JavaScript by Vetri

Creating a Programmable Web Page

Tutorial Objectives

- Understand basic JavaScript syntax
- Create an embedded and external script
- Work with variables and data
- Work with data objects and extract values from dates
- Work with expressions and operators
- Create and call a JavaScript function
- Work with arrays and conditional statements
- Learn about program loops

Server-Side Programs

- a user must be connected to the Web server to run the server-side script
- only the programmer can create or alter the script
- the system administrator has to be concerned about users continually accessing the server and potentially overloading the system

Client-Side Programs

- solve many of the problems associated with server-side scripts
- can be tested locally without first uploading it to a Web server
- are likely to be more responsive to the user
- can never completely replace server-side scripts

Introduction to JavaScript

- **JavaScript** is an interpreted programming or script language from Netscape.
- JavaScript is used in Web site development to such things as:
  - automatically change a formatted date on a Web page
  - cause a linked-to-page to appear in a popup window
  - cause text or a graphic image to change during a mouse rollover

Java vs. JavaScript

- Requires the JDK to create the applet
- Requires a Java virtual machine to run the applet
- Applet files are distinct from the XHTML code
- Source code is hidden from the user
- Programs must be saved as separate files and compiled before they can be run
- Programs run on the server side

- Requires a text editor
- Required a browser that can interpret JavaScript code
- JavaScript can be placed within HTML and XHTML
- Source code is made accessible to the user
- Programs cannot write content to the hard disk
- Programs run on the client side

