PT()

LAB II/VI

SCHEDULE

Lab I

- **Project intro**
- **Context & Interaction**
- **GitHub**
- **Git**
- **Markdown**

Lab II (today)

I. Homework

II. Command line

III. Demo GH Pages

IV. Local machine

V. Job story

VI. Static site

Assignments

- Run
- GH Pages
- (Profile)
- Research

Schedule project tech

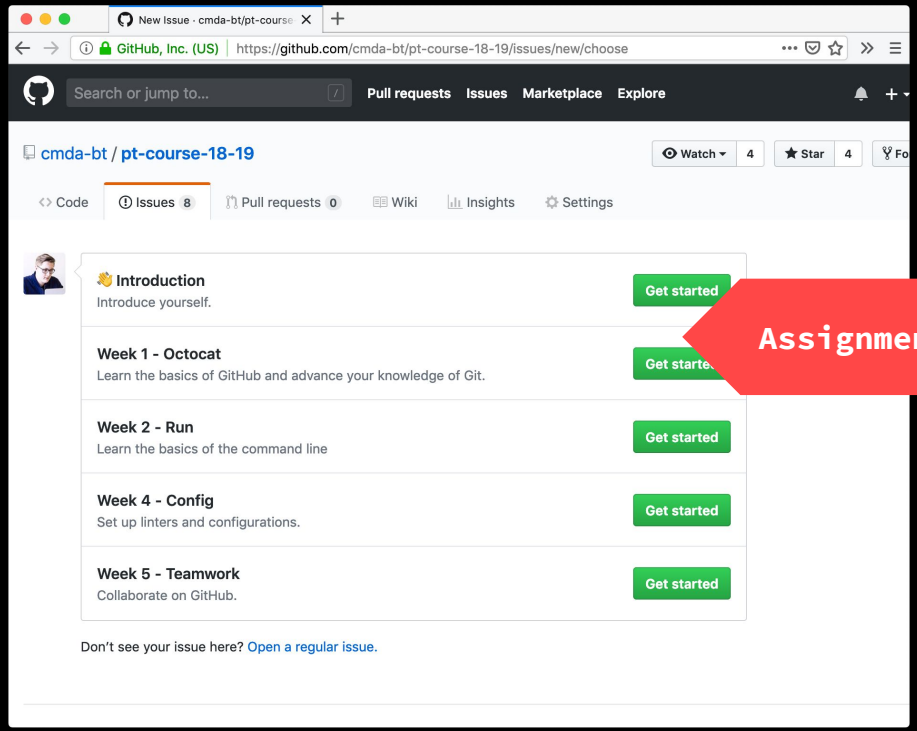
Wk 1:	20-24 Apr	Kick-off + Lab 1
	Free ~	
Wk 2:	04-08 May	Lab 2
Wk 3:	**14-18 May**	**Lab 3 - Present job story + static prototype (feedback)**
Wk 4:	18-22 May	Lab 4
Wk 5:	25-29 May	Lab 5
Wk 6:	**01-05 Jun**	**Lab 6 - Present indiv. assignment (grade indiv.)**
Wk 7:	08-12 Jun	Lab 7
Wk 8:	15-19 Jun	Lab 8
Wk 9:	**22-26 Jun**	**Final presentation (grade team)**
	...	...
Wk 10:	29-03 Jul	Resit (herkansing)

Homework

I/VI

HAND-IN

GITHUB



Assignments

https://github.com/cmda-bt/pt-course-19-20

COMMAND LINE

I/VI

II CLI

?

A command-line interface (CLI) […] is a means of interacting with a computer program where the user […] issues commands to the program in the form of successive lines of text […]. A program which handles the interface is called a […] shell

wikipedia.org

II CLI

?

A command-line interface [illegible] ns of interacting with a computer program where the user […] issues commands to the program in the form of successive lines of text […]. A program which handles the interface is called a […] shell

Alternative to **GUI** and the like

wikipedia.org

II CLI

?

A command-line interface […] is a means of interacting with a computer program where the user […] issues commands to the program in the form of successive lines of text […]. A program which handles the interface is called a […] shell

"Thing" where you talk to

wikipedia.org

II CLI

?

A command-line interface […] is a means of interacting with a computer program where the user […] issues commands to the program in the form of successive lines of text […]. A program which handles the interface is called a […] shell

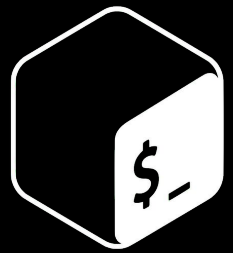
Note: Servers often have a CLI (not a GUI).
To control a server you need elementary knowledge of CLIs.

II CLI

BASH?

Bash is a Unix shell and command language […] it has been distributed widely as the default login shell for most Linux distributions and Apple's macOS […]. A version is also available for Windows 10.

wikipedia.org



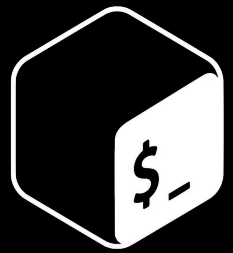
II CLI

BASH?

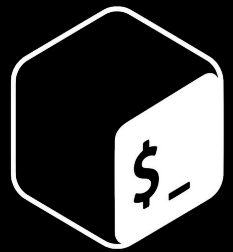
Bash is a Unix shell and command language […] distributed widely as the default login shell for most Linux distributions and Apple's macOS […]. A version is also available for Windows 10.

