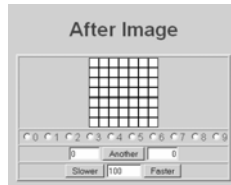




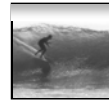
Announcements

Project 2 is posted.



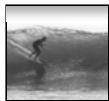
Tip of the Day: "Programming" is like brain surgery to many -- something that takes a genius to understand. But anyone who got into UW can learn it.

1



Programming Basics

When it comes to being precise about an algorithm, a programming language is better than English



The Plan

We will learn JavaScript over the next few lectures

- JavaScript is used with HTML in Web pages
- JavaScript is a contemporary programming language -- we will learn only its basics
- You will program in TextPad (or your favorite editor) and run your program with your browser

JavaScript is the way to make HTML "active"

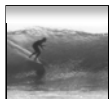
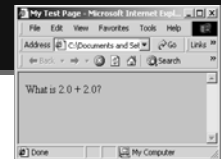
3



Begin with HTML

HTML is static ... the contents of the file are displayed as given

```
<html><head><title>My Test Page</title></head>
<body> <!-- No JavaScript yet, just HTML text -->
  What is 2.0 + 2.0?
</body>
</html>
```



JavaScript Needs HTML

JavaScript must be surrounded by <script> tags in a Web page ...

```
<html><head><title>My Test Page</title></head>
<body>
  What is 2.0 + 2.0?
  <script language="JavaScript">
    Put your JavaScript code here
  </script>
</body>
</html>
```

Script tags can be used anywhere where white space is OK, so use them as needed

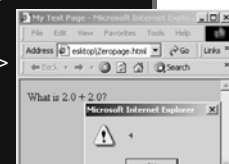


Browsers Process JS

When the browser comes to JavaScript, it processes it immediately

```
<html><head><title>My Test Page</title></head>
<body>
  What is 2.0 + 2.0?
  <script language="JavaScript">
    alert(2.0 + 2.0);
  </script>
</body>
</html>
```

Page not fully loaded





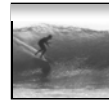
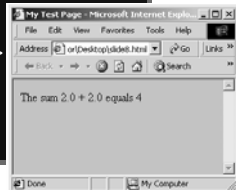
JS Can Build Pages

JavaScript can add to a page using the `document.write` command ...

```

<html><head><title>My Test Page</title></head>
<body>
  The sum 2.0 + 2.0 equals
  <script language="JavaScript">
    document.write(2.0 + 2.0);
  </script>
</body>
</html>

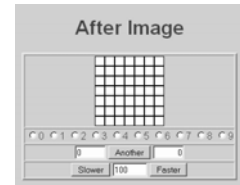
```



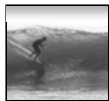
JavaScript is Cool

JavaScript has many slick applications so it's worth taking a couple of lectures to learn it

* We move on now to the basics, but first ...



8



Names In Programming

In normal language, names, and the things they name -- their values -- usually cannot be separated

- In programming most names change values ... a consequence of finite specification
- Titles (US_Open_Champ), Offices (Mayor), Roles (Juliet), etc. are familiar examples of names that change values
- Rules, Processes and Directions exploit the variable value: "Juliet moves to the window"

9



Variables

• Names in programming are *identifiers*

• The things they name are their *values*

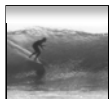
The package -- identifier & value -- is a *variable*, implying a possible change

- Identifiers have a specific structure in every programming language
- JS: letters, digits, _ start with letter, case sen.

x x_textOut MI5 long_variables_are_OK rate

hyphens-not-OK 007 no spaces end

10



Declarations

To *declare variables* is to state what variables will be used

- Required ... put declarations first in program
 - Use the word: `var`
 - Follow with a list of variables separated by ,
 - Terminate all statements with a semicolon ;
- ```
var x, input1, input2, rate;
```
- Give variables an initial value with =
- ```
var interestRate = 4, pi = 3.14159;
```

11



Values

Programming languages allow several *types* of values: numeric, strings of letters, Boolean

- numbers: 1 0 -433 6.022e+23 .01
- not numbers: 1,000 10⁶ 5% 7±2
- strings: "abc" 'efg' " " "B&B's" ""
- not strings: ' '<tab>' 'a' '\'
- Boolean: true false
- not Boolean: T F yes no

12



Assignment

The universal form of assignment:

`<variable> <assignment symbol> <expression>`

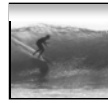
For example ...

`day = hours/24;`

- value of the variable on the left is changed to have the new value of expression on right
- read "=" as "is assigned" "becomes" "gets"
- right-to-left value flow

`=` is different in math and programming

13

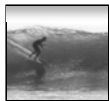


Expressions

Expressions are like "formulas" saying how to manipulate existing values to compute new values, e.g. `hours/24`

- Operators: + - * / % produce numbers
- Operators: < <= == != >= > on numbers (or strings for == and !=) produce Booleans
- Operators: && || ! on Booleans produce Booleans
- Grouping by parentheses is OK and smart
`seconds = ((days*24 + hours)*60 + min)*60`

14



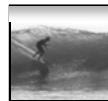
Overloading Plus

The + can be used to add numbers or join strings (concatenate)

- `5 + 5 ⇔ 10`
- `"a" + "b" + "c" ⇔ "abc"`
- `'5' + '5' ⇔ '55'`
- The operand type determines the operation
- Combine a number and string???
- `5 + '5' ⇔ '55'`
- Rule: With an operand of each type, convert number to string, concatenate

`⇔` is the symbol for "has the value"

15



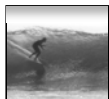
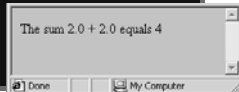
First JS Program, Revisited

Rewrite earlier code with new concepts

```

<html><head><title>My Test Page</title></head>
<body> The sum 2.0 + 2.0 equals
<script language="JavaScript">
var anumber = 2.0, another, answer;
another = 2.0;
answer = anumber + another;
document.write(answer);
</script>
</body>
</html>

```

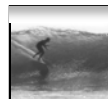


Conditional

Conditionals test if an expression is true or not

- General form ...
`if (<Boolean expression>)`
 `<Then statement>;`
- For example
`if (day == "Friday")`
 `evening_plan = "party";`

17



If-Then-Else

Branch both ways with If-Then-Else

- ```

if (<Boolean expression>)
 <Then statement>;
else
 <Else Statement>;

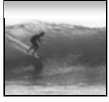
```
- Example ...  

```

if ((year%4)== 0) { ←
 leapYear = true;
 febDays = febDays+1;
}
else
 leapYear = false;

```

18



## Summary

Programming is the exact  
specification of an algorithm

JavaScript is typical ... with many rules

\* Learning strategy

- **Do the reading first**
- **Practicing is better than memorizing for learning the rules**
- **Use the program-save-reload-check plan**
- **Precision is your best friend**

19