Navigating in Terminal

https://www.rithmschool.com/courses/terminal/terminal-navigation

* What is the difference between / and ~? What do we call each of these directories?  
  // ‘/’ represents a root diectory. ‘~’ represents the home directory.
* What command do we use to change directories?  
  // ‘cd’ + directory name to move in to a subdirectory.  
  // ‘cd ..’ to move up a directory.
* What is the difference between an absolute and relative path?  
  // a absolute path starts with ‘/’ and relative path does not.

Working with Files and Folders.

https://www.rithmschool.com/courses/terminal/terminal-files

1. Create a file called name.txt.  
   // cd Desktop  
   // touch name.txt
2. Try renaming the file to rename.txt using the mv command. What does this tell you about the command?  
   // mv name.txt rename.txt
3. Using the cp command, make a copy of rename.txt and call it copy.txt.  
   //cp rename.txt copy.txt
4. Remove the file copy.txt.  
   //rm copy.txt
5. Create a folder called questions.  
   //mkdir questions
6. Change directories to the questions folder.  
   //cd questions
7. Create a file called first.txt.  
   //touch first.txt
8. Create a file called second.txt.  
   //touch second.txt
9. Go back a directory and make a copy of the questions folder and call it questions\_copy.  
   //cd ..  
   //cp -r questions questions\_copy
10. When using cp -r what is the -r called? What does it do?   
    // ‘-r’ is a flag. It copy directories recursively
11. Delete the original questions folder and the copy.  
    // rmdir –rf questions

<https://www.rithmschool.com/courses/terminal/terminal-basics-exercises>

Terminal Basics Exercises.

### **Part I**

1. make a directory called first  
   //cd ~/Desktop  
   //mkdir first
2. change directory to the first folder  
   //cd first
3. create a file called person.txt  
   //touch person.txt
4. change the name of person.txt to another.txt  
   //mv person.txt to another.txt
5. make a copy of the another.txt file and call it copy.txt  
   //cp another.txt copy.txt
6. remove the copy.txt file  
   //rm copy.txt
7. make a copy of the first folder and call it second  
   //cd ..  
   //cp –r first second
8. delete the second folder  
   // rm –rf second

### **Part II**

1. What does the man command do? Type in man rm.   
   How do you scroll and get out? (Currently using a PC, will test after purchasing a Mac)  
   //The man command lists the manual for a command - D or Control ⌃-D to advance half a page - B or Control ⌃-B to go back one page - U or Control ⌃-U to go back half a page  
     
   //’man’ abbreviation for manual. It’ll list out option for a particular command.
2. Look at the man page for ls. What does the -l flag do? What does the -a flag do?  
   //’-l’ use a long listing format. ‘-a’ do not ignore entries starting with .(show hidden files)
3. Type the following command to download and save the contents of google.com: curl https://www.google.com > google.html
4. Use less to look at the contents of google.html.
5. Look at the man page for less. Read the section on /pattern. Search for the text **hplogo** in the google.html file.  
   // inside of less /hplogo  
   // in directory less –p hplogo google.html
6. How do you jump between words in the terminal?  
   (Currently using a PC, will test after purchasing a Mac)  
   //option left/right
7. How do you get to the end of a line in terminal?  
   //control + e
8. How do you move your cursor to the beginning in terminal?  
   //control + a
9. How do you delete a word (without pressing backspace multiple times) in terminal?  
   (Currently using a PC, will test after purchasing a Mac)  
   //option + delete
10. What is the difference between a terminal and shell?  
    //terminal is the interface for users to interact with.  
    //shell is the backend.
11. What is an absolute path?  
    // a absolute path starts with ‘/’ (aka current location)
12. What is a relative path?  
    //a relative path does not start with ‘/’. The entire patch needs to be printed out starting from the root directory.
13. What is a flag? Give three examples of flags you have used.  
    //a flag is additional option for a command. The flag starts with ‘command\_name –‘ then the flag. Three examples of flags used : –a, –l, - rf
14. What do the r and f flags do with the rm command?  
    // ‘r’ remove directories and their contents recursively, ‘f’ force removal

Permissions, Redirection, and Piping Exercise.

