DCI - Webdevelopment course

**09.12.2109 - introduction**

we will use terminal in these two weeks and github

blending learning→ there is online material and the teacher will share it in the class, in order to practice etc… freecodecamp

<https://www.freecodecamp.org/>

**TESTS:**

after javascrip there will be tests. it is more to check at which point we are and where we need help

---

→ command line is the only way to communicate with the server

→ we will use visual studio (visual code), it is an application from microsoft

<https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>

------------------------------------------------------------------------------------------------------------------------

**01.12.2019**

**TERMINAL**

→ **cd** → change directory

→ **.** current folder

→ **..** level up directory

→ **pwd** (where I am) → pass working directory

→ **mkdir** → make directory → create directory

→ **ls →** short list→ to show the directories (menu)

→ **clear**

→ **ls -l** → it gives some information about the directories → will show bunch of details →

→ **ls -a** → to show the hidden folders

→ **ls -R** shows all directories inside the directories

→ **--help →** shows information about the directories

→ **super user do** → super user will do whatever you want it to do (sudo)

→ **info** (+ls or + pwd, etc…) → it will show the info

→ files starting with a dot are files that are hidden

→ there is no difference between info and help (just different people who made it)

→ **man** → will also show the info (how to get out of manual→ with q)

v → **ll** → will show information (its longetŕ list than the short list or the ls -l)

→ **nano** will open up an editor to write in the file (text editor for terminal)

→ **less** → showing the editor pad (nano) but no option to modify → for files you don’t use cd you just open the file with **less**

**→ mv** to move files and directories

→ **rm** to remove files and directories → **rm -i** (you get asked again if you really want to remove it, otherwise it is removed forever and ever → **rmdir** to delete a directory

→ **../..** 2 levels up

→ to create a new text→ go to the right directory → type nano and name of the new file→ write something and save with Strg+X and Y→ to open it type less and the file name

rules for filesnames

→ no symbols

→ no capitals

→ no spaces!!

**if you want to create a file you have to determine the extensions:**

**txt**

**.html**

**.css**

**etc…**

→ to create a file in the terminal → **touch** → file name.extension

→ write nano and the name of the file → the editor will open→ to save ctr+X and to save just y and enter

→ if the directory name is too long then start the title and type tab and the rest of the name will automatically appear

→ ls -a serves to show hidden directories, it is important to have hidden directories when working with github etc…

→ you can use the arrows to see the commands recently used, but only the ones that were executed

**github.com/hnsreeny/terminal** (there is a list of commands)

**→ nano --help** will show all the details of the files and directories

same with **touch --help**

mkdir --help

ls --help

→ you can use it with any command to see the information

→ super user do → super user will do whatever you want it to do (**sudo**)

→ sudo apt-get install npm (for ex)

→ how to get out of wherever you are

→ if you are inside of nano is Strg

→ otherwise Strg+C

→ to download an application → **sudo apt-get install npm** (or whatever program)

there is a library (at) for applications ( a folder with react, npm, everything we need)

so the sudo is connecting to the folder and downloading from there

→ **sudo apt-get update** (it would update all the applications in the directory) (apt is referring to the applications in the local machine) → you always have to use sudo

all again explained here: https://help.ubuntu.com/community/UsingTheTerminal

→ how to move a file or directory → mv file name ../ directory name/

→ example mv thisisatext.txt ../gabriela/

→ the file thisisatext.txt was moved to the directory called gabriela from music

→ to go 2 levels up ../..

**Check for all the keys in the computer:**

<https://www.computerhope.com/keys.htm>

→ single quote and back-tick are different, don’t mix them up

Terminal

<https://github.com/hnsreeny/terminal>

------------------------------------------------------------------------------------------------------------------------

**11.12.2019**

**markdown language** → base language of html

its basic and easy to use, we will use it to upload applicatons to github→ they will always read it when checking the applications from the HHRR team

# is the way of making H1, ## for H2 and so on and so forth

*\_this is for italic\_*

**\*\*this is bold\*\***

zto do lists: star and space →

\* item 1

\* item 2

\* item 3

unordered lists:

1- item 1

2- item 2

3- item 3

---

visual studio code

→ download markdown preview enhancer

→ shift+cntr+V to open the preview mode

in order to write code in the markdown:

```

```

→ everything insed those will be understood as code

<https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf>

<https://github.com/hnsreeny/markdown>

<https://www.webfx.com/tools/emoji-cheat-sheet/>

<http://linuxcommand.org/>

<https://slack-files.com/files-pri-safe/TRAQBRYNQ-FR69UJMKL/cli-cheat-sheet.pdf?c=1576073769-d69e8d8b5bdcab9e5e80218333ab005d2a35a4ac>

------------------------------------------------------------------------------------------------------------------------

**12.12.2019**

**SEARCH AND FIND**

find app –name \*

in the past, if i didnßt remember the full name of a file you can use a question mark as place holder for the letters you donßt remember ??? → three question marks

→ if you don’t know how many characters then the first letters and then \*. The extensions are very important.

you can use \*.\* if you don’t know the extension.

