Introduction to JavaScript programming

Web Programming

SCRPTING

- a scripting language is a simple, interpreted programming language
  - scripts are embedded as plain text, interpreted by application
  - *simpler execution model:* don't need compiler or development environment
  - *saves bandwidth:* source code is downloaded, not compiled executable
  - *platform-independence:* code interpreted by any script-enabled browser
  - *but:* slower than compiled code, not as powerful/full-featured

Benefits

- Adding dynamic features to Web pages
  - validation of form data
  - image rollovers
  - time-sensitive or random page elements
  - handling cookies
- defining programs with Web interfaces
  - buttons, text boxes, clickable images, prompts, frames

JavaScript Parts

- JavaScript comes in 3 parts:
  - Core language
    - Types, expression, control statements
  - Client-Side
    - Supports features related to the browser
  - Server-Side
    - Support features related to the server side

JavaScript and HTML

- Scripts are included in HTML files by placing them inside an HTML script *container* inside either the head or body section of the document.

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

- The HTML comment markers are to hide the JavaScript from browsers that don't understand the HTML script element.
- You can use language="JavaScript" instead type="text/javascript"

Simple JS program

```
<html>
<head>  <title>First JavaScript
    example</title> </head>
<body>
<h1>First JavaScript Page</h1>
<script type="text/javascript">
<!--
document.write("<hr>");
document.write("Hello World ");
// this a comment
document.write("<hr>");
//-->
</script>
</body>
</html>
```

view

- document.write displays text in page
- text to be displayed can include HTML tags
- as in C++/Java, statements end with ;
- JavaScript comments similar to C++/Java
  - // starts a single line comment
  - /*…*/ enclose multi-line comments

Communicate with the user

- JavaScript provides 3 functions (messageboxes)
  - alert()
  - confirm()
  - Prompt()

Using the alert() Method

- It is the easiest methods to use amongst alert(), prompt() and confirm().
- You can use it to display textual information to the user (simple and concise).
- The user can simply click “OK” to close it.

```
<head>
<script type="text/javascript">
        alert(“An alert triggered by JavaScript”);
</script>
</head>
```

view

Using the confirm() Method

- This box is used to give the user a choice either OK or Cancel.
- It is very similar to the “alert()” method.
- You can also put your message in the method.

```
<head>
<script type="text/javascript">
        confirm(“Are you happy with the class?”);
</script>
</head>
```

Using the prompt() Method

- This is the only one that allows the user to type in his own response to the specific question.
- You can give a default value to avoid displaying “undefined”.

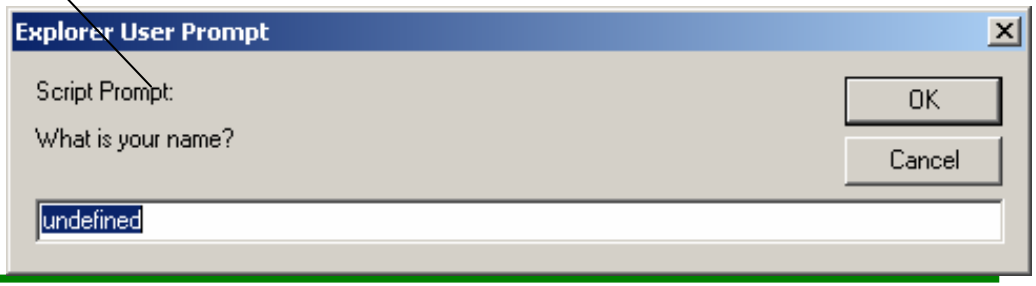
```
<head>
<script type="text/javascript">
        prompt(“What is your student id number?”);
        prompt(“What is your name?”,”No name”);
</script>
</head>
```

Three methods

```
<html>
<head>
  <title>alert</title>
</head>
<body>

<script language="JavaScript">
alert("This is an Alert method");
confirm("Are you OK?");
prompt("What is your name?");
prompt("How old are you?","20");
</script>
</body>
</html>
```

view



Data Types

- JavaScript allows the same variable to contain different types of data values.
- Primitive data types
  - Number: integer & floating-point numbers
  - Boolean: logical values “true” or “false”
  - String: a sequence of alphanumeric characters
- Composite data types (or Complex data types)
  - Object: a named collection of data
  - Array: a sequence of values
- Special data types
  - Null: an initial value is assigned
  - Undefined: the variable has been created by not yet assigned a value

Data types

- JS supports five primitive types:
  - Number, String, Boolean, Undefined, Null

```
var x=3.14;       // number
 x="a string";    // string
 x=true;          // Boolean
```

- The keyword var is used to declare variables since no typing is required.