Often used, often available

wikipedia.org



```
[~]$ rm -f foo.txt
```

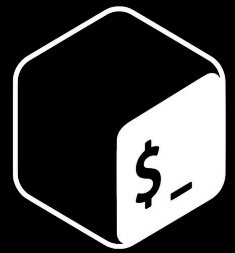


II CLI

BASH

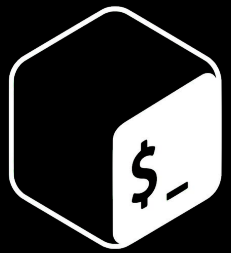
```
[~]$ rm -f foo.txt
```

Prompt



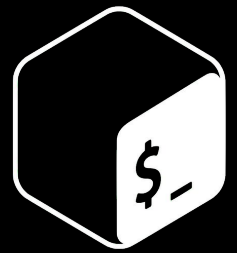
```
[~]$ rm -f foo.txt
```

Command

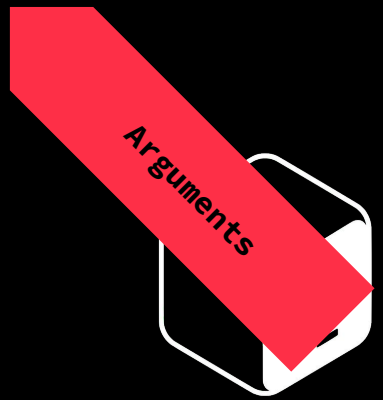


```
[~]$ rm -f foo.txt
```

Options



```
[~]$ rm -f foo.txt
```



II CLI

COMMANDS

- Navigate
- Files
- Apps
- Rights

II CLI

RECAP

pwd	navigate	print working directory
ls	navigate	list directory contents
cd	navigate	change working directory
touch	files	create file or change file access time
mkdir	files	make directory
rm	files	remove
mv	files	move
cp	files	copy
cat	files	concatenate files
echo	files	print
curl	files	transfer data
less	apps	read
vim	apps	write
man	apps	read the manual
sudo	rights	do something as someone else

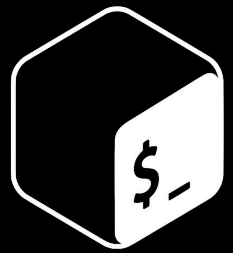
II CLI: FI

bash

```
[examples] $ echo "Hello World!"
Hello World!
[examples] $
```

ECHO

Print



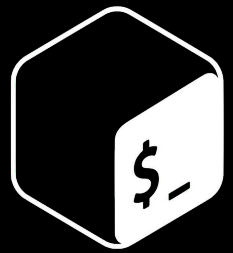
II CLI: N

bash

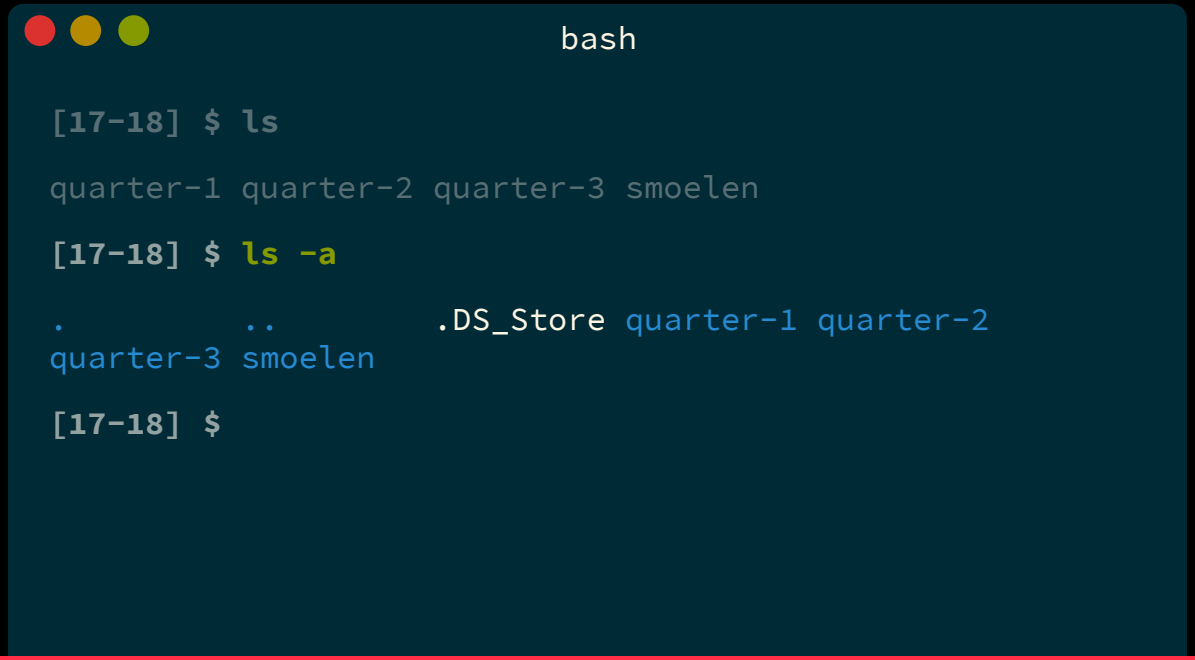
```
[17-18] $ pwd
/Users/tilde/Dropbox/hva/17-18
[17-18] $
```

PWD

Print working directory



II CLI: N



LS

Note: files starting with a **.** are hidden by default
ls -a shows them.
On macOS, Finder creates **.DS_Store**s: ignore them.

On macOs in finder, cmd+Shift⇧+. to show then

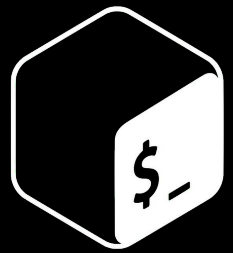
II CLI: N

CD

bash

```
[17-18] $ cd quarter-1
quarter-1 quarter-2 quarter-3 smoelen
[quarter-1] $ pwd
/Users/tilde/Dropbox/hva/17-18/quarter-1
[quarter-1] $ cd ../../16-17/quarter-3
[quarter-3] $ pwd
/Users/tilde/Dropbox/hva/16-17/quarter-3
[quarter-3] $ cd ~
[tilde] $ pwd
/Users/tilde
```

User directory



II CLI: N

CD

Tip: use lower-dash-case for file names: it’s easier to type. Definitely don’t use anything other than letters, periods, underscores, or dashes

