# 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

Or If using socket.io just put whatever port we’re using for the server

Then run ngrok http 9000

## 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;

}

}

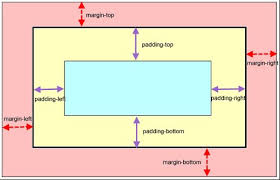
## List of attributes

### 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;

# JAVASCRIPT - JAVASCRIPT - JAVASCRIPT -

Listens for events: move mouse, hover, click, scroll, press key

Script tag just before the closing </body> <script> </script>

Close the tag or brower will think ALL the html is a javascript

$(element).on(event-type, thing-to-be-done);

$("button").on("click", function(){ alert("clicked!") } )

When I click on the button, the function alert will display “clicked!”

/\* Comment \*/ vs // single line comment

## Logics and Basics

### Operators, Variables

Modulus: reste de la division euclidienne: 15 % 4 // 3

NOT: ! OR : || AND: &&

&& true if everything is true

false && false -> false

or returns the first falsy value (either false, or null, NaN, etc)

null && ‘user’ -> null (is first falsy value)

|| true if one of them is true

or returns the first truthy value

null || ‘user’ -> ‘user’

! unitary operator, returns opposite of truthy/falseness

!!(stuff) -> original truthiness of stuff

=== fully equal in type and value; == looks at value but not type

5 === ‘5’ is false; 5==’5’ is true

!== is strict inequality, != looks at value not type

<http://dorey.github.io/JavaScript-Equality-Table/>

Recursive:

**Long-Hand Syntax Short-Hand Syntax**

**x = x + 1 x += 1**

**x = x - 5 x -= 5**

**x = x \* 2 x \*= 2**

**x = x / 10 x /= 10**

**x = x % 10 x %= 10**

**x = x + 1 x = x++**

null: created but not assigned a value

### If, then, ternary operator

Ternary operator: for IF, THEN, ELSE shortcut

expression ? value\_if\_truthy : value\_if\_falsy

(10>7) ? “bien sur” : “impossible” gives …. “Bien sur”

Syntax

if (x > 10) {

x += 10;

y += 10;

} else if (x > 5) {

x += 5;

} else {

whatever;

}

Or switch mode

switch (2 \* x) {

case 2:

y = 49;

break;

case 4:

y = 37;

break;

default:

y = 1;

}

Estimate switch… case = if/then … default = else (therefore not necessary)

### Loops: while & for

var x = 10;

while (x > 5) {

x -= 2;

}

Will return 4

for (initialization; condition; finalExpression) {

// A block of code.

}

var x = 10;

for (var i = 0; i < x; i += 1) {

console.log('HELLO');

}

Will write HELLO 10 times:

Starting with i = 0,

While i<x, execute instruction (write HELLO),

Then do i+=1

Break -> stop the loop. Same way as return works, see below

### Functions

Declaration:

function name(x;y;z) {

// Body;

}

Return -> stop the function now -> useful when having an if, so you stop the function if the condition is true

### On Numbers

Integers have the upper limit. Use Long or BigInt

NaN is still of type ‘number’ -> use isNan(x)

Math.pow, Math.sqrt, Math.random

Sort in ascending order:

var points = [40, 100, 1, 5, 25, 10];  
points.sort(function(a, b){return a-b});

The result of points will be:1,5,10,25,40,100

Use b-a to get descending order

Round 8.111111 to 3 decimals

NEW 8.111.toFixed(3)

var result=Math.round(8.111111\*1000)/1000 //returns 8.111

* multiply by 10^3 gets 8111.111, and Math.round of this gives 8111, which divided by 10^3 is what we want

### On text

String.toLowerCase() // str.toUpperCase()

Str.charAt returns character in a specific place

Substring character at a place, or between 2 places

Number(‘1’) -> convert string to a number

Split strings returns an array

‘this text to split’.split(‘ ‘) -> [this, text, to, split]

‘hey’.split(‘e’)[0] -> ‘h’ // ‘hey’.split(‘e’)[1] -> ‘y’

## Array, Object, Prototype

### Array

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array>

var myFriends = ['ellen', 'mary', 'doug', 'pat'];

myFriends[0] -> ‘ellen’

myFriends.length -> 4

To get the (nth) last item: myFriends[myFriends.length-1] -> ‘pat’

var arrayOfArrays = [['a', 'b', 'c'], ['d', 'e', 'f'], ['g', 'h', 'i']]; arrayOfArrays[0][0] = ‘a’

; arrayOfArrays[1][2]= ‘f

.indexOf(): searching in arrays

var animals = ['bear', 'beetle', 'boa'];

animals.indexOf('boa'); // Evaluates to 2 (index 2 = 3rd position)

Push and Pop: add and remove the last element

Animals.push(‘rat’) -> add ‘rat’ as 4th item (index 3)

Animals.pop() -> remove last item

Shift: take form first position

Unshift: put in the first position

Splice: remove from an array to put to a new one

friends.splice(1, 2): from position 1 (=2nd), remove 2 elements and put them in a new array

array,reverse reverse the array

Split: transform string into an array, splitting according to a rule

Var string = ‘I am a string’

String.split() -> [‘I am a string’]

String.split(‘ ‘) -> ["i", "am", "a", "string"]

But careful, string.split('a') -> ["i ", "m ", " string"]

**To reverse from Split,**

**Join**() method joins all elements of an **array** into a string.

Array.sort: sort the array

Map: change every element of array according to a formula;

function square(x) { return x \* x; }

var resultingArray = [1, 2, 3].map(square);

/\*\*\* Must use RETURN otherwise function will return undefined \*\*\*/

ForEach will execute a function, for each element in the array

It will NOT return a new array but just an undefined. Use forEach if you want to log stuff in console

For each is same as doing a for-loop on each element of the array

for (var i=0; i<friends.length; i++) {

console.log('Hello ' + friends[i]) }

=== friends.forEach(function(buddy)

{console.log('Hello' + buddy) })

var resultingArray = [1, 2, 3].forEach(square)

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach>

Every wil do foreach and stop the first a value is Falsy

Some the same and stops whem value is TRuthy

- var THRESHOLD = 12; var v = [5, 2, 16, 4, 3, 18, 20]; var res; res = v.every(function(element, index, array) { console.log('element:', element); if (element >= THRESHOLD) { return false; } return true; }); console.log('res:', res); // logs: // element: 5 // element: 2 // element: 16 // res: false

Filter

Loops through an array and create a new array keeping only elements matching the condition

Ex: let’s keep only items > 50

var numbers = [12, 5, 8, 130, 44];

var biggers = numbers.filter(function(item) {

return item > 50

});

-> result is array [130]

Reduce

Will perform an operation using , successively, the value of each element in the array

<https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce>

WARNING!! It will **always** use the first 4 parameters as below, they are native to reduce, this is how it works

[0, 1, 2, 3, 4].reduce(function(previousValue, currentValue, index, array) {

console.log(previousValue, currentValue, index, array);

return previousValue + currentValue;

});

0 1 1 [0, 1, 2, 3, 4] *// is the console.log being looped*

3 1 2 2 [0, 1, 2, 3, 4] *// is the console.log being looped*

3 3 3 [0, 1, 2, 3, 4] *// is the console.log being looped*

3 6 4 4 [0, 1, 2, 3, 4] *// is the console.log being looped*

RETURNS 10 (summed 1+2+3+4)

### Associative Arrays / Objects

Associate a key to each element of the array.

