



Lesson #4:

Introduction to CGI/Perl

and XHTML Forms



About Perl

- Perl stands for *Practical Extraction and Report Language*.
- Created in 1986 by Larry Wall.
- It used to make web pages interactive.
- It is a very powerful text manipulation tool.
- Easily moved from one platform to another.



About CGI

- CGI stands for *Common Gateway Interface*.
- CGI is a protocol, not a programming language. It determines how servers talk to programs.
- CGI scripts are not all written in Perl, some are written in C, TCL or VB.
- For web programming, the CGI/Perl combination is preferred.



Perl and XHTML

- Perl programs are executed on the server to generate dynamic XHTML code.
- Specific Perl instructions are used to generate the code.
- A Perl application reacts with data submitted by the user to create a different page accordingly.



Perl data

- Data in Perl can be classified as number or string.
- Data in Perl can be a constant or a variable.
- Data in Perl can be a scalar, an array or a hash.
- Numbers are the simplest. An expression in Perl is just represented as it is: $2 + 2$.



Numbers and strings

- A string is a collection of characters.
- Strings are enclosed between “ or ‘.
- Strings can be concatenated using the . (dot) operator. 'ice'.'cream' results in 'icecream'.
- Perl automatically assume the correct operands. **3+'a'** is 3 and **3.'a'** is '3a'.



Constants and variables

- Like in all programming languages, variables are containers of values.
- Variable names always start with \$, @ or % depending if the variable is a scalar, an array or a hash.
- Variables are usually not declared and are never typed.

```
$address = '3 Main Street';
```

```
$x = $x + 10;
```



Scalars and arrays

- A scalar is an individual piece of data. 'hello' and 3 are scalar constants. A scalar variable always begin with a \$.

```
$x = 10;
```

- An array consist of multiple values. An array in Perl can contain numbers and strings.
- ```
@array = (10, 'Toyota', 'Prius', 2005);
print $array[1];
print @array;
```



# Hashes

- A hash is made up of pairs of scalars. The first element is called the key, the second element is the value. Keys and values are separated by `=>`. Pairs are separated by a comma `,`.

```
%info = (age=>45, name=>'Murray');
```

- You can also use commas only (use quotes for keys)

```
%info2 =
('age', 45, 'name', 'Murray');
```



# Operators and functions

- Basic operations are **+**, **-**, **\***, **/**, **%** and **.**

```
$sum = 3 + 5;
```

```
$modulus = 3 % 5;
```

```
$quotient = 3 / 5;
```

- Functions have their arguments enclosed in parentheses.

```
$item1 = shift (@array);
```



# Operators and functions

- The result of a function is usually different than its return value. For `shift ('a','b','c')`, the result is `('b','c')`, the return value `'a'`.

```
@array1 = ('a', 'b', 'c');
@array2 = shift (@array1);
print @array1;
print @array2;
```

**bca**



# Quotations marks

- There are 2 kind of quotations marks in Perl.
- The single quote (') is used when you want the string to be taken literally.
- `$word = "World";`
- `print 'Hello\n$word';` will be displayed as `Hello\n$word`.
- The double quote (") is used when you want the string to be evaluated.
- `print "Hello\n$word";` will be displayed as
- `Hello`  
`World`



# Quotation functions

- There are 2 functions, one for each kind of quotations marks.
- Q for a single quote, qq for double quotes and qw for individual words.

```
$a = q(winner's circle);
```

```
$b = 'winner\'s circle';
```

```
$c = qq();
```

```
$d = "<img src=\"lamp.gif\"";
```

```
@e = qw(cats dogs elephants);
```

```
@f = ('cats', 'dogs', 'elephants');
```



# Perl statements

- As you may have already noticed, Perl statements are very similar to C statements.
- Like in C, the semicolon indicates a statement end.
- Also like C, curly braces { } are used to group blocks of statements.
- Unlike C, however, variables do not need to be declared (unless you need private variables).



# Private variables

- By default, all variables are global, they are visible through the entire program.
- It is possible to declare private variables, variables that are visible only inside their current block `{ }`. These private variables declarations must occur inside the intended block. The **my** function declares private variables. Private variables are highly recommended if you borrow blocks of code not written by you.

```
my $car;
```

```
my ($make, $model, $year);
```



# The shebang line

- Equivalent to C preprocessor directives, the shebang line (sharp(#) and bang(!)), indicates the location of the Perl interpreter. The location is usually given to you by your hosting company.

- Typical UNIX shebang line (mandatory):

```
#!/usr/bin/perl
```

- Typical Windows shebang line (optional):

```
#!C:\Perl\bin\perl.exe
```



# Program skeleton

- Programs are written using a simple text editor. Mandatory statements are in red, optional statements in blue.

```
#!/usr/bin/perl
```

```
use CGI ':standard' ;
```

```
use strict;
```

```
print "Content-type: text/html\n\n";
```

```
rest of program here...
```

- All you need after that is to save your file with a .cgi or .pl extension.



# Program hosting

- Always name your files with all-lower-case names. Underscores are OK. No spaces in file names.
- Upload the files to your cgi-bin directory on your host server (usually inside your public\_html or www folder). Make sure that the directory protection is set at 711 or 755.
- Make sure your .cgi or .pl file has a protection of 755.



