CS201 - PL'S
Script Programming

Aravind Mohan

Allegheny College

November 15, 2021

Scripting Languages:

- Scripting languages have always been important in computer systems
  - They are the glue that ties the different elements of the system together
  - Their origins go back to the days of card-based operating systems
    – JCL (OS360 JCL)
    – GEORGE II, GEORGE III
- And they were much used in minicomputer operating systems
  - Data General's AOS
  - Unix

Scripting Languages:

- Scripting is about producing simple very-high-level-languages that are friendly to the programmer.
- Scripting languages are relatively simple, and often allow users to do complex things.
- Java, C++, C#, etc. are extremely complex
  – they have a nasty tendency to get bigger and bigger as designers add more and more useful facilities, and interface components, and bells and whistles
  – take a long time to learn to use (but are wonderful when you really understand them).

Scripting Languages:

- shell languages (e.g., “bash”, “csh”, “zsh”, “tcsh”, and many others)
- text-processing languages (e.g., “awk”, “perl”, and others)
- “glue” and general-purpose languages (e.g., Python, Perl, Ruby, etc.)
- “extension” languages (e.g., JavaScript, Visual Basic, VimScript, etc.)

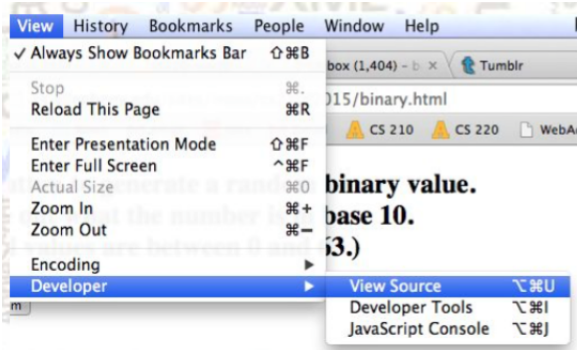
Some languages fall under several categories

Scripting Languages:

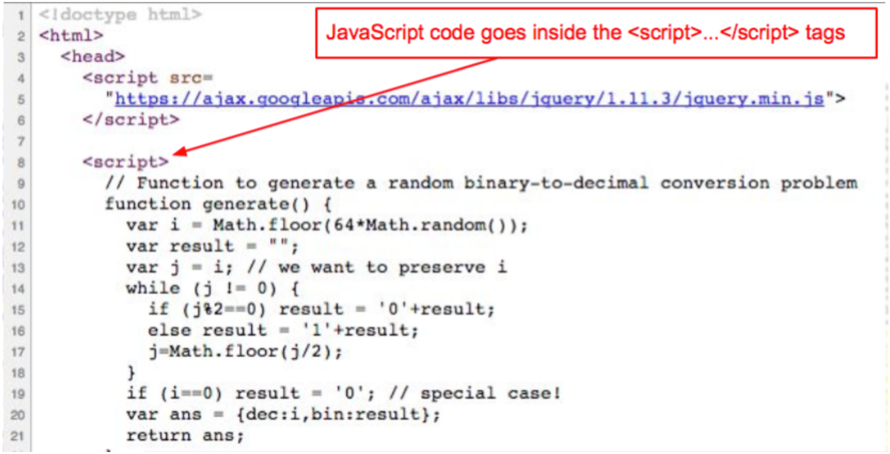
- Mostly we have focused on features of the language itself rather than its use in “extending” the features of HTML, CSS, etc. in web pages.
- In Chrome and just about any other browser, search for a menu item called “Developer” or “Tools” or “View Source” and look at the underlying code.

Scripting Languages:

Here is what it looks like on my laptop:



Scripting Languages

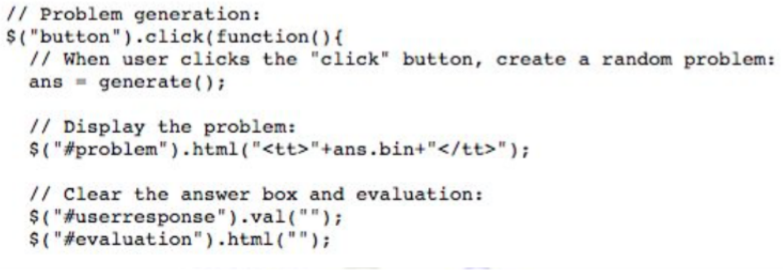


```
<!doctype html>
<html>
  <head>
    <script src=
      "https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js">
    </script>

    <script>
      // Function to generate a random binary-to-decimal conversion problem
      function generate() {
        var i = Math.floor(64*Math.random());
        var result = "";
        var j = i; // we want to preserve i
        while (j != 0) {
          if (j%2==0) result = '0'+result;
          else result = '1'+result;
          j=Math.floor(j/2);
        }
        if (i==0) result = '0'; // special case!
        var ans = {dec:i,bin:result};
        return ans;
```

Scripting Languages:

We can augment the behavior of HTML elements “callbacks”, i.e., functions that get passed into event handlers such as the one that handles a “button click”



```
// Problem generation:
$("button").click(function(){
  // When user clicks the "click" button, create a random problem:
  ans = generate();

  // Display the problem:
  $("#problem").html("<tt>"+ans.bin+"</tt>");

  // Clear the answer box and evaluation:
  $("#userresponse").val("");
  $("#evaluation").html("");
```

What is client-side and server-side?

- Any machine can play the role of either a client or a server
  – You could even have a machine being both
- Some languages, e.g. Javascript, are said to be client-side
  – Run on the user's browser/web client
- Other languages, e.g. PHP, are said to be server-side
  – Run on the server that is delivering content to the user

Static Web Model

- You (the client) send a request to the server for a web page.
- The server looks up the web page using part of the URL you have sent it, then returns the HTML page which your browser subsequently displays on your machine.

A More Dynamic Web Model

- You (the client) send a request to the server and it dynamically determines the HTML that is to be returned.
- The dynamics of the reply is achieved through extending the web server with a program (script) that does some data processing and creates HTML output based on the data you sent (e.g. contents of a form).
- The process of generating the HTML response is performed server-side.

Server-side scripting

- One approach is the Common Gateway Interface (CGI) where we have a separate program that can be executed.
- An alternative is to have extra code in the HTML that can be executed on the server to determine the HTML that is to be returned. That is how PHP works.

Client-side scripting

- The other (complementary) approach is to do the work on the client machine.
  – Again we have extra code in the HTML, but now it is executed by the user's browser (i.e. client-side). Most common client side script is Javascript.
  – An example of its use is when a web page has a form. We can use Javascript to validate the input data client-side before it is sent to a server.
- If we do the validation on the client, this reduces the work that the server has to do and reduces the time taken to respond to the user.
- HTML5 essentially includes Javascript elements to enhance its power.

Client-side scripting

Javascript can also be used to create dynamic web page content.
For example:

- We could change the content based on the fact that you visited the web page before.
- Time of day.
- JavaScript popup menus.

Scripting Languages

Back to the Bash

`https://en.wikibooks.org/wiki/Bash_Shell_Scripting`

Reading Assignment

PLP Chapter 13

Questions

Do you have any questions from this class discussion?