Regular Expression Cheat Sheet

- `[]` defines a range of characters.
- `.` matches any character.
- `\` is used to escape the following character when that character is a special character. So, for example, a regular expression that found '.com' would be `\\.com` because `.` is a special character that matches any character.
- `\d` matches any single digit.
- `\D` matches a non-digit. Opposite of `\d`
- `\w` matches any word character (equivalent to `[A-Za-z0-9_]`).
- `\W` matches any non-word character. Opposite of `\w`.
- `\s` matches any space, tab, or newline.
- `\S` matches a character that is not a space, tab, nor newline. Opposite of `\s`.
- `^` asserts the position at the start of the line. So what you put after it will only match if they are the first characters of a line.
- `$` asserts the position at the end of the line. So what you put before it will only match if they are the last characters of a line.
- `\b` adds a word boundary. Putting this either side of a stops the regular expression matching longer variants of words.
- `*` matches the preceding element zero or more times. For example, `ab*c` matches 'ac', 'abc', 'abbbc', etc.
- `+` matches the preceding element one or more times. For example, `ab+c` matches 'abc', 'abbbc' but not 'ac'.
- `?` matches when the preceding character appears zero or one time.
- `{VALUE}` matches the preceding character the number of times define by VALUE; ranges can be specified with the syntax `{VALUE,VALUE}`.
- `|` means or.
- `(...)` matches expression inside the parentheses, defining a group that can later be retrieved, such as for use with replacement, using a `\number` reference (backlash followed by group number)