II CLI: N

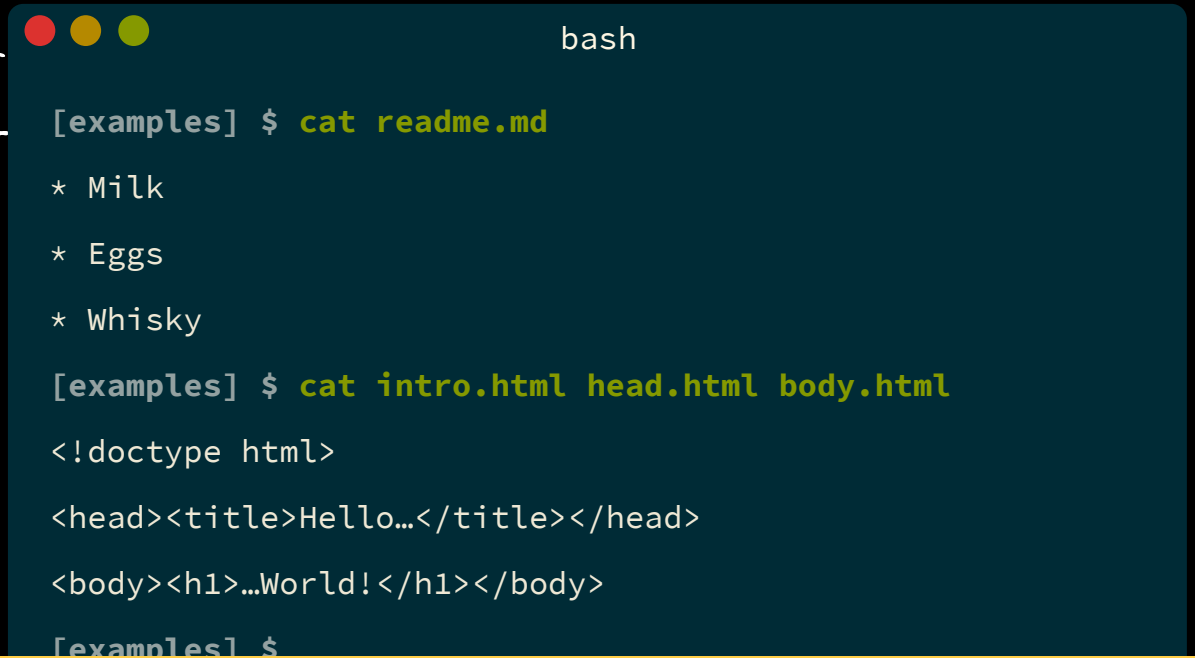
CD

Tip: type a tab (⇥) for autocomplete!

cd A⇥

cd Applications/

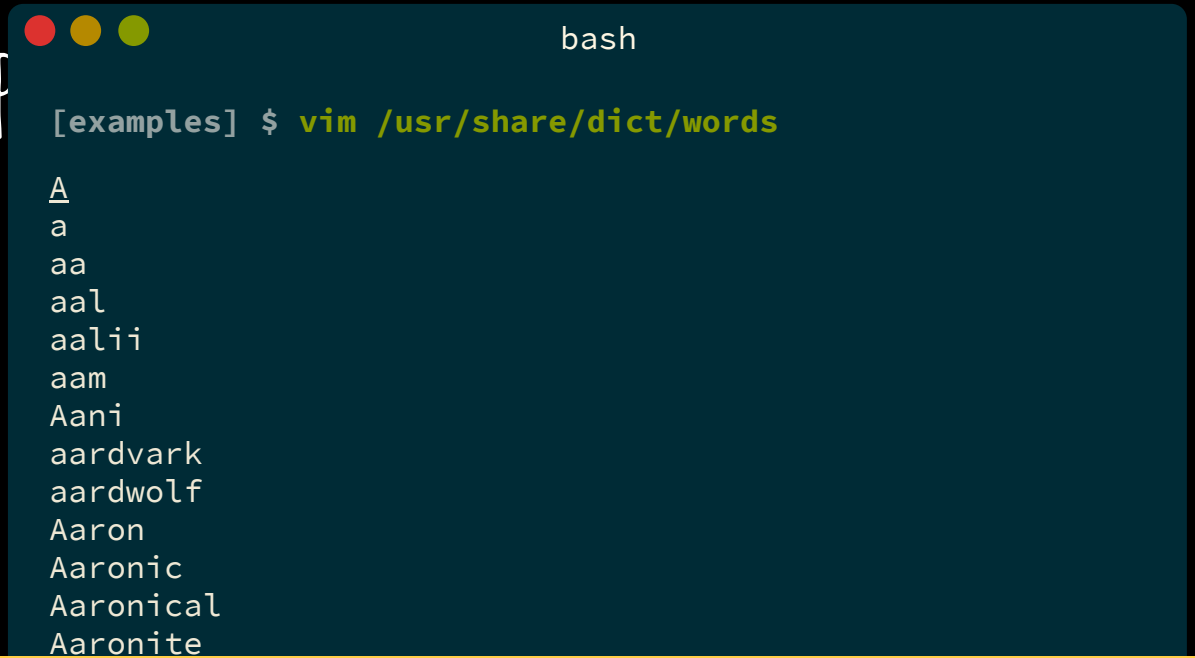
II CLI: FI



CLEAR

Tip: `clear` or `cmd+k` to clear the screen.