Create using {}, combination of key-value pairs

var lunches = { 'Josh' : 'pasta','Floyd' : 'salad','Matt' : 'tuna sandwich','Shannon' : 'soup' };

var candidateData = 'name' : "John Doe",'age' : 32,'isFullTime' : true, 'pastEmployers' : ['Microsoft','Google','Amazon'],yrsExperience' : {'ruby' : 3,'java' : 6,'javascript' : 5}}

Then access using normal array[] , or array.key

There is no indexing anymore, we access values based on the key:

candidateData[‘name’] === ‘John Doe’

Same as candidateData.name

candidateData['yearsExperience']['javascript'] === 5

And to add a new key-value pair there is no concept of number or order, just type in the new pair:

candidateData[‘Hobby’]=’Basket’

### Objects, Prototypes

-> Object is unique, repeating for each new house is long !!

var house1 = { sqfoot: 3000, bathrooms: 8, bedrooms: 10 }

var house2 = etc…

-> Create various instances of an object

function House(**sqfoot**, **bathrooms**, **bedrooms**) {

this.sqfoot = **sqfoot**;

this.bathrooms = **bathrooms**;

this.bedrooms = **bedrooms**;

}

To create 2 houses I simply do

var house1 = new House(3000, 8, 10); //*CONVENTION IS TO CAPITALISE*

var house2 = new House(1500, 1, 2);

var house3 = new House(2500, 4);

var house4 = new House(undefined, 1, 1);

house1 -> House {sqfoot: 3000, bathrooms: 8, bedrooms: 10}

house2 -> House {sqfoot: 1500, bathrooms: 1, bedrooms: 2}

house3 -> House {sqfoot: 2500, bathrooms: 4, bedrooms: undefined}

house4 -> House {sqfoot: undefined, bathrooms: 1, bedrooms:1 }

to populate the bedrooms in house3….

house3.bedrooms = 6

house1 is an INSTANCE of House

house1 instanceof String -> true

Bit different when one parameter of the object is a function

// CF below for faster way using Prototype //

var Person = function(firstName, lastName) {

OR

function Person(firstName, lastName) {

this.firstName = firstName;

this.lastName = lastName;

this.fullName = function() {

return ('Hello '+ firstName + ' ' + lastName);

}

}

var jeremy = new Person(‘Jeremy’, ‘Marer’)

jeremy.fullName**()** -> Jeremy Marer ***// fullName is a function so use ()***

jeremy is an instance of Person

jeremy instanceof Person -> true

**More efficient using Prototype**

**Also enables to add new key/value pairs to an object**

function Person(firstName, lastName) {

**this.firstName** = firstName;

**this**.**lastName** = lastName;

}

Person.prototype.fullName = function() {

return ('Hello '+ **this**.**firstName** + ' ' + **this.lastName**);

}

*// proto is a function! Don’t forget to RETURN something*

### Iterating through object keys and values

Var test = {“name”: \j};

Object.keys(test) list the keys of an object

For…in iterates through properties of an object

for (var prop in obj) {

do xxx

*// prop always refer to the property currently being examined*

*// obj always refer to the object*

*// obj[property]*

}

cf pre-made functions

## Debugger

*// Only works if the console is open !!*

Write debugger; under the line you want to stop at,

Then load the console in Chrome

Put break points on lines where you want the code to stop

Play will load code until next break

* Step over will run the next function and not go line by line
* Step into will send you *inside* the next function so you can test it
* step over will run the function line by line

Step out will send you out of the function, when you know the rest is good

## DOM Document Object Model

On html page one div has id='js-repo-pjax-container'.

Rigt-click ‘inspect element’ -> go to console elements list in the page tree

Use Javascript to assign this id to a var called element:

var element = document.getElementById('js-repo-pjax-container')

To change the text and background color inside the div

element.innerHTML = 'this has been changed by DOM manipulation'

element.style.backgroundColor = 'red';

**Create elements on a page, = in the DOCUMENT**

var element = document.createElement('div') *//declare a div*

var parent = document.getElementById('body'); *//create parent element, here we gave <body =id”body”>*

parent.appendChild(element); // put our div inside the body

OR declare a var for this child: var child = parent.appendChild(element);

**Find all elements with class=”students”**

**var students = document.getElementsByClassName('students');**

body.children list all children within body

body.childNodes similar with more info

**Use children to “navigate”through the tags.**

EG A form has an id, and then 2 inputs inside = 2 children

Text is the value of the first input form

var former = document.getElementById('former');

former.addEventListener('submit', function(e) {

alert(this.children[0].value);

Where am I (url): window.location.href

Find image with id=’diagram’ then change the source of the image

var element = document.getElementById('diagram')

element.setAttribute('src', `img/dom\_basic.png`)

### Local Storage and JSON objects

Set, Create, Remove key-value pairs

localStorage.setItem('myCat', 'Tom') 🡪 Storage {myCat: "Tom", length: 1}

localStorage.getItem('myCat') 🡪 "Tom"

localStorage.removeItem('name')

localStorage.clear() 🡪 RESET

Replace value by doing new setItem

***!! Only pass strings as values will be stored as Strings***  (except Booleans)

localStorage.setItem('myArray', [1,2,3]) 🡪 {myArray: "1,2,3", length: 1}

**Object will be messed up, hence we use JSON methods – stringify**

**JSON to Add objects to local values**

var bob = {name: 'Bob', age: '81'}

var stringBob = JSON.stringify(bob) 🡪 "{"name":"Bob","age":"81"}"

localStorage.setItem('coolPerson', stringBob)

🡪Storage {coolPerson: "{"name":"Bob","age":"81"}", myArray: "1,2,3", length: 2}

**JSON parse to get the Object back, and to get values from the object**

var stringBob = localStorage.getItem('coolPerson') 🡪 "{"name":"Bob","age":"81"}"

(normal, everything in localStorage is a string!)