→ ls -la → information of a file → there are some numbers at the beginning. they refer to the owner of the file, the groups using these machines and the last three numbers are for public

XXXXXXXXX → - (file), d (directory), l (link), r (read) w (write) x (execute-change)

→ sometimes there are numbers instead of letters for the permissions, there is a website tat calculates that for you.

<https://chmod-calculator.com/>

<https://www.pluralsight.com/blog/it-ops/linux-file-permissions>

--

sudo apt-get install npm

→ **node package manager**

to upgrade→ sudo apt-get upgrade npm

to remove → sudo apt-get remove nmp

to upgrade globally → sudo apt-get upgrade -g npm

→ if I use update without pointing which application it will update all the applications in the machine→ in order to update just an application you have to point which one

→ better to not update all applications regularly. better check what needs to be updated

→ these applications are in a library from linux, not in the internet, the library is called apt. we will learn how to upload applications to the apt library

**GITHUB**

→ new repository

→ cerate new file→ README.md

→ this file is the only file in which I will always use markdown language

→ write something

-->EVERY STEP IN GITHUB HAS TO BE COMMITTED

→ commit new file

there are branches→ main branch is the master→ later on the master will have a different file and the owner branches will have other pages. now we all work with the same branches.

<https://github.github.com/gfm/>

→ you can delete the files opening them and clicking on the trash can → commit

commit→ every step is part of an achievement list → it acts as a list of things that have been done. → it shows in a different color in the preview → always write comments that make sense.

---> you can see how many changes have been done and there is a number on the right side

→ every github account has settings but also every repository

→ you can rename there. the name has to always be different, names can not double

→ Danger Zone

→ those changes can only be done once

→ if you create a file with the option of creating a readme file automatically the title with appear automatically with headed title, otherwise it is the same

task→ do a readme file with everything: image, table, lists, etc...

→ check our team githum repository often, there will be constant updates:

[**https://github.com/FBw-26**](https://github.com/FBw-26)

→ to see the version → git --version

→ from the terminal, how to clone a repository

→ git clone + link (<http://github.com>..

→ create new repository→ test → branch → new

→ so changes are different according to the branch selected.

→ in order to have both in both branches it has to be merged

→ to copy the information the new branch has to be created from the sub-branch → it copies automatically when creating a new branch from the sub-branch (nnot from the master)

→ **generating SSH key** → to create security

→ git is local whereas github is in the server

→ we connect git and github through SSH keys

→ ssh-keygen -t rsa -b 4096 -C "*your\_email@example.com*

→ write your email address inside the quotes

→ enter → enter as many times as necessary

more info: <https://help.github.com/en/github/authenticating-to-github/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

<https://www.ssh.com/malware>

------------------------------------------------------------------------------------------------------------------------

**17.12.2019**

**MORE GITHUB**

→ create new repository

→ do you wanna create .gitignore? → file to control other hidden files for git → add node option → later on we will create our own file but for now auto generate .gitignore

→ keep license → none

→ upload some images → you can download them or delete→ no way to get back what you delete

→ always commit all the changes

→ to have an image uploaded to github allows to link the readme file to the image in the local repository

→ GIT is the local place to work in the files and GITHUB is online→ distributed version control system.

→ git is a version of svcs→ usually used for open source applications→ everyone has access to it → allow full access to every file, branch and iteration of a project

→ allows every user to access to a full and self contained history of changes

→ allows collaboration, everyone has access to the same files

→ if 2 people are working on the same file at the same time github will show the changes and you should be able to choose which one you want to commit

→ we will never work like this i the future, uploading and deleting individually, but for now it is just to learn the features

→ git init, git commit, git pull, git push , git status, git add, git branch, git clone, are the usual features available in github

--

**→ you can check what’s going on in github from the terminal.**

→ create a directory called github-test

→ create a repository without the readme file

→ at terminal go to

→ to configure email → git -- global user.name “Gabriela Acha”

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela$ mkdir github-test

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela$ cd github-test

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ ls

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ ls-la

ls-la: command not found

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ ls -la

total 8

drwxr-xr-x 2 dci-st124 dci-st124 4096 Dez 17 10:46 .

drwxr-xr-x 10 dci-st124 dci-st124 4096 Dez 17 10:46 ..

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git init

Initialized empty Git repository in /home/dci-st124/gabriela/github-test/.git/

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ ls -la

total 12

drwxr-xr-x 3 dci-st124 dci-st124 4096 Dez 17 10:47 .

drwxr-xr-x 10 dci-st124 dci-st124 4096 Dez 17 10:46 ..

drwxr-xr-x 7 dci-st124 dci-st124 4096 Dez 17 10:47 .git

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git status

On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ ^C

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ ^C

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ touch README.md

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ ls

README.md

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

README.md

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git commit -m "fist commidci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git commit -m "first commit"

\*\*\* Please tell me who you are.

Run

git config --global user.email "you@example.com"

git config --global user.name "Your Name"

to set your account's default identity.

Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'dci-st124@dcist124-Lenovo-V130-15IKB.(none)')

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git config -- global user.name "Gabriela Acha"

error: key does not contain a section: global

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git config ––global user.name "Gabriela Acha"

error: key does not contain a section: ––global

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git config –– global user.name "Gabriela Acha"

usage: git config [<options>]

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git config --global user.name "Gabriela Acha"

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git config --global user.email "gabriela.gabriela.acha.acha@gmail.com"

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

README.md

nothing added to commit but untracked files present (use "git add" to track)

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git commit -m "first commit"

On branch master

Initial commit

Untracked files:

README.md

nothing added to commit but untracked files present

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git remote add origin <https://github.com/gabrielaacha/github-test.git>

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git push origin master

error: src refspec master does not match any

error: failed to push some refs to 'https://github.com/gabrielaacha/github-test.git'

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git push origin master

error: src refspec master does not match any

error: failed to push some refs to 'https://github.com/gabrielaacha/github-test.git'

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$

→ to validate a newly created file:

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git add index.html

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git add README.md

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: README.md

new file: index.html

→ in order to commit to master in github →  **git commit -m “second commit”**

→ **push to master:**

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git push origin master

Username for 'https://github.com': gabrielaacha

Password for 'https://gabrielaacha@github.com':

Enumerating objects: 4, done.

Counting objects: 100% (4/4), done.

Delta compression using up to 4 threads

Compressing objects: 100% (2/2), done.

Writing objects: 100% (4/4), 286 bytes | 286.00 KiB/s, done.

Total 4 (delta 0), reused 0 (delta 0)

To https://github.com/gabrielaacha/github-test.git

\* [new branch] master -> master

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ ^C

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$

→ to pull from master, if you have been away from some time from the project:

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git status

On branch master

nothing to commit, working tree clean

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$ git pull origin master

remote: Enumerating objects: 5, done.

remote: Counting objects: 100% (5/5), done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), done.

From https://github.com/gabrielaacha/github-test

\* branch master -> FETCH\_HEAD

babc735..a33639c master -> origin/master

Updating babc735..a33639c

Fast-forward

index.html | 1 +

1 file changed, 1 insertion(+)

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/github-test$

→ to initialise the repository from the terminal → **git init**

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela$ mkdir test-random

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela$ cd test-random

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/test-random$ ls -la

total 8

drwxr-xr-x 2 dci-st124 dci-st124 4096 Dez 17 11:42 .

drwxr-xr-x 11 dci-st124 dci-st124 4096 Dez 17 11:42 ..

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/test-random$ git init

Initialized empty Git repository in /home/dci-st124/gabriela/test-random/.git/

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/test-random$ ls -la

total 12

drwxr-xr-x 3 dci-st124 dci-st124 4096 Dez 17 11:42 .

drwxr-xr-x 11 dci-st124 dci-st124 4096 Dez 17 11:42 ..

drwxr-xr-x 7 dci-st124 dci-st124 4096 Dez 17 11:42 .git

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/test-random$ touch README.md

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/test-random$ git add .

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/test-random$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: README.md

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/test-random$

we need to define who is controlling → **git remote add origin git@github.com:gabrielaacha/test-random.git**

→ after commit changes → PUSH to master (**git push origin master**)

→ **git remote add origin** [**git@github.com**](mailto:git@github.com)**:gabrielaacha/test-test.git** → this means that github will have remote access to that repo

→ **git log** is to check what happened and who committed the changes→ history of the repo

→ before pushing everything happens **locally**, so you need internet to push it to github, **globally**

------------------------------------------------------------------------------------------------------------------------

**18.12.2019**

→ go to the directory lessons in the directory fbw26 and type **git remote -v** → to see who has the control

git remote -v

origin https://github.com/FBw-26/lessons.git (fetch)

origin https://github.com/FBw-26/lessons.git (push)

→ we will add another origin → **git remote rename origin upstream**

→ copy line : git remote add origin git@github.com:gabrielaacha/lesson-from-fbw26.git

→ and then **git remote -v** to see who is in control

git remote -v

origin git@github.com:gabrielaacha/lesson-from-fbw26.git (fetch)

origin git@github.com:gabrielaacha/lesson-from-fbw26.git (push)

upstream https://github.com/FBw-26/lessons.git (fetch)

upstream https://github.com/FBw-26/lessons.git (push)

→ **git fetch upstream**

remote: Enumerating objects: 11, done.

remote: Counting objects: 100% (11/11), done.

remote: Compressing objects: 100% (6/6), done.

remote: Total 9 (delta 2), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (9/9), done.

From https://github.com/FBw-26/lessons

00b421c..6b7531d master -> upstream/master

**→** now i wann push to my github repo → **git add .** and **git commit -m “bla bla**”

→ **git push origin master** → in order to push it to our repository

---

→ we want to clone a repo from fbw26

→ we go to the repo and clone

→ git clone and copy the git hub link <https://github.com/FBw-26/clone-test.git>

→ git remote rename origin upstream

→ git remote add origin git@github.com:gabrielaacha/clone-test.git

→ git add . and git commit …

→ push

→ whenever you are working in foreign repo then **fetch** and not **push**

**--**

→ **git fetch upstream**

→ will push the updates to the original repository

remote: Enumerating objects: 5, done.

remote: Counting objects: 100% (5/5), done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), done.