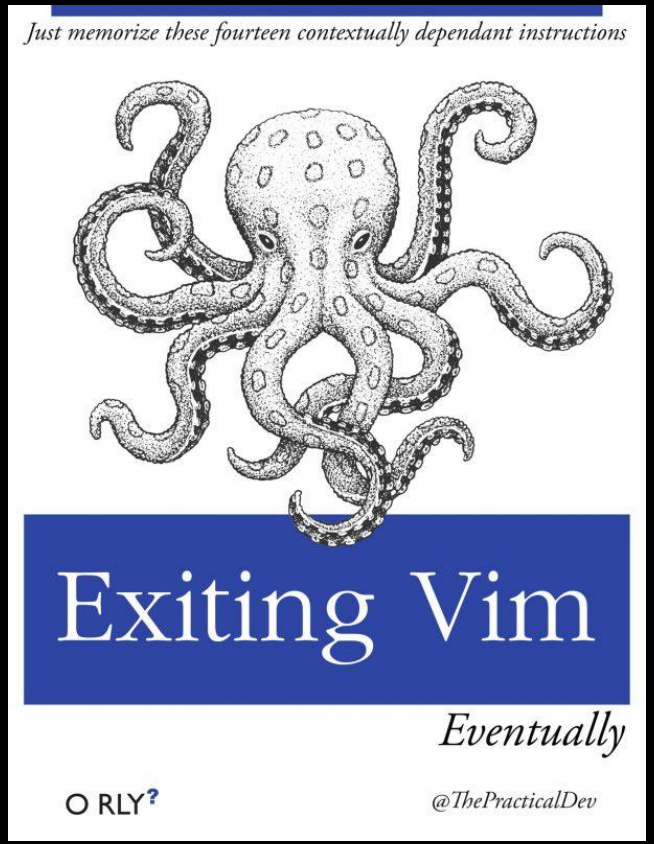
II CLI: AP



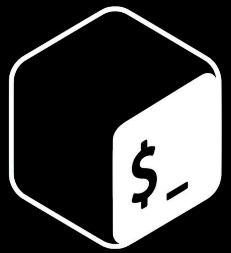
VIM

Tip: `:q!` to stop!

II CLI: APPS



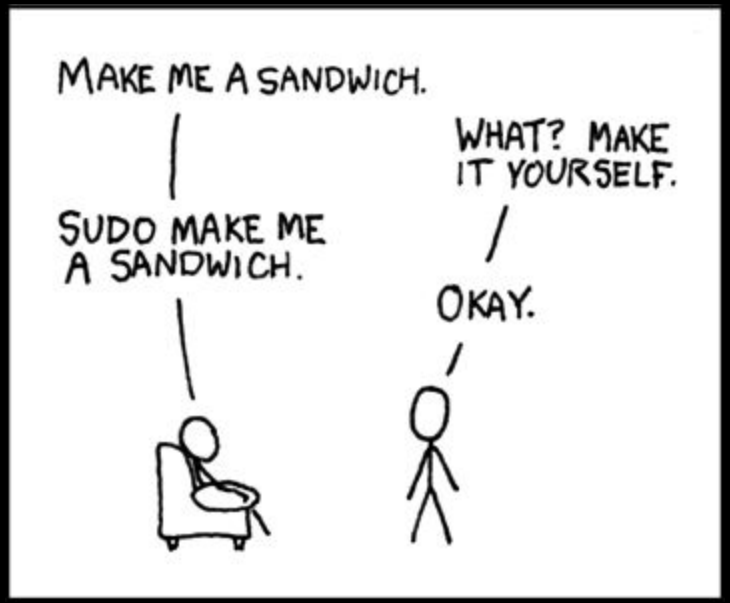
VIM



II CLI: RI

SUDO

Note: things often are not allowed because they are dangerous.
Sudo is **very** dangerous.



“Sandwich” via xkcd

II CLI

RECAP

pwd	navigate	print working directory
ls	navigate	list directory contents
cd	navigate	change working directory
touch	files	change file access time
mkdir	files	make directory
rm	files	remove
mv	files	move
cp	files	copy
cat	files	concatenate files
echo	files	print
curl	files	transfer data
less	apps	read
vim	apps	write
man	apps	read the manual
sudo	rights	do something as someone else

DEMO

COMMAND LINE

Assignments

Run the Command Line

In this assignment you'll learn the basics of the command line.

Synopsis

- **Practice**
- **Time**: 0:30h
- **Due**: before lab 3

Step A

Create a directory on your computer, `run`. In it, create a file `tutorial` the Gist into it:

`tutorial.sh gist`

> ⚠️ If you are on Windows, make sure the select LF line endings (Uni saving the file. In Atom, you can click on CRLF in the status bar and switch to LF. In SublimeText, go to the View menu, and click Line Endings.

In your terminal, go to the `run` directory and enter it (**hint**: use the `cd` command to "change directories").

If you now run `ls` (to print out files) in the directory, you should see the tutorial:

```
$ ls
# tutorial.sh
```

TAKE 30M TO BEGIN THE RUN ASSIGNMENT ON GITHUB

Demo GH Pages

III/VI

MACHINE

IV/VI

II MACHINE

PRETTY

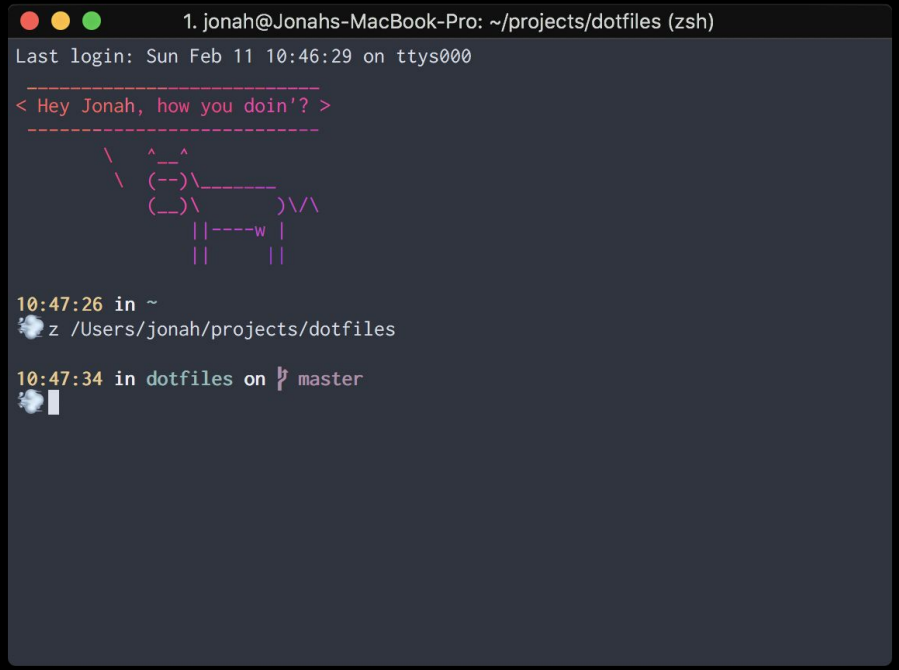
In this course, you'll find yourself staring at that black screen with green letters a lot. You'll type the same commands over and over again. **As you'll spend so much time there, it makes sense to make the command line more useful, and prettier.**

