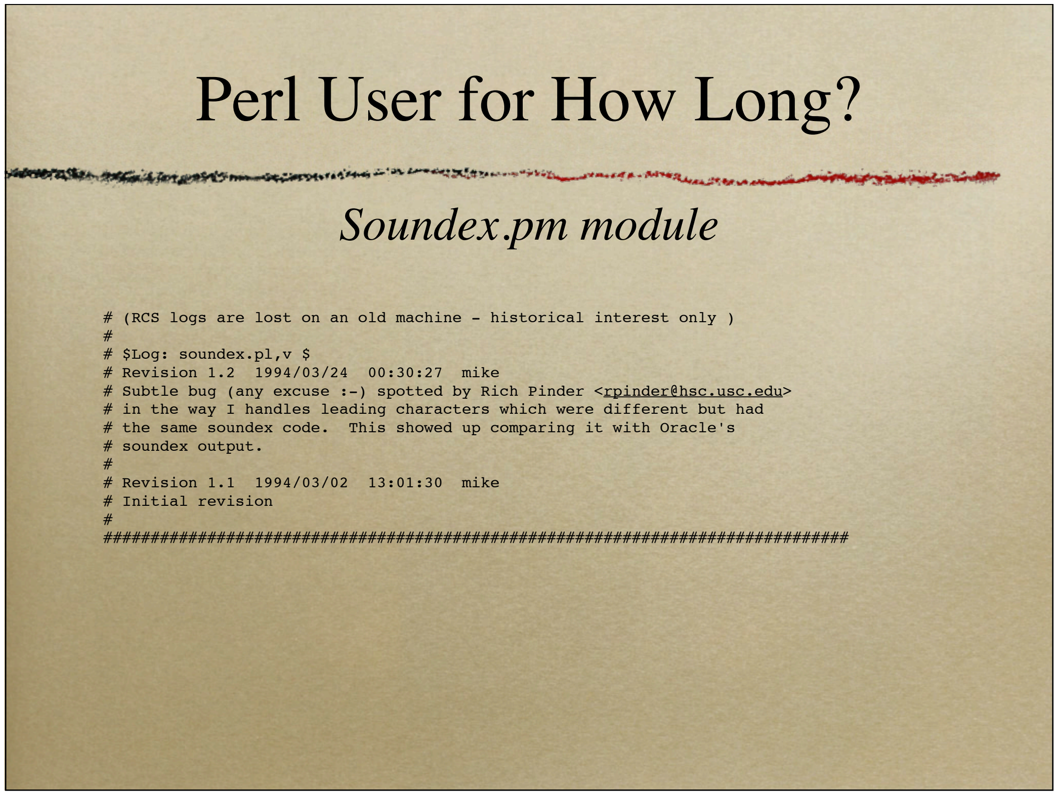
'Don\'t Quote Me On That!'

"mike\@stok.ca"



Perl User for How Long?

Soundex.pm module

```
# (RCS logs are lost on an old machine - historical interest only )
#
# $Log: soundex.pl,v $
# Revision 1.2  1994/03/24  00:30:27  mike
# Subtle bug (any excuse :-) spotted by Rich Pinder <rpinder@hsc.usc.edu>
# in the way I handles leading characters which were different but had
# the same soundex code.  This showed up comparing it with Oracle's
# soundex output.
#
# Revision 1.1  1994/03/02  13:01:30  mike
# Initial revision
#
##############################################################################
```