<https://www.rithmschool.com/courses/terminal/terminal-permissions-redirection-piping-exercises>

### **Part I (Permissions and links)**

Write terminal commands to do the following:

1. Create a file called restricted.txt.  
   //touch restricted.txt
2. Change the permissions on the restricted.txt file to allow the owner to read and write to the restricted.txt file. Do this using the **Octal** Notation.  
   //chmod 600 restricted.txt
3. Change the permissions on the restricted.txt file to only allow the owner, group and everyone to read and write and execute the restricted.txt file. Do this using the **Symbolic** notation.  
   //chmod a+rwx restricted.txt
4. Create a folder called secret\_files. Inside the secret\_files folder create a file called first\_secret.txt and another folder called classified. Inside of the classified folder create a file called super\_secret.txt.  
   //mkdir secret\_files  
   //cd secret\_files  
   //touch secret\_files/first\_secret.txt  
   //mkdir secret\_files/classified  
   //touch secret\_files/classified/super\_secret.txt
5. Change the permissions on the secret\_files to only allow the owner and group to read, write and execute in all the files and folders inside of secret\_files. Do this using the Octal Notation.  
   //chmod –R 770 secret\_files
6. Create a hard link for the restricted.txt called hard\_link.  
   //cd secret\_files  
   //ln restricted.text hard\_link
7. Create a symbolic link for the classified folder called classified\_link.  
   //ln -s classified classified\_link

### **Part II (Piping and Redirection)**

For the following exercises, create a text file called vegetables.txt with the following text:

Lettuce

Amaranth

Beet

Celery

Kale

Dill

Cabbage

Broccoli

Lettuce

Amaranth

Beet

Spinach

Chard

Broccoli

Cabbage

Dill

Write the following terminal commands to do the following

1. Sort vegetables.txt.  
   //sort vegetables.txt
2. Count the number of lines in vegetables.txt.  
   //wc –l vegetables.txt
3. Create a file called vegetables\_sorted.txt which contains all the unique vegetables sorted in ascending order in vegetables.txt (do this without the touch command).  
   //sort vegetables.txt | uniq > vegetables\_sorted.txt
4. Create a file called last\_three.txt which contains the last three vegetables in the vegetables.txtfile (do this without the touch command).  
   //vegetables.txt | tail –n 3 > last\_three.txt
5. Count the number of lines the word "Broccoli" appears on (using wc and grep).  
   //cat vegetables.txt | grep “Broccoli” | wc -l

Intermediate Terminal Exercises.

<https://www.rithmschool.com/courses/terminal/terminal-intermediate-terminal-exercises>

### **Part I**

Answer the following questions:

1. Create an environment variable called FIRST\_NAME and set it equal to your first name (this does not need to be permanent)  
   // cd ~/Desktop  
   // export FIRST\_NAME=Renwick
2. Print the FIRST\_NAME variable  
   // echo $FIRST\_NAME
3. Print out the $PATH variable  
   // echo $PATH
4. What is the $PATH variable?  
   // paths for environment to find where to run commands
5. Why would you want to create an environment variable?  
   //Securing info. Quickly go to enviro locaction
6. How do you permanently save environment variables?  
   //paste in the export line into .zshrc or .bash\_profile.
7. What is a process?  
   //a computer execution currently running
8. How do you list all processes running on your machine?  
   // ps aux
9. What is a PID?  
   //process ID. Unique ID for each process running.
10. How do you terminate a process?  
    //kill PID
11. What is the difference between kill and kill -9?  
    //Sometimes kill PID will not terminate the process then kill -9 is used to end the process.
12. What grep flag allows for case insensitive search?  
    // grep -i
13. What grep flag allows for a certain number of lines before the match?  
    // grep –B num
14. What grep flag allows for a certain number of lines around the match?  
    //grep –C num
15. What grep flag allows for a certain number of lines after the match?  
    //grep –A num
16. What grep flag allows for full word search?  
    //grep -w
17. What grep flag shows you the line number of a match?  
    //greo -n