From https://github.com/FBw-26/lessons

6b7531d..f89bae3 master -> upstream/master

→ **git pull upstream master**

→ will update our repository with what is new at upstream, at the original repository: fbw26

From https://github.com/FBw-26/lessons

\* branch master -> FETCH\_HEAD

Updating 00b421c..f89bae3

Fast-forward

README.md | 28 ++++++++++++++++++++++++++++

1 file changed, 28 insertions(+)

→ **git add** . and **git commit -m “new update”**

→ **git push origin master**

---

echo"# github-tets" >> README.md

this line will create a readme file with this text in it

---------------

if we imagine the github projects as trees we imagine them as a tree and every branch is a copy of the main

→ create new branch (**new-branch**)

→ go to pull requests (to merge the new work with the master)

→ create pull request → **pull new update**

→ choose base and compare it with (a branch for ex.)

→ if nothing is conflicting the branches will be able to merge

→ merge pull request

→ confirm

→ when merging always add a comment to the master→ write what you have done

→ **merge and pull request**

---------------------------------

→ to see the branches in terminal → **git branch**

→ **git checkout -b logout**

→ **git branch** (again)

→ this will switch the brach → in order to change the branch → git checkout (name of the other branch)

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/fbw26$ cd github-test-2000/

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/fbw26/github-test-2000$ less READMEmd

READMEmd: No such file or directory

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/fbw26/github-test-2000$ git branch

\* master

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/fbw26/github-test-2000$ git checkout -b logout

Switched to a new branch 'logout'

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/fbw26/github-test-2000$ git branch

\* logout

master

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/fbw26/github-test-2000$ nano README.md

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/fbw26/github-test-2000$ git checkout master

M README.md

Switched to branch 'master'

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/fbw26/github-test-2000$ git checkout logout

M README.md

Switched to branch 'logout'

dci-st124@dcist124-Lenovo-V130-15IKB:~/gabriela/fbw26/github-test-2000$

→ when pushing → push to logout (the branch) not to the master!!

git push origin logout

-----------------------------------

to delete repos:

→ settings

→ danger zone

--------------------------------

→ **git branch** will show you in which branch you are → which branch you are working on

→ to merge → git merge ddd (you have to be in the master branch to merge it with the master branch)

-------------------------------

echo '# Apple# >> README.md → echo will write whatever is inside the quotes in the README.md file

------------------------------

**Github**

→ settings

→ collaborators

→ add a collaborator

→ then you can work on the other persons repository

→ create a new branch to work on

→ when merging you should need the verification of the other person

-------

------------------------------------------------------------------------------------------------------------------------

**08.01.2020**

**Java Script**

in terminal : **node index js** → in order to see the code in the terminal within visual code

→ var mostly used for hoisting and global

→ bookmark for java script

**const** → another way to define a variable

constant value, you won’t be able to change it anymore

with **let** i am still able to change the value of x, with const you can’t, no one can touch the function. we won’t see the benefit of it until we see it inside of a function

let can be used everywhere and also var

→ **global variables** → i use them anywhere in my code. i need them because i need to access them any time in my code

→ **local variables** → they live inside their scope

→ **\** in order to ignore the character that comes after, for instance a **‘**

---

let kid = 2;

console.log(kid>16 ? "Party":"Home Party");

→ if the kid is younger than 16 then home party

→ if older then party

--

Bracket notation, to chose which character we wanna choose

→ **[ ]**

// `'\`, `\"`, `\n` for a new line

console.log("Hi I am \n Hadi");

// .trim() delets spaces beginning and end

let str4 = "Hadi";

console.log(str4.trim());

→ to create your own function in or the function to avoid writing console.log all the time. you can put it at the beginning of the page and write print instead of console.log:

let str4 = "Hadi";

console.log(str4.trim());

function print (str){

console.log(str);

}

print ("Hadi");

**// NUMBERS**

There are different systems:

→ regular: 1-9

→ Binary: 1 and 0

→ decimal → 1,2 etc...

→ Hexadecimal: 1-9 and ABCDEF

→ Octal

--

**excercises:**

you can use + - / or \* and if you use % it will give you the remainder

→ how to apply remainder in java script?

**// 1. Create two variables. One variable should contain a string value and the other should contain a number. Concatenate the string and the number.**

**// 2. Check if the following numbers are even numbers. Use a ternary and if the number is even print 30 is even, else print that it is odd.**

**// 9. Create a new string from a given string by changing the position of first and last characters. The string length must be greater than or equal to 1. eg. of output JavaScript => tavaScripJ**

--

in order to see the results all the time download nodemon, so you don’t have to type nodemon all the time

if you wanna use terminal normally aǵain exit nodemon with cntr + C

--

for resources:

<https://developer.mozilla.org/>

or write in the browser mdn and the question : math.max MDN

---------------------------------------------------------------------------------------------------------------------------

**09.01.2020**

**more Javascript**

let x, y, z; //

x = 1; //

y = "T";

console.log(x);

console.log(Math.max(33, 44, 52, 111111, 76, 7, 0));

console.log(Math.min(33, 47111, 4, 5, 6, 0.3, 7, y));

//we create this short cut to not have to use console.log all the time

function display(str) {

console.log(str);

}

display("Hi this is a new function");

// every information entered by the user will first appear as string, it has to be turned into number to be calculated

let strInt = "12333";

let strInt2 = "999";

console.log(strInt + strInt2);

console.log(parseInt(strInt) + parseInt(strInt2));