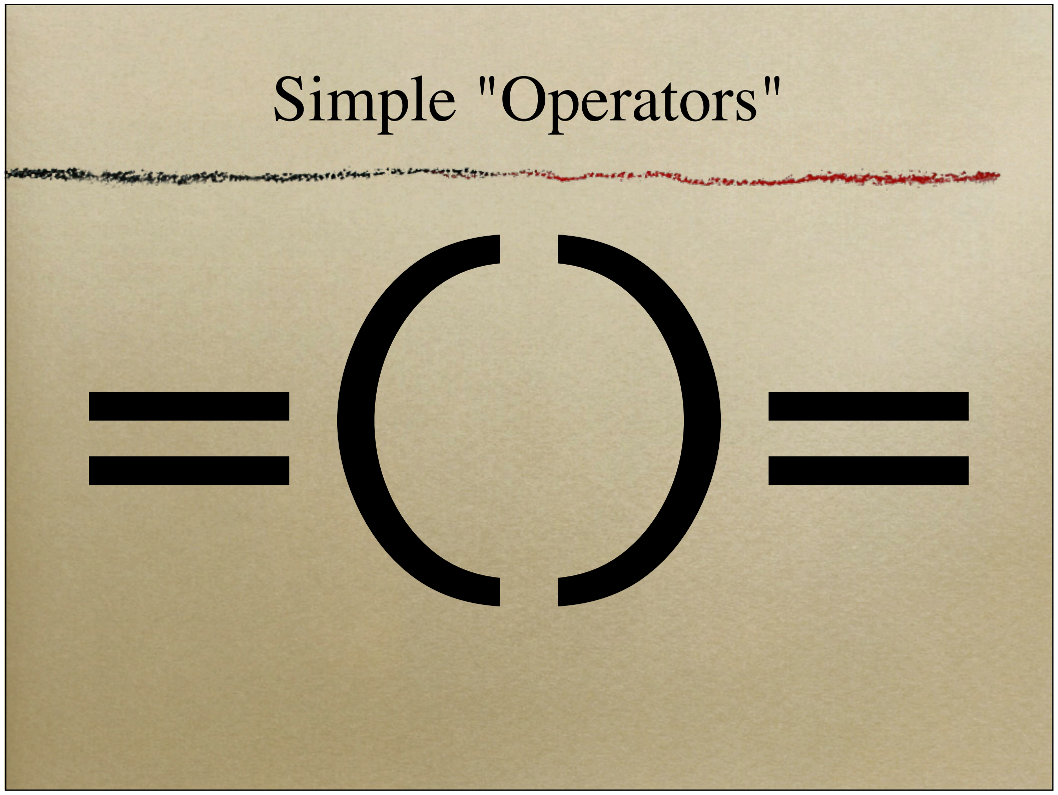
What is this presentation - a filler for a few minutes! I wanted to get the ball rolling. The overarching metaphor for this presentation is that quotes could be viewed as the "white bread" around a sandwich filling - they have a useful function but the important thing is the stuff filling the sandwich.

Who am I? Husband, aeronaut, competent shot, people person (?), poor typist, reasonable and cheap baby sitter, food & beer lover, accidental programmer, Marmite eater, Ruby enthusiast, bad drummer, relatively long time perl user... http://cpansearch.perl.org/src/MIKESTOK/Text-Soundex-2.00/Soundex.pm - that's at least 16 1/2 years ago, but wait, my copy of "Programming perl" was bought in 1991, hmmm, soon 20 years. My it was amazing what you could do with Perl 4.019 once you'd downloaded it over the 24,400 bpm modem!

I used Perl to get jobs done and to avoid learning C++. In the "good old days" perl was small and fast and small enough to fit in my head (including libraries.) It was a practial tool seemingly with no theoretical axe to grind, and it had a great community in Boston so it involved beer. Computer science as currently conceived doesn't do it for me. So that's my orientation.

A lot of people seem to know a bundle about OO, but don't have a clue about Perl fundamentals. They end up writing plain ugly code, and this talk is my opinions only (like assholes... most people have one) and it talking about some very simple stuff.

Thinking of assholes...