II MACHINE

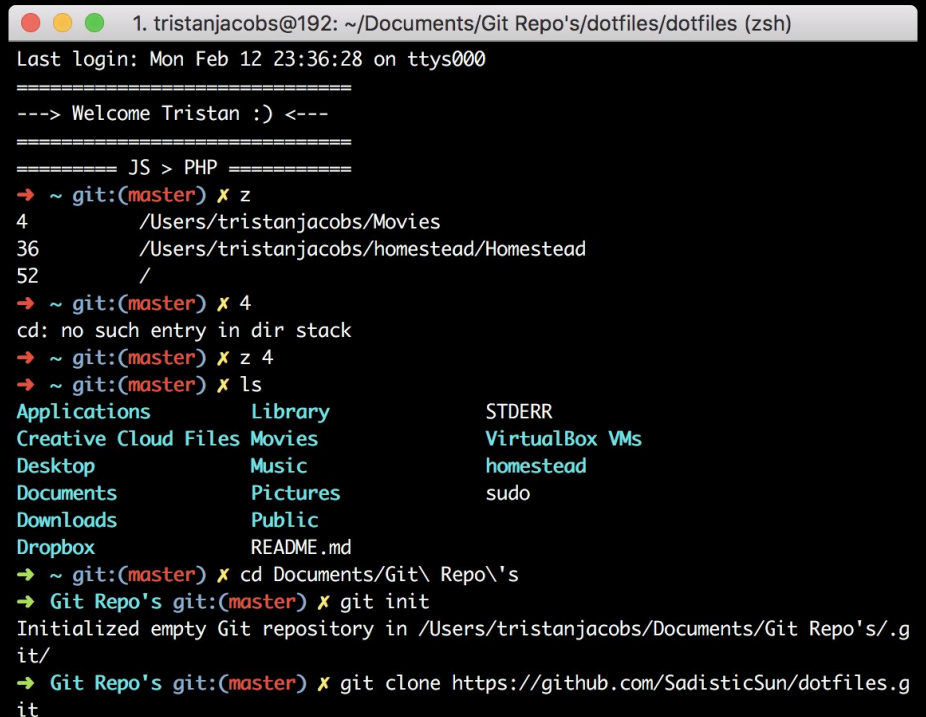
PROFILE

You can customise your command line by changing a **config file**. The file in question, called your **profile**, is often named *.bash_profile, .profile, or .bashrc*, depending on your operating system.

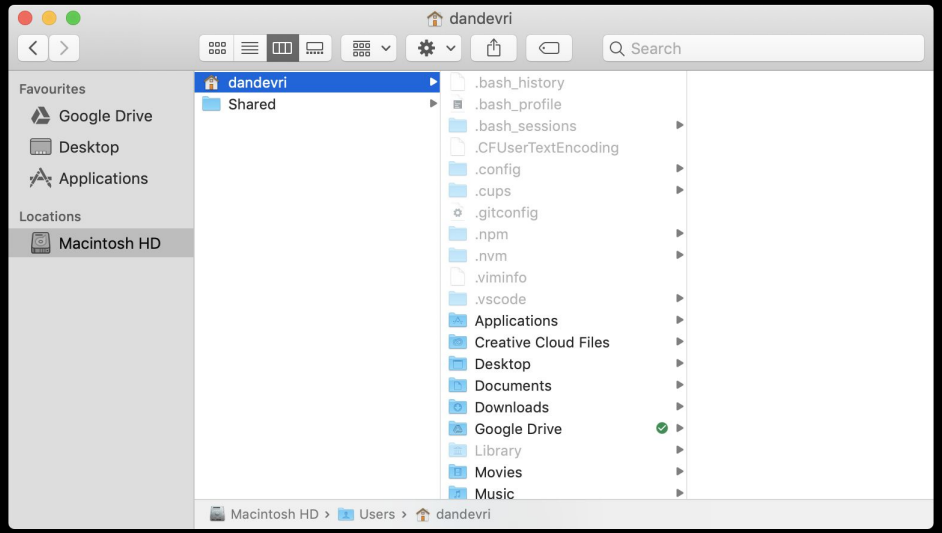
Karinliu/karinliu-dotfiles



theonejonahgold/dotfiles



SadisticSun/dotfiles



Note: files starting with a **.** are hidden by default

ls -a shows them.

On macOS, Finder creates **.DS_Store**s: ignore them.

On macOs in finder, cmd+Shift⇧+. to show then

Job stories

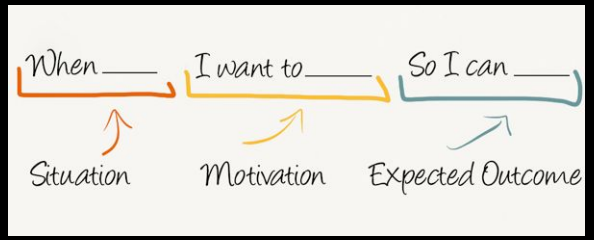
V/VI

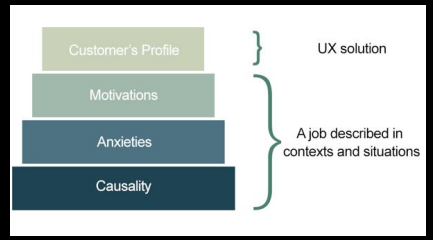
III JOB STORIES

Job Stories

When in teamwork you'll need to be able to cooperate in an efficient way. One aspect of efficiency is **clarity** for all members of the team. Job Stories provide an excellent way of dividing tasks within teams in an efficient focused way.