**// same with decimals, we use the US system but since european users will enter data with commas we have to be aware of that**

let strFloat = "333.553";

console.log(parseFloat(strFloat));

**// round up and down**

let float1 = 2.4;

console.log(Math.ceil(float1)); // will round up the number

console.log(Math.floor(float1)); // will round down the number

let str1 = "this is a new text";

console.log(str1.toUpperCase());

**//Addition**

**// x + y \* s**

console.log(((x + y) / (z + 2.3)) \* 29 + x);

**// String + Number**

let d, f, r;

d = 5;

f = "Hi I'm Gabriela";

console.log(f + d);

**// Concatenation**

**// String + Boolean**

r = true;

console.log(r + d); // true in javascript is 1 and false is 0

**// String + String**

let str2 = "Hi";

let str3 = " I am Gabriela";

console.log(str2 + str3);

**// typeof (Objects, functions, console...)**

console.log(typeof str3); // I would use it when I am not sure what type of data I am dealing with (whether it is a string or a number), often you don't know the type of data because it is coming from the backend

console.log(typeof x);

console.log(typeof console);

console.log(typeof console.log);

console.log(typeof Math.floor);

**// (0.1 \* 0.2)**

let floatNum = 0.1;

let floatNum1 = 0.2;

console.log((floatNum \* floatNum1).toFixed(3)); // this makes the result be limited to 3 digits after the comma

let floatNum2 = 0.1;

let floatNum3 = 0.2;

let num = 3;

let result = floatNum2 \* floatNum3;

let lastResult = result.toFixed(num);

console.log(lastResult);

**// 1.23e+5 some number have an e (big number that is replaced by e)**

**// x = 3.444**

let intStr = "10 Hi this is cool"; // if the 10 is at the end it is not interpreted as a number

let floatStr = "1033.3 hi this is noooo";

console.log(parseInt(intStr));

console.log(parseFloat(floatStr));

//console.log(x.toFixed(2));

**// && || (and or)**

let var1 = true;

let var2 = true;

let var3 = true;

console.log(var1 && var2 && var3 ? "cool" : "not cool"); // if ALL variables are true then "cool", else "not cool"

let var4 = 1 > 4; // false

let var5 = false;

let var6 = true;

console.log(var3 && var4 && var5 ? "cool" : "not cool"); // if ALL variables are true then "cool", else "not cool"

console.log(var3 || var4 || vcar5 ? "cool" : "not cool"); // if ONE of the variables is true then "cool", else "not cool"

**// = Assigning**

console.log(ohNo);

var ohNo;

ohNo = x; // if it would be LET it would apply only to this function and the variable would have to be declared before the console.log

console.log(ohNo);

console.log(ohNo2);

var ohNo2;

let xx = "1";

ohNo2 = 1;

console.log(ohNo2);

**// == vs ===**

**// 2 times = to check if one number is equal to the other**

**// 3 times = to ???**

console.log(1 == 1 ? "yeah it is" : "nope it is not");

console.log(1 == 2 ? "yeah it is" : "nope it is not");

console.log(x == 1 ? "yeah it is" : "nope it is not");

// console.log(x == OhNo ? "yeah it is" : "nope it is not");

// console.log(xx === OhNo2 ? "yeah it is" : "nope it is not");

**// Truthy and falsy values: false, undefined, null, 0, ''**

console.log(true ? "yes" : "Noooooo"); //every number but 0 is true in javascript

console.log(1 ? "yes" : "Noooooo");

console.log(0 ? "yes" : "Noooooo");

console.log("any word" ? "yes" : "Noooooo"); // any content in the string is true

console.log("" ? "yes" : "Noooooo"); // empty string is always false

console.log(null ? "yes" : "Noooooo"); // false

console.log(undefined ? "yes" : "Noooooo"); // falthy

console.log(Math.random ? "yes" : "Noooooo"); // I wanna check if these things exists

**// what is not a number**

**// https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/isNaN**

isNaN(NaN); // true

isNaN(undefined); // true

isNaN({}); // true

isNaN(true); // false

isNaN(null); // false

isNaN(37); // false

**// strings**

isNaN("37"); // false: "37" is converted to the number 37 which is not NaN

isNaN("37.37"); // false: "37.37" is converted to the number 37.37 which is not NaN

isNaN("37,5"); // true

isNaN("123ABC"); // true: parseInt("123ABC") is 123 but Number("123ABC") is NaN

isNaN(""); // false: the empty string is converted to 0 which is not NaN

isNaN(" "); // false: a string with spaces is converted to 0 which is not NaN

var NotaNumber = undefined;

console.log(isNaN(NotaNumber) ? "true" : "false");

console.log(isNaN(NotaNumber) ? "this is NOT a number" : "this is a number");

var NotaNumber2 = true;

console.log(isNaN(NotaNumber2) ? "true" : "false");

var NotaNumber3 = null;

console.log(isNaN(NotaNumber3) ? "true" : "false");

var NotaNumber4 = "37";

console.log(isNaN(NotaNumber4) ? "true" : "false");

var NotaNumber5 = "";

console.log(isNaN(NotaNumber5) ? "true" : "false");

