Information Infrastructure II

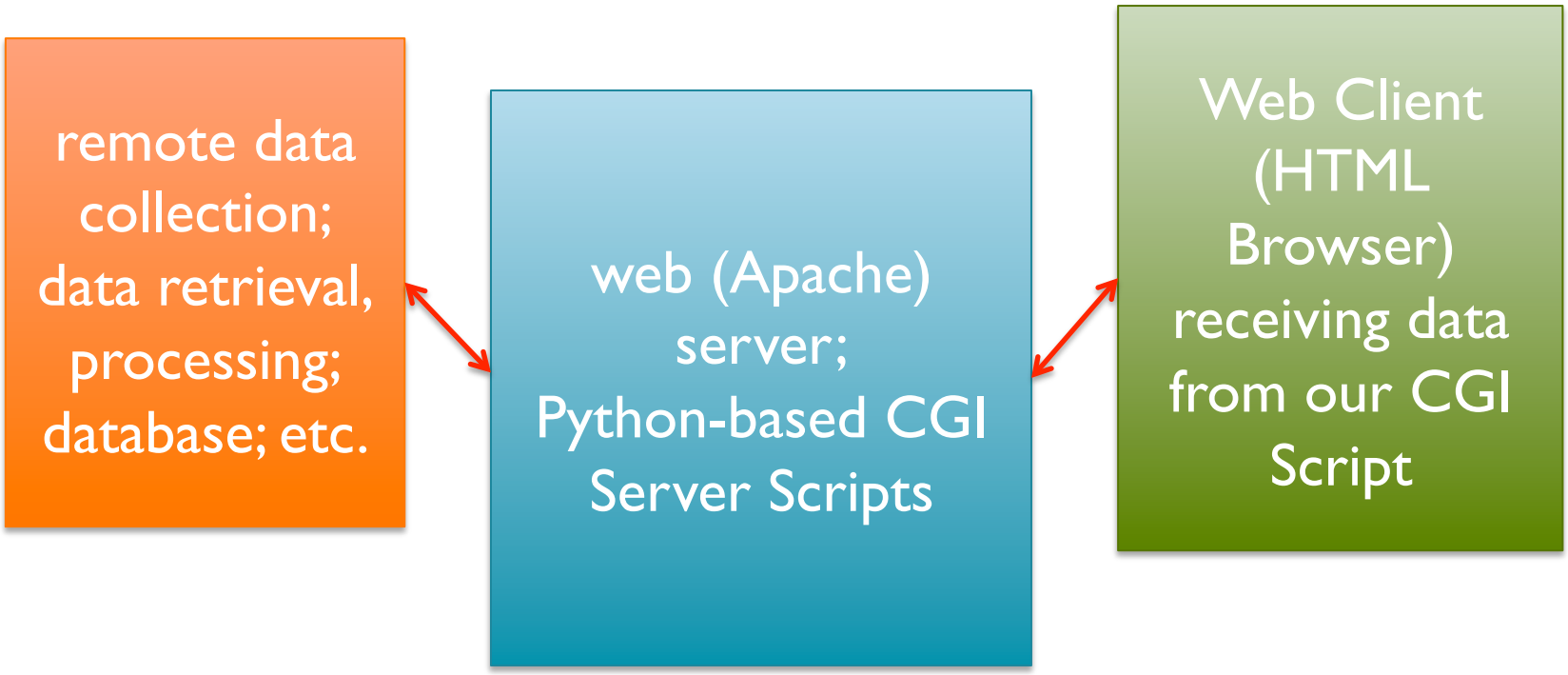
INFO I211 – Spring 2014 – Sections 18530 & 22719

Lecture 20 – 2014.04.07 & 2014.04.08

Instructor:
Mitja Hmeljak,
http://mypage.iu.edu/~mitja
mitja@indiana.edu

Building a Distributed System

a possible distributed system model:



CGI server-side scripts

- how do we know where and how our Python-based CGI scripts run?
- we can use various debugging techniques:
  - run the program from command-line before accessing it through the client-server (i.e. URL from a web browser)
  - catch exceptions in our script, and print them out
  - ...
- figure out the runtime environment: *printenv*!

web (Apache) server; Python-based CGI Server Scripts

CGI server-side scripts: printenv.cgi

```
#!/usr/bin/python
import os
print "Content-type: text/html\n\n"
print "<html><head><title>printenv.cgi</title></head><body>\n"
print "<h3>printenv.cgi</h3><hr>\n\n"

itemslist = []
for (key, val) in os.environ.items():
    itemslist.append( (key, val) )
itemslist.sort()

for (key, val) in itemslist:
    print key+" = "+val+"<br>"

print "\n<hr></body></html>"
```

if everything else fails, save this script as **index.cgi** and place it in your **~/cgi-pub/** directory on silo...

What about programming the client?

~~Web Page Scripting~~

Programming Client Side

JavaScript: is it Java ... in a script?

(hint: that's not it)

Web Client (HTML Browser)
receiving data from our CGI Script... couldn't we run our programs *right here*?!?

JavaScript became available in 1995, when Netscape added a *lightweight interpreted language* to its web browser.

