new Array();  
    Day[0]="Sunday";  
    Day[1]="Monday";  
    Day[2]="Tuesday";  
    Day[3]="Wednesday";  
    Day[4]="Thursday";  
    Day[5]="Friday";  
    Day[6]="Saturday";  
  
    return Day[DayNumber];  
}
```

## Creating MonthTxt() function

```
function MonthTxt(MonthNumber) {  
    var Month = new Array();  
    Month[1]="Jan.";   
    Month[2]="Feb.";   
    Month[3]="Mar.";   
    Month[4]="Apr.";   
    Month[5]="May";   
    Month[6]="Jun.";   
    Month[7]="Jul.";   
    Month[8]="Aug.";   
    Month[9]="Sept.";   
    Month[10]="Oct.";   
    Month[11]="Nov.";   
    Month[12]="Dec.";   
    return Month[MonthNumber];  
}
```

HTML - JavaScript, Math 279,  
Fall 2013

45

## Calling DayTxt(), MonthTxt() function

```
var MonthName=MonthTxt(ThisMonth);  
var DayName=DayTxt(ThisDay);  
  
document.write(DayName+', '+MonthName+ ' '+ThisDate  
+', ' +ThisYear);
```

HTML - JavaScript, Math 279,  
Fall 2013

46

## FINAL PROJECT Part I

This project is due by 11:59pm 12/23/2013 !!

Remember to put your name in a comment line of your code.

## FINAL PROJECT Part I (cont.)

- ❑ create `js_game.html` in your flash drive
- ❑ build ten buttons in `js_game.html` using HTML form
- ❑ index these ten buttons by numbers 0, 1, 2,... 8, 9



```
.....  
<body>  
<form>  
<input type="button" name="bt0" value="0">  
<input type="button" name="bt1" value="1">  
<input type="button" name="bt2" value="2">  
<input type="button" name="bt3" value="3">  
<input type="button" name="bt4" value="4">  
<input type="button" name="bt5" value="5">  
<input type="button" name="bt6" value="6">  
<input type="button" name="bt7" value="7">  
<input type="button" name="bt8" value="8">  
<input type="button" name="bt9" value="9">  
</form>  
</body>  
</html>
```

HTML - JavaScript, Math 279,  
Fall 2013

49

## Using **onLoad** attribute

- ❑ the **onLoad** attribute is used in **<body>** tag
- ❑ can be used to load an initial dialog window
- ❑ example

**<body onLoad="greeting()">**

where, **greeting()** is a user defined function

HTML - JavaScript, Math 279,  
Fall 2013

50

## Using **alert()** method

- ❑ the **alert()** method is used to display a dialog box
  - input: a message
  - output: a pop-up dialog box with
    - ❑ 1. the message that is defined by the designer;
    - ❑ 2. an OK button
- ❑ example: **alert("Hello there.")**
- ❑ in `js_game.html`, build function **greeting()**
  - using JavaScript within the `<head> ... </head>` tags
  - input: none
  - output: a pop-up dialog box shows a greeting message (use `alert()` method)

## Build **game()** function in `js_game.html`

- ❑ input: a number - the number that the user chose
- ❑ output: a dialog box telling the user whether the number is correct

```

function game(attempt){
    trys--;
    if (attempt==myNumber){
        alert("You won!");
        resetNumbers();
    }else{
        if (trys==0){
            alert("You Lose! My number is "+myNumber+".");
            resetNumbers();
        }else{
            if (attempt<myNumber){
                alert("Try a larger number!");
            }
            if (attempt>myNumber){
                alert("Try a smaller number!");
            }
        } //end inner else
    } //end outer else
} //end function

```

HTML - JavaScript, Math 279,  
Fall 2013

53

- add the following variables and function before game() function

```

var trys;
var myNumber;
function resetNumbers(){
    trys=3;
    myNumber=Math.floor(Math.random()*10);
}

```

- add the following command in greeting() function  
`resetNumbers();`
- add `onClick="game()"` in each of the button tags, example

```

<input type="button" name="bt0" value="0" onClick="game(0)">
<input type="button" name="bt1" value="1" onClick="game(1)">
<input type="button" name="bt2" value="2" onClick="game(2)">
... ..

```

HTML - JavaScript, Math 279,  
Fall 2013

54

The End!