### **Part II**

Write the following terminal commands to do the following (assume that instructors.txt is an imaginary file):

1. Find all files inside the Desktop folder that have a name of "learn."  
   //find -name “learn”
2. Find all files inside the Desktop folder that start with a "P."  
   //find -name “P\*”
3. Find all files inside the Desktop folder that end with .txt.  
   //find -name “\*.txt”
4. Find all files inside the Desktop/views folder that have the name data somewhere in their filename.  
   //cd views  
   //find –name “\*data\*”
5. Inside of the instructors.txt file, output the number of times the word "Elie" appears.  
   //cat instructors.txt | grep –c “Elie”  
   //grep –c “Elie” instructors.txt
6. Inside of the instructors.txt file, list all matches for any full word that starts with a capital "P."  
   //cat instructors.txt | grep –w “P\*”  
   // grep –w “P\*” instructors.txt
7. Inside of the instructors.txt file, list all the line numbers for any full word that starts with a "z" (it should match regardless of upper or lower case).  
   //cat instructors.txt | grep –ni “z.\*”  
   // grep –ni “z.\*” instructors

SSH – waiting for MAC.

Sdd

Advanced Terminal Exercises.

<https://www.rithmschool.com/courses/terminal/terminal-advanced-terminal-exercises>

Use the following text file to answer the questions

Elie-Schoppik-sushi

Tim-Garcia-gummybears

Janey-Keig-bagels

Colt-Steele-tacos

Matt-Lane-pizza

1. Replace all of the - with : using sed  
   // sed –ie ‘s/-/:/g’ filename
2. Return a file with just the first name and last name separated by a space (you can do this with cutand sed or just sed)  
   // cut -d- -f1-2 something.txt | sed 's/-/ /g'   
   //cut -d- -f1 -f2 instructors.txt | sed 's/-/ /g' solution did not work

1>>>>2

2>>>>3

3>>>>4

4>>>>5

1. Using cut print out just the numbers 2, 3, 4, 5. Use xargs to print them all on 1 line  
   // cut -c 6 something.txt | xargs
2. Using xargs in the ./Desktop directory, find all of the files that include the text Welcome  
   // find ~/Desktop -name '\*' | xargs grep Welcome
3. Write a shell script called access\_file.sh which accepts one parameter and changes the permissions to 755  
   //touch access\_file.sh  
   //chmod 755 access\_file.sh
4. Write a shell script called unaccessible\_sh.sh which accepts one parameter and changes the permissions to 300  
   //touch unaccessible\_sh.sh  
   //chmod 300 unaccessible\_sh.sh
5. Using sed write the command to replace all instances of the name "foo" with the string "bar" in a file called baz.txt  
   //sed ‘s/foo/bar/g’ > baz.txt
6. Write the command to only print out all of the pids using awk  
   //ps aux | awk ‘{print $2}’
7. Type in the df -h command - it will show you much space you have on your hard drive. Using the awk command, print out **only** the first percentage capacity.  
   //df –h | awk ‘FNR == 2 {print$6}’

Basic Git Commands.

https://www.rithmschool.com/courses/git/git-github-basic-commands

1. Create a folder called learn\_git. (Make sure you do this from a folder that isn't a git repository!)  
   //mkdir learn\_git
2. cd into the learn\_git folder.  
   //cd learn\_git
3. Create a file called first.txt.  
   //touch first.txt
4. Initialize an empty git repository.  
   //git init
5. Add first.txt to the staging area.  
   //git add .
6. Commit with the message "adding first.txt".  
   //git commit –m “adding first.txt”
7. Check out your commit with git log.  
   //git log
8. Create another file called second.txt.  
   //touch second.txt
9. Add second.txt to the staging area.  
   //git add .
10. Commit with the message "adding second.txt".  
    //git commit –m “adding second.txt”
11. Remove the first.txt file.  
    //rm first.txt
12. Add this change to the staging area.  
    //git add -A
13. Commit with the message "removing first.txt".  
    //git commit –m “removing first.txt”
14. Check out your commits using git log.  
    //git log

Git Basics Exercises.  
  