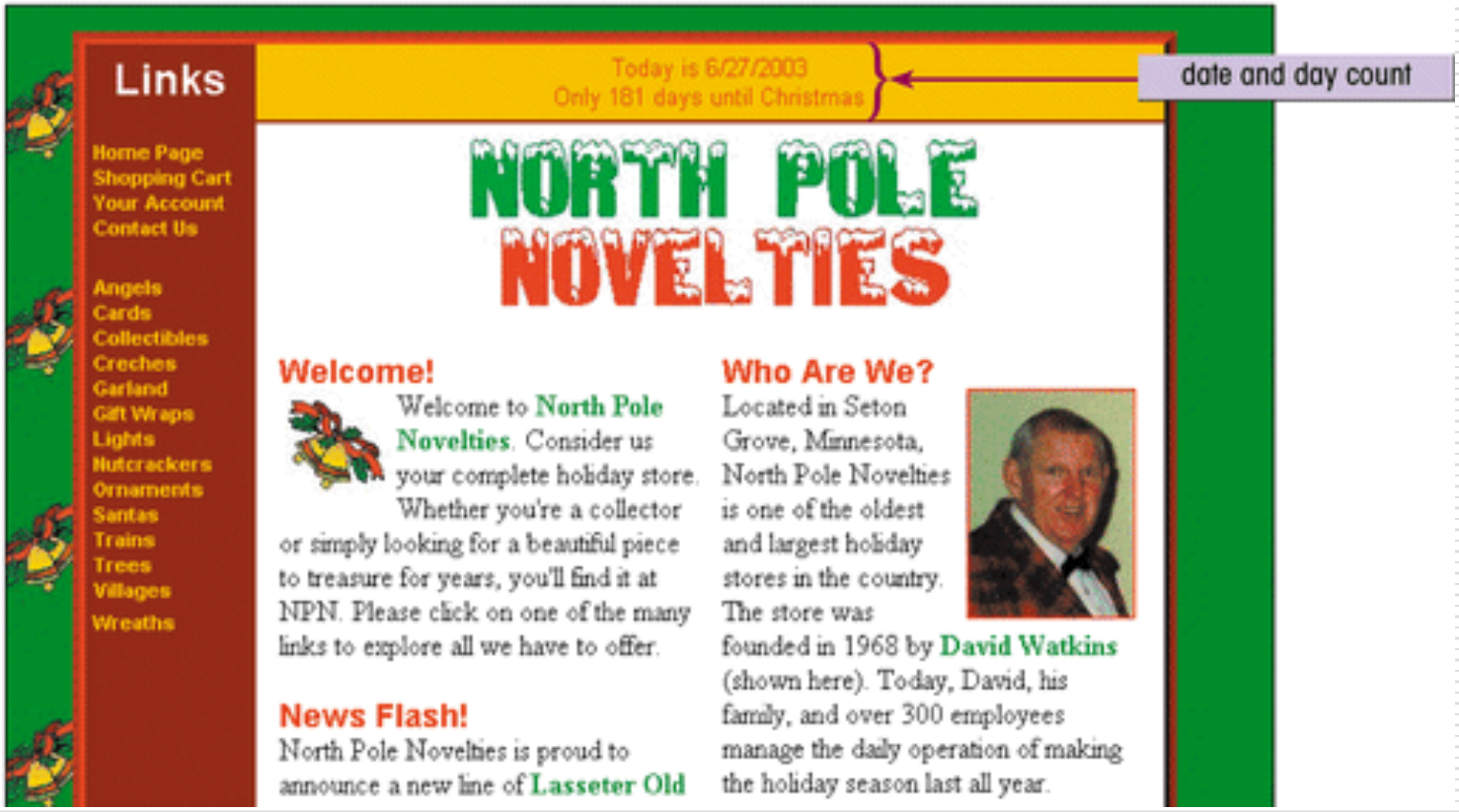
ECMAScript

- The responsibility for the development of a scripting standard has been transferred to an international body called the European Computer Manufacturers Association (ECMA).
- The standard developed by the ECMA is called ECMAScript, though browsers still refer to it as JavaScript.
- The latest version is ECMA-262, which is supported by the major browsers.

Other Client-side Languages

- Internet Explorer supports JScript.
- JScript is identical to JavaScript, but there are some JavaScript commands not supported in JScript, and vice versa.
- Other client-side programming languages are also available to Web page designers, such as the Internet Explorer scripting language, VBScript.

Example of Web Site using JavaScript



Writing a JavaScript Program

- The Web browser runs a JavaScript program when the Web page is first loaded, or in response to an event.
- JavaScript programs can either be placed directly into the HTML file or they can be saved in external files.
  - placing a program in an external file allows you to hide the program code from the user
  - source code placed directly in the HTML file can be viewed by anyone

Writing a JavaScript Program

- A JavaScript program can be placed anywhere within the HTML file.
- Many programmers favor placing their programs between **<head>** tags in order to separate the programming code from the Web page content and layout.
- Some programmers prefer placing programs within the body of the Web page at the location where the program output is generated and displayed.

Using the <script> Tag

- To embed a client-side script in a Web page, use the element:

  <script type="text/javascript" >

  script commands and comments

  </script>

- To access an external script, use:

  <script src="*url***"**

  type="text/javascript">

  script commands and comments

  </script>

Comments

- The syntax for a single-line comment is:
  // *comment text*
- The syntax of a multi-line comment is:
  /*
  comment text covering several lines
  ***/**

Hiding Script from Older Browsers

- You can hide the script from these browsers using comment tags:

  <script type="text/javascript">

  <!-- *Hide from non-JavaScript browsers*

  JavaScript commands

  // *Stop hiding from older browsers* **-->**

  </script>

- When a Web browser that doesn't support scripts encounters this code, it ignores the **<script>** tag.

Writing Output to a Web Page

- JavaScript provides two methods to write text to a Web page:
  - **document.write("text");**
  - **document.writeln("text");**
- The document.writeln() method differs from document.write() in that it attaches a carriage return to the end of each text string sent to the Web page.

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

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 & Data

- 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
- Variable names are case-sensitive.
- **document.write(Year);**

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 you can use a variable in your program, you need to declare a variable using the **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 = "December";

  Month = "December";

Working with Dates

- There are two ways to create a date object:
  variable = **new Date**("month day, year, hours:minutes: seconds")
  variable = **new Date**(year, month, day, hours, minutes, seconds")
  - *variable* is the name of the variable that contains the date information
  - *month*, *day*, *year*, *hours*, *minutes*, and *seconds* indicate the date and time

var Today=new Date("October 15, 2006");

var Today=new Date(2006, 9, 15);

Retrieving the Day & Time Values

- JavaScript stores dates and times as the number of milliseconds since 6 p.m on 12/31/69.
- Use built in JavaScript date methods to do calculations.
- If you want the ThisDay variable to store the day of the month. To get that information, apply the **getDate()** method.

