Bash Profile, Environmental Variables

Bash Profile

- We use the Terminal application on a Mac or git-bash on Windows to access the terminal
- By default, we've both been using bash, a command line application to interact with our computer
- If we want to customize our terminal experience or store variables accessible in any program run from the command line on Windows or Mac OS, we need to create a **bash profile** file

`~/.bash_profile`

- The file itself is a hidden file, known as a dotfile because its name is prepended with a dot
- It must live in your "home" directory, on a Mac this is /Users/yourname but no matter what system you're on you can access it using a tilde
- Using vim or the `subl` Sublime Text command line utility, open up your bash profile:

```
$ vim ~/.bash_profile
$ subl ~/.bash_profile
```

What goes in a bash profile?

- Custom prompt settings to make your prompt look exactly how you'd like it to
- "Environmental variables", special variables accessible to all command line programs you run. A great way to protect sensitive information like API keys.
- Lines to add directories to your $PATH - meaning to give Bash additional directories to scan for command line programs to include on startup of the command line, such as `cat` and `git`

Customizing your bash prompt

An example:

```
PS1="\W \u \$ "
```

If we add this line to our ~/.bash_profile, save/exit vim, and then source it:

```
source ~/.bash_profile
```

then our prompt should now read:

```
Location username $
Desktop zachfeldman $
```

Sourcing

- In order for your changes to be reflected in any open Terminal windows, you must **source** your bash_profile from the currently open session:

```
source ~/.bash_profile
```

- You can also simply open a new terminal window, which will automatically source your `~/.bash_profile`

Environment Variables

- While developing software, you'll often need to reference sensitive information in your source code
- Obviously it's not good to put API keys and password inside of version control and eventually up on GitHub
- **Even putting your credentials up on GitHub once can comprise them, even if you "wipe them" from Git**
- Instead, we can store these values locally or on the server in **environment variables** and then reference these variables in our code

Adding Environment Variables to our Bash Profile

To set an environment variable in your bash profile:

```
export VAR_NAME=var_value
```

make sure you **source** your bash profile once you've added the variable!

To be sure the variable "stuck", echo it out in the terminal:

```
echo $VAR_NAME
```

Environment Variables in Ruby

- To use your environment variables in a Ruby program, just use the ENV hash
- This hash contains all of the environment variables in the system in key value pairs
- For instance, to access the variable VAR_NAME in a Ruby program, use:

```
ENV['VAR_NAME']
```

Exercise: Understanding Env Variables

- Add an environmental variable to your `bash_profile`
- Source your `bash_profile`
- Create a Ruby program that outputs your new environmental variable