<https://www.rithmschool.com/courses/git/git-github-git-basics-exercises>

1. Create a folder called learn\_git\_again.  
   //mkdir learn\_git\_again
2. cd into the learn\_git\_again folder.  
   //cd learn\_git\_again
3. Create a file called third.txt.  
   //touch third.txt
4. Initialize an empty git repository.  
   //git init
5. Add third.txt to the staging area.  
   //git add .
6. Commit with the message "adding third.txt".  
   //git commit –m “adding third.txt”
7. Check out your commit with git log.  
   //git log
8. Create another file called fourth.txt.  
   //touch fourth.txt
9. Add fourth.txt to the staging area.  
   //git add .
10. Commit with the message "adding fourth.txt"  
    //git commit –m “adding fourth.txt:
11. Remove the third.txt file  
    //rm third.txt
12. Add this change to the staging area  
    //git add -A
13. Commit with the message "removing third.txt"  
    //git commit –m “removing third.txt”
14. Check out your commits using git log  
    //git log
15. Change your global setting to core.pager=cat - you can read more about that [here](https://git-scm.com/book/en/v2/Customizing-Git-Git-Configuration).  
    //git config –global core.pager “cat”
16. Write the command to list all of the global configurations for git on your machine. You can type git config --global to find out how to do this  
    //git config –global -l

Checkout and Reset.

<https://www.rithmschool.com/courses/git/git-github-checkout_reset>

1. Create a folder called destruction.  
   //cd ~/Desktop  
   //mkdir destruction
2. cd into that folder.  
   //cd destruction
3. Initialize an empty git repository.  
   //git init
4. Create a file called done.txt.  
   //touch done.txt
5. Remove that file from the working directory (remember you can not use git checkout).  
   // git clean –df
6. Create a file called stage\_me.txt.  
   //touch stage\_me.txt
7. Add stage\_me.txt file to the staging area.  
   //git add .
8. Move stage\_me.txt file from the staging area to the working directory.  
   //git rm - - cached stage\_me.txt
9. Add stage\_me.txt file to the staging area.  
   //git add .
10. Remove stage\_me.txt from the staging area **and** the working directory.  
    //git reset --hard
11. Create a new file called commit\_me.txt.  
    //touch commit\_me.txt
12. Add commit\_me.txt to the staging area.  
    //git add .
13. Commit with the message "adding commit\_me.txt".  
    //git commit –m “adding commit\_me.txt”
14. Create another file called second.txt.  
    //touch second.txt
15. Add second.txt to the staging area.  
    //git add .
16. Commit with the message "adding second.txt".  
    //git commit –m “adding second.txt”
17. Check out your previous commits using git log --oneline to see the unique identifier or SHA for each of your commits.  
    //git log –oneline
18. Using git reset, undo the previous commit and move your changes back to the **working directory**.  
    //git reset - - mixed 6a6f798
19. Add second.txt again.  
    //git add .
20. Commit with the message "Trying to commit again".  
    //git commit –m “Trying to commit again”
21. Using git reset undo the previous commit and move your changes back to the **staging area**.  
    //git reset – soft 6a6f798
22. Commit with the message "Trying to commit again and again".  
    //git commit –m “Trying to commit again and again”
23. Using git reset undo the previous commit so that any changes are **not** part of the working directory.  
    //git reset –hard 6a6f798
24. Pat yourself on the back! You just went through a pretty complex git workflow!

Branching

<https://www.rithmschool.com/courses/git/git-github-branching>

1. Create a folder called branch\_time.  
   //mkdir branch\_time
2. cd into that folder.  
   //cd branch\_time
3. Initialize an empty git repository.  
   //git init
4. Create a file called first.txt, then add and commit the file.  
   //touch first.txt
5. Create a new branch called amazing\_feature.  
   //git checkout –b amazing\_feature
6. Create a file called best.txt.  
   //touch best.txt
7. Add the file.  
   //git add .
8. Commit the file with the message -m "added best.txt"  
   //git commit –m “added best.txt”
9. Switch back to the master branch.  
   //git checkout master
10. Merge your changes from the feature branch into master.  
    //git merge amazing\_feature
11. Delete the feature branch.  
    //git checkout –D amazing\_feature

Branching and Merging Exercises.

