# MISCs – MISCs – MISCs – MISCs - MISCs – MISC s

## Chrome Console

Start developer tools: cmd+alt+J

Console on/off: Esc

See all my soure code in new tab: cmd+alt+U

## Terminal Command Line

Ruby pry

Puts `clear` to clear the screen

Back-tick says we run command in the command line

Clear is the command line to clear the screen

### Folders

. current directory

.. parent directory

Pwd where I am now

Cs change directory

Cd .. go to parent directory (..)

Cd ~ go to home

Cd / go to hard drive

Cd - go to previous directory

Cd -2 go to where I was 2 steps ago

Tree shows folder tree (after brew install tree)

Ls list items in the directory

Ls –a all including hiddens

Ls –l list files plus info on them (can be –la for hidden files)

Mv file to move multiples: into the last one named

Mv folder/file . move file into “.” = current working directory

Cp copy

Cp –r recursive copy: copy all subfolders and content

Touch file create a file

Open

Mkdir create folder

Mkdir –p this/path to create a directory struture

Rm remove **cannot retrieve file, deleted forever!**

Rmdir remove directory if empty **cannot retrieve file, deleted forever!**

Rm –r : recursive remove: remove folder and all its **content - CANNOT RETRIEVE!**

Permissions on file: chmod change mod

### Text

Echo add text to a file

Printf “blabla” >> file.txt

Use > to overwrite the file with the new text

Cat <file> display text that’s on a file

Grep to search for text

env | grep "^T" >> t-vars.env search, within result of printenv, for words that start with T, and add that to t-vars.env

^T has to begin with T

T$ has to end with T

Wc count, words, lines, bytes

wc –l file.txt count lines in a file (wc =

Find . –name “.git” find, in current working dir, stuffs with name ending .git

find ~ -name "\*.txt" | wc -l >> text-files-count.txt find, in home dir, files with name ending .txt, then count number of lines for this result, then add that to the text\_file\_count

sort sort alphabetically

head -5 show first 5 lines

head show first 10 lines

head file1 file2 show first 10 of these 2 files

<http://www.linfo.org/redirection.html>  
As an example of the separate redirection of the two output streams from a single command, the wc command can be used to attempt to count the lines, words and characters in two files: *nonfile*, which does not exist, and *file10*, which does exist. Standard output is redirected to the file*wordcount* and standard error is redirected to the file *errorlog*:

wc nonfile file10 > wordcount 2>> errorlog

Subl . open current working dir in Sublime

### Environment variable

Create a variable as global

echo 'export WDI13\_TEST=homework' >> ~/.zshrc

source ~/.zshrc

. ~/.zshrc

### Instal other ‘Gems’

Always run rbenv rehash afterwards to restart terminal

gem list –local to see the installed gems

**NGROK**

Type python or py.

Create webserver on a port 9000 : python -m SimpleHTTPServer 9000

Then run ngrok http 9000

## Rake Tasks

touch Rakefile - it will be a ruby file

desc 'make some coffee'

task :make\_coffee, :cups do |task, args|

# anything coming after the task if an argument

puts task

puts args

puts “making #{args[:cups]} coffee"

end

call it in the terminal… weird syntax!! watch the “ “

**➜ rake** rake 'make\_coffee[5]'

make\_coffee

{:cups=>"5"}

making 5 coffee

### Create our own Gemfile that we will then pass to any Rails project

# this has to be run FROM THIS RAKEFILE FOLDER

# Pass in as arguments, e.g /Users/Guiton/Desktop/WDI13/classwork/w11/rake\_test\_app

I

# rake 'custom\_gemfile[/Users/Guiton/Desktop/WDI13/classwork/w11/rake\_test\_app]'

desc 'Moves our pre-config Gemfile to any Rails project we want'

task :custom\_gemfile, :rails\_app\_location do |task, args|

destination = args[:rails\_app\_location]

source = "#{Dir.pwd}/Gemfile\_template"

FileUtils.copy(source, destination)

end

## Sublime Text

Cmd+d highlight all instances of a word, able to make changes on all occurences at the same time

Cmd+click : multiple cursos

## GIT

<https://github.com/ga-students/WDI_LDN_13_CLASSNOTES/blob/master/week_1/d2/git_and_github.md>