**// Math.random // we will constantly use it in react**

**console.log(Math.floor(Math.random() \* 10) + 1); // you multiply in order to move the comma and get some control // Math.floor is used to turn the number integer**

**// EXERCISE**

**// 1. Minimum and maximum**

**// a. Lowest Number**

**// Print out the lowest number between -1 and 4.**

console.log(Math.min(-1, 0, 1, 2, 3, 4));

**// b. Highest Number**

**// Print out the highest number between -1 and 4.**

console.log(Math.max(-1, 0, 1, 2, 3, 4));

**// 2. Rounding**

**// a. Round up**

**// Round up the following numbers: 3321, 326.76, -9.78, 43.342.**

let roundup1 = 3321;

let roundup2 = 326.76;

let roundup3 = -9.78;

let roundup4 = 43.342;

console.log(Math.ceil(roundup1));

console.log(Math.ceil(roundup2));

console.log(Math.ceil(roundup3));

console.log(Math.ceil(roundup4));

**// b. Round down**

**// Round down the following numbers: 3321, 326.76, 76788.7, -9.78,**

let rounddown = 3321;

let rounddown2 = 326.76;

let rounddown3 = 76788.7;

let rounddown4 = -9.78;

console.log(Math.floor(rounddown));

console.log(Math.floor(rounddown2));

console.log(Math.floor(rounddown3));

console.log(Math.floor(rounddown4));

**// 3. Dice Roll!**

**// Create a program that prints a random integer from 1 - 6.**

console.log(Math.floor(Math.random() \* 6) + 1);

-------------

function getBaseLog(x, y) {

return Math.log(y) / Math.log(x);

}

// 7 x 7 = 49

console.log(getBaseLog(7, 49));

// expected output 2

// 4 x 4 x 4 x 4 x 4 x 4 = 4096

console.log(getBaseLog(4, 4096));

// expected output 6

---------------------------------------------------------------------------------------------------------------------------

**13.01.2020**

**JAVA SCRIPT**

**// console.log(x.toFixed(4));**

**let old, nice, crazy;**

old = true;

nice = false;

crazy = true;

console.log(!crazy); // exclamation mark wil give you the opposite to this situation

console.log(1 !== 1); // if the program thinks 1 is not equal to 1 you are telling it it is

console.log(old !== crazy); // not true because old is equal to crazy

console.log(old !== nice); // true because old and crazy are ot equal

**// boolean inversion with !**

**//console.log(crazy ! "Yeah" : "nope");**

**// givenName || firstName || "ABC";**

console.log("Abc");

let givenName, firstName;

firstName = "Hadi";

console.log(givenName || firstName || "Abc"); // if all of them are undefined then its undefined, if two are truthy it will take the first one to console.log

**// let's do some algorithms**

**// Nancy's income is 500€ per week. how much does Nancy makes every year?**

**const nancySalary = 500 \* 52;**

**// console.log(nancySalary)**

console.log(`Nancy Salary ${nancySalary} Euro in one year`);

**// the amount of hours Nancy worked over the last two weeks are 8, 6, 5, 9, 8, 2, 1, 8.5, 7, 4 what is nancy's average hour?**

const nancyAverage = (8 + 6 + 5 + 9 + 8 + 2 + 1 + 8.5 + 7 + 4) / 10;

console.log(nancyAverage);

console.log(`Nancy average hours of ${nancyAverage} in a week`);

**// hadis version**

const nancyWorkingHours = 8 + 6 + 5 + 9 + 8 + 2 + 1 + 8.5 + 7 + 4;

let nancyAverage2 = nancyWorkingHours / 10;

console.log(

`Nancy worked last two weeks ${nancyWorkingHours}, the average is ${nancyAverage2}`

);

**// if (<boolean>){...}**

console.log(true ? "yeah cool" : "nope sorry");

let a, b, c;

a = true;

b = false;

c = true;

let counter = 0;

let logIn = false;

if (a == b && c) {

// false && true is understood as false

console.log("Nice");

counter = 2; // if statement is truth assign 2 to the counter and true to the logIn

logIn = true;

} else {

// is the statement is true then it will print Nice, otherwise it will print Not cool

console.log("Not cool");

counter = 3;

logIn = false;

}

**// `if` vs. ternary operator | if (true) {condition} curly brackets for the truthy situation {} if this is not true java script will ignore it**

**// preserved words can't be used as variables in java script, for example if**

if (true) {

console.log("yeah it is true");

}

if (false) {

let superHero = 44; // usually variables are declared outside the curly brackets not inside

console.log("no it's not true"); // if the declaration is nor true java script will ignore it

}

if (1 < 4 || (1 < 55 && false)) {

console.log("yeah its true");

}

**//EXERCISES**

**// A math student scored 75, 70, 85, 90, 100 on the first five tests he took . After he took his sixth math test, the average is now 85. What did he score on the sixth test?**

const mathStudent = 75 + 70 + 85 + 90 + 100;

const studentAverageTest5 = mathStudent / 5;

const studentAverageTest6 = 85 \* 6;

console.log(studentAverageTest6 - mathStudent);