<https://www.rithmschool.com/courses/git/git-github-branching-merging-exercises>

### **Part I**

Answer the following questions:

* What does git clean do?  
  //git clean removes untracked files that are not in the staging area.
* What do the -d and -f flags for git clean do?  
  //-d Remove untracked directories in addition to untracked files. If an untracked directory is managed by a different Git repository, it is not removed by default. Use -f option twice if you really want to remove such a directory.  
    
  //-f If the Git configuration variable clean.requireForce is not set to false, git clean will refuse to delete files or directories unless given -f, -n or -i. Git will refuse to delete directories with .git sub directory or file unless a second -f is given.
* What git command creates a branch?  
  //git checkout –b NEW\_BRANCH\_NAME
* What is the difference between a fast-forward and recursive merge?  
  //fast-forward merging with original branch only when original branch have no commits  
  //recursive is when both have commits
* What git command changes to another branch?  
  //git checkout BRANCH\_NAME
* How do you remove **modified or deleted** files from the working directory?  
  // git checkout BRANCH\_NAME
* What git command deletes a branch?  
  //git branch –D BRANCH\_NAME
* What does the git diff command do?  
  //compare two different commits
* How do you remove files from the staging area?  
  //git rm –cached FILE\_NAME
* How do merge conflicts happen?  
  //Open conflicted file. Edit and save the file.  
  //Stage and commit newly saved file

### **Part II**

Create your own merge conflict! Work on the same file on two separate branches and merge the two branches together. Fix the conflicts and finish your merge. In the real world you will never intentionally try to create merge conflicts, but it is important to understand how and why they are created. Most importantly, it's important not to be intimidated by merge conflicts, and to be able to fix them with confidence!

GitHub Exercises

<https://www.rithmschool.com/courses/git/git-github-github-exercises>

### **Part I**

Let's start by taking a bit of time to practice the git workflow below. It is so valuable to just practice this workflow a couple times, since you will most likely doing it professionally as well as in your individual projects and open source contributions. Here are some things to do.

1. Create a local repository and add and commit some files  
   //mkdir GitHub\_Exercise  
   //cd GitHub\_Exercise  
   //git init  
   //echo "something" > test.txt  
   //git add .  
   //git commit –m “First Commit”
2. Create a remote repository and push your code from the local repo to the remote  
   //github.com/new  
   //create new repository : github\_exercise  
   //git remote add origin GITHUB URL  
   //git push –u origin master  
   //git push origin
3. Fork the repo <https://github.com/rithmschool/git_practice> - clone it and submit a pull request  
   //Click the Fork button  
   //copy SSH url  
   //mkdir more\_practice && cd more\_practice   
   //git clone SSH\_URL  
   //git remote upstream ORIG\_REP\_URL
4. Create a new branch locally and push it to GitHub  
   // git checkout –b “new\_branch”
5. Submit a pull request with your new branch against the master branch on the git\_practice repo.  
   //git pull origin master