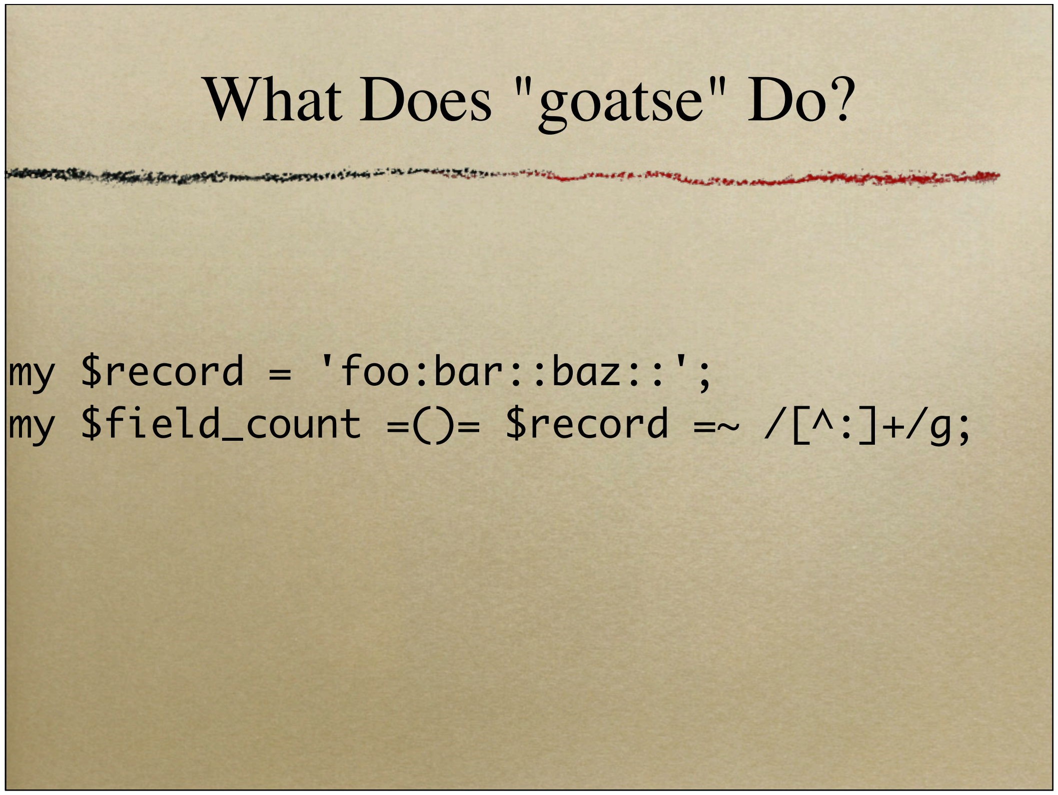


Simple "Operators"

=()=

This is the "goatse operator". Simple combination of operators which is actually useful! No I don't have the old goatse.cx image!

This is here because it can befuddle people, and if you know goatse the bread and filling metaphor could be extended to buns and no filling... but I won't dwell on that.



What Does "goatse" Do?

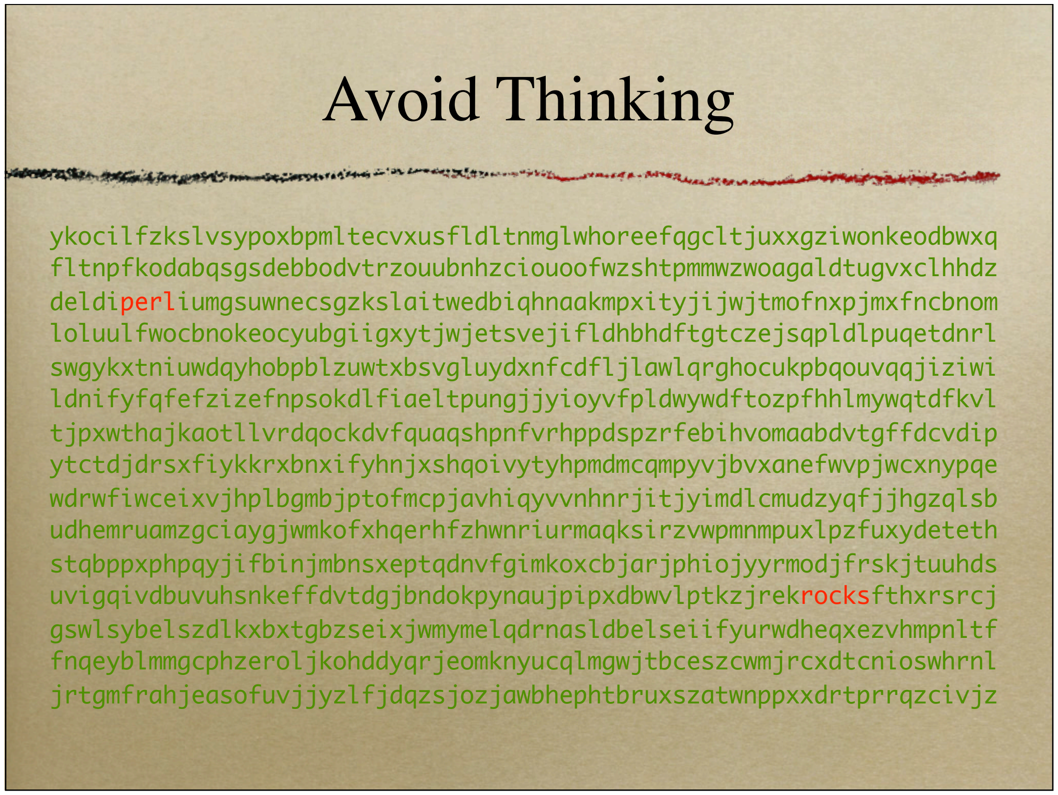
```
my $record = 'foo:bar::baz::';
my $field_count =()= $record =~ /[^:]+/g;
```

Useful for getting the length of a list of results without having to assign it to some intermediate array. Back to what the talk is about... being a perl user one of my virtues ought to be laziness...