JavaScript is *unrelated* to the Java programming language developed by Sun Microsystems.

JavaScript ← → Python ?

- JavaScript ← → Python, similarities and differences

- JavaScript basics & example programs

Hello World in JavaScript

a JavaScript one-line program:

```
<html> <head><title>Javascript Hello World 01</title></head>
<body>

    <script language=javascript>

        document.write("<h1>Hello World, this is Javascript!</h1>")

    </script>

</body>
</html>
```

How are we going to edit JavaScript?

On Windows systems:

http://notepad-plus-plus.org

On Mac OS X systems:

http://www.barebones.com/products/textwrangler/

JavaScript ← → Python: Similarities

Python and Javascript are programming languages

(unlike HTML, which is a markup language)

Python and Javascript are interpreted:

Python is interpreted by the Python interpreter, invoked either from IDLE or command-line

JavaScript is interpreted by a web browser (there are also – less commonly used– command-line implementations)

Python and Javascript are imperative and object-oriented languages used to implement algorithms

(unlike HTML, which is used to describe page content)

JavaScript ← → Python: Similarities

Python is a standard. There are versions (2.x and 3.x being the most common variants), but each version provides a cross-platform compatible language and run-time environment.

JavaScript is supported differently by different web browsers. Basic syntax is the same, but there are many non-standard extensions that vary.

JavaScript source code (unlike Python source code) can be **stored inside a web page** and even **intermixed with HTML**.

JavaScript code debugging in a web browser is not typically practical, but a separate command-line JavaScript interpreter can be used for this purpose.

Python and JavaScript syntax are quite different:

one obvious difference: in JavaScript, blocks must be enclosed in {curly brackets}

How to add JavaScript to a Web Page

A special <script> tag is used to embed JavaScript inside HTML:

```
<script laguage=javascript>
      javascript code
      javascript code
      javascript code
</script>
```

To call a JavaScript function from HTML, some JavaScript can be made part of HTML, for example to call a function from an HTML button:

```
... onClick="myFunction()" ...
```

The output from a JavaScript program can be presented to the user as HTML rendered by the web browser. In that, it is used similarly to Python CGI scripts that generate HTML pages.

But CGI scripts generate HTML code and *then* send it to the browser over an HTTP connection, whereas JavaScript code can generate HTML *within* the browser itself.

JavaScript ← → Python: Syntax Comparison

In JavaScript, indentation does *not* delimit blocks.

Curly brackets {...} are used instead, just like in C languages, in Java, etc.

Python:

```
def myFunc(aVar):
    if (aVar == “test”):
        return( 0)
    else:
        return(-1)
```

JavaScript:

```
function myFunc(aVar) {
    if (aVar == 'test') {
        return(0)
    } else {
        return (-1)
    }
}
```

Indentation does *not* matter to the JavaScript language interpreter....

This code :

```
function myFunc(aVar) {
        if (aVar == 'test') {
            return(0)
         } else {
             return (-1)
         }
   }
```

Works just the same as this code:

```
function myFunc(aVar) {
    if (aVar == 'test') {
    return(0)
    } else {
    return (-1)
    } }
```

...but which one is more *readable*?

... but indentation *does* matter to the (human) reader!

Proper indentation is es*sential* to maintain JavaScript code that is readable and understandable:

```
function myFunc(aVar)  {
      if (aVar == 'test') {
          return(0)
      } else {
           return (-1)
      }
}
```

Functions and return values in JavaScript

Very similar to Python. Defining a function however requires the `function` keyword:

```
function myFunctionName(aParameter) {
    if (aParameter == 'test') {
        return(0)
    } else {
        return (-1)
    }
}
```

Then, calling the function is most similar, too:

```
myFunctionName ("a string value")
```

Variables in JavaScript – and Arrays

In JavaScript, variables ***must*** be declared before being used. The syntax is:

```
var VARIABLENAME = VALUE
```

For example:

```
var myName = "Pat"
```

The "***var***" keyword tells the JavaScript interpreter that a variable is being ***declared***, for subsequent use. Then a value is assigned, completing the variable ***definition***.

Unlike Python, Javascript has built-in arrays. To declare an array and its number of elements, the syntax is:

```
var ARRAYNAME = new Array(SIZE)
```

For example:

```
var myArray = new Array(50)
```

To access one element in an array, use [square brackets] for indexing:

```
var oneElement = myArray[1]
```

advanced (!warning!) side-note **about JavaScript and Arrays**
JavaScript has *both* mutable and immutable arrays

For the curious: the JavaScript *Array* type is actually *not* a C-style array of immutable lenght.

- It is actually a JavaScript *object* type that associate keys to values
- *key* → *value* ...not dissimilar to Python dictionaries
- keys that are numeric strings can be specified directly (without "" quotes)
- Array objects have a *length* property, etc. and can have new values appended to them.

The JavaScript Typed Array (as from the newest ECMA specifications for JavaScript) is a viceversa a C-style array of immutable length.