**// Expected output: Score in the sixth test: --.**

let studentGrade = studentAverageTest6 - mathStudent;

console.log(`the expected score in the 6th test should be ${studentGrade}`);

**// For James to get a 1st class degree, he needs to get an average of 80% in all 9 of his assessments. He has taken 8 assessments and his average is 78%. What is the minimum percentage he must get in his last assessment to ensure he gets a first class?**

const jamesAverage8 = 78 \* 8;

const jamesRequiredGrade = 80 \* 9;

const jamesAverage9 = jamesRequiredGrade - jamesAverage8;

console.log(jamesRequiredGrade - jamesAverage8);

if (jamesAverage9 <= 96) {

console.log("1st class degree");

} else {

console.log("2nd class degree");

}

**//Expected output: James needs a minimum of --% to get an 80% average.**

console.log(`James needs a minimum of ${jamesAverage9}% to get an 80% average`);

**// Check if two given integer values are in the range 50 to 99 (this includes both 50 and 99). Print true if either of them are in the range.**

let intMin = 50;

let intMax = 99;

let x = 55;

let z = 192;

if ((x >= 50 && x <= 99) || (z >= 50 && x <= 99)) {

console.log("integer values are between 50 and 99");

}

**// Check if three given integer values are in the range 50 to 99 (inclusive). Print true if one or more of them are in the range.**

let intNumNum1 = 67;

let intNumNum2 = 88;

let intNumNum3 = 1921;

if (

(intNumNum1 >= 50 && intNumNum1 <= 99) ||

(intNumNum2 >= 50 && intNumNum2 <= 99) ||

(intNumNum3 >= 50 && intNumNum3 <= 99)

);

console.log(true);

**// Declare the variables a, b and c, and give them number values. Check which one out of the three variables has the largest value and print it to the console. Then change the values of the variables to see if your conditional still works.**

let a1 = 3;

let b1 = 9;

let c1 = 58;

console.log(Math.max(a1, b1, c1));

**// Create a new string adding “Py” in front of a given string. If the given string begins with “Py” then print the original string.**

let str1 = "Py";

let str2 = "thon";

console.log(str1 + str2);

console.log(str1 + "thon");

console.log(`${str1}thon`);

let str = "Python";

if (str.substring(0, 2) === "Py") {

console.log(str);

}

**// Calculate the sum of the two integers. If the sum is in the range 50 to 80, print 65, otherwise print 80.**

let intNum1 = 65;

let intNum2 = 7;

let y = intNum1 + intNum2;

if (y > 59 && y < 80) {

console.log(65);

} else {

console.log(80);

}

**// Check whether the sum of two integers is 8, or whether their difference is 8. If one of these is the case, print true.**

let intInt1 = 3;

let intInt2 = 5;

let intDiff = Math.abs(intInt1 - intInt2);

let intSum = intInt1 + intInt2;

console.log(intDiff);

if (intSum == 8 || intDiff == 8) {

console.log(true);

}

**// Check whether either one of two integers is 15, or if their sum is 15. If one of these is the case, print true.**

let intNumero1 = 5;

let intNumero2 = 10;

let intSum2 = intNumero1 + intNumero2;

if (intNumero1 == 15 || intNumero2 == 15 || intSum2 == 15) {

console.log(true);

}

**// Check whether one of two integers is a multiple of 7 or 11. If so, print true.**

let intMul1 = 45;

let intMul2 = 78;

if (

intMul1 % 7 == 0 ||

intMul1 % 11 == 0 ||

intMul2 % 7 == 0 ||

intMul2 % 11 == 0

);

{

console.log("intMul is multiple of 7 or 11");

}

**// Calculate the sum of the two given integers. If the two values are the same, then print triple their sum.**

let givenInt1 = 23;

let givenInt2 = 23;

let givenIntTrip = (givenInt2 + givenInt1) \* 3;

if (givenInt1 === givenInt2) {

console.log(givenIntTrip);

}

**// Calculate the difference between a specified number and 19. Print double the difference if the specified number is greater than 19.**

let num1 = 3;

let numDiff = Math.abs(num1 - 19);

let numDo = numDiff \* 2;

if (numDiff < 19) {

console.log(numDo);

}

**// Mark and John are trying to compare their BMI (Body Mass Index),**

**// which is calculated using the formula: BMI = mass / (height \* height). (mass in kg, i.e. 72.5, and height in metres, i.e. 1.65).**

**// Store Mark’s and John’s mass and height in variables.**

**// Calculate both their BMIs and store their BMIs in variables.**

**// Store Mark’s and John’s mass and height in variables.**

**// Calculate both their BMIs and store their BMIs in variables.**

**// Create a boolean variable containing information about whether Mark has a higher BMI than John.**

**// Print a string to the console containing the variable from step 3 using string interpolation.**

**// (e.g. “Is Mark’s BMI higher than John’s? Why yes, it’s [boolean], it is”).**

**// Create an if statement which prints the name and BMI of the person with the highest BMI**

let guy1 = "Mark";

let guy2 = "John";

let massMark = 82;

let massJohn = 75.5;

let heightMark = 1.83;

let heightJohn = 1.88;

let bmiMark = massMark / heightMark;

let bmiJohn = massJohn / heightJohn;

console.log(bmiMark);

console.log(bmiJohn);

if (bmiMark > bmiJohn) {

console.log(`${guy1}'s BMI is higher than ${guy2}`);