Data Types

```
<html>
<head>
  <title>Data Types and
Variables</title> </head>

<body>
<script type="text/javascript">

   x = 1024;
document.write("<p>x = " + x
                  + "</p>");

  x = “hello";
  document.write("<p>x = " +
                  x + "</p>");
  </script>
</body>
</html>   view
```

- assignments are as in C++/Java
  - message = "howdy";
  - pi = 3.14159;
- variable names are sequences of letters, digits, and underscores: *start with a letter*
- variables names: case-sensitive
- *you don't have to declare variables, will be created the first time used*
- *variables are loosely typed, can assign different types of values*

Null & Undefined

- An “undefined” value is returned when you attempt to use a variable that has not been defined or you have declared but you forgot to provide with a value.
- Null refers to “nothing”
- You can declare and define a variable as “null” if you want absolutely nothing in it, but you just don’t want it to be “undefined”.

Operator and control statemetns

- standard C++/Java operators & control statements are provided in JavaScript
  - +, -, *, /, %, ++, --, …
  - ==, !=, <, >, <=, >=
  - &&, ||, !
  - if, if-else, while, do, …

Operators

- The concatenation operator is +.

```
x="3";
y="4";
z=x+y;          // "34"
a=y+x;          // "43"
```

- But + is also used for addition of numbers.

```
x=3;
y=4;
z=x+y;          // 7
```

- This duality is a problem since most data stored in HTML form elements is string data. We can parseFloat() (or parseInt()) data to force addition .

```
x="3";
y="4";
z= parseFloat(x)+parseFloat(y);  // 7
```

Expressions

- It is a set of literals, variables, operators that merge and evaluate to a single value.
  - Left_operand *operator* right_operand
- By using different operators, you can create the following expressions.
  - Arithmetic, logical
  - String and conditional expressions.

Operators

- Arithmetic operators
- Logical operators
- Comparison operators
- String operators
- Bit-wise operators
- Assignment operators
- Conditional operators

Arithmetic operators

- left_operand “operator” right_operand

Operator	Name	Description	Example
+	Addition	Adds the operands	3 + 5
-	Subtraction	Subtracts the right operand from the left operand	5 - 3
*	Multiplication	Multiplies the operands	3 * 5
/	Division	Divides the left operand by the right operand	30 / 5
%	Modulus	Calculates the remainder	20 % 5

Unary Arithmetic Operators

- Binary operators take two operands.
- Unary type operators take only one operand.
- Which one add value first, and then assign value to the variable?

Name	Example
Post Incrementing operator	Counter++
Post Decrementing operator	Counter--
Pre Incrementing operator	++counter
Pre Decrementing operator	--counter

Logical operators

- Used to perform Boolean operations on Boolean operands

Operator	Name	Description	Example
&&	Logical and	Evaluate to “true” when both operands are true	3>2 && 5<2
||	Logical or	Evaluate to “true when either operand is true	3>1 || 2>5
!	Logical not	Evaluate to “true” when the operand is false	5 != 3

Comparison operators

- Used to compare two numerical values

Operator	Name	Description	Example
==	Equal	Perform type conversion before checking the equality	“5” == 5
===	Strictly equal	No type conversion before testing	“5” === 5
!=	Not equal	“true” when both operands are not equal	4 != 2
!==	Strictly not equal	No type conversion before testing nonequality	5 !== “5”
>	Greater than	“true” if left operand is greater than right operand	2 > 5
<	Less than	“true” if left operand is less than right operand	3 < 5
>=	Greater than or equal	“true” if left operand is greater than or equal to the right operand	5 >= 2
<=	Less than or equal	“true” if left operand is less than or equal to the right operand	5 <= 2

String operator

- JavaScript only supports one string operator for joining two strings.

Operator	Name	Description	Return value
+	String concatenation	Joins two strings	“HelloWorld”

```
<script language=“JavaScript”>
        var myString = “”;
        myString = “Hello” + “World”;
        alert(myString);
</script>
```

Bit Manipulation operators

- Perform operations on the bit representation of a value, such as shift left or right.

Operator	Name	Description
&	Bitwise AND	Examines each bit position
|	Bitwise OR	If either bit of the operands is 1, the result will be 1
^	Bitwise XOR	Set the result bit, only if either bit is 1, but not both
<<	Bitwise left shift	Shifts the bits of an expression to the left
>>	Bitwise signed right shift	Shifts the bits to the right, and maintains the sign
>>>	Bitwise zero-fill right shift	Shifts the bits of an expression to right

Assignment operators

- Used to assign values to variables

Operator	Description	Example
=	Assigns the value of the right operand to the left operand	A = 2
+=	Add the operands and assigns the result to the left operand	A += 5
-=	Subtracts the operands and assigns the result to the left operand	A -= 5
*=	Multiplies the operands and assigns the result to the left operand	A *= 5
/=	Divides the left operands by the right operand and assigns the result to the left operand	A /= 5
%=	Assigns the remainder to the left operand	A %= 2

Order of Precedence

- Similar to C++/Java

Scope of a Variable

- When you use a variable in a JavaScript program that uses functions.
- A global scope variable is one that is declared outside a function and is accessible in any part of your program.
- A local variable is declared inside a function and stops existing when the function ends.

