

Week 10 Homework

New stuff learned this week:

Regular Expressions

- the `[]` surrounding characters creates a *character class*, (or *character set*) which match any of the characters in the set, so `[aeiouy]` would match any vowel
- character sets are a *logical unit* so they can be modified by the metacharacters `*` and `+`, so for example `[ae]+` means “match one or more characters that are either `a` or `e`”
- character classes can take *ranges*, like `[A-Z]` or `[0-9]`.
- character class ranges can be combined, like `[a-z0-9]`
- you can mix ranges and individual characters in a character class, so for example `[0-9_]` would match any *digit* or the *underschore character* `_`
- the *case insensitive flag* `/i` applies to character classes, so `/[ab]/i` is the same thing as writing `/[AaBb]/`
- the `^` character at the *beginning* of a character class *negates*. That means, that the character set matches *everything EXCEPT* what it would normally match. So `[^abc]` would match everything except the characters a, b, and c, and `[^0-9]` would match everything except digits.
- inside of a character class, most normal metacharacters represent their literal value: like `.` and `*` and `+` have no special meaning, they mean their actual character, and do not need to be *escaped*.

HTML

- HTML is a *markup* language, but not a *programming* language
- HTML is extremely *forgiving*, it doesn't throw errors, and tries to make sense of whatever you write, even if it's not technically correct.
- whitespace* (spaces, tabs, and newlines) in HTML is basically ignored. A single space, tab, or return is all treated as a single space, and adding more spaces or tabs has no effect. So `Foo bar` is exactly the same as `Foo bar`
- HTML is composed of *tags*, which generally open, have some inner content (or “children”) and then close, like `<i>foobar</i>`
- some HTML tags don't have any inner content, so they *self close*, like so: `
`
- HTML tags can have pairs of variable-like data attached to them called *attributes*. For instance `` denotes an `img` tag with an attribute `src` which has the *value* of `cat.jpg`
- the `p` HTML tag means a *paragraph*: `<p>Hello world!</p>`
- the `h1` - `h6` tags mean *headers* with decreasing levels of importance, like `<h1>Main title!</h1>` and `<h3>Not as important</h3>` and `<h6>Goat banjo unimportant</h6>`

- the `i` tag basically means *italic*, the `b` tag basically means **bold**.
 - a correctly formed HTML document has:
 - a *doctype* on the first line: `<!DOCTYPE html>`
 - then a `<html>...</html>` section with everything else inside it
 - then a `<head></head>` section (inside the `html` tag)
 - then a `<body>...</body>` section (also inside `html`)
-

Touch Typing Links:

- <http://touchtype.co>
 - <https://www.how-to-type.com>
-

Homework plan:

- 1 day reviewing and creating a few more flash cards
- 2 days CLI practice
- 1 day `vim` practice
- 2 days touch-typing practice
- 2 days **WEB** practice
- watch [CCCS#9](#) – just once

Homework day 1:

- do flashcard assignment (see below)
- touch typing practice
- `vimtutor` - Everything except Lesson 7

Homework day 2:

- CLI practice #1
- Web practice #1

Homework day 3:

- touch typing practice
- CLI Practice #2

Homework day 4:

- Web practice #2
 - watch [CCCS#9](#)
-

Flash Card Assignment

- Review all of your old cards
 - Make two new REGEX flash cards for *character classes* — one for normal usage, and one for *negated* usage
 - Make a new set of cards for `HTML` (put that in the upper left) covering these tags
 - `<html>`
 - `<head>`
 - `<title>`
 - `<body>`
 - `<h1>` ... through `<h6>` (one card can cover all six)
 - `<p>`
 - `<i>`
 - ``
 - ``
-