JSON.parse(stringBob).age 🡪 "81"

On Arrays

Students 🡪 ["bob", "jo", "sally"]

stringStudents = JSON.stringify(students) 🡪 "["bob","jo","sally"]"

JSON.parse(stringStudents) 🡪 ["bob", "jo", "sally"]

JSON.parse(stringStudents)[0] 🡪 "bob"

## Javascript Events

Syntax: Element.addEventListener(event, function, useCapture)

Create event when we click on an element with id=’event-click’

var eventClickButton = document.getElementById('event-click');

eventClickButton.addEventListener('click', function() {

alert('Ive been clicked!');

});

Same if working on an object. Example change picture on mouse hover

var billMurray = document.getElementById(‘bill’);

**billMurray**.addEventListener(‘mouseover’, function(e) { *//when hovering over pic*

**this**.setAttribute(‘src’, ‘new url/300/200’);

**//***better* ***this****.src = new url*

})

var billMurray = document.getElementById(‘bill’);

**billMurray**.addEventListener(‘mouseout’, function(e) { *//when stop hovering*

**this**.setAttribute(‘src’, ‘new url/300/200’);

})

***//Note the (e) inside function; shortcut for “event”, it links back to this locally specific event for this specific object – eg above, refers to mouseout event on billMurray***

***// If calling an existing function inside an event, DON’T use ()***

thingy.addEventListener('click', oneFunction, false);

function oneFunction (fgfg) {fgh} ;

***// Can ONLY create events on one element at a time –***

***// getElementsByClassName or Tag returns an ARRAY, se select getELxxx[i]***

**To program multiple events, we need to use a for loop**. Say we have multiple radio buttons with class=’radio\_Event’ and want to uncheck them

Var radios = document.getElementsByClassName(‘radio-event’) *// all elements with this class*

For var (i=o; i<radios.length; i++) {

***// individually adding the events to each element: radios is an array***

Radios[i].addEventListener(‘change’, function(e) {

(for var k=0; k<radios.length; k++) {

radios[k].checked=false;

}

})

}

Event target to check what item is impacted by the event.

EG an <ul> with a <li> of vegetables. Wanna now what item in the list I clicked on

var vegetables = document.getElementById('vegetables');

vegetables.addEventListener('click', function(e) {

alert(event.target.textContent)

})

Functions to make Jscript asynchronous

setTimeout(function() {

alert(‘I will run after 3,500 milliseconds’);

}, 3500);

setInterval(function() {

alert(‘I will run after 1,000 milliseconds’);

}, 1000)

### Clear the timers: clearTimeout() and clearInterval()

### List of possible events

<https://developer.mozilla.org/en-US/docs/Web/Events>

‘click’, ‘dblclick’, ‘mouseup’, ‘mousedown’, ‘mouseover’, ‘mouseout’

‘onscroll’

‘focus’ when you click into an input field

‘blur’ when outside of a submit text box

**On forms and buttons**

**‘change’ when a form is changed**

Submitting forms: submit

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

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

e.preventDefault();

})

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

## jQuery

jQuery is a load of functions to make shortcuts on javascript

***Always returns an ARRAY – works like CSS seclectors***

Always put the script to jQuery before all other js files (still bottom of body)

Navigating through elements

$(‘thing’) = document.getElementsByTagName / ClassName / Id

$(‘tag’) - $(‘.class’) - $(‘div .class’) - $(‘#id’)

***Can only do a jQ function on a jQelement!!! (ie an array)***

var cities = $(‘li’) -> [‘barcelona’, ‘madrid’]

Console.log( cities[0].text() )

🡪 ERROR because citie[0] is NOT jQuery object, while .text() is

Console.log( $(cities[0].text() ) ) -> will work!

Find an element, even within another element

$('section').find('span')

Next element nested **at same level //** Prev returns the one before, **nested at same level**

$('ul').next() // $('ul').prev()

Navigating through “family”

sibling() .parent() .children()

***//mind the syntax and space on children***

$('p').first() -> fisrt paragraph; // $(‘p’).last()

$(‘section:first) -> first of type section

${‘section :first-child’) -> first-child of the parent, in this case section

$('ol :nth-child(2)') -> second child in the <ol>

$('li:nth-child(2)') -> ALL <li> which are a second child

If we omit the argument, .index() will return the position of the first element within the set of matched elements in relation to its siblings

**EACH to loop through elements**

$('li').each(function(index, element) {

/*/ index ->index of the element in the array*

*// element -> the jQuery element*

console.log(index, $(element).text() ) ;

***// .text() to convert to text – remember all $ elements are arrays***

})

Find all the classes, or attributes, of an object

$('.secret\_message').attr('class')

Actions

APPEND / PREPEND

inserts the specified content as the l**ast child**

<ul class="cities">

<li>Madrid</li>

<li>Barcelona</li>

</ul>

var newCity = 'Granada';

$('.cities').append('<li>' + newCity + '<li>')

Can also select element on page and append to another

$( ".container" ).append( $( "h2" ) );

List all text inside the <li> above

$(‘.cities li).each( function(index, element) {

console.log( $(element).text() )

***// only call a jQuery function on a jQuery element !!***

Remove element .remove()

Empty “inside”an element .empty()

$foo.empty() === $foo.children().remove()

Remove/add class

$('section').removeClass('no\_js') / $('.message').addClass('hidden')

t class on/off

$('.secret\_message').toggleClass('hidden') -> if not there, add it; if there, remove it

Change text: $('header > h1').text('My Page');

Change the html content : $('blockquote').html('<span>no quote</span>');

Change parameters… use CSS!

$('h2').css('text-decoration', 'underline');

Combine jQuery and Jscript!

$(this).attr('src', "http://url") === this.src="http:/url”);

Change text to uppercase in an input box

Calling Events and Listerners

var setUpEventListeners = function() {

$('#helloButton').on('click', showPortfolio);

}

**!!! DON’T PASS ARGUMENTS INTO FUNCTION WHEN DEFINING EVENTS !!!**

**NO!! $('.greeting').on('click', showGreeting($(this).attr('id')));**

**YES !! $('.greeting').on('click', function(){ showGreeting($(this).attr('id') );**

BEST PRACTICE

1. create a Var/function to set all event listeners
2. define the functions for each events separately, for clarity
3. wrap the function for events into the doc.ready event, to make sure the events are NOT run before the doc is loaded **// Cannot call docs from the DOM… until the DOM is loaded**!

