JavaScript

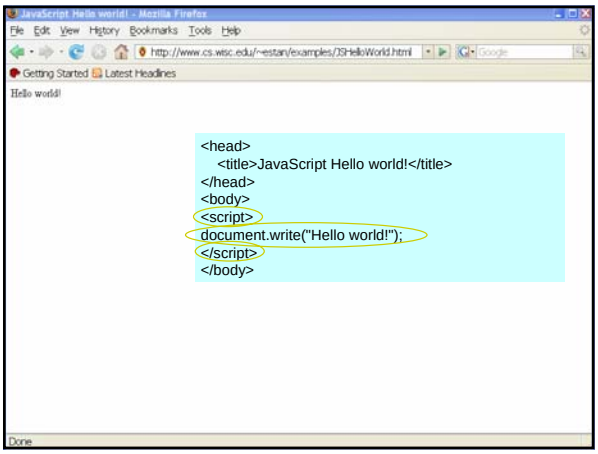
CS 640, Lecture 7

Overview of lecture

- The core language
- JavaScript and the browser

JavaScript: the big picture

- Interpreted, object-oriented, dynamic typing
- Not related to Java
- Introduced by Netscape with Netscape 2.0
- Microsoft's version called JScript
- Standardized by ECMA (European Computer Manufacturers Association) as ECMAScript
- Implemented by all modern browsers
- Good at user interaction, page manipulation
  - Relies heavily on event-driven programming



Syntax similar to C++

- Statement block delimited by “{“ and “}”
- Statements separated by ;
  - If single command per line, the ; can be left out
- Spaces, indentation have no semantic value
- Comments after // or between /* and */
- Identifiers contain $, _, letters and digits
  - Cannot start with a digit
  - Variable names and keywords case sensitive

The if and switch statements

```
if(expression)
  statement

if (expression1)
  statement1
else if (expression2)
  statement2
…
else if (expressionk)
  statementk
else
  catch_all_statement
```

```
switch(n){
case 1: // if n==1
  statements1
  break;
case 2: // if n==2
  statements2
  break;
default: // otherwise
  statements3
}
```

- The labels can be strings or even expressions

Loops

```
while (expression)
  statement

do
  statement
while (expression);

for (init;test;incr)
  statement
```

- `break` ends the innermost loop or `switch`
- `continue` jumps to the end of loop
- Labels can be used with nested loops

```
outer: for(i=0;i<5;i++){
  for(j=0;j<5;j++){
      a[i][j]++;
      if(a[i][j]<0)
          break outer;
  }
}
```

Functions and exceptions

- Defined as

```
function fname (args){
  statements;
}
```

- Invoked as

```
fname(args)
```

- Execution of function can be terminated with

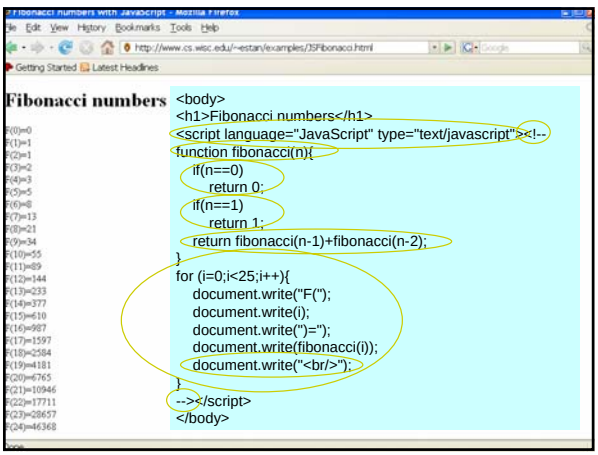
```
return expression;
```

- To throw an exception

```
throw expression;
```

- To catch an exception

```
try{
  statements1
}catch (e){
  statements2
}finally{
  statements3
}
```