CLI Homework #1

1. carefully review the “New Stuff Learned this Week” Regular Expressions section of this document 😊
 2. `ssh` into your home dir and create a new directory called `week10` and then `cd` into it
 3. copy the file `char.txt` from the root directory of the computer down into your `week10` file, and `cat` it out so you can see what it says.
 4. using `cat` and `sed` plus a *character class*, print out the text of `char.txt` so that all the usages of the word `gray` or `grey` are changed to `blue` except for the last two words `graey` and `greay` (do not use the `|` alternation, only use `[...]` character classes)
 5. repeat step 4, but this time also include the last two misspellings.
 6. Repeat step 5, but this time, change the words to `r<something>d`, keeping the original vowels, so that the line reads `Rad can be spelled red or rad but not raed or read.`
 7. Change your `sed` expression so that line two gets fixed and the fourth word becomes `stepped` — you must use a character class with a *range*
 8. Change your `sed` expression so that lines 4 and 5 (with the pointers) both become `FOOBAR`
 9. Change your `sed` expression so that it strips out all of the *digits* from the last line, revealing the secret message!
 10. Change your `sed` expression so that everything that is *NOT* a digit is stripped from the last line, resulting in `5992432312114233433398992882`
-

CLI Homework #2

1. `ssh` into your home dir and `cd` into the `week10` dir

- copy the file `urls.txt` from the root directory of the computer down into your `week10` file, and `cat` it out so you can see what it says.
- using `cat` and piping to `sed` write a regular expression so that line 1 becomes `Web page (FOOBAR) (FOOBAR)`
- change your expression and use *backreferences* to make line one now become `Web page (./index.html) (./bar.html)`
- change your expression now so line 2 becomes `PDF: [] banana`
- change your expression now so line 2 becomes `PDF: none`
- change your expression now so line 2 becomes *blank*
- change your expression now so that the `href` and `src` attributes on lines 3 and 4 are both changed to `F00` like this `href="F00"` and `src="F00"` but make sure that *NONE of the other lines are changed!*
- make a new `sed` expression so that line 5 reads `Images: Foo.jpg cat.gif cat.png`
- Extra credit:** ✨ make a new `sed` expression so that line 5 reads `Images: <i>Foo.jpg</i> <i>cat.gif</i> <i>cat.png</i>` (hint: you'll need to escape the `/` character in your replacement string).
- Using a *negated character class* make a `sed` expression that turns the last line into `<p>Film flam</p>`

Web Homework #1


- carefully review the “New Stuff Learned this Week” HTML section of this document 😊
- `ssh` into your home dir
- create a new directory called `www` and move into that dir
- use `echo` and a redirect to create a file called `index.html` with the text: `Testing, 1, 2, 3`
- open a browser and navigate to `http://<yourname>.howtocomputer.link` (substituting your lowercase slack name for `<yourname>`) — you should see your message in the browser!
- now, open the `index.html` file with `vim` and rebuild the `html` page from scratch, so that it is a totally correct HTML page, be sure to include:
 - doctype
 - `html` tag
 - `head` tag with a `title` tag (check that you can see your title in the browsers tab)
 - `body` tag with some content
- now, edit the `index.html` page (remember, you don't need to close `vim` when you're editing the file and checking how it looks in your browser, you can just do `:w<enter>` to *write without closing*) — so that it has 6 *headings*, levels from `<h1>` through `<h6>`, each with the text `I am a level <number> heading!` where `<number>` is the heading level, like `<h3>I am a level 3 heading!</h3>`. View the outcome in the browser to see what the different headings look like.
- next, in between each heading, add a *paragraph tag* with some text in it. Save, and view the page again. you should see a block of text in between each heading.
- next, wrap bits of your paragraph text in 3 different tags, save, and view in the browser to see what affect they have on the appearance of your webpage. The three tags are: `<i>` `` and