To *declare* a typed array and the number of *bytes* it occupies, for example:

```
var buffer = new ArrayBuffer(16)
```

To access one element in an array, declare a *view type* first, then access:

```
var int32View = new Int32Array(buffer);
int32View[i] = i*2;
```

if statements in JavaScript

In JavaScript, *if* statements are similar to – but not the same as – Python *if* statements. There is an *else* clause, but no *elif* clause.

Instead of the *elif* clause, JavaScript provides other conditional statements, such as *case* / *switch*, inherited from C-style languages (but we won't look at those right now).

while loop statements in JavaScript

In JavaScript, while statements are similar to those in Python:

```
while (BOOLEAN EXPRESSION) {
 STATEMENT;
 STATEMENT;
}
```

Javascript however also provides do/while statements, where the boolean expression test is performed *after* the loop is executed, and not before:

```
do {
   STATEMENT;
   STATEMENT;
 } while (BOOLEAN EXPRESSION)
```

for loop statements in JavaScript

for loops in JavaScript are based on C and C++ syntax, and are quite different than in Python. The format is:

```
for( INITIALIZE; BOOLEAN EXPRESSION; ACTION) {
   STATEMENT;
   STATEMENT;
}
```

For example:

```
for (var i = 0; i < 10; i = i+1) {
   document.write("i is equal to " + i)
}
```

for loop statements in JavaScript

for loops in JavaScript are based on C and C++ syntax:

```
for( INITIALIZE; BOOLEAN EXPRESSION; ACTION) {
  STATEMENT;
}
```

There are three parts in the definition of a *for* loop, all three placed in parentheses, separated by semicolons, *before* the loop *statement block*:

1. the *initialization* code (first part) is executed *once* at the beginning, before the execution of the *statement block*.
2. the *boolean expression* (second part) is *evaluated every* time the loop repeats, before the execution of the *statement block* and then the statements in the loop are executed *only* if the resulting value is TRUE.
3. the *action* code (third part) is executed *after* the loop executes. It is typically used to increment a counter variable.

for loop example in JavaScript

```
for (var i = 0; i < 10; i = i+1) {
   document.write("i is equal to " + i)
}
```

In the above example, the document.write() statement executes 10 times.

The variable *i* will have value 0,1,2,3 ... 9 in the statement block.

After the 10th execution, the i=i+1 action executes again, at which point the test i < 5 evaluates to FALSE, and the *for* loop ends.

The document.write() therefore does not execute when i = 10.

Hello World (in JavaScript) version 2

Type this code in a plain-text file, save it as "hello2.html", and open it in a browser!

```
<html> <head><title>Javascript Hello World 02</title></head>
<body>

    <script language=javascript>
        var myArray = new Array(50)


        document.write("<h3>Hello World, this is a Javascript array:</h3>")

        document.write("<br>")

        for (var j = 0; j < 5; j = j+1) {
            myArray[j] = j * j;
            document.write("j is: " + j + ",    ")
            document.write("myArray["+j+"] is: " + myArray[j] + "<br>")
        }

        document.write("<br>")

    </script>

</body>
</html>
```

JavaScript input/output

JavaScript code typically executes inside a web browser.

Therefore, most the input/output and user interaction to and from a JavaScript program will be handled within a web browser's page.

Input: the user will provide input using HTML <form> elements, buttons, clicking on links, etc.

Output: a JavaScript program's output is displayed typically as part of a web page, as HTML rendered by the web browser.

JavaScript output on a web page

The document.write("text string") built-in JavaScript function can be used to add content to a web page. For example:

```
<html>
    <body>
    <h1>Example:</h1>
    <script language=javascript>
        document.write("<h2>Here is my website!</h2>")
    </script>
    </body>
</html>
```

The one-line JavaScript program inside this HTML page will be executed as it is encountered when the web browser parses and renders the HTML code.

The JavaScript one-line program outputs some HTML code, which will then be rendered by the web browser.

JavaScript output on a web page

Since the JavaScript program inside <script> tags is run *as the web browser renders the page*, the program can produce output that is calculated right at that moment:

```
<html>
    <body>
    <h1>Clock Example:</h1>
    <script language=javascript>
            var date = new Date()
            var hours = date.getHours()
            var min  = date.getMinutes()
            document.write("Time: <b>" +hours+":"+min+"</b>")
    </script>
</body>
</html>
```

Group Work:

Modify the Javascript program in this HTML page, so that it outputs seconds as well.

(hint: the Javascript function to obtain seconds is... getSeconds())

```
<html>
    <body>
    <h1>Clock Example:</h1>
    <script language=javascript>
            var date = new Date()
            var hours = date.getHours()
            var min  = date.getMinutes()
            document.write("Time: <b>" +hours+":"+min+"</b>")
    </script>
</body>
</html>
```

Save your code in a plain-text file named "hello3.html", and open it in a browser.

Group Work Solution

```
<html>
    <body>
        <h1>
            Clock Example (with seconds):
        </h1>

<script language=javascript>
        var date = new Date()
        var hours = date.getHours()
        var min  = date.getMinutes()
        var sec  = date.getSeconds()
        document.write("Time: <b>" +hours+":"+min+":"+sec+"</b>")
</script>

    </body>
</html>
```