$(document).ready(function(){

setUpEventListeners();

initialize();

})

var setUpEventListeners = function() {

$('#helloButton').on('click', showPortfolio);

$('.aClass').on('click', anotherFunction);

}

var showPortfolio = function() {

console.log(event);

alert('hello')

var anotherFunction () {}

}

var initialize = function() {

$('section').removeClass('no\_js');

$('.message').addClass('hidden');

}

# AJAX – AJAX – AJAX - AJAX

<http://api.jquery.com/jquery.ajax/>

Asynchronous javascript and xml

EG youtube: only load the comments when you scroll down past a point

API: Point is to receive a JSON object from another source and use it in our own code

All in JSON – good to use “double-quotes”

*// Open python server python -m SimpleHTTPServer 9000*

*// Create an AJAX request (AJAX built-in in Javascript)*

***This is the crude way to understand the logic, see at bottom MUCH FASTER WITH JQUERY***

Week5 day 4, rest\_countries

function ajaxRequest(method, url){

var request = new XMLHttpRequest();

request.open(method, url);

request.send();

return request;

}

*// we only want request status 4 and ok*

function successfulRequest (request){

return if (request.readyState === 4 && request.statusText == 'OK')

}

function getRegions(event) {

var regions = []; *// will be used for storage*

var request = ajaxRequest('GET', 'https://restcountries.eu/rest/v1/all');

*// Remember JS is synchronous but we need to wait until the request is fully processed in order to do something*

request.onreadystatechange = function() {

*// console.log(request); => Loads of various request objects, all looking similar (a hash) with long JSON objects*

if successfulRequest(request)) {

regionSelect.innerHTML = ''

*// regionSelect.innerHTML ='<option value="default">-- Select your region --</option' => was much quicker than creating the optionDefault below...*

var response = JSON.parse(request.response) *// is a long JSON string => parse it to get an Array: each array[i] lists infos on a country -> we need the region, and we store it in the regions array*

for (var i=0; i < response.length; i++) {

*// need to avoid duplicate regions*

if (regions.indexOf(response[i].region) === -1 && response[i].region.length > 0) {

regions.push(response[i].region);

*// now add it to the dropdowm menu: "select" made of "options"*

var optionRegion = document.createElement('option');

optionRegion.setAttribute('value', response[i].region);

optionRegion.innerHTML = response[i].region;

regionSelect.appendChild(optionRegion);

}

}

*// just add a "select region" to the dropdown*

var optionDefault = document.createElement('option');

optionDefault.setAttribute('value', 'default');

optionDefault.innerHTML = '-- Select your region --';

regionSelect.insertBefore(optionDefault)

} *// end if readyState == 4*

} // *end request.onreadystatechange*

} *// end getRegion*

var getRegionsButton;

var regionSelect;

document.addEventListener('DOMContentLoaded', function() {

console.log('heyyyy');

*// Define our variables*

getRegionsButton = document.getElementById('get-regions');

regionSelect = document.getElementById('region-select');

*// Add Event Listeners*

getRegionsButton.addEventListener('click', getRegions);

});

***AJAX IS MUCH FASTER WITH JQUERY***

Week5 day 4, rest\_countries

function getRegions(event) {

var regions = [];

// Create an AJAX request (AJAX shortcut in jQuery)

$.ajax({

type: 'GET',

url: 'https://restcountries.eu/rest/v1/all'

}).done(function(response) {

// .done is like the "onreadystatechange thing" + check successful request

// jQ already does JSON.parse on response so it is an objet => JQUERY OBJECTS ARE ARRAYS!!

$('#region-select').empty();

$('#region-select').append('<option value="default"> -- Select region -- </option>')

$.each(response, function(index, item) {

if ($.inArray(item.region, regions) == -1 && item.region.length >=1) {

regions.push(item.region);

$('#region-select').append('<option value="' + item.region + '">' + item.region + '</option>')

}

})

})

}

function getCountries(event) {

var regionName = $(this)[0].value // remember jQuery returns an Array !

$.ajax({

type: 'GET',

url: "https://restcountries.eu/rest/v1/region/"+ regionName

}).done(function(response) {

$('#country-select').empty();

$('#country-select').append('<option value="default"> -- Select country -- </option>')

$.each(response, function(index, item ) {

$('#country-select').append( '<option value="' + item.name + '">' + item.name + '</option>')

})

})

}

function getCountryData() {

var countryName = $(this)[0].value;

$.ajax({

type: 'GET',

url: 'https://restcountries.eu/rest/v1/name/' + countryName

}).done(function() {

$('#results').empty();

$('#results').append("<h4>" + countryName + "</h4>");

$('#results').append("<ul id='description'></ul>");

})

}

$(document).ready(function(){

console.log('heyyyy');

$('#get-regions').on('click', getRegions);

$('#region-select').on('change', getCountries);

$('#country-select').on('change', getCountryData)

})

# RUBY – RUBY – RUBY – RUBY – RUBY

Everything in Ruby is an OBJECT

Command line

Irb to test lines in the terminal (just like JS console)

ruby file.rd to run the file

Qui / exit / ctlr+d

Ruby –v for current version

Pry

Can’t execute script inside pry though - quit

## Logics and Basics

### Data types

Sth.class -> tell me what it is

Convert data types: '5'.to\_i => string to integer

(1..3).to\_a => [1,2,3]

[1,2,3].map { |i| i.to\_s } => [‘1’,’2’,’3’]

!! ‘hello’.to\_i -> 0 !

'4.56fg'.to\_f -> 4.56

is\_a? to compare data types (Return true / false)

b.is\_a? (String) / same as .kind\_of? String

Equality: == // !=

Boolean and logical operators

Similar to JS

Nil && true -> nil

5 || 2 -> 5 (stops at first true) ; 5 && 2 -> 2 (checks all conditions so runs the last one too)

### Number

2\*\*3 : 2 power of 3 ; % rest in Euclidian division

7/2 -> 2 ! Dividing a fixed number by a fixed number returns a fixed number

way around simply write one float

7.odd? => true

Number.round(n)

Round UP : .ceil / round dowm .floor

(self.to\_f).round

Also found very weird (self.to\_f +0.5 ).to\_i

=> ensure self is a float, and adding 0.5 will ensure the corresponding integer is right

### Strings

Print : display (like console.log)

**Puts** : same & adds a new line underneath

Get input from user - always returns a string!

gets (get string)

gets.chomp to avoid recording the “enter key” at the end, cause it adds a line

.downcase / .upcase / .capitalize (first letter only) / .swapcase (swap for every letter)

multiply strings: ‘gui’\*3 -> guiguigui

text.include? 'string’

.start\_with? / .end\_with? / String.index(‘letter’)

‘string’.index(‘t’) -> 1

‘string’.chars[1] -> ‘t’  *!! inside [ ]*

Slice strings (and arrays)

String[1,3] : from index, return array with 3 elements

**Change text**

.split(‘where’)

Concat: +, or << (append to existing word), or word1.concat(word2)

“blab li blu”.sub(‘bla’, ‘boum’) // or gsub for all occurrences of bla

delete all capital letters => string.delete "/[A-Z]/"

Partition to separate text

Searches sep or pattern (regexp) in the string and returns the part before it, the match, and the part after it. If it is not found, returns two empty strings and str.