Git –version

Git init to initalise the project on the folder we want

Git status

**Git add <file>**

git add . will add all files created since last commit

git rm –cache <file> to remove from staging

**git commit** –m “optional message after –m”

Git diff to see what’s changed since last commit

Git diff <file/directory> for a specific one

git reset HEAD <file> remove from staging (after having done git add)

git checkout <file> remove changes made to file since last commit

git log to see history

git log –p see log plus changes on each commit

Get from Github

1. Fork
2. Open terminal navigate to folder you want
3. git clone <clone url>
4. make changes locally and commit
5. push them to site, git push origin master
6. submit a pull request on the main website

<https://help.github.com/articles/fork-a-repo/>

git remote add upstream <url> to create upstream remote

<https://help.github.com/articles/syncing-a-fork/>

Check all branches

git remote –v check the origin of my repo

**Create another branche and put it online**

First create a new repository (see below),: gperrin01.github.io

This is my page that people can see

Push my content into this new repo

git remote add pages [git@github.com:gperrin01/gperrin01.github.io.gi](mailto:git@github.com:gperrin01/gperrin01.github.io.gi)

* add another “remote” to my repo, named pages
* doing git remote shows ‘origin’ + ‘pages’

Then you can push content on to the new branch

git push pages master

# HTML – HTML - HTML – HTML - HTML – HTML –

## HTML Basics

<!DOCTYPE html>

<html>

<head>

<title></title>

<style> </style>

</head>

<body>

<header> </header>

<main> </main>

<article>

<h1> </h1>

<p> </p>

</article>

<footer> </footer>

</body>

</html>

Ordered list: <ol> <li> </li> </ol>

Unordered list (bullet points): <ul> </ul>

Line break <br>

Comment: <!-- text as comment -->

Link: <a href=”URL” target=“blank”> Text to click on </a>

Target=‘blank’ opens it on a new tab

Image: <img src=”URL” alt=”sth that would be spoken” title=”it will show when you hover over it”>

*// ALT is used by search engines*

*Head is important for search engines*

## HTML Tables:

<table border="1px">

<thead>

<tr>

<th colspan=”2”> Full names in the group </th>

</tr>

<tr> <th>First Name</th> <th>Last Name</th> </tr>

</thead>

<tbody>

<tr> <td> Guillaume </td> <td> Perrin </td> </tr>

<tr> <td> Cedric </td> <td> Rozier </td> </tr>

</tbody>

</table>

Styling the table:

<table style="border-collapse:collapse”>

<tr style="border-bottom:1px solid black;">

<th style="padding:5px;border-left:1px solid black;"> (padding = indent I think)

## HTML Divisors <div>, <articles>

Used to divide page into pieces we can style individually – visual objects such as sidebars, menus

Div are **block** elements – they take the whole width of the page

Easy to wrap in <a> so they are clickable

<body>

<a href="http://www.nba.com">

<div style="width:50px; height:50px; background-color:yellow"> click for NBA </div>

</a>

</body>

<Articles> do the same with texts. Good to group an h2 and its p

## HTML Styling

Attributes, for headings, paragraphs, lists, links, etc

<h1 style=”color:red; font-family:arial; font-size:12px; background-color:yellow; text-align:left”> blabla </h1>

Bold and Italic are simple tags: <bold> <em> ***your word*** </em> </bold>

<bold> is same as <b> same as <strong>

<em> same as <i>

<SPAN> tag to style only a part of a sentence or paragraph etc

<p> this text is <span style=”color:red”> red </span> <p>

To get faster, create <style> tags in the header

<!DOCTYPE html>

<html>

<head>

<style>

p {

color: purple;

}

</style>

<title> Blabla </title>

</head>

Of course much better by calling CSS file in the head

## Others

Inputs – to ask info from users

Input Email <input type="email" placeholder="your email">

Submit button <input type="submit">

<input id="amount1" type="text" placeholder="enter an amount" />

<input id="deposit1" type="button" value="Deposit" />

**Select for dropdowns / Options are the itms in the dropdown**

<select name='station\_start' class='station' id='station\_start'>

<option>Please Select</option>

<option value="value1"> This is Value 1</option>

</select>

🡪 $(‘#station\_start).val() = ‘value1’ !! it looks at value= !!