Control

```
if (Boolean expression) {

  block of statements;

}
```

```
if (Boolean expression) {

  block of statements;

}
else {

  block of statements;

}
```

```
if (Boolean expression) {

  block of statements;

}
else if (Boolean expression) {

  block of statements;

}
.
.
.
else if (Boolean expression) {

  block of statements;

}
else {

  block of statements;

}
```

```
for (var x=1 ; x<=100 ; x=x+1) {

   block of statements;

}
```

```
while (Boolean expression) {

   block of statements;

}
```

Functions

- function definitions are similar to C++/Java, except:
  - no return type for the function (since variables are loosely typed)
  - no types for parameters (since variables are loosely typed)
  - by-value parameter passing only (parameter gets copy of argument)

What is an Object?

- An object is a thing, anything, just as things in the real world.
  - E.g. (cars, dogs, money, books, …)
- In the web browser, objects are the browser window itself, forms, buttons, text boxes, …
- Methods are things that objects can do.
  - Cars can move, dogs can bark.
  - Window object can alert the user “alert()”.
- All objects have properties.
  - Cars have wheels, dogs have fur.
  - Browser has a name and version number.

Array

- An Array contains a set of data represented by a single variable name.
- Arrays in JavaScript are represented by the Array Object, we need to "new Array()" to construct this object.
- The first element of the array is "Array[0]" until the last one Array[i-1].
- Example:

```
myArray = new Array(5)
```

Array Example

```
<html>
<head>
  <title>array</title>
</head>
<body>

<script language="JavaScript">
            Car = new Array(3);
            Car[0] = "Ford";
            Car[1] = "Toyota";
            Car[2] = "Honda";
            document.write(Car[0] + "<br>");
            document.write(Car[1] + "<br>");
            document.write(Car[2] + "<br>");
</script>

</body>
</html>
```

- You can also declare arrays with variable length.
  - arrayName = new Array();
  - Length = 0, allows automatic extension of the length.
  - Car[9] = "Ford"; Car[99] = "Honda";

Conditional Statement

- “if” statement
- “if … else” statement
- “else if” statement
- “if/if … else” statement
- “switch” statement

“if” statement

- It is the main conditional statement in JavaScript.
- The keyword “if” always appears in lowercase.
- The condition yields a logical true or false value.
- The condition is true, statements are executed.

```
if (condition) { statements; }
```

“if.. else” statement

- You can include an “else” clause in an if statement when you want to execute some statements if the condition is false.

```
if (condition) { statements; }
  else { statements; }
```

```
if (condition)
{ statement; }
  else if (condition)
{ statement; }
  else { statement; }
```

“switch” statement

- Allows you to merge several evaluation tests of the same variable into a single block of statements
- Like C++/Java.

```
switch (expression) {
        case label1:
                statements; break;
        default:
                statements;
}
```

Looping Statement

- “for” Loops
- “for/in” Loops
- “while” Loops
- “do … while” Loops
- “break” statement
- “continue” statement

“for” statement

- One of the most used and familiar loops is the for loop.
- It iterates through a sequence of statements for a number of times controlled by a condition.
- The change_exp determines how much has been added or subtracted from the counter variable.
- Like C++/Java

```
for (initial_expression; test_exp; change_exp)
{ statements; }
```

“for/in” statement

- When the for/in statement is used, the counter and termination are determined by the length of the object.
- The statement begins with 0 as the initial value of the counter variable, terminates with all the properties of the objects have been exhausted.
  - E.g. array → no more elements found

```
for (counter_variable in object)
{  statements;  }
```

“while” statement

- The while loop begins with a termination condition and keeps looping until the termination condition is met.
- The counter variable is managed by the context of the statements inside the curly braces.
- Like Java/C++

```
initial value declaration;
while (condition) {
        statements;
        increment/decrement statement;
}
```

“do … while” statement

- The do/while loop always executes statements in the loop in the first iteration of the loop.
- The termination condition is placed at the bottom of the loop.
- Like C++/Java

```
do {
        statements;
        counter increment/decrement;
 } while (termination condition)
```

Example

- A simple calculator
  - View

Order of precedence

Precedence	Operator
1	Parentheses, function calls
2	, ~, -, ++, --, new, void, delete
3	*, /, %
4	+, -
5	<<, >>, >>>
6	<, <=, >, >=
7	==, !=, ===, !==
8	&
9	^
10	|
11	&&
12	||
13	?:
14	=, +=, -=, *=, …
15	The comma (,) operator