"hello".partition("l") #=> ["he", "l", "lo"]

"hello".partition("x") #=> ["hello", "", ""]

"hello".partition(/.l/) #=> ["h", "el", "lo"]

**Regular expressions: match text by its patter**

“replace All text in Capitals”.gsub(/[A-Z]/, '0')

Regular expressions are put between /…/

[A-Z] is pattern for capital letters

“Find character next to a whitespace”.match(/ ./, index to start search) -> I

.match + regEx: is this word following the pattern?

array.keep\_if { |i| /[aeiou]/.match(i.chars[0]) }

**Sring interpolation – must have double-quotes**

**puts "the number you want is #{number}. How cool"**

Below are some of the more common escape sequences that can appear inside of double quotes.

* \" – double quote
* \\ – single backslash
* \a – [bell/alert](http://en.wikipedia.org/wiki/bell_character" \o "w:bell character" \t "_blank)
* \b – [backspace](http://en.wikipedia.org/wiki/backspace" \o "w:backspace" \t "_blank)
* \r – [carriage return](http://en.wikipedia.org/wiki/carriage_return" \o "w:carriage return" \t "_blank)
* \n – [newline](http://en.wikipedia.org/wiki/newline" \o "w:newline" \t "_blank)
* \s – [space](http://en.wikipedia.org/wiki/space_character" \o "w:space character" \t "_blank)
* \t – [tab](http://en.wikipedia.org/wiki/tab_key" \o "w:tab key" \t "_blank)

**puts** "Hello\t\tworld"

**puts** "Hello\b\bGoodbye world"

**puts** "Hello\rStart over world"

**puts** "1. Hello\n2. World"

The result:

$ **double-quotes.rb**

Hello world

HeGoodbye world

Start over world

1. Hello 2. World

Comment: #

**Dates**

Convert to 31/10/2013 format

date.strftime "%d/%m/%Y"

**Range**

(1..20).max = (1..20).last = 20

(1…20).max = 19 but last = 20

**Access a file: IO (input / ouput)**

# count the number of words in a file

IO.read(file\_path).split.length

### LOOPS

Ctrl+c to exit

**If then else**

print 'Enter a number'

number = gets.to\_i

if number > 0

puts 'you entered a positive number'

elsif number < 0

puts 'you entered a negative number'

else

puts 'you entered zero'

end

# don’t forget the end statement!

**Unless**, just like if !xx

user\_registered = false

puts 'Please sign up to continue' unless user\_registered

Or 1 line:

if true then puts 'Hello!' else puts 'Bye' end

**Case statement**

puts 'Exit the program? (yes or no): '

answer = gets.chomp.downcase

case answer

when 'yes'

puts 'Goodbye!'

when 'no'

puts 'Ok. Continue!'

else

puts 'Not sure what you mean by that'

end

**While / until**

number =1

while number < 11

puts number

number = number + 1

end

//

until number >10

puts number

number = number + 1

end

**For loops are NOT recommended**

for i in 1..10 !!! .. up to and including last one /// … up to and not including last one

break if i > 5

puts i

end

**100.times do / up to / down to**

**end**

10.times do |k| puts "Number #{k+1}"   
# times will start at 0, so on the 10th iteration, k is equal to 9 end

1.upto(10){ |k| puts "Number #{k}"}

(1..100).each do |k|

puts "#{k}. This is Ruby preferred way of doing loops, when possible"

end

(1..100).each{ |k| puts "#{k}. Curly braces make it even shorter"}

## Functions / Methods

String.methods => all methods linked to strings

.sort.inspect to list them in a condensed look

def say\_hello

puts 'Hello World'

end

*# no need for () to call it*

def add(number1, number2)

“ the result is #{number1 + number2}”

end

***## by defaut RB will “return” the last line of the function***

Unlike JS, methods MUST be declared before being called

***!! Scoping in Ruby !!***

***All variables must be declared INSIDE the method. It won’t understand variables declared outside***

🡪 Pass all variables you’ll need as arguments of the function

x=0

def doStuff(x)

foo = x

puts foo

end

doStuff(x)

Use default value if no argument are passed when calling methods

def hello\_students(students=’james’)

puts students

end

🡪 hello\_student(‘lauren’) returns ‘lauren’

🡪 hello\_student() returns ‘james’

### Blocks

10.times { puts ‘heyyy’ }

10.times do

|n|

*#more than one line*

end

Shorthand for a block

array.reject(&:!)

same as array.reject { |e| e==alse } since e==false same as !e same as !

Other example: array.map {|i| i.reverse } = array.map(&:reverse)

array.map(&:reverse)

**You can always add a block to a method**

* **use YIELD to declare the elements on which to pass the block on**
* call the function using the block

def **doge**(word1, word2)

phrases = [**'**wow**'**, **"**much #{word1}**"**, **"**so #{word2}**"** ]

for phrase in phrases

yield phrase

end

end

doge(**'**ruby**'**, **'**syntax**'**) {|str| puts str.upcase }

* WOW
* MUCH RUBY
* SO SYNTAX

You get an error if you don’t’ pass a block

block\_given? to check if a block was passed on

way around:

for phrase in phrases

yield phrase if block\_given?

end

### Enumerables

Each is a method on the enumerable module

**%w(**jeremy lauren mathilda lexie**)**.each { |name| print\_name(name) }

* jeremy lauren mathilda lexie

On a hash by default each uses key-value pair

hash.each {|key, value| puts **"**#{key} score was #{value}**"** }

On hash can use each\_with\_index to play with the index:

**%w(**jeremy mathilda lauren**)**.each\_with\_index do |name, index|

[28] pry(main)\* **#code**

[28] pry(main)\* end

See more in arrays/hashes

.detect => finds 1st item to match condition

.select / .find\_all

.reject /.drop

.keep\_if / .delete\_if

**.reduce / .inject takes |sum, n| as arguments**

[**1**, **2**, **3**, **4**, **5**, **6**, **8**, **9**, **10**].reduce { |sum,n| sum + n }