This tells us there are 3 fields in the record which have some content.

Probably not that clear to all readers, and it's a bizarre construct which stands out in code and might distract the maintenance programmer from the business at hand. Most people can have their attention grabbed by things that stand out or are unexpected, and my point in this talk is to see how to minimise distractions from poor quote choice.

What do I mean by distraction?



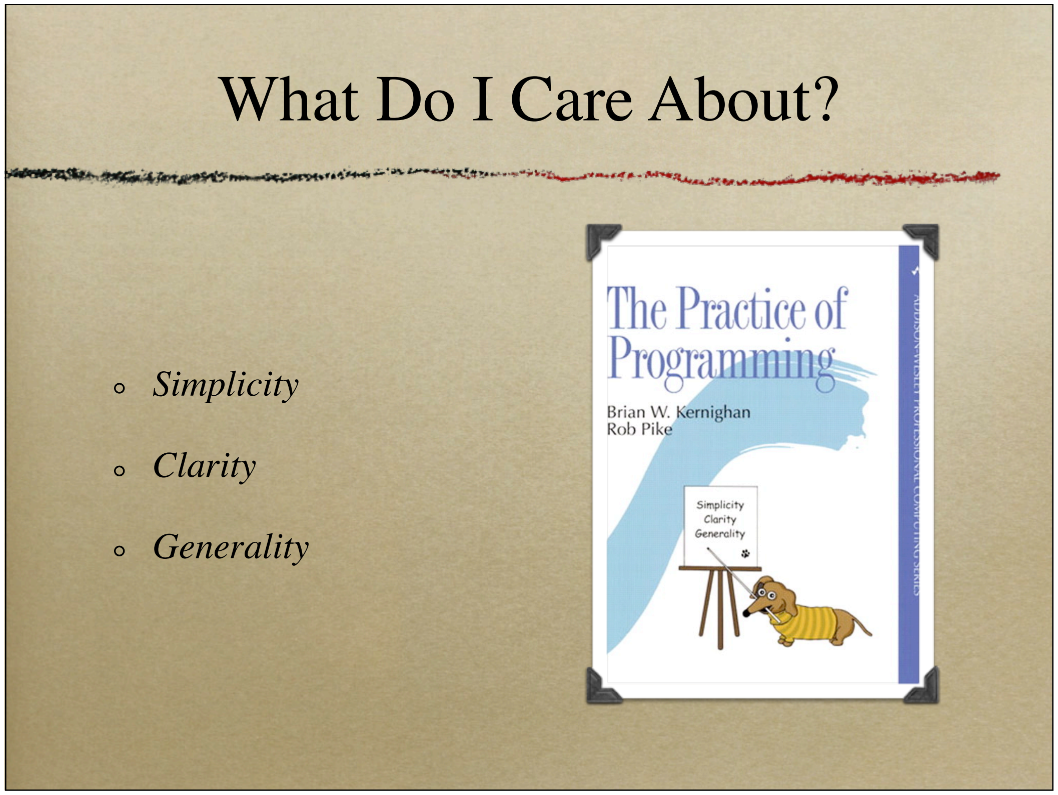
Avoid Thinking

```
ykocilfzkslvsypoxbpmltecvxusfldltnmglwhoreefqgcltjuxxgziwonkeodbwxq
fltnpfkodabqsgsdebbodvtrzouubnhzciouoofwzshtpmmwzwoagaldtugvxclhhdz
deldiperliumgsuwnecsgzkslaitwedbiqhnaakmpxityjijwjtmofnxpjmxfncbnom
loluulfwocbnokeocyubgiigxytjwjetsvejifldhbhdftgtczejsqpldlpuqetdnrl
swgykxtniuwdqyhobpblzuwtxbsvgluydxnfcdfljlawlqrghocukpbqouvqqjiziwi
ldnifyfqfefzizefnpsokdlfiaeltpungjjyioyvfpldwywdftozpfhhlmywqtdfkvl
tjpxwthajkaotllvrdqockdvfquaqshpnfvrhppdspzrfebihvomaabdvtgffdcvdip
ytctdjdrsxfiykkrxbnxifyhnjxshqoivytyhpmdmcqmpyvjbvxanefwvpjwcxnypqe
wdrwfiwceixvjhplbgmbjptofmcpjavhiqyvvnhnrjitjyimdlcmudzyqfjjhgzqlsb
udhemruamzgciaygjwmkofxhqerhfzhwnriurmaqksirzvwpmnmpuxlpzfuxydeteth
stqbppxphpqyjifbinjmbnsxeptqdnvfgimkoxcbjarjphiojyyrmodjfrskjtuuhds
uvigqivdbuvuhsnkeffdvtdgjbndokpynaujpipxdbwvlptkzjrekrocksfthxrsrcj
gswlsybelszdlkxbxtgbzseixjwmymelqdrnasldbelseiifyurwdheqxezvhmpnltf
fnqeyblmmgcphzeroljkohddyqrjeomknyucqlmgwjtbceszcwmjrcxdtcnioswhrnl
jrtgmfrahjeasofuvjjyzlfjdqzsjozjawbhephtbruxszatwnppxxdrtprrqzcivjz
```