In order to define good job stories you'll need to know a bit more about the context of product use and look at the user's motivations. Together with a causality argument, they define how a product specifically needs to perform in a specific context. So a job story defines the job to be done, by focusing on **context, causality** and **motivations**.



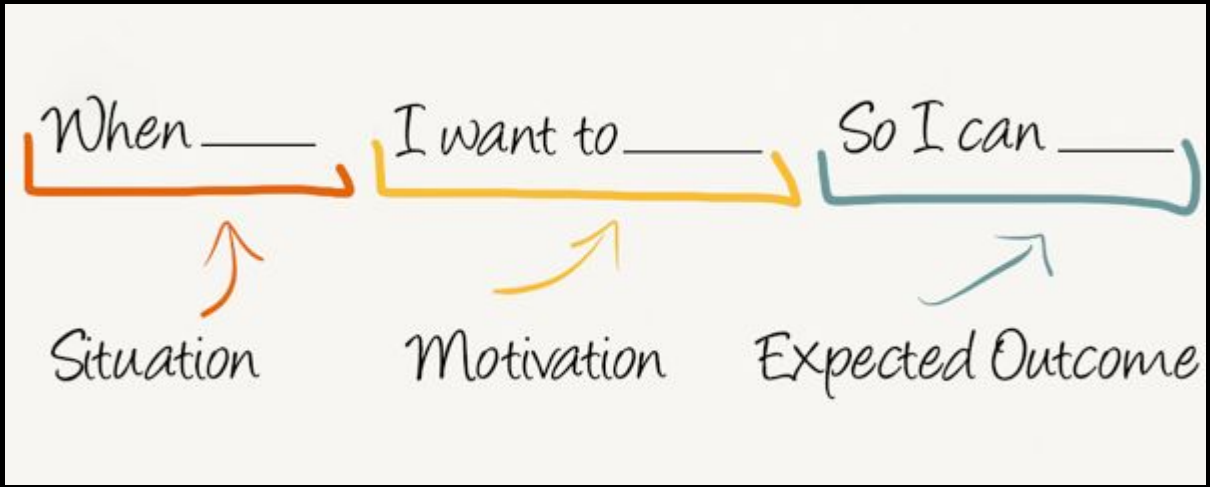


Inspiring article: https://www.intercom.com/blog/using-job-stories-design-features-ui-ux/

III JOB STORIES

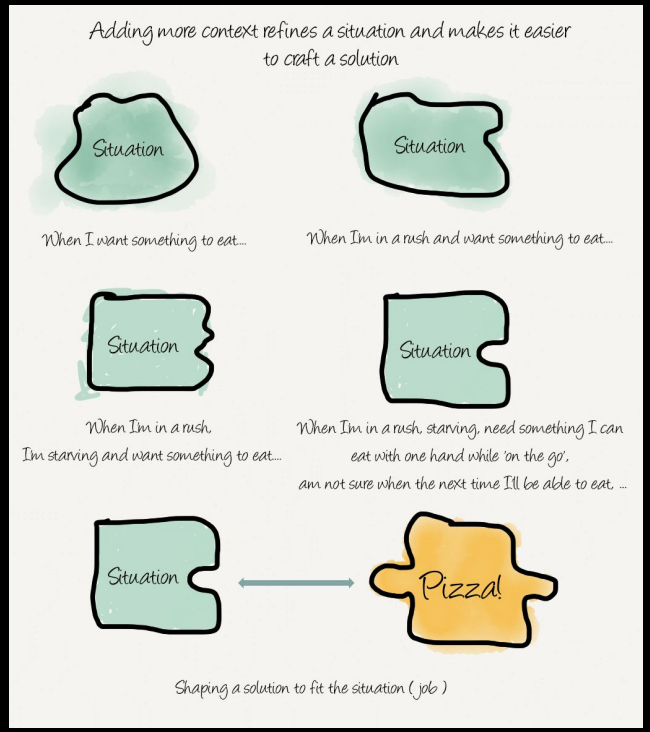
Job Stories

Defining the job to be done, by focusing on context, causality and motivations.



Inspiring article: https://www.intercom.com/blog/using-job-stories-design-features-ui-ux/

III JOB STORIES



Job Stories tip #1: Refine A Situation By Adding Contextual Information

*When I want something to **eat…***

*When I'm in a **rush** and want something to eat…*

*When I'm in a rush, I'm **starving** and want something to eat…*

*When I'm in a rush, need something I can eat with **one hand while 'on the go'**, am not sure when the next time I'll be able to eat...*

https://jtbd.info/5-tips-for-writing-a-job-story-7c9092911fc9

III JOB STORIES

Job Stories tip #2: Come From Real People Not Personas

Job Stories can only come from real customer interviews. Before designing a feature or new product, you must talk to real people and uncover all the anxieties and contexts which were in play when they used your or a competitor's product.

Good interview explanation:
https://soundcloud.com/jobstobedone/mattress-interview-live-jtbd-interview-debrief-analysis-jasonfried

III JOB STORIES

Job Stories tip #4: Add Forces To Motivations

Situation**:** *When I'm using my tablet and encounter a problem….*

Motivation**:** *I want to get help right away…*

***Force A**: I'm irritated because I was in the middle of something…*

***Force B**: I get nervous asking for help…*

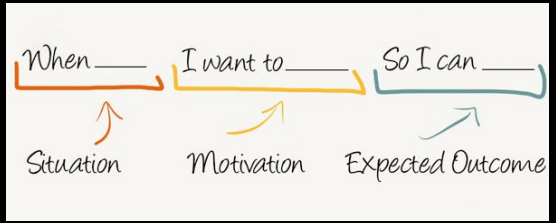
***Force C**: Asking for help might make me look stupid…*

Expected Outcome**:** So I can finish what I started.