* sum=1, n=2 ; next sum = 3, n=3; etc

Short-form: (**1**..**10**).inject(:+)

Can also provide an initial value, put it under bracket: .inject(set init value)

EG: counting the occurrence of each word in a sentence

* it will be stored in a new hash, which has default value zero
* first round: {}
* second round grabs first word so hash = {‘word’ => 1 }

long\_sentence.split(**'** **'**).inject(**Hash**.new(**0**)) do |hash, word|

hash[word.downcase] += **1**

hash

end

### Splat Operator – methods with variable list of parameters

def introduction(age, gender, \*names)

"Meet #{names.join(' ')}, who's #{age} and #{gender}"

end

**def** add(\*numbers)

numbers.inject(0) { |sum, number| sum + number }

**end**

**def** add\_with\_message(message, \*numbers)

"#{message} : #{add(\*numbers)}"

**end**

puts add\_with\_message("The Sum is", 1, 2, 3)

### Passing Blocks {} as parameters – options

def release\_if (&block)

output = @bikes.select &block // gonna pass our block in the .select method

output.each { |bike| @bikes.delete(bike) }

end

docking\_station.release\_if ( { :broken? } )

* broken? is a method on class Bike -> bike.broken? returns true or false
* if I run release\_if { :broken? }
  + bike.select &block becomes bike.select(&:broken?)
* if I run release\_if{ : \*\*\*\*\* need help for syntax to do releas\_if bike.broken? == false

***& says ‘hey I’m gonna pass an argument to you’***

def add(a\_number, another\_number, options = {})

sum = a\_number + another\_number

sum = sum.abs if options[:absolute]

sum = sum.round(options[:precision]) if options[:round]

sum

end

puts add(1.0134, -5.568) -> -4.5546

puts add(1.0134, -5.568, absolute: true) -> 4.5546

puts add(1.0134, -5.568, absolute: true, round: true, precision: 2) -> 4.55

You can't use both splatted arguments and last-parameter-is-a-hash at the same time through Ruby, so you'll have to work on the arguments inside of calculate. There's no neat way to do this - you have to check if the last argument to calculate is a Hash, then remove it from the list before calling add or subtract.

def add(\*numbers)

numbers.inject(0) { |sum, number| sum + number }

end

def subtract(\*numbers)

current\_result = numbers.shift

numbers.inject(current\_result) { |current\_result, number| current\_result - number }

end

def calculate(\*arguments)

# if the last argument is a Hash, extract it

# otherwise create an empty Hash

options = arguments[-1].is\_a?(Hash) ? arguments.pop : {}

options[:add] = true if options.empty?

return add(\*arguments) if options[:add]

return subtract(\*arguments) if options[:subtract]

end

## Arrays & Hashes

Array to a Hash

Hash[\*array] (same as hash = Hash.new

# some say it can crash on loads of data so use => Hash[array.each\_slice(2).to\_a]

If array = [**'**a**'**, **'**b**'**, **'**c**'**, **'**d**'**]

array.each\_slice(**2**).to\_a

* [[**"**a**"**, **"**b**"**], [**"**c**"**, **"**d**"**]]

### Arrays

<http://ruby-doc.org/core-2.1.5/Array.html>

arr.methods

**Create array**

my\_array = [1,2,3,4,5,6]

.first / .last /

go [-1] to get the last one

(1..10).to\_a => [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

%w(monday tuesday wednesday) => ["monday", "tuesday", "wednesday"]

Array << item // array.push(item) #at the end // Array.unshift(item) #at the beginning

arr.unshift(arr.pop) -> take the last one and put in first position

arr << 9 << 7 << 6 : insert multiple

arr.insert(3, 'orange', 'pear', 'grapefruit') : at index 3, add all the following

**Delete**

.pop : removes last and returns it (does not alter the original array)

.shift : removes first and returns it (does not alter the original array)

arr.delete(2) : delete element 2

arr.delete\_at(2) : delete on index 2

[1,2,3,4,5,6,7,8,9].delete\_if{ |i| i%2 == 0 } : delete IF following condition is met

**Look for elements**

Array.count(item) : how many times in item present

array.include? 'item'

array.emptty? => true or false

Index

a = [ "a", "b", "c" ]

a.index("b") #=> 1

Intersection with &

Set Intersection — Returns a new array containing elements common to the two arrays, excluding any duplicates. The order is preserved from the original array

**Finding elements, according to criteria (length, being odd, etc**

.max, .min

item in array with max number of words: array.max\_by(&:length) // or .min

.sort /.sort!

sort by last element array.sort\_by {|e| e[-1]}

**Operations on arrays**

.sample : returns random element

.suffle : shuffles everything

.reverse : does not alter the original array, so need to store it into another variable

.reverse! : WILL alter the original array

.uniq : remove duplicates ( and .uniq!)

.flatten array if there is an array inside / .flatten!

.compact to remove the nils / .compact!

**Iterating thourgh arrays**

Array.each { |n| do something on n }

.each\_with\_index { |item, index| }

.reverse\_each : iterate in reverse order

.map will return a new array

.map! will modify the array

.map.with\_index

Non-destructive selection : select if matches criteria; reject removes those who don’t match

arr = [1, 2, 3, 4, 5, 6]

arr.select { |a| a > 3 } #=> [4, 5, 6] // same as .find\_all

arr.reject { |a| a < 3 } #=> [3, 4, 5, 6]

arr.drop\_while { |a| a < 4 } #=> [4, 5, 6]

arr #=> [1, 2, 3, 4, 5, 6]

.drop(n) : Drops first n elements from ary and returns the resulting array

.take Returns first n elements from the array.