Buttons: <button>Like</button>

**On forms and buttons**

**‘change’ when a form is changed**

Submitting forms: submit

Note: a form only has one submit button

var formSubmitEvent = document.getElementbyId(‘form\_event’);

forSubmitEvent.addEventListener(‘submit’, function(e) {

e.preventDefault();

})

select forms with type submit in jQuery: $(‘:submit’) good to link it with the form id: $(‘#form1 :submit’)

Amazing pseudo-selector to fnd the selected option in a dropdown: $(‘#form1 :selected’)

**Data attributes on Html**

V good to give data-id to the elements, eg <input type=’text’, data-id=’3’>

('.completed input').data('id')

!! Default behavior of a form is to refresh the page!!

Event.preventDefault to prevent refresh and keep adding stuff on the page

<https://developer.mozilla.org/en/docs/Web/HTML/Element/form>

# CSS – CSS – CSS – CSS – CSS – CSS

Open in Browser, open console alt+cmd+j, go to elements

See the nesting of all <>

Magnifying glass to see detail of a specific item on the page

*// You cannot save the code written here, only for testing*

<!DOCTYPE html>

<html>

<head>

<title>Blabla </title>

*// Make sure normalize.css is above our own style.css*

<link type="text/css" rel="stylesheet" href="/normalize.css>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

CSS comment: /\*I'm a comment\*/

Versus HTML comment <!—I’m a comment -->

## Custom selectors

See [http://flukeout.github.io/#](http://flukeout.github.io/)

### SPECIFICITY

Over ruling by number of points

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Important | Inline | ID | Class | Tag |
| 10,000 | 1,000 | 100 | 10 | 1 |

(funny thing being that 11 tags will override a class…)

### General rule:

div p -> apply to p that is “child” of a div tag

div > p -> apply to p nested directly inside of <div>; hence, in the body, good idea to put some <p> inside <div> so you can differentiate them from the <p> which are directly inside body (body > p versus all p

Same for <ul>, create a <p> within the <li> so you can grab the li > p

H1 + p - choose all p directly following after an h1 -> for siblings = same level of indentation

H1 ~ p : same as “++ but will get all following p, not just the first one

### CLASS to take elements of different types and give them the same styling.

Html: <p class=”red”> xxx </p>

CSS: .red { color: red; }

* apply to any type of elements (p, h1, h3, etc) and will make it red
* call it with a span tag: <p> this word is <span class=”red”> red </span> </p>

Item can have multiple classes: <div class=”box1 item1”>

Define in CSS: .box1 { } and .item1 { }

### ID to grab a specific item

Html: <p id="intro"> xxx </p>

CSS: #intro { font-weight: bold; }

ID will override Class

### Advanced

\* { -> apply rule to ALL elements in the page

h1 \* - all elements inside h1 ; ul.fancy \*

h1.red -> grabs all h1 with class red,

#intro.red grabs all ids intro with class red

p, h1, .red - select all p and h1 and all .red

div:empty selects all empty <div> elements.

:not(#fancy) selects all elements that do not have id="fancy".

div:not(:first-child) selects every div that is not a first child.

:not(.big, .medium) selects all elements that do not have class="big" or class="medium".

#### Select an element by one of its attributes

input[type="submit"] { color: white; background: red; }

will ensure I get the input box for submit, and not the one for email

#### First child, nth child , last child, siblings

p:first-child { color: red } OR p:nth-child(2) { font-weight: bold }

<div> <p> I’m first child </p>

<p> **I’m second child** </p>

</div>

:first-child will get all first-child elements

div p:first-child : all of them within a div

li:only-child : select li elements who are only childs of something, e.g ul li:only-child

li:last-child will select the last li element

:nth-last-child(A)

Selects the children from the bottom of the parent. This is like nth-child, but counting from the back!

:nth-last-child(2) selects all second-to-last child elements

#### First of type, Nth of type

span:first-of-type -> selects the first <span> in any element.

div:nth-of-type(2) selects the second instance of a div

.example:nth-of-type(odd) selects all odd instances of a the class .example

span:nth-of-type(6n+2) selects every 6th instance of a span, starting from (and including) the second instance.

p span:only-of-type selects a span within any p if it is the only span in there.

div:last-of-type selects the last <div> in every element.

p span:last-of-type selects the last <span> in every <p>.