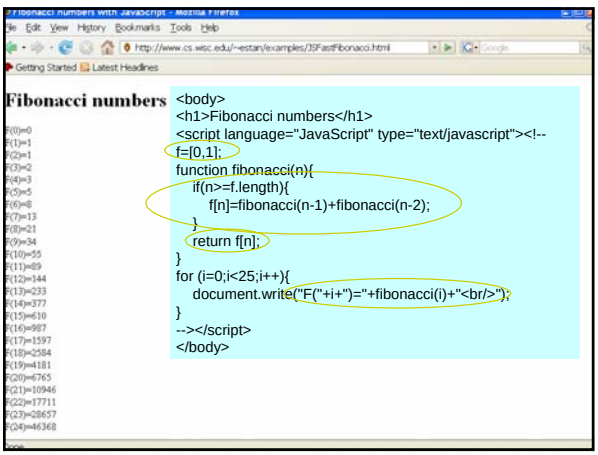


Most important types

- Numbers – `15` or `0xff` or `5.19` or `5.7e15`
- Strings – `'abc'` or `"abc"` or `'a\'\nb'`
  - Concatenation using +
- Booleans – `true` or `false`
- Functions – `sq=function(x){return x*x;}`
- Arrays – `[1,2,3]` or `['abc',[2,5],8]`
  - Variable size, can be sparse
- Objects – `{x:2,y:3}` or `{color:"blue",age:5}`
  - Very different from C++ and Java
  - Functions and arrays are also objects

JavaScript Arrays

- Elements' indices start with 0
- Initialization
  - Assigning an array literal `a=[]`
  - Using constructor `a=new Array(1,2,3)`
- Can grow by assignment to inexistent element
- The `length` property gives length – `a.length`
- Sparse: `a=['first']; a[99]='hundredth'` gives a two elements and a `length` of 100
- Special value `undefined` returned when inexistent element read



Useful array methods

- `join()` combines elements into a single string
  - Default separator ',' but can with argument to join
  - `String.split()` method gives array of substrings
- `reverse()` reverses the order of elements
- `sort()` sorts elements in alphabetic order
- `slice(`*`from`*`,`*`to`*`)` returns elements between indices *`from`* and *`to`*
  - Negative "indices" give distance from end of array
- `pop()` deletes and returns last element
- `push(`*`e`*`)` adds *e* to end of array, returns new length
- `shift()` deletes and returns first element (shifts others)
- `unshift(`*`e`*`)` adds *e* to beginning of array (shifts others)

JavaScript objects

- Objects are little more than associative arrays
  - `obj.property` is same as `obj["property"]`
  - To create a new property, just assign to it
  - Object methods are properties whose values are functions
    - They can use the keyword `this` to refer to object
- Prototype-based inheritance (no classes)
  - Object x also inherits all properties of `x.prototype`
  - The constructor is just a function that uses the keyword `this` to access the new object
  - "Class properties" simulated with properties of constructor

C++ vs. JavaScript objects

```
class Circle{
private:
    int x,y,radius;
    static const double pi;
public:
    Circle(int xc, int yc, int r){
        x=xc;y=yc;radius=r;};
    double getArea(){
        return radius*radius*pi;};
    double getDist(){
        return sqrt((double)(x*x+y*y));};
};
const double Circle::pi=3.14;
int main(int argc, char* argv[]){
    Circle c(2,4,3);
    printf("Area is %f\n",c.getArea());
    printf("Distance to origin is %f\n" ,c.getDist());
}
```

```
function Circle(xc,yc,r){
    this.x=xc;
    this.y=yc;
    this.radius=r;
}
Circle.prototype.getArea=function(){
    return this.radius*this.radius*Circle.pi;
}
Circle.prototype.getDist=function(){
    return Math.sqrt(this.x*this.x+this.y*this.y);
}
Circle.pi=3.14;
c=new Circle(2,4,3);
document.write("Area is
        "+c.getArea()+"<br/>");
document.write("Distance to origin is ");
document.write(c.getDist()+"<br/>");
```

Traversing an object

The `for/in` loop

```
point={x:2,y:4};
for (p in point)
  document.write(p);
```

- Does not go through inherited properties
- For arrays, the `for/in` loop goes through the indices of elements