`<code>`

- exit vim and copy the `index.html` file into a file called `foo.html` then open `http://<yourname>.howtocomputer.link/foo.html` in your browser. It should look exactly the same as the other file.
- open the `foo.html` file in `vim` and make a few modifications to it, save your changes, and view in a browser.
- close vim, and then *in one command* make 3 nested directories inside of `www` of `herp/derp/goat`
- `cd` down into the `goat/` dir
- copy the `foo.html` file you made and edited in steps 9-10 into your current directory (`goat/`)
- with `vim` edit the `foo.html` file you just copied into your current dir, changing the `title` tag and the `h1` tag to include the word `GOAT` .
- open your browser and type an address in that will let you see this new webpage.
- Extra Credit** ✨: if you know a bunch of HTML/CSS from Khan or somewhere else, make a new web-page and try to get a couple of things working:
 - a `<style>` tag in the `head` element
 - some fancy css styling to make your page look snazzy!
 - an external css stylesheet
 - an ordered list
 - an unordered list
 - a link to one of your other pages (using a FULL url including `http://`)
 - a link to one of your other pages (using a relative url 🤔)

Web Homework #2

- `ssh` into your home dir, and `cd` into the `www` dir
- list out the contents of the computers *root dir* and then, list out the contents of the `www-assets/` dir *inside of the root dir*
- in one command, copy *all three files* from the `www-assets/` dir down in your current working directory.
- now, make a copy of the `boilerplate.html` file (which should now exist in your `www/` dir) and name it `cat.html`
- open the `cat.html` file with `vim` and change the `title` tag to “Cats are Cute”
- in the `body` section, make a headline that reads `This cat is a cutie.`
- below your headline, insert an `img` tag that displays the `cat.jpg` file you copied into your `www/` dir in step 3. (hint: review the “new stuff learned this week” if you need a clue how to do this)
- save your `cat.html` and enter a URL into your browser that will let you see your webpage about the cute cat (it will start with `http://<yourname>.howtocomputer.link`)
- Next, edit the `cat.html` file again and add a paragraph of text below the image with some text about the cat from the picture. Save and view in a browser again.
- Exit `vim` and make a new directory called `animals` inside your current working directory.
- Still from the `www/` dir, move *ONLY* the `cat.html` file into `animals/` dir you just created
- Now, change the URL in your browser so you can view your `cat.html` webpage at it's new location. When you do, the image will be broken, it won't display.
- Figure out why the image doesn't load, and fix it.

14. `cd` into the `animals/` dir
15. use `cat` and a pipe and `sed` to copy the contents of the `cat.html` file into a new file called `goat.html` using `sed` to change all the instances of `cat` or `Cat` in the HTML to `goat` and `Goat`. You should not use `vim` OR the `cp` command to do this, just `cat sed` and a redirect, with some pipes. (you'll actually need TWO `sed` expressions piped together to preserve the uppercase/lowercase)
16. `cat` out the contents of your new `goat.html` file and see if your `sed` expression worked. If it doesn't look right, `rm` the file and repeat step 14 till you get it right.
17. view the `goat.html` file in your browser, you should see a picture of a goat!
18. copy the `boilerplate.html` file from your `~/www` folder down into your current working directory, renaming it `animals.html` in the process.
19. edit the `animals.html` file in `vim` and edit the `body` tag so that you have a single `p` tag that has the sentence "I like cats and goats". Save the file and view `animals.html` in your browser.
20. continue editing `animals.html` and now wrap the word `cats` with a `a` tag — the `a` tag creates a *hyperlink*  between webpages. The syntax is like this `some text`. Wrap the word `cats` with an `a` tag, and have the URL (inside the `href` attribute) point to the full web URL of your `cats.html` webpage. Save the file and view it in a browser. You should be able to click on the word `cats` and have your browser change to your `cat.html` webpage.
21. Repeat step 20, but this time, make the word `goats` link to your `goat.html` file. Save and view in your browser, testing that the link works.
22. Edit the `foo.html` file you made in steps 13-14 of the first Web homework, adding a paragraph that says `Check out my web page about animals!`. Make the words `web page` link to your `animals.html` page. Save and test in a browser.
23. Edit the `foo.html` file from step 22 again, and change the `href` attribute of the `a` tag so that it uses a *relative path* to the `animals.html` file. That means the `href` attribute should not start with `http://...` (hint: use your *relative path* skills from CLI to solve this, it works the same!)