https://jtbd.info/5-tips-for-writing-a-job-story-7c9092911fc9

III JOB STORIES

Job Stories: specific steps for creating job stories



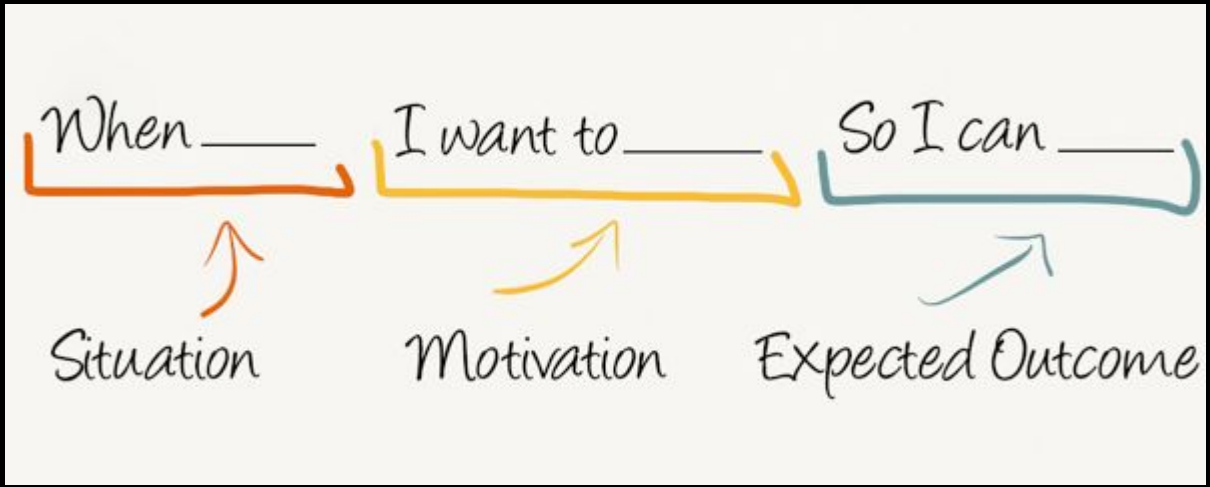
1. Start with the **high level** job.
2. Identify a **smaller** job or jobs which help *resolve* the higher level job.
3. Observe how people solve the problem **now** (i.e. which job do they currently use).
4. Come up with a **Job Story**, or Job Stories, that investigate the causality, anxieties, and motivations of what they do now. (If you are able, look at the *forces (the emotional state)* as well
5. Create a **solution** (usually in the form of a feature or UI change) which resolves that Job Story.

Inspiring article: https://www.intercom.com/blog/using-job-stories-design-features-ui-ux/

III JOB STORIES

Job Stories

Defining the job to be done, by focusing on context, causality and motivations.

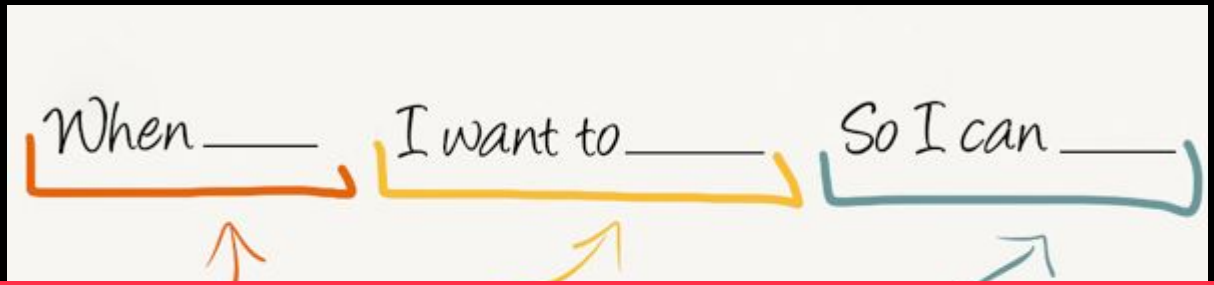


Inspiring article: https://www.intercom.com/blog/using-job-stories-design-features-ui-ux/

III JOB STORIES

Job Stories

Defining the job to be done, by focusing on context, causality and motivations.

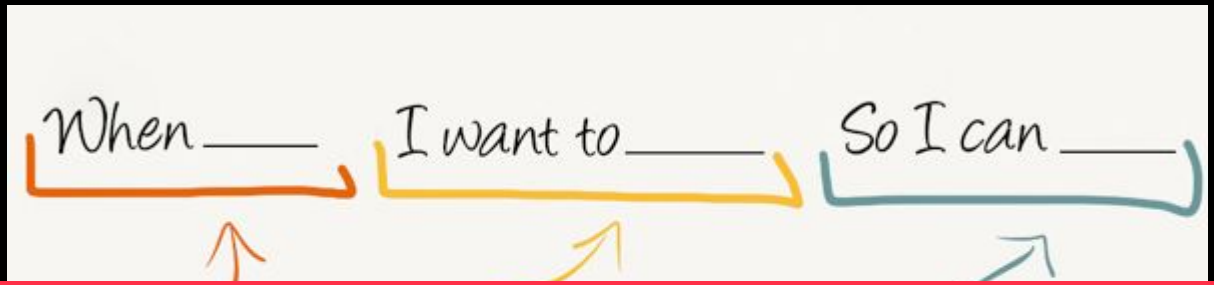


Note: These Job Stories are quite important for the other courses! It will determine which feature you are going to work on.

III JOB STORIES

Job Stories

Defining the job to be done, by focusing on context, causality and motivations.



Note: Make sure you have 1 well-defined Job Story before next lecture.

STATIC SITE

VI/VI

IV STATIC SITE

STATIC PAGES

Create a sitemap of all the relevant pages of your feature. You can use google drawings or omnigraffle to create an overview

Sketch some wireframes (or wireflows) of the interface. These can be rough sketches or more hi-fi. Find a fidelity that suits you.

IV STATIC SITE

STATIC PAGES

Turn your wireflow into static HTML pages. Do a HTML breakdown of your wireframe to see which semantic HTML elements you need. Ten create the HTML page for your feature.