### Pseudo-selectors,

If I hover over a div or paragraph

div:hover { color: #cc0000; font-weight: bold; text-decoration: none; }

More with Links:

a:link: An unvisited link.

a:visited: A visited link.

a:active when you click on it

## Responsive styling

To display well on types of screen

Replace width by max-width

Media query: only activate CSS when browser meets certain conditions

Eg for narrow browsers (phones), we make text smaller and easy to display

@media (max-width: 500px) {

h1 {

font-size: 36px;

}

li {

display: block;

padding: 5px;

}

}

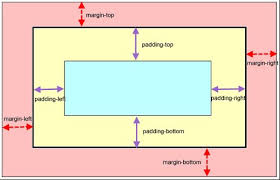
## List of attributes

### General positioning

### Default Displays

|  |  |  |
| --- | --- | --- |
| Block | Inline.Block | Inline |
| Takes 100% of width of its parent element  (= of the page if parent is body)  Can set width + height  Margin on 4 sides | Margin 4 sides  Width + height  Block elements can sit next to each other | Stays inline with flow of the page  Can set width + height  Can margin left/right  CANNOT margin top/bottom |
| Div  Section  Article  Ul / ol  H1… h5  Audio  Nav  Aside |  | **Img**  A  Button  input  Span  strong |

***!!! Img are INLINE and cannot be set width or margins!! need to be put display block***

****

Magin create space outside an elements; padding does it inside

**Margin top/right/bottom/left**

**0 Auto -> center text on the page**

**0 margin on top and bottom**

**Auto on left and right, meaning centered (stretching all the way L&R)**

**Postion**:

Default = static

Position Relative -> moving relative to where it was

Then left/right.top/bottom: add a margin -> left 2px will push to the right

Position Absolute: relative to its parent which has any position other than Static=Default

🡪 if position absolute, look up the parent thas has a specific Position

🡪 if parent has no position mentioned, then it will look at the grand parent

**🡪 useful to have the overall background in a div with pos absolute so that everything else will be positioned according to this one**

Position fixed:

### Text

color:red;

or color: #hexcode (hexadecimal color code)

rgba(0,0,0,0.5), 0-255 scale; last one controls transparency

font-family: Arial; (default value can be serif, sans-serif, cursive)

font-size: 12px; or font-size: 1em

font-weight: bold

text-align: left

font-weight: bold;

padding-top: 25px;

padding-bottom: 25px;

background-color: yellow;

### Links:

a {color: #cc0000, blue; text-decoration: underline, none; }

### Lists

ul { padding: 10px; } to offset padding to the left of the bullet point

1 value -> all sides

4 values: top right bottom left

li { display: inline; }

Inline = exists with natural flow of the text

Block = stretch whole width of the page with line breaks before and after

By default block: lists, headers, paragraphs

### Background image

Body { background: url("http://dash.ga.co/assets/anna-bg.png");

background-size: cover;

background-position: center;

color: white;

}

if doing background-image: url(‘fv’) 10px 10px no repeat

### Image

***!!! Img are INLINE and cannot be set width or margins!! need to be put display block***

height: 170px;

width: 170px;

border: 1px solid red

box-shadow: rgba(0,0,0,0.2) 10px 10px; or none, etc

display: block

margin: auto

*Img need to be put display bock if we want to use margin auto to center it on the page*

### Input

border: 0 to remove some of the default styling

### Tables

border-collapse: collapse

border-bottom: 1px solid black

border-left: 1px solid black;">

padding: 5px; (padding = indent I think)

Table image

td img {

height: 75px;

width: 75px;

box-shadow: rgba(0,0,0,0.2) 10px 10px; or none;

}

### Divs

background-color: #3c4543;

height: 100px

width: 100px

display: inline-block ; inline (not as blocks, can’t choose dimensions)

position: relative/fixed

top: -10px; (if relative)

border: 2px solid red

border-style: solid; dashed

border-width: 2px;border-top-left-radius: 15px;

border-radius: 5px

border-top-right-radius: 15px;

left:50%;

margin-left:-254px, auto

vertical-align: top/center/bottom

ul{

list-style-type: none;

position: fixed;

margin: -10px;

}

li {

display: inline;

border: 2px solid #000000;

border-radius: 5px 5px;

}

Call a style<span class="bold"></span >

.bold{

font-family: tahoma;

font-weight: bold;

font-size: 1.2em;

font-variant: small-caps;

color: #ffffff;