Take\_while

a = [1, 2, 3, 4, 5, 0]

a.take\_while { |i| i < 3 } #=> [1, 2]

.delete\_if and .keep\_if

[1,2,3,4,5,6,7,8,9].delete\_if{ |i| i.odd? } : delete IF following condition is met

**.reduce / .inject takes |sum, n| as arguments**

[**1**, **2**, **3**, **4**, **5**, **6**, **8**, **9**, **10**].reduce { |sum,n| sum + n }

* sum=1, n=2 ; next sum = 3, n=3; etc

Short-form: (**1**..**10**).inject(:+)

Can also provide an initial value, put it under bracket: .inject(set init value)

EG: counting the occurrence of each word in a sentence

word.split(**'** **'**).inject(**Hash**.new(**0**)) do |hash, word|

[32] pry(main)\* hash[word.downcase] += **1**

[32] pry(main)\* hash

[32] pry(main)\* end

Combinations: yields all combinations of length n of elements from the array and then returns the array itself.

a = [1, 2, 3, 4]

a.combination(1).to\_a #=> [[1],[2],[3],[4]]

a.combination(2).to\_a #=> [[1,2],[1,3],[1,4],[2,3],[2,4],[3,4]]

### Hashes

<http://ruby-doc.org/core-2.1.5/Hash.html>

**Create**

NEW SYNTAX:

Capital = {england: **'**London**'**, france: **'**Paris**'**}

Access: apital[:england] => ‘London’

Add/modify: capital[:italy] = **"**Rome**"**

Or capital = **Hash**.new(**'**This country is not listed**'**)

* Default value if you access a key that doesn’t exist – will return msg instead of neg
* Eg capital[:usa] => ‘this country is not listed’

Chuck = { “punch” => 99, “kick”=> 98, etc}

chuck = Hash[:punch, 99, :kick, 98, :stops\_bullets\_with\_hands, true]

my\_friends = Hash.new

my\_friends[:sophie] = 'likes cinema'

**Frequent confusion to access HASH and CLASSES**

**Hash with hash[thing] // class with class.thing**

*Add a sub-hash to all keys in the hash*

students.keys.each {|i| students[i][:test\_subhash] = [**'**test**'**, **'**array**'**]}

*:Symbol vs “String”*

Symbol cannot be changed. Use it when you want to prevent the element to be changed (it would return an error)

Symbol is stored in memory while string created each time -> symbol more memory efficient

List all symbols currently stored in memory: Symbol.all\_symbols

#### Playing with key-value pairs

options.fetch(:capacity, DEFAULT\_CAPACITY)

Hash.fetch method retrieves the value for the given key ( options[:capacity] in this case) and if the key is not found, it returns the second argument ( DEFAULT\_CAPACITY ). So if the capacity is passed, it's used, otherwise the default one will be assigned.

* works same as options[:capacity] || DEFAULT\_CAPACITY
* use CAP LETTERS to signify constant variable

h = { "a" => 100, "b" => 200, "c" => 300, "d" => 300 }

h.key(200) => "b"

or .assoc

h.assoc("letters") #=> ["letters", ["a", "b", "c"]]

h.assoc("foo") #=> nil

.invert => Returns a new hash created by using *hsh*’s values as keys, and the keys as values

Flatten hash to put all in array – needs (n) to tell how recursive you want it (flatten array is recursive, not hash)

a = {1=> "one", 2 => [2,"two"], 3 => "three"}

a.flatten # => [1, "one", 2, [2, "two"], 3, "three"]

a.flatten(2) # => [1, "one", 2, 2, "two", 3, "three"]

**Delete by accessing the key**

{:england=>"Liverpool", :france=>"Paris", :australia=>"Canberra"}

capital.delete(:england)

{}.each do |key, value|

xxx

end

{}.keys / {}.values

## Object Oriented Programming

file:/Users/Guiton/Desktop/WDI\_13/classwork/week\_4\_ruby/

* obj\_orient\_prog/person.rb has lesson and explanations
* /oop\_rental\_appINCLASS has exo and homework correction

**Don’t forget to list the attribute accessors or nothing will work!!**

Use “options” to simplify input of long arguments

class Building

def initialize(options = {})

# options is a hash which will hold the attributes - no need to worry about the order

@floor = options[:floor]

@address = options[:address]

@apartments = options[:apartments]

@age = options[:age]

@concierge = options[:concierge]

end

end

If using DEFAULT\_CAPACITY, put it right under the class name

class DockingStation

DEFAULT\_CAPACITY = 10

def initialize(options={})

@capacity = options.fetch(:capacity, DEFAULT\_CAPACITY)

@bikes = []

end

To override the msg coming out when typing “puts” on an object

Add method to the Class, then run it on an instance

def to\_s

"My name is #{@name} and I study #{@course}"

end

=> dave = Student.new(‘Dave’, ‘wdi’)

=> puts dave => my name is Dave and I study wdi

**List ALL objects of a class**

def self.all

ObjectSpace.each\_object(self).to\_a

end

# Debugging and Test Driven Development

## Debugging

### Pry for Ruby

Require ‘pry’ at beginning of code

Potentially require ‘pry-byebug’

If using a main.rb to put require ‘fry’

* require\_relative ‘other files.rb’

Put binding.pry on a line to act like a debugger

Need code after binding.pry, just add nil if end of page

Ls to list all variables I have access to

Can look at what they are, play with them, etc

‘step’ to the next point

## Test Driven Development

### RSPEC

Better: :/Users/Guiton/Desktop/WDI\_13/homework/week\_4\_ruby/boris\_bike

Test files must be named xxx\_spec.rb

Good to write code in lib folder, and test in spec folder

Ex: write code on **greeter**.rb

Create **bike**\_rspec.rb

require\_relative ‘**bike’** // path relative to where the file is

I will focus on a CLASS called **Bike**:

describe ‘**Bike’** do

let(:bike) {Bike.new} => whenever I call ‘bike’ it will refer to this bike.new

it 'should not be broken when created' do

expect(bike.broken?).to be false

end

end

if using xit (Xit), the test will not run and will show as pending

Useful to create a spec\_helper.rb file

* put require\_relative ‘spec\_helper’ at top of each test file
* then in spec\_helper write all the files required for every test file
  + require ‘pry’, require\_relative ‘bike’, require\_relative ‘van’, etc

If all tests are on the same directory (spec) => run rspec

run full test on one file => spec greeter\_spec.rb

or run test on line 12 => spec greeter\_spec.rb:12

**Install**

If you want to use rspec-expectations with rspec, just install the rspec gem and RubyGems will also install rspec-expectations for you (along with rspec-core and rspec-mocks):

gem install rspec

Want to run against the master branch? You'll need to include the dependent RSpec repos as well. Add the following to your Gemfile:

%w[rspec-core rspec-expectations rspec-mocks rspec-support].each do |lib| gem lib, :git => "git://github.com/rspec/#{lib}.git", :branch => 'master' end

If you want to use rspec-expectations with another tool, like Test::Unit, Minitest, or Cucumber, you can install it directly:

gem install rspec-expectations

**Basic usage**

Here's an example using rspec-core:

RSpec.describe Order do it "sums the prices of the items in its line items" do order = Order.new order.add\_entry(LineItem.new(:item => Item.new( :price => Money.new(1.11, :USD) ))) order.add\_entry(LineItem.new(:item => Item.new( :price => Money.new(2.22, :USD), :quantity => 2 ))) expect(order.total).to eq(Money.new(5.55, :USD)) end end

The describe and it methods come from rspec-core. The Order, LineItem, Item and Money classes would be from *your* code. The last line of the example expresses an expected outcome. If order.total == Money.new(5.55, :USD), then the example passes. If not, it fails with a message like:

expected: #<Money @value=5.55 @currency=:USD> got: #<Money @value=1.11 @currency=:USD>

**Built-in matchers**

Equivalence

expect(actual).to eq(expected) # passes if actual == expected expect(actual).to eql(expected) # passes if actual.eql?(expected) expect(actual).not\_to eql(not\_expected) # passes if not(actual.eql?(expected))

Note: The new expect syntax no longer supports the == matcher.