  DayValue **=** *DateObject***.getDate()**

Retrieving the Month Value

- The **getMonth()** method extracts the value of the current month.
- JavaScript starts counting months with 0 for January, you may want to add 1 to the month number returned by the getMonth() method.
- *ThisMonth* **=** *Today*.**getMonth()+1;**

Retrieving the Year Value

- The **getFullYear()** method extracts the year value from the date variable.
- ThisYear **=** Today**.getFullYear();**

Working with Expressions and Operators

- Expressions are JavaScript commands that assign values to variables.
- Expressions are created using variables, values, and operators.
- The + operator performs the action of adding or combining two elements. For example,
  - var ThisMonth = Today.getMonth()+1;

OPERATOR	DESCRIPTION	EXAMPLE

+	Adds two values together	var Men = 20;
		var Women = 25;
		var TotalPeople = Men + Women;

–	Subtracts one value from another	var Price = 1000;
		var Expense = 750;
		var Profit = Price - Expense;

*	Multiplies two values together	var Width = 50;
		var Length = 25;
		var Area = Width*Length;

/	Divides one value by another	var People = 50;
		var TotalCost = 200;
		var CostperPerson = TotalCost/People;

%	Shows the remainder after dividing one value by another	var TotalEggs = 64;
		var CartonSize = 12;
		var EggsLeft = TotalEggs % CartonSize;

++	Increases a value by 1 (unary operator)	var Eggs = 12; var BakersDozen = Eggs++;
- -	Decreases a value by 1 (unary operator)	var Eggs = 12; var EggsIfOneIsBroken = Eggs- -;
–	Changes the sign of a value (unary operator)	var MyGain = 50; var YourLoss = – MyGain;

Operators

- Binary operators work on two elements in an expression.
- Unary operators work on only one variable.
  - unary operators include: the increment (++), decrement (--), and negation (-) operators.
- An increment operator is used to increase the value of the *x* variable by one.

```
x = 100;
y = x++;
```

Operators

- The decrement operator reduces the value of a variable by 1.

```
x = 100;
y = x--;
```

- The negation operator changes the sign of a variable:

```
x = -100;
y = -x;
```

Assignment Operators

- Expressions assign values using assignment operators. "=" is the most common one.
- Additional includes the += operator
- The following create the same results:

  x = x + y;

  x += y
- Either of the following increase the value of the *x* variable by 2:

  x = x + 2;

  x += 2

Assignment Operators

OPERATOR	DESCRIPTION
=	Assigns the value of the variable on the right to the variable on the left (x = y)
+=	Adds the two variables and assigns the result to the variable on the left (equivalent to x = x + y)
-=	Subtracts the variable on the right from the variable on the left and assigns the result to the variable on the left (equivalent to x = x – y)
*=	Multiplies the two variables together and assigns the result to the variable on the left (equivalent to x = x*y)
/=	Divides the variable on the left by the variable on the right and assigns the result to the variable on the left (equivalent to x = x/y)
%=	Divides the variable on the left by the variable on the right and assigns the remainder to the variable on the left (equivalent to x = x % y)

The Math Object & Math Methods

- Another way of performing a calculation is to use the JavaScript built-in Math methods.
- These methods are applied to an object called the Math object.
- The syntax for applying a Math method is:

  value **= Math.***method*(*variable*);
- For example,
  AbsValue = **Math.abs**(NumVar);

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

Creating JavaScript Functions

function *function_name*(*parameters*) **{**

JavaScript commands

}

- *parameters* are the values sent to the function (note: not all functions require parameters)
- **{** and **}** are used to mark the beginning and end of the commands in the function.

Creating JavaScript Functions

- Function names are case-sensitive.
- The function name must begin with a letter or underscore (_) and cannot contain any spaces.
- There is no limit to the number of function parameters that a function may contain.
- 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>");
}
```

- there is one line in the function's command block, which displays the current date along with a text string

Performing an Action with a Function

To call the ShowDate function, enter:

```
var Today = "3/9/2006";
ShowDate(Today);
```

- the first command creates a variable named "Today" and assigns it the text string, "3/9/2006"
- the second command runs the ShowDate function, using the value of the Today variable as a parameter
- result is "Today is 3/9/2006"

Returning a Value from a Function

To use a function to calculate a value use the return command along with a variable or value.

```
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

Placing a Function in an HTML File

- The function definition must be placed before the command that calls the function.
- One convention is to place all of the function definitions in the <head> section.
- A function is executed only when called by another JavaScript command.
- It's common practice for JavaScript programmers to create libraries of functions located in external files.

Home Page
Shopping Cart
Your Account
Contact Us

Angels
Cards
Collectibles
Creches
Garland
Gift Wraps
Lights
Nutcrackers
Ornaments
Santas
Trains
Trees
Villages
Wreaths

Today is 10/15/2006
Only 71 days until Christmas

NORTH POLE NOVELTIES

WELCOME TO OUR ONLINE store. Consider us your complete holiday store. Whether you're a collector or simply looking for a beautiful piece to treasure for years, you'll find it at NPN. Please click on one of the many links to explore all we have to offer.

