What do converted templates look like?
String templates

- Prior to Scribunto, we have three “functions” for string processing
  
  ```
  {{#ifeq: str1|str2|if-true|if-false}}
  ```

- Return `if-true` or `if-false`, depending on whether `str1` and `str2` are the same.

  ```
  {{padleft:string|len}}
  ```

- This will add characters to `string` if it's shorter than `len`.

  ```
  {{padleft:|len|string}}
  ```

- This will truncate `string` to `len` characters, if it is longer.

- You can build a lot on this, with some work...
String templates - strlen

- How do you measure the length of a string with only “pad”, “truncate”, and “test equality” operations?
- You truncate to each possible length and check for equality.
- But remember that templates don't handle loops or recursion, either. So you have to unroll the loop.
- Yes, people *actually did this.*
Enwiki's Template:Str len, simplified

```latex
{{#ifeq: {{{1|}}} | {{padleft:{{{1|}}}]|500}}
| 500
| {{str len/core
|{{{1|}}}|
| {{str len/core
|{{{1|}}}|
| {{str len/core
|{{{1|}}}|
| | hundreds
|}}| tens
|}}| ones
}}
```

- For performance, it calculates the ones, tens, and hundreds digits separately.
Enwiki's Template:Str len/core, Just the part for the ones digit

• {{1}} is the string, {{2}} is the high digits of the length.

• Binary search:
  - If padding to length xx4 is equal to the original string, then it must be xx4-xx9
    • If padding to length xx6 is equal to the original string, then it must be xx6-xx9
      - If padding to length xx8 is equal to the original string, then it must be xx8 or xx9
        • So pad to xx9, and return xx8 or xx9 depending on if that's equal too.
      - Otherwise, it must be xx6 or xx7
        • So pad to xx7, and return xx6 or xx7
    • Otherwise it must be xx4-xx5
      - So pad to xx5, and return xx4 or xx5 depending.
  - Otherwise it must be xx0-xx3
    • Etc...
  • The tens and hundreds are similar, although for performance they do a linear search for 0-4 before doing a binary search for 5-9.
Strlen in Scribunto

• So what does it look like in Scribunto?
• Template:Str len

{{#invoke:String|len|s=>{{{1|}}}}}

• Module:String

local p = {}

function p.len( frame )
    return mw.ustring.len( frame.args.s )
end

return p

• That's it!
• Although someone has made the module on enwiki more complicated, because they want to support either trimming or not trimming whitespace from the string.
String templates - substr

• How do you extract a substring of a string with only “pad”, “truncate”, and “test equality” operations?

• You can do it if you can find the characters at positions \( i, i+1, i+2 \), and so on. But how do you do that?

• To find the character at position \( i \), you can see if the string truncated to length \( i-1 \) followed each possible character is equal to the string truncated to length \( i \).

• And remember that templates don't handle loops or recursion, either. So you have to unroll the loop.

• Yes, people actually did this.
Template: str sub, simplified

{{#ifeq:{{{2|0}}}|0|{{str_left |nocategory={{{nocategory|}}} |{{{1}}}|{{{3|0}}}}}|<noinclude>!--
--></noinclude>{{#ifexpr:{{{2|0}}} < 1 and {{{2|0}}} + {{{3|0}}} >= 1|<noinclude>!--
--></noinclude>{{str_index |nocategory={{{nocategory|}}} |{{{1}}}|1}}}}<noinclude>!--
--></noinclude>{{str_index |nocategory={{{nocategory|}}} |{{{1}}}|2}}}}<noinclude>!--
--></noinclude>{{str_index |nocategory={{{nocategory|}}} |{{{1}}}|3}}}}<noinclude>!--
--></noinclude>{{#ifexpr:{{{2|0}}} < 50 and {{{2|0}}} + {{{3|0}}} >= 50|<noinclude>!--
--></noinclude>{{str_index |nocategory={{{nocategory|}}} |{{{1}}}|50}}}}<noinclude>!--
--></noinclude>{{#ifexpr:{{{2|0}}} >= 50 or {{{2|0}}} + {{{3|0}}} > 50|{{FormattingError | nocategory={{{nocategory|}}} |max index is 50 for str_sub}}}|<noinclude>!--
--></noinclude>}}
**Template:** str index, simplified

```
{{str index/logic
 |*{{str left|{{1}}}{{2}}}}*
 |{{str left |{{1}}}{{2}}}|
{#expr:{{2}}-1}}
}}
```

- That's deceptive, it just does the initial truncations and then passes on to a helper template.

- Remember, the truncations and the giant switch are run for every position in the substring.
Substr in Scribunto

• So what does it look like in Scribunto?
  • Template:Str sub
    {{#invoke:String|sublen
    |s={{{1|}}}
    |i={{{2|0}}}
    |j={{{3|0}}}
    }}
  • Module:StringLength
    local p = {}
    function p.sublen( frame )
      local i = tonumber( frame.args.i ) or 0
      local len = tonumber( frame.args.len )
      return mw.ustring.sub( frame.args.s, i + 1, len and ( i + len )
    end
    return p

• That's it!
• Ok, those examples were a bit silly, since these string operations are built into Lua.
Other templates

• Citation templates!
  – Still being worked on
  – Still complex, but much easier to read
  – Already several times faster than the old templates, and probably could be optimized

• Convert template
  – Instead of huge numbers of subtemplates, will store conversion data in a submodule
  – The new mw.loadData() function is ideal for this purpose