We have a lot of "free" visual processing which we can't but help use. This can be used to help focus attention – preattentive processing happens effortlessly and guides our attention to things. On top of this processing I believe (and haven't done a search) that we see patterns in code before we read it, particularly with well laid out code in a syntax colouring editor.

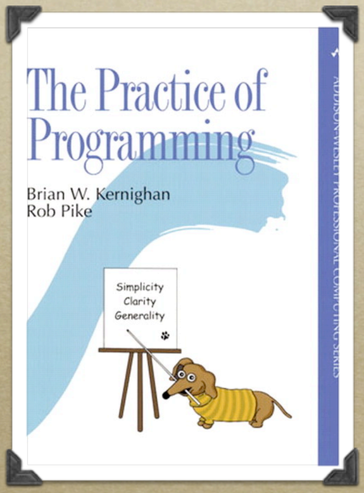
Point of this is to see if we can make as much fall into the background as possible, there's enough in code to worry about without distracting the programmer with "odd" constructs. We can help or hinder people by (mis)directing them.

Letter sequence was generated using for (1 .. 1000) { print +('a' .. 'z')[rand 26] } and inserting perl rocks, don't know where "whore", "wonk", "zero" came from.



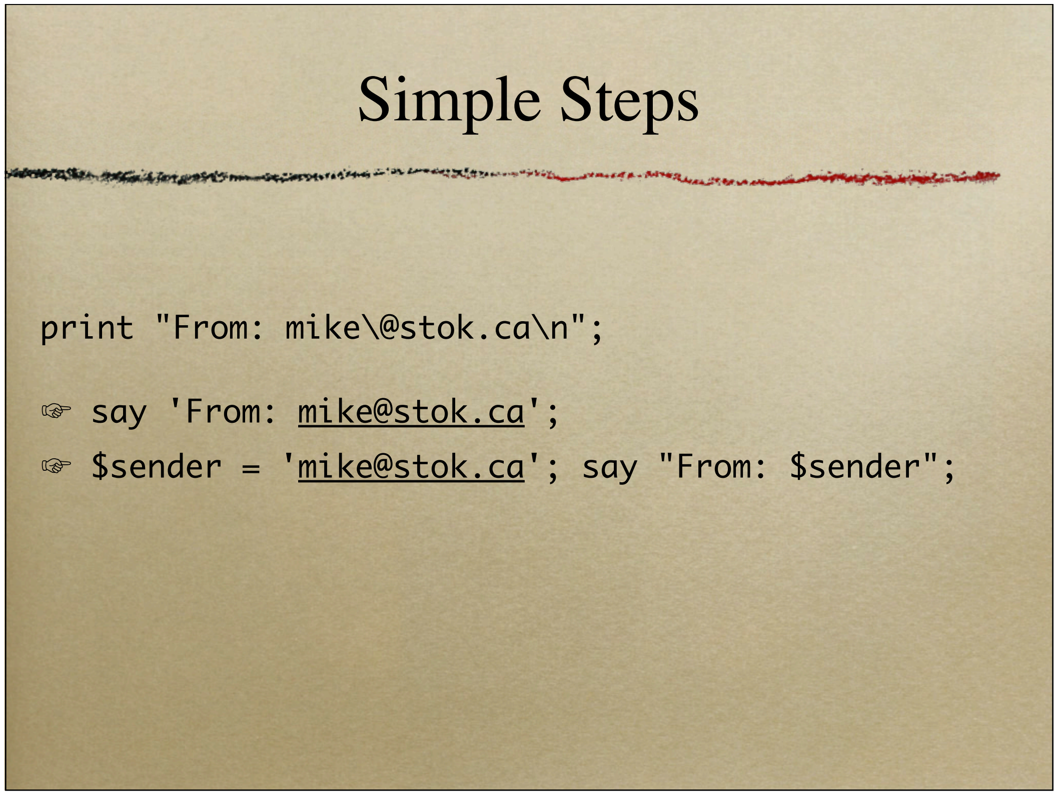
What Do I Care About?