The `in` operator

```
point={x:2,y:4};
has_x="x" in point;
has_z="z" in point;
```

- Checks inherited properties
- For arrays, the `in` operator tests whether the element with the given index exists

JavaScript variables

- All variables untyped – `x=4;x='xyz';x=[]`
- Variables are either global or local to function
  - No block scope!!!
- Variables are declared using the `var` keyword
  - Assigning to unused name implicitly declares it as a global
  - Reading an undeclared variable causes an error

Types	Passed/copied by	Comparison
boolean, number	value	value
string	immutable	value
object (arrays, functions)	reference	reference

Some JavaScript operators

Operator(s)	Operand type(s)	Operation performed
. [] ()	object, identifier array, integer function, argument	Property access Array index Function call
++,-- !	lvalue bolean	Increment, decrement Logical complement
*,/,%	number, number	Multiplication, division, remainder
+,-	number, number	Addition, subtraction
<<,>>,>>>	integer, integer	Shift operations
<,<=,>,>=	num./string, num./string	Comparisons (numeric or lexicographic)
==,===,!==,!===	any, any	(in)equality, (non)identity
Bitwise operators, boolean operators, conditional operator		
=	lvalue, any	Assignment
+=,-=,*=,…	lvalue, any	Assignment with operation

Overview of lecture

- The core language
- JavaScript and the browser

Adding JavaScript to a page

- Using the <script> </script> tag
  - Text between tags is JavaScript program
  - Can specify external file using src attribute
  - Executed as the document is loading
- Value of an attribute such as onclick
  - This type of code is called event handler
  - Executed when event happens
- Body of a URL using the javascript: “protocol”
  - Executed when browser loads URL

The context for JavaScript

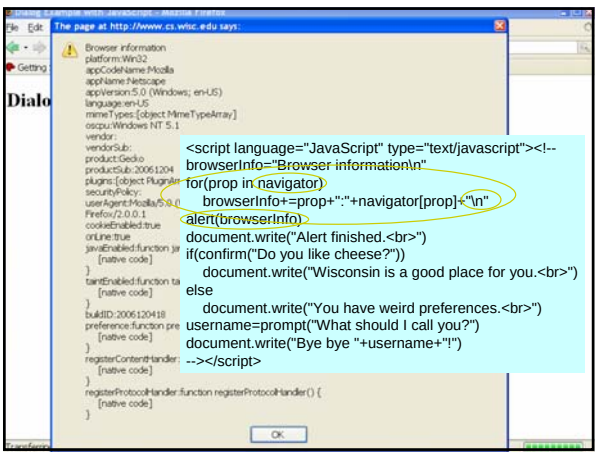
- All scripts on the same page share the same global variables, even if they are read using the `src` attribute
- All global variables are just properties of a “global object”
- The keyword `this` (unless inside object methods) and the variable `window` refer to the “global object”

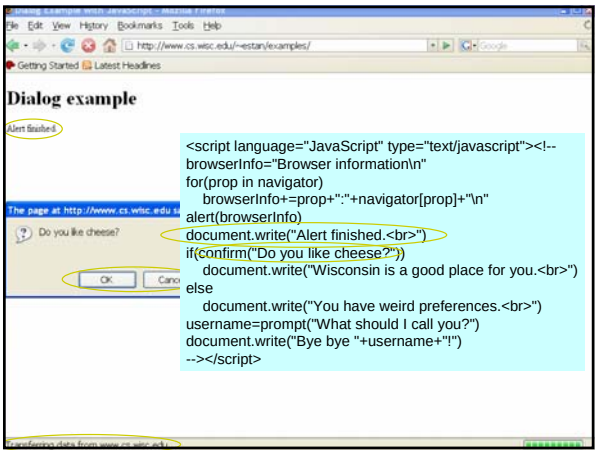
Some properties of `window`

- `document` holds the internal representation of the document displayed in the window
- `location` holds the URL of the current page
  - Can instruct the browser to load a new page by assigning its URL to `location`
- `status` (a.k.a. `defaultStatus`) holds the current message in the status bar
  - Typically shows the URL pointed to by link under mouse
- `navigator` has details about the browser software (application name, version, platform, etc.)

