Announcements

- **Friday 3 pm recitation will be held in Peabody 218**
  - 11 am recitation is not moving (will be in SN 014)
- **HW 4 due today**
  - 10% penalty for up to 24 hours late
  - Additional 25% penalty for every additional 24-hours after that
- **HW 5 expected to be posted by tomorrow**
- **2-min presentations by:**
  - Dillon Cockrell
  - Mallory Roseman

Quiz

// Trace the document.write statements
```
var fs = new Array(0,0,0,0,0);
var i;
var j;
for (i= 1; i <= 4; i++)
{
    for (j = 4; j >= 2; j-- )
        fs[j] = fs[j-1] + fs[j];
    fs[1] = 1;
    for (j = 1; j <= 4; j++)
        document.write(" "+ fs[j]);
    document.write("<br>");
}
```
Local variables

function examAvg(ex1, ex2, ex3)
    {  var total = ex1 + ex2 + ex3;
       var avg = total/3;
       return avg;
    }

- A function can have local variables defined in function
  - Created when you enter function; destroyed on exit
  - Value not preserved across function calls
  - Unknown outside function

- What if global variable name is same as local variable (or formal parameter) name?
  - No problem! These are still distinct.
  - Inside function,
    - Variable is first interpreted as local/formal (if defined)
    - Then, global

Example

function foo()
    {  var age = 20;
        ...
    }
    ...
    var age = 10;
    foo();

- What is age after call to foo()?
- What happens if we remove “var” here?
- Guideline?
  - Some recommend not to reuse names
  - Some recommend choosing the best mnemonic name
Can a function change its parameters?

```javascript
function f(x)
    { document.write("Before " + x + "<br>");
        x++;
        document.write("After " + x + "<br>");
    }

var age = 10;
document.write("Before call " + age + "<br>");
f(age);
document.write("After call " + age + "<br>");
```

- **Call by value!**
  - Function call creates a copy of age
  - The copy goes away once the function returns

Built-in functions

- **eval(string):**
  - Returns value of string, evaluated as expression
  - `eval("3 + 4 * 5")`  23
  - `eval("3 == 2")`  false
  - `eval("x = 10+5;")`  executes and returns 15

- **String properties:**
  - `var s = "ijklmno";`
  - `s.length`  7
  - `s.charAt(0)`  "i"
  - `s.charAt(1)`  "j"
  - `s.charAt(2)`  "k"
  - `s.charAt(3)`  "l"
  - `s.substring(2,5)`  "klmn"
  - `s.substring(3,s.length)`  "lmno"
**Built-in Math functions**

- **Already seen:**
  - Math.abs
  - Math.random
  - Math.ceil
  - Math.floor

- **Some more:**
  - Math.max(x, y) \(\text{Max of x and y}\)
  - Math.min(x, y) \(\text{Min of x and y}\)
  - Math.pow(x, y) \(x^y\)

---

**Recursion**

- **Can a function call another function?**
  - Yes!
    ```javascript
    function triple(x)
    {
      return x + double(x);
    }
    ```

- **Can a function call itself?**
  - Yes! This is known as “recursion”
    - Very powerful and concise way to specify some algorithms
  - Put away laundry
    1. If no items in basket, you’re done! Else continue below.
    2. Take 1 item from basket
    3. Fold it and put it away
    4. Put away laundry
  - Why does this work?
Will this work?

A.E. Neuman’s Spaghetti Sauce
(makes 2 cups)

1. Get a 1-quart Pan
2. Put on stove
3. Add 2 cups A.E. Newman’s Spaghetti Sauce
   - If (you need more than n>2 cups)
     - Get 2 cups from ready-made bottle
     - Add n-2 cups A.E. Neuman’s Spaghetti Sauce
   - Else
     - Pour from ready-made bottle
4. Heat
5. Cool

Try this recursion on your own

- Write a recursive function to compute n!
  - Hints:
    - n! = n(n-1)(n-2)...1 = n * (n-1)!
    - 0! = 1 (by definition)
  - Example:
    - 0! = 1
    - 1! = 1
    - 2! = 2 = 2 * 1!
    - 3! = 6 = 3 * 2!
    - 4! = 24 = 4 * 3!
    - 5! = 120 = 5 * 4!
    - 6! = 720 = 6 * 5!
    - ...

© Copyright: Jasleen Kaur. 2011.
Introduction to Programming