- *Simplicity*
- *Clarity*
- *Generality*


The Practice of Programming
Brian W. Kernighan
Rob Pike
Simplicity
Clarity
Generality
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Communicating with other programmers. Writing code like poetry, but form is still subservient to function. Make the code only as complicated as it needs to be, strip it down as much as possible without abandoning Perl's power.

My preference is for small scale things.

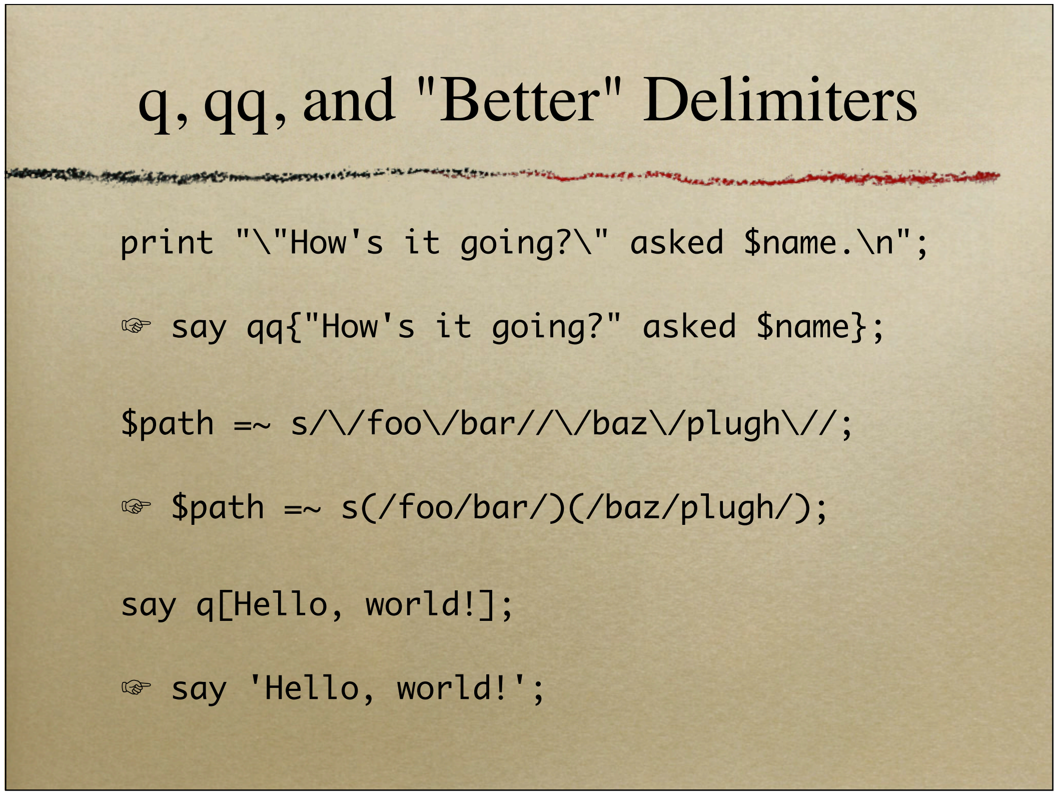


Simple Steps

```
print "From: mike\@stok.ca\n";
```

```
☞ say 'From: mike@stok.ca';
☞ $sender = 'mike@stok.ca'; say "From: $sender";
```

Get rid of backslashes. Maybe add semantic value.



