

Perl Introduction

`www.perl.org`

EIW - Perl Introduction

1

Perl & CGI

- Perl is often used for CGI programs
 - portable
 - great text handling capabilities
 - lots of sample code, libraries, extensions, etc.
- Perl is used for lots of things, not just CGI!

EIW - Perl Introduction

2

Perl and EIW

- When studying the Perl language, I'll show some examples that run as simple Perl programs.
- For our programming (labs, etc.) we will focus on Perl CGI (the programs are run by the web server, not from the DOS prompt).

EIW - Perl Introduction

3

History

- Perl was developed by Larry Wall as a replacement for AWK.
- "Practical Extraction and Report Language"
- Perl has been around for a long time (much longer than the WWW).
- The language has changed a lot...

EIW - Perl Introduction

4

Overview of The Perl Language

- Compared to C/C++:
 - variables are not similar at all!
 - simple variables are untyped,
 - there are major syntactic differences
 - control structures very similar
 - Perl has a few extra
 - operators are very similar
 - Perl has a few extra

EIW - Perl Introduction

5

A Simple Perl Script

```
print("Hello World\n");
```

- Can also be written like this:

```
print "Hello World\n";
```

EIW - Perl Introduction

6

Running A Perl Script

- Put your Perl program in "foo.pl".
- From the Command (DOS) prompt:
 - perl foo.pl

```
C:\eiw> perl foo.pl
Hello World

C:\eiw> c:\perl\bin\perl foo.pl
Hello World
```

EIW - Perl Introduction

7

Perl Variables - Scalars

- Simplest type of Perl variable is called a scalar.
 - Can hold one *thing*.
 - Can be anything, but only one of them!
 - integer
 - floating point
 - string

EIW - Perl Introduction

8

Scalar Variable Names

- All scalar variable names start with "\$".
- The rest of the name can be made from letters, numbers and '_' (underscore).
- These are valid scalar variable names:

```
$foo    $foo_blah    $x100y200z
$foofoofoofoofoofoofoofoofoofoo
$HiDave    $Very_Descriptive_Name
```

EIW - Perl Introduction

9

Scalar Assignment

- Assigning a value to a scalar is just what you would expect:

```
$pi = 3.141593;  
$foo = "foo";  
$Foo = 27;
```

Semicolons!

EIW - Perl Introduction

10

Other kinds of Perl Variables

- There are lists (arrays) that can hold more than one *thing*.
- Hashes (Associative arrays) hold more than one thing.
- References hold *another variable*

- We will cover some of these eventually...

EIW - Perl Introduction

11

Constants (Literals)

- Scalar constants are just like in C/C++/JavaScript

```
3  
3.141593  
6.02E23  
"foo"  
032
```

Don't start numeric constants with a leading 0 (in Perl this means the number is an Octal number!)

EIW - Perl Introduction

12

String Literals

- Must be enclosed in double quotes or in single quotes.
- Double quotes:
 - \ means something special
 - There are other special characters.
- Single Quotes:
 - nothing is special!

EIW - Perl Introduction

13

Double Quotes and *backslash*

- Inside a doubly quoted string, the following are *special sequences*:

<code>\n</code>	newline
<code>\r</code>	return
<code>\t</code>	tab
<code>\\</code>	a single backslash
<code>\"</code>	a double quote

EIW - Perl Introduction

14

String Literal Examples

<code>"Hello\n"</code>	Hello	<i>followed by a newline</i>
<code>'Hello\n'</code>	Hello\n	
<code>"12\t6\t3"</code>	12 6 3	
<code>'12\t6\t3'</code>	12\t6\t3	
<code>"He said \"Hi\""</code>	He said "Hi"	
<code>'He said "Hi"'</code>	He said "Hi"	

EIW - Perl Introduction

15

Perl Math Operators

- The usual: + - * / %
- Exponentiation is **: 10**3
- Examples:

```
$y = $m * $x + $b;  
$radians = $degrees * (3.141593/180.0);  
$seconds_per_year=365*24*60*60;
```

EIW - Perl Introduction

16

Perl String Operators

- Concatenation operator is '.'

```
$myname = "Dave" . " " . "Hollinger";
```

```
$myname = $first . $blank . $last;
```

We don't have to worry about the "+" operator ambiguity we had in JavaScript

EIW - Perl Introduction

17

String Repetition Operator

- the letter **x** is a string repetition operator (a string *times* operator):

<u>Expression</u>	<u>Value</u>
"M" x 4	"MMMM"
"Hello" x 2	"HelloHello"
"Joe" x (5-2)	"JoeJoeJoe"

EIW - Perl Introduction

18

Perl Comparison Operators

- Two different kinds:
 - for comparing numbers.
 - for comparing strings.
- Since a Perl scalar can be either a number or a string you must use the right operator, or you won't get what you expect!