# Getting data

- A dynamic web page created by Perl becomes truly dynamic by processing information submitted by visitors.
- The easiest way is to submit information via XHTML forms.

```
<form
 action="http://www2.scs.ryerson.ca/
 ~dhamelin/cgi-bin/form1.cgi"
 method="get">
 First name: <input type="text"
 name="first" />

 Last name: <input type="text"
 name="last" />

 <input type="submit">
</form>
```



# The `<form>` tag

- The form element creates a form for user input. A form can contain text fields, check boxes, radio-buttons and more. Forms are used to pass user-data to a specified URL.
- *Action* is a required attribute. Its value is a URL that defines where to send the data when the submit button is pushed. That URL points to a server-side (Perl) program.
- *Method* and *name* are also commonly used attributes.



# The method attribute

- The HTTP method for sending data to the action URL. The default method is **get**.
- **method="get"**: This method sends the form contents in the URL: `URL?name=value&name=value`. Note: If the form values contains non-ASCII characters or exceeds 100 characters you **MUST** use `method="post"`.
- **method="post"**: This method sends the form contents in the body of the request. Note: Most browsers are unable to bookmark post requests.

# The <input> tag



- The <input> tag defines the start of an input field where the user can enter data.
- There are different types of input fields which are specified by the type attribute.
- **type="text"**: The default attribute. A simple text field.
- **type="password"**: A text field with invisible characters. Not recommended with get method.
- **type="submit"**: A button that will submit the form to the program.



# The <input> tag

- **type="reset"**: Brings back a blank form.
- **type="checkbox"**: With check boxes, more than one option can be selected.
- **type="radio"**: With radio buttons, only one option can be selected.
- ```
<form action="cgi-bin/test.cgi">  
Name:<input type="text" name="name"  
size="20" />  
</form>
```

Name:



Password, submit and reset

```
<form action="cgi-bin/test.cgi">
```

```
Password: <input type="password"  
          name="password" size="20" />
```

```
<br />
```

```
<input type="submit">
```

```
<input type="reset">
```

```
</form>
```

Password:



Check boxes

```
<form action="cgi-bin/test.cgi">  
Form of transportation?<br />  
Bike: <input type="checkbox"  
      name="Bike" />  
Car: <input type="checkbox"  
     name="Car" /><br />  
Foot: <input type="checkbox"  
     name="Foot" />  
Subway: <input type="checkbox"  
       name="Subway" /><br />  
</form>
```

```
Form of transportation?  
Bike:  Car:   
Foot:  Subway: 
```

Radio buttons



```
<form action="test.cgi">  
Gender?<br />  
Female: <input type="radio"  
      name="gender" value="F" />  
Male: <input type="radio"  
      name="gender" value="M" />  
</form>
```

Gender?

Female: Male:



value and checked

- The **value** attribute just places a default value in a text field or assigns a value to a check box or radio button item.
- The **checked** attribute will check a radio button or check box by default.

```
<form action="test.cgi">
```

```
Section?<br />
```

```
011: <input type="checkbox" name="section"
      value="011" checked="checked" />
```

```
021: <input type="checkbox" name="section"
      value="021" />
```

```
</form>
```

Section?

011: 021:



The `<textarea>` tag

- Defines a text-area (a multi-line text input control). A user can write text in the text-area. In a text-area you can write an unlimited number of characters. The default font in the text-area is fixed pitch.
- The required attributes are rows and cols for the numbers or rows and columns visible.
- Other attributes are name, and readonly.



The <textarea> tag

```
<form action="cgi-bin/test.cgi">
```

```
<textarea rows="10" cols="30"  
  readonly="readonly">
```

```
The cat was playing in the garden.
```

```
</textarea>
```

```
</form>
```

A screenshot of a web browser's text area. The text area is a rectangular box with a blue border. It contains the text "The cat was playing in the garden." in a monospaced font. On the right side of the box, there is a vertical scrollbar with a small arrow pointing up at the top and a small arrow pointing down at the bottom.

The cat was playing in the
garden.



Menus and combo boxes

- Menus and combo boxes are created with the `<select>` and `<option>` tags.
- Attributes include `name`, `size` (the number of visible elements) and `multiple="multiple"` (allowing multiple items to be selected).

```
<select name="car" size="2" multiple="multiple">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="opel">Opel</option>
  <option value="audi">Audi</option>
</select>
```





Submit and reset buttons

- By default, the submit button label is *Submit Query*. You can change it with the value attribute.
- By default, the reset button label is *Reset*. You can change it with the value attribute.

```
<input type="submit" value="Go!" />
```

```
<input type="reset" value="Erase" />
```





Using a query string

- An alternative to forms exist to submit data to a CGI/Perl program. We can use the query string.
- A query string item consists of a key and a value. The key corresponds to the name of a form item and the value, its value.

[http://www.living-net.org/cgi-bin/test.cgi?
city=Toronto&day=Monday](http://www.living-net.org/cgi-bin/test.cgi?city=Toronto&day=Monday)



End of lesson