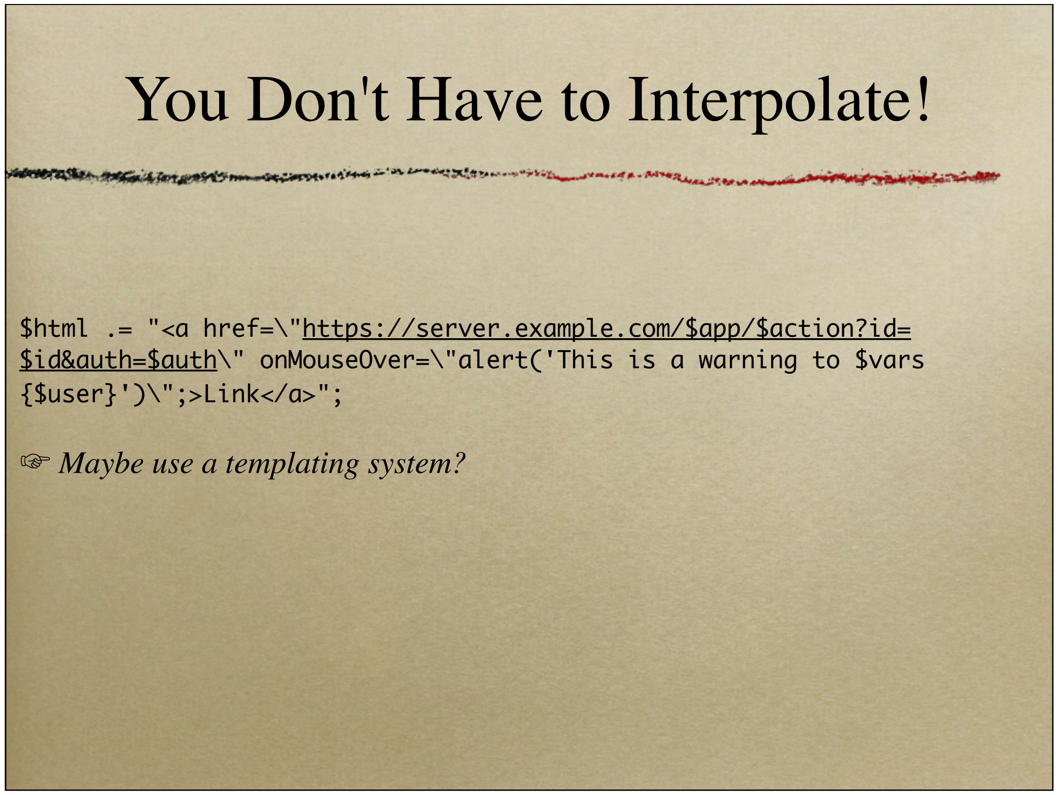
q, qq, and "Better" Delimiters

```
print "\"How's it going?\" asked $name.\n";

☞ say qq{"How's it going?" asked $name};

$path =~ s/\/foo\/bar//\/baz\/plugh\//;

☞ $path =~ s(/foo/bar/)(/baz/plugh/);

say q[Hello, world!];

☞ say 'Hello, world!';
```

Remember that you can pick {}, <>, [], () and they'll let you use balanced pairs inside the "quotes". For example say qq("Hi there $victim!" said $perp (quietly)\n); works.

My preference is to stick to " and ' where possible as people are used to seeing them, and try and use {} as string delimiters to avoid "unnecessary variety" in the code (though Perl over-uses {})

Context matters.



You Don't Have to Interpolate!

```
$html .= "<a href=\"https://server.example.com/$app/$action?id=
$id&auth=$auth\" onMouseOver=\"alert('This is a warning to $vars
{$user}')\";>Link</a>";
```

☞ *Maybe use a templating system?*

What needs to be escaped? How? If the JavaScript is complicated then can it be pulled out into a separate file and. TT has good modules for creating URLs.

It's The Filling That Matters

- *The filling is probably more important than the bread.*
- *Can we remove unnecessary "noise" from the filling?*
- *Does changing the quotes make the code clearer & simpler?*

http://www.che.lt/uploads/posts/thumbs/1211847802_1204207538_04_pics.jpg