### **Part II - Command Line Murder Mystery**

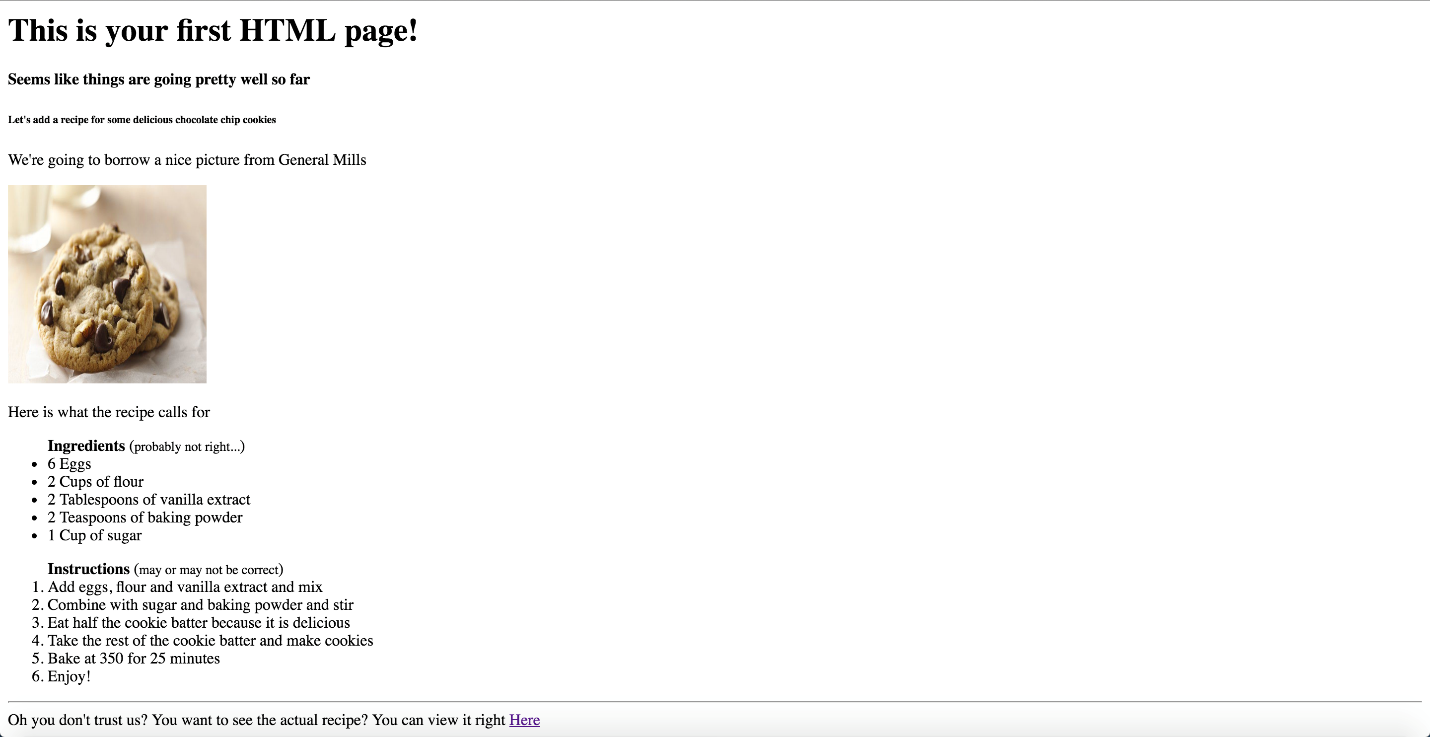
If you have gone through our Terminal and UNIX course, take the time to combine your knowledge of the Terminal, Git, and GitHub to complete a murder mystery! You can find the exercises [here](https://github.com/veltman/clmystery). Fork and clone the repository and try to solve the mystery!

7/14/2017 – will complete later

HTML Basics Exercise

<https://www.rithmschool.com/courses/html-css-fundamentals/html-basics-exercise>

Write the necessary HTML to make your page look just like this one:



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

</head>

<body>

<h1>This is your first HTML page!</h1>

<h2>Seems like things are going pretty well so far</h2>

<h3>Let's add a recipe for some delicious chocolate chip cookies</h3>

<p>We're going to borrow a nice picture from General Mills</p>

<img src="https://images-gmi-pmc.edge-generalmills.com/eb52c020-c145-440c-8445-911f133c0096.jpg" alt="chocole chip cookies">

<p>Here is what the recipe calls for</p>

<ul><b>Ingredients</b> (probably not right...)