This is a common source of problems for people learning Perl!

EIW - Perl Introduction

19

Comparison Operators

Comparison	Numeric Operator	String Operator
Equal	==	eq
Less Than	<	lt
Greater Than	>	gt
Less Than or Equal	<=	le
Greater Than or Equal	>=	ge

EIW - Perl Introduction

20

String comparisons

- Perl uses the ASCII value of each character as the basis for comparison.
 - compares the ASCII value of the first char of each string.
 - if necessary checks the seconds char, and so on.

'a' < 'b' < 'z' 'A' < 'B' < 'Z'

EIW - Perl Introduction

21

Comparison Operator Trouble

This is true: `1876 > 4`

So is this: `"1876" > "4"`

This is false: `"1876" gt "4"`

This is false: `"1876" > 4`

EIW - Perl Introduction

22

Data Conversion

- Perl automatically converts scalars between numeric and string depending on the context.

`2 * "3.14"` converts `"3.14"` to a number

`(117 lt 23)` converts both to string

`$x == "POST"` converts both to numbers

EIW - Perl Introduction

23

String to number conversion

- Perl looks at the left part of the string for numeric characters - uses as many as it can until it finds something that cannot be part of a number.

`"13.5Joe"` becomes 13.5

`"Hello"` becomes 0

`"Joe13"` becomes 0

EIW - Perl Introduction

24

Exercises - What is the value?

```
"HelloWorld" x "1"  
17 + "13"  
17 + "thirteen"  
("hi" . "dave") == "hidave"  
("hi" . "dave") eq "hidave"  
10 x 2  
"987654321" gt "9871654321000"
```

EIW - Perl Introduction

25

Variable Interpolation

- Inside *double* quotes, a scalar variable is *interpolated* - it is replaced by the *value* of the variable.

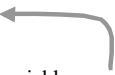
```
$num=13;  
$foo = "The number is $num";  
# now foo is "The number is 13"
```

EIW - Perl Introduction

26

Potential Interpolation Problem

```
$num=13; $number=17;  
$foo = "The number is $number";
```

- What is the value of `$foo`?
 - "The number is 13ber"
 - "The number is 17" ← 
- Answer: Perl will take the *longest* variable name it can match (to a real variable), so the second one is the value of `$foo`

EIW - Perl Introduction

27

Avoiding Interpolation

- You don't always want interpolation, how could we generate this:?

- "The number in \$num is 13"

```
$foo = "The number in \$num is $num";
```

EIW - Perl Introduction

28

Reading from STDIN

- You can read a line from Standard Input with:

```
<STDIN>
```

- Examples:

```
$foo = <STDIN>;  
$y = 17 + <STDIN>;
```

EIW - Perl Introduction

29

<STDIN>

- Each time <STDIN> is called for, *an entire line* is read from standard input.

```
print 'Enter your age\n';  
$age = <STDIN>;  
print 'Enter your weight\n';  
$weight = <STDIN>;
```

- If the user types the line "17 30" and hits Enter, the variable `$age` will have the value "17 30"!

EIW - Perl Introduction

30

<STDIN> newlines and chop

- <STDIN> returns the entire line - including a newline.
- Some times you don't want the newline.
- Use the chop operator:
 - `chop variablename`
 - `chop $foo` will remove the last character from `$foo`.

EIW - Perl Introduction

31

Chop Example

```
print "Enter First Name\n";
$name = <STDIN>;
print "Enter Last Name\n";
$lname = <STDIN>;
chop $fname; chop($lname);
print "Your name is: $fname $lname\n";
```

With the chop

```
Enter First Name
Marco
Enter Last Name
Polo
Your name is Marco Polo
```

Without the chop

```
Enter First Name
Marco
Enter Last Name
Polo
Your name is Marco
Polo
```

EIW - Perl Introduction

32

chop vs. chomp

- `chop` removes the last character (no matter what it is).
- `chomp` removes the last character, but only if it is a newline (`'\n'`)

```
chomp $fname;
```

EIW - Perl Introduction

33

Perl `printf`

- If you know `printf`, perl has this function!
 - Usage is just like in the C programming language:

```
printf("Hello %s World %d\n","cruel", 13);
```

```
Hello cruel World 13
```