News Flash!

North Pole Novelties is proud to announce a new line of **Lasseter Old Towne Village collectible houses**.

Who Are We?

Located in Seton Grove, Minnesota, North Pole Novelties is one of the oldest and largest holiday stores in the country. The store was founded in 1968 by **David Watkins** (shown here). Today, David, his family, and over 300 employees manage the daily operation of making the holiday season last all year.

```
document.write("Today is "+ThisMonth+"/"+
ThisDay+"/"+ThisYear+"<br />");
document.write("Only "+DaysLeft+
" days until Christmas");
```

```
<head>
<script src="library.js" type="text/javascript">
</script>
</head>
```

```
<script type="text/javascript">
var Today=new Date("October 15, 2006");
var ThisDay=Today.getDate();
var ThisMonth=Today.getMonth()+1;
var ThisYear=Today.getFullYear();
var DaysLeft=XmasDays(Today);
</script>
```

```
document.write("Today is "+ThisMonth+"/"+
ThisDay+"/"+ThisYear+"<br />");
document.write("Only "+DaysLeft+
" days until Christmas");
```

library.js

```
function XmasDays(CheckDay) {
   var XYear=CheckDay.getFullYear();
   var XDay=new Date("December, 25, 2006");
   XDay.setFullYear(XYear);
   var DayCount=(XDay-CheckDay) /(1000*60*60*24);
   DayCount=Math.round(DayCount);
   return DayCount;
}
```

Setting Date Values

JavaScript functions that allow you to set or change the values of date objects

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

Working with Conditional Statements

if (*condition***) {**

JavaScript Commands

}

- ***condition*** is an expression that is either true or false
- 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

Comparison, Logical, and Conditional Operators

To create a condition, you need one of three types of operators:

- a **comparison operator** compares the value of one element with that of another, which creates a **Boolean expression** that is either true or false
- a **logical operator** connects two or more Boolean expressions
- a **conditional operator** tests whether a specific condition is true and returns one value if the condition is true and a different value if the condition is false

An Example 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;**
  - the ***y*** variable must have an exact value of 20 for the expression to be true
  - comparison operator uses a double equal sign (**==**)

Comparison Operators

OPERATOR	DESCRIPTION
==	Returns true if variables are equal (x = y)
!=	Returns true if variables 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 less than the variable on the right (x < y)
>=	Returns true if the variable on the left is greater than or equal to the variable on the right (x >= y)
<=	Returns true if the variable on the left is less than or equal to the variable on the right (x <= y)

A Logical Operator

- The logical operator **&&** returns a value of true only if all of the Boolean expressions are true.

OPERATOR	DESCRIPTION	EXAMPLE
In the following examples, assume that x = 20 y = 25		
&&	Returns true when both expressions are true.	(x == 20) && (y == 25) returns true (x == 20) && (y == 20) returns false
||	Returns true when either expression is true.	(x == 20) || (y == 20) returns true (x == 25) || (y == 20) returns false
!	Returns true if the expression is false and false if the expression is true.	!(x == 20) returns false !(x == 25) returns true

A Conditional Operator

tests whether a specific condition is true and returns one value if the condition is true and a different value if the condition is false.

- **Message = (mail == "Yes") ? "You have mail": "No mail";**
- tests whether the **mail** variable is equal to the value "**Yes**"
  - if it is, the **Message** variable has the value "**You have mail**";
  - otherwise, the **Message** variable has the value "**No mail**".

Using an If...Else Statement

if (*condition***) {**

JavaScript Commands if true

} else

JavaScript Commands if false

}

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

Home Page
Shopping Cart

Today is 12/28/2006
Happy Holidays from Nroth Pole Novelties

if…else Conditional Statement

```
document.write("Today is " + ThisMonth +
      "/"+ThisDay+"/"+ThisYear+"<br />");
if (DaysLeft > 0) {
      document.write("Only "+DaysLeft+
            " days until Christmas");
} else {
      document.write("Happy Holidays from
            Nroth Pole Novelties");
}
```

Using Arrays

- An **array** is an ordered collection of values referenced by a single variable name.
- The syntax for creating an array variable is:

  var *variable* **= new Array(***size***);**
  - *variable* is the name of the array variable
  - *size* is the number of elements in the array (optional)
- To populate the array with values, use:
  *variable***[**i**]=***value***;**
  where i is the i^{th} item of the array. The 1^{st} item has an index value of **0**.