<li>6 Eggs</li>

<li>2 Cups of flour</li>

<li>2 Teaspoons of vanilla extract</li>

<li>2 Teaspoons of baking poweder</li>

<li>Cup of sugar</li>

</ul>

<ol><b>Instructions</b> (May or may not be correct)

<li>Add Eggs</li>

<li>Combine</li>

<li>Eat half the cookie batter</li>

<li>Take the rest of the cookie</li>

<li>Bake at 350</li>

<li>Enjoy!</li>

</ol>

<hr>

Oh you don't trust us? You want to see the actual recipe? You can view it right <a href="something.html">Here</a>

</body>

</html>

Intermediate HTML Exercise.

<https://www.rithmschool.com/courses/html-css-fundamentals/intermediate-html-exercise>

### **Part 1** Write the necessary HTML to make your page look just like this one:

|  |  |
| --- | --- |
| html mockup blog with table and form | <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Non-semantic HTML</title>  </head>  <body>  <header>Inside of header - let's add a navbar witd some essential links</header>  <nav>  <a href="#">Home </a>  <a href="#">About</a>  <a href="#">Contact</a>  </nav>  <hr>  <h2>Here is tde main section of our page - it consists of a new article</h2>  <article>  <h2>Article 1 -1995</h2>  <h3>Don't believe me?</h3>  <table>  <tdread>  <tr>  <th>Language</th>  <th>Designed By</th>  <th>First Appeard</th>  </tr>  </tdread>  <tbody>  <tr>  <td>C</td>  <td>Bjarne</td>  <td>1995</td>  </tr>  <tr>  <td>C</td>  <td>Bjarne</td>  <td>1995</td>  </tr>  <tr>  <td>C</td>  <td>Bjarne</td>  <td>1995</td>  </tr>  <tr>  <td>C</td>  <td>Bjarne</td>  <td>1995</td>  </tr>  <tr>  <td>C</td>  <td>Bjarne</td>  <td>1995</td>  </tr>  </tbody>  <tfoot>  <tr>  <td>C</td>  <td>Bjarne</td>  <td>1995</td>  </tr>  </tfoot>  </table>  </article>  <article>  <h2>Article 2- Comming soon....</h2>  </article>  <h2>Contact</h2>  <form action="">  <h2>General Information</h2>  <div><label for="#">header</label><input type="text"></div>  <div><label for="#">header</label><input type="text"></div>  <div><label for="#">header</label><input type="text"></div>  <div><label for="#">header</label><input type="text"></div>  <br>  <h2>Essential Questions</h2>  <span>What kind of work do you need?</span>  <div><label for="#">test</label><input type="checkbox"></div>  <div><label for="#">test</label><input type="checkbox"></div>  <div><label for="#">test</label><input type="checkbox"></div>  <div><label for="#">test</label><input type="checkbox"></div>  <br>  <span>How interested are you in our design skills?</span>  <div><label for="#">test</label><input type="radio"></div>  <div><label for="#">test</label><input type="radio"></div>  <div><label for="#">test</label><input type="radio"></div>  <div><label for="#">test</label><input type="radio"></div>  <div><input type="submit" value="Submit me!"></div>  </form>  <footer>Here is our footer &copy; </footer>  </body>  </html> |

### **Part 2** Write the necessary HTML to make your page look just like this one Don't worry about the song / video being the same - you can use whatever video you'd like!

|  |  |
| --- | --- |
| html mockup video from youtube | <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>Non-semantic HTML</title>  </head>  <body>  <header>  <h1>  Welcome Back!  </h1>  </header>  <article>  <h3>Here's a greate video from YouTube!</h3>  <iframe src="https://www.youtube.com/embed/dQw4w9WgXcQ" width="560" height="315" frameborder="0" allowfullscreen> </iframe>  </article>  <article>  <h3>Here's some music from Spotify!</h3>  <iframe src="https://open.spotify.com/embed?uri=spotify:user:spotify:playlist:3rgsDhGHZxZ9sB9DQWQfuf" width="300" height="380" frameborder="0" allowtransparency="true"></iframe>  </article>  <article>  Want to stay informed? Put your info here!  <div>  <form action="#">  <label for="email">Email: </label>  <input type="email" name="email" id="email">  <input type="submit" value="Join OurMailing List">  </form>  </div>  </article>  </body>  </html> |