Identity

expect(actual).to be(expected) # passes if actual.equal?(expected) expect(actual).to equal(expected) # passes if actual.equal?(expected)

Comparisons

expect(actual).to be > expected expect(actual).to be >= expected expect(actual).to be <= expected expect(actual).to be < expected expect(actual).to be\_within(delta).of(expected)

Regular expressions

expect(actual).to match(/expression/)

Note: The new expect syntax no longer supports the =~ matcher.

Types/classes

expect(actual).to be\_an\_instance\_of(expected) # passes if actual.class == expected expect(actual).to be\_a(expected) # passes if actual.is\_a?(expected) expect(actual).to be\_an(expected) # an alias for be\_a expect(actual).to be\_a\_kind\_of(expected) # another alias

Truthiness

expect(actual).to be\_truthy # passes if actual is truthy (not nil or false) expect(actual).to be true # passes if actual == true expect(actual).to be\_falsy # passes if actual is falsy (nil or false) expect(actual).to be false # passes if actual == false expect(actual).to be\_nil # passes if actual is nil expect(actual).to\_not be\_nil # passes if actual is not nil

Expecting errors

expect { ... }.to raise\_error expect { ... }.to raise\_error(ErrorClass) expect { ... }.to raise\_error("message") expect { ... }.to raise\_error(ErrorClass, "message")

Expecting throws

expect { ... }.to throw\_symbol expect { ... }.to throw\_symbol(:symbol) expect { ... }.to throw\_symbol(:symbol, 'value')

Yielding

expect { |b| 5.tap(&b) }.to yield\_control # passes regardless of yielded args expect { |b| yield\_if\_true(true, &b) }.to yield\_with\_no\_args # passes only if no args are yielded expect { |b| 5.tap(&b) }.to yield\_with\_args(5) expect { |b| 5.tap(&b) }.to yield\_with\_args(Fixnum) expect { |b| "a string".tap(&b) }.to yield\_with\_args(/str/) expect { |b| [1, 2, 3].each(&b) }.to yield\_successive\_args(1, 2, 3) expect { |b| { :a => 1, :b => 2 }.each(&b) }.to yield\_successive\_args([:a, 1], [:b, 2])

Predicate matchers

expect(actual).to be\_xxx # passes if actual.xxx? expect(actual).to have\_xxx(:arg) # passes if actual.has\_xxx?(:arg)

Ranges (Ruby >= 1.9 only)

expect(1..10).to cover(3)

Collection membership

expect(actual).to include(expected) expect(actual).to start\_with(expected) expect(actual).to end\_with(expected) expect(actual).to contain\_exactly(individual, items) # ...which is the same as: expect(actual).to match\_array(expected\_array)

Examples

expect([1, 2, 3]).to include(1) expect([1, 2, 3]).to include(1, 2) expect([1, 2, 3]).to start\_with(1) expect([1, 2, 3]).to start\_with(1, 2) expect([1, 2, 3]).to end\_with(3) expect([1, 2, 3]).to end\_with(2, 3) expect({:a => 'b'}).to include(:a => 'b') expect("this string").to include("is str") expect("this string").to start\_with("this") expect("this string").to end\_with("ring") expect([1, 2, 3]).to contain\_exactly(2, 3, 1) expect([1, 2, 3]).to match\_array([3, 2, 1])

**should syntax**

In addition to the expect syntax, rspec-expectations continues to support the should syntax:

actual.should eq expected actual.should be > 3 [1, 2, 3].should\_not include 4

See [detailed information on the should syntax and its usage.](https://github.com/rspec/rspec-expectations/blob/master/Should.md)

**Compound Matcher Expressions**

You can also create compound matcher expressions using and or or:

expect(alphabet).to start\_with("a").and end\_with("z") expect(stoplight.color).to eq("red").or eq("green").or eq("yellow")

**Composing Matchers**

Many of the built-in matchers are designed to take matchers as arguments, to allow you to flexibly specify only the essential aspects of an object or data structure. In addition, all of the built-in matchers have one or more aliases that provide better phrasing for when they are used as arguments to another matcher.

Examples

expect { k += 1.05 }.to change { k }.by( a\_value\_within(0.1).of(1.0) ) expect { s = "barn" }.to change { s } .from( a\_string\_matching(/foo/) ) .to( a\_string\_matching(/bar/) ) expect(["barn", 2.45]).to contain\_exactly( a\_value\_within(0.1).of(2.5), a\_string\_starting\_with("bar") ) expect(["barn", "food", 2.45]).to end\_with( a\_string\_matching("foo"), a\_value > 2 ) expect(["barn", 2.45]).to include( a\_string\_starting\_with("bar") ) expect(:a => "food", :b => "good").to include(:a => a\_string\_matching(/foo/)) hash = { :a => { :b => ["foo", 5], :c => { :d => 2.05 } } } expect(hash).to match( :a => { :b => a\_collection\_containing\_exactly( a\_string\_starting\_with("f"), an\_instance\_of(Fixnum) ), :c => { :d => (a\_value < 3) } } ) expect { |probe| [1, 2, 3].each(&probe) }.to yield\_successive\_args( a\_value < 2, 2, a\_value > 2 )

Usage outside rspec-core

You always need to load rspec/expectations even if you only want to use one part of the library:

require 'rspec/expectations'

Then simply include RSpec::Matchers in any class:

class MyClass include RSpec::Matchers def do\_something(arg) expect(arg).to be > 0 # do other stuff end end

Also see

* <http://github.com/rspec/rspec>
* <http://github.com/rspec/rspec-core>
* <http://github.com/rspec/rspec-mocks>
* <http://github.com/rspec/rspec-collection_matchers>

# Blog posts

MVP vs minimum scalable product

Experience from JS tictactoe project

Build what they ask you to do exactly. Or build so that you can scale later without having to re-do all the code.

Save time now or spend time later, waste time now because the scalable feature will never be used