Using Arrays

To create and populate the array in a single statement, use:

var *variable* = new Array(*values*);

- ***values*** are the array elements enclosed in quotes and separated by commas
- **var** MonthTxt**=new Array(“”, “**January**”, “**February**”, “**March**”, “**April**”, “**May**”, “**June**”, “**July**”, “**August**”, “**September**”, “**October**”, “**November**”, “**December**”);**
  - January will have an index value of “1”.

```
<script type="text/javascript">
   var Today=new Date();
   var ThisDay=Today.getDate();
   var ThisMonth=Today.getMonth()+1;
   var ThisYear=Today.getFullYear();
   var DaysLeft=XmasDays(Today);
   var MonthTxt = new Array("", "January", "February", "March",
         "April", "May", "June", "July", "August", "September",
         "October","November", "December");
   document.write("Today is "+MonthTxt[ThisMonth]+" " +
         ThisDay+", "+ThisYear+"<br />");
   if (DaysLeft > 0) {
         document.write("Only "+DaysLeft+" days until Christmas");
   } else {
         document.write("Happy Holidays from North Pole
                                                Novelties");
   }
</script>
```

Creating the MonthText Function in *library2.js*

```
function MonthTxt(MonthNumber) {
        var Month=new Array();
        Month[0]="";
        Month[1]="January";
        Month[2]="February";
        Month[3]="March";
        Month[4]="April";
        Month[5]="May";
        Month[6]="June";
        Month[7]="July";
        Month[8]="August";
        Month[9]="September";
        Month[10]="October";
        Month[11]="November";
        Month[12]="December";
        return Month[MonthNumber];
}
```

Calling the MonthTxt Function

use the ThisMonth variable
to call the MonthTxt function
and then stores the result in
a new variable named "MonthName"

```
<head>
<script src="library2.js"
        type="text/javascript"></script>
</head>


var MonthName=MonthTxt(ThisMonth);
```

Working with Program Loops

- A **program loop** is a set of instructions that is executed repeatedly.
- There are two types of loops:
  - loops that repeat a set number of times before quitting
  - loops that repeat as long as a certain condition is met

The For Loop

- The **For loop** allows you to create a group of commands to be executed a set number of times through the use of a **counter** that tracks the number of times the command block has been run.
- Set an initial value for the counter, and each time the command block is executed, the counter changes in value.
- When the counter reaches a value above or below a certain stopping value, the loop ends.

The For Loop Continued

for (*start*; *condition*; *update*) {

JavaScript Commands

}

- ***start*** is the starting value of the counter
- ***condition*** is a Boolean expression that must be true for the loop to continue
- ***update*** specifies how the counter changes in value each time the command block is executed

```
<table border>
<tr>
<script>
   for (num = 1; num <=4; num++) {
      document.write("<td>"+num+"</td>");
   }
</script>
</tr>
</table>
```

For loop

resulting table

```
<table border>
<script>
for (rownum = 1; rownum <=3; rownum++) {
   document.write("<tr>");
   for (colnum = 1; colnum <=4; colnum++) {
      document.write("<td>" + rownum + "," + colnum + "</td>");
   }
   document.write("</tr>");
}
</script>
</table>
```

nested For loop

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4

resulting table

Specifying Counter Values in a For Loop

FOR LOOP	COUNTER VALUES
for (i = 1; i <= 5; i++)	i = 1, 2, 3, 4, 5
for (i = 5; i > 0; i--)	i = 5, 4, 3, 2, 1
for (i = 0; i <= 360; i += 60)	i = 0, 60, 120, 180, 240, 300, 360
for (i = 2; i <= 64; i *= 2)	i = 2, 4, 8, 16, 32, 64

The While Loop

- The **While loop** runs a command group as long as a specific condition is met, but it does not employ any counters.
- The general syntax of the While loop is:

while (*condition*) {

JavaScript Commands

}

  - ***condition*** is a Boolean expression that can be either true or false

```
<table border>
<tr>
<script>
    var num = 1;
    while (num <= 4) {
        document.write("<td>"+num+"</td>");
        num++;
    }
</script>
</tr>
</table>
```

While loop

resulting table

```
<table border>
<script>
var rownum = 1;
var colnum = 1;
while (rownum <=3) {
    document.write("<tr>");
    while (colnum <=4) {
        document.write("<td>" + rownum + "," + colnum + "</td>");
        colnum++;
    }
    document.write("</tr>");
    colnum = 1;
    rownum++;
}
</script>
</table>
```

nested While loop

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4

resulting table