COMP 302 Lecture 13 14 October 2016

Ambiguity

- In Wikitext, '' (two single quotes) indicates italics and ''' (three single quotes) to indicate bold. Therefore, what does it mean to write any multiple of 2 and 3 quotes?
- In templates, {{-> invocation, {{{-> parameter. "{{{{{{{" 6 curly braces): is this two parameters? Three invocation?
    - In the outer environment, there are no parameters. Therefore, the only way to parse this is three nested invocations.
    - When parameters and invocations mix, we need to decide.
        - {{{: parameters
        - {{{{: 1 parameter and a single character
        - We may decide to prioritize parameters. Therefore to get three nested invocations, we may use {{ {{ {{ (using spaces).

Now we can parse our template language.
Convert our string of characters -> AST.
Now we need to evaluate it

- "1pass" compiler -> evaluates as it parses. Most basic but not flexible.
- Multi-pass compiler -> processes the code repeatedly. Doesn't necessarily parse it repeatedly.

- Evaluate every node in the AST.
    - Starting from the root.
- Evaluate normal text (not template code, not parameters, not definitions) → we get the normal text.
    - eval("fooo") → "fooo"
        - "fooo" is supposed to be an AST node, the function converts the AST node to a string.
- When we see "{:" (definition), "{{" (invocation), "{{{" (parameter)

1. Evaluating a definition {:
    - {:name | param1 | param2 | … | body:}
    - Evaluate all the parts → not quite.
    - Evaluate the name. If it's just a string, we return the string. If it's something else, we recursively evaluate that until we eventually get a string.
    - Evaluate all the parameters.
    - Extract (not evaluate) the body.
    - With the above, we now have a function declaration that we can represent in an environment.
        - Internally, we have a structure to represent this:
            - Name: ___
            - Parameters: ___
            - Body → AST
        - Record this structure somewhere
    - For now, let's not worry about nested definitions. Assume none of them. Scope is not a concern.
        - Record this in a global array.

- evaluated({:…:}) → recorded definition.
    - return “” (empty string).

2. Evaluate an invocation {{
    - {{name | arguments | …}}.
    - Evaluate the name → string.
    - Evaluate each of the arguments.
    - Look up the name in our environment.
        - Find its list of parameters
        - Create a set of bindings mapping parameters to arguments
            - The environment we now execute the body in.
    - Evaluate the body in that environment.
    - e.g. *see 13c302*
    -