IV STATIC SITE

Add some presentational CSS. You've already created a style guide so you're already able to set-up some basic styles. Think about colors, fonts etc.

IV STATIC SITE

STATIC PAGES

- Create a sitemap
- Sketch some wireframes
- Make (semantic) static HTML pages
- Add presentational CSS
- If you feel comfortable enough, you can also choose to create your site using express and a template engine

HOMEWORK

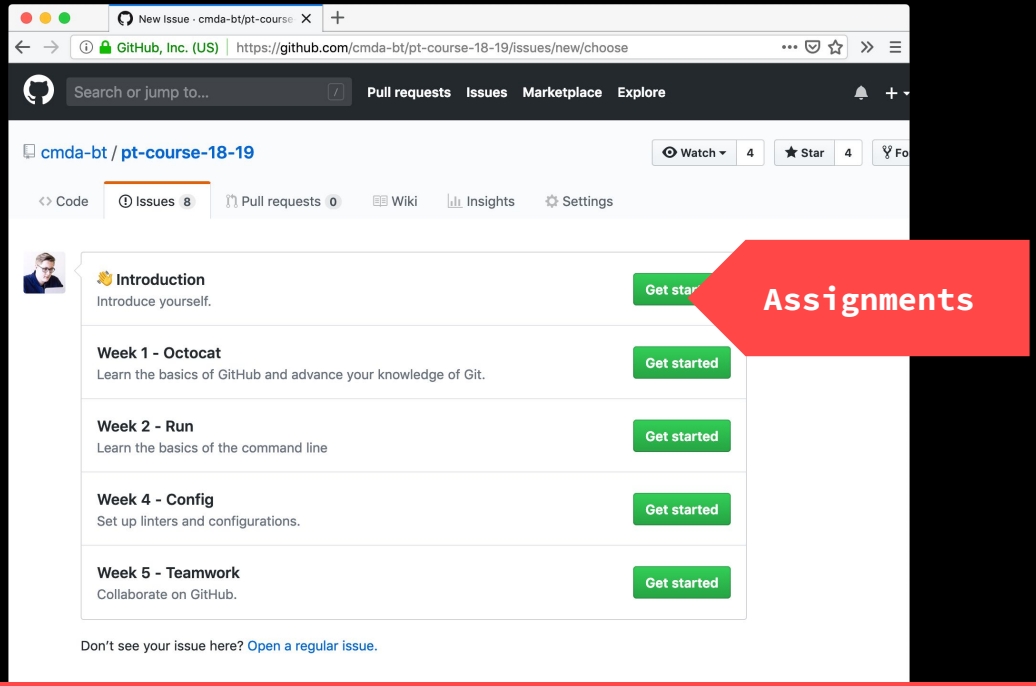
V/V

Schedule project tech

Wk 1:	20-24 Apr	Kick-off + Lab 1
	Free ~	
Wk 2:	04-08 May	Lab 2
Wk 3:	**14-18 May**	**Lab 3 - Present job story + static prototype (feedback)**
Wk 4:	18-22 May	Lab 4
Wk 5:	25-29 May	Lab 5
Wk 6:	**01-05 Jun**	**Lab 6 - Present indiv. assignment (grade indiv.)**
Wk 7:	08-12 Jun	Lab 7
Wk 8:	15-19 Jun	Lab 8
Wk 9:	**22-26 Jun**	**Final presentation (grade team)**
	...	...
Wk 10:	29-03 Jul	Resit (herkansing)

HAND-IN

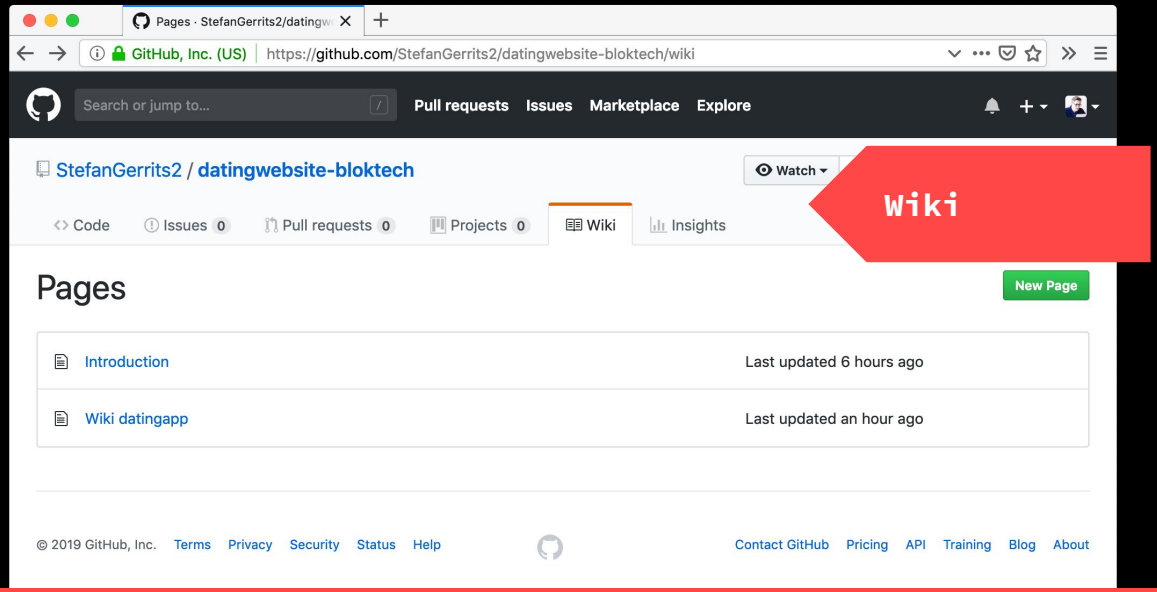
GITHUB



Hand in job story + static site before next project lesson

HAND-IN

GITHUB



Update wiki!

Time for working on new assignments

. . .

asking questions

. . .

EXIT;

SEE YOU IN LAB III/VI!