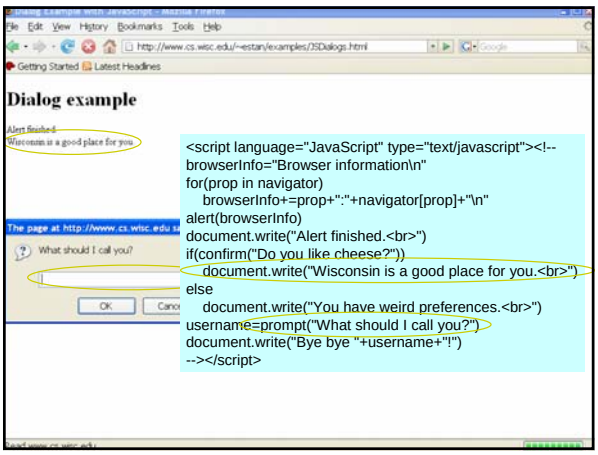
Some methods of `window`

- For interacting with user through dialog boxes
  - `alert()` displays a message
  - `confirm()` asks user to confirm or cancel
  - `prompt()` asks user to enter string
- For manipulating timers
- For manipulating windows (opening a new one, closing, moving, resizing, printing, etc.)
- Browser configuration may disallow certain functions (or the writing of certain properties)

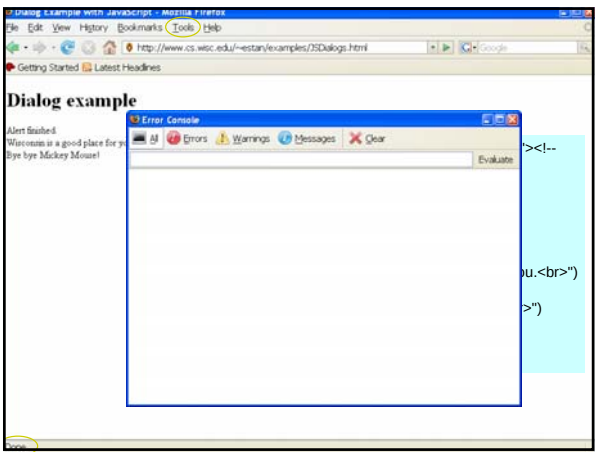





```
<script language="JavaScript" type="text/javascript"><!--
browserInfo="Browser information\n"
for(prop in navigator)
    browserInfo+=prop+":"+navigator[prop]+"\n"
alert(browserInfo)
document.write("Alert finished.<br>")
if(confirm("Do you like cheese?"))
    document.write("Wisconsin is a good place for you.<br>")
else
    document.write("You have weird preferences.<br>")
username=prompt("What should I call you?")
document.write("Bye bye "+username+"!")
--></script>
```




```
<script language="JavaScript" type="text/javascript"><!--
browserInfo="Browser information\n"
for(prop in navigator)
    browserInfo+=prop+":"+navigator[prop]+"\n"
alert(browserInfo)
document.write("Alert finished.<br>")
if(confirm("Do you like cheese?"))
    document.write("Wisconsin is a good place for you.<br>")
else
    document.write("You have weird preferences.<br>")
username=prompt("What should I call you?")
document.write("Bye bye "+username+"!")
--></script>
```



Some simple JavaScript events

- Event handler attributes supported by many tags
  - The specific handlers supported vary
  - Value of attribute is interpreted as JavaScript code

Handler	Triggered when	Comments
onclick	Mouse click on element	Return `false` to cancel default action
onmouseover	Mouse moves over el.	
onmouseout	Mouse moves off el.	
onmousedown	Mouse button pressed	
onmouseup	Mouse button released	
onkeydown	Key pressed down	Used for form elements and <body> Return `false` to cancel
onkeypress	Key pressed and released	
onkeyup	Key released	Used for form elements and <body>
onload	Document load complete	Used for <body> and <img>