} else {

console.log(`${guy2}'s BMI is higher than ${guy1}`);

}

let bmiHighest = Math.max(bmiMark, bmiJohn);

console.log(bmiHighest);

**// Make a variable for firstName and age and give each variable values.**

**// Create an if/else statement to check whether the person’s age is less than 13.**

**// If so, print “firstName is a child”.**

**// If the age is equal to or more than 13 and less than 20, print “firstName is a teenager”.**

**// If the person’s age is equal to 20 and less than 30, then print “firstName is a young adult”.**

**// If none of these conditions apply, print “firstName is a adult”.**

let firstName2 = "Micky";

let age = 15;

if (age < 13) {

return console.log(`${firstName2} is a child`);

}

if (age >= 13 && age <= 20) {

return console.log(`${firstName2} is a teenager`);

}

if (age >= 20 && age <= 30) {

return console.log(`${firstName2} is a young adult`);

} else {

return console.log(`${firstName2} is an adult`);

}

**// Hadi's solution**

const firstName3 = "Peter";

const age2 = "13";

if (age2 < 13) {

console.log(`${firstName3} is a boy`);

} else {

if (age2 < 20) {

console.log(`${firstName3} is a teenager`);

} else {

if (age2 <= 30) {

console.log(`${firstName3} he is an adult`);

} else {

console.log(`is not a number`);

}

}

}

**// it's same but omitting the curly brackets // technically it is the same but this is more common**

const age3 = "13";

if (age3 < 13) {

console.log("he is a boy");

} else if (age3 < 20) {

console.log("he is a teenager");

} else if (age3 >= 30) {

console.log("he is an adult");

} else {

console.log("is not a number");

}

homework → check substring

---------------------------------------------------------------------------------------------------------------------------

**14.01.2020**

**JAVA SCRIPT**

questions:

→ how to download nodemon again.

**→ multipple if statements → nested if/else**

→ if x < 20 → teenager

→ if x > 13 → …

const age = "13"

if (age < 13) {

console.log ("he is a boy");

} else {

if (age < 20){

console.log("he is a teenager");

}else{

if (age <= 30){

console.log("he is an adult");

}else{

console.log("is not a number");

}

}

}

→ it is better to use else if it is technically the same but more common:

const age3 = "13";

if (age3 < 13) {

console.log("he is a boy");

} else if (age3 < 20) {

console.log("he is a teenager");

} else if (age3 >= 30) {

console.log("he is an adult");

} else {

console.log("is not a number");

}

**---> switch (<expression>) { …}**

→ it is used for choosing countries or telephone prefixes, etc, usually in forms

**// switch (<expression>){ ... } // it is used for ex. for currencies, for phone prefixes, etc…**

const color = "Pink";

switch (color) {

case "Pink":

console.log("cool");

break;

case "Black":

console.log("Nice");

break;

case "white":

console.log("soooo nice");

break;

default:

console.log("Noooo that's crazy");

}

----

→ if else when i compare a raise, unpredicted answers, I don’t wanna list all th ecases, in the range.

→ switch case whenever i have a concrete answer, street name, color, country, instruments, etc…

→ **switch** questions the value given and checks if it matches with the cases:

**// an example of switch in order to choose one month of the year:**

const month = 9;

switch (month) {

case 0.1:

console.log("Jan");

break;

case 2:

console.log("Feb");

break;

case 3:

console.log("Mar");

break;

case 4:

console.log("Apr");

break;

case 5:

console.log("May");

break;

case 6:

console.log("Jun");

break;

case 7:

console.log("Jul");

break;

case 8:

console.log("Aug");

break;

case 9:

console.log("Sept");

break;

case 10:

console.log("Oct");

break;

case 11:

console.log("Nov");

break;

case 12:

console.log("Dec");

break;

default:

console.log("This is not real");

}

→ swith case vs if else:

**// another example for switch:**

const x = 22.22;

switch (x) {

case 1999.33:

console.log("fine");

break;

case 22.22:

console.log("this is wrong");

break;

case 1000000001:

console.log("cool");

break;

default:

// you always need a default in case the user types something that is not listed

console.log("so this is not for fun or is it!");

}

**// switch` vs. `else if`**

const num = 1;

if (num <= 10 || num >= 0) {

console.log("it is num in the range of 0 and 10");

}

if (num == 1) {

console.log(num);

}

if (num == 2) {

console.log(num);

}

if (num == 3) {

console.log(num);

}

if (num == 4) {

console.log(num);

} else {

console.log("It is new num");

}

const num2 = 1;

switch (num2) {

case 1:

console.log(num2);

break;

case 4:

console.log(num2);

break;

case 3:

console.log(num2);

break;

default:

console.log("this is fun");

}

**// the order is very important, the statement that is on the top gets the priority // if the string is empty it would be 0 and divisible by 3 and 5 so it would print FizzBuzz**

------------------------------------------------------------------------------------------------------------------------

**15.01.2020**

**JAVA SCRIPT**

// 1. Calculate the perimeter of a square. Assume each side is 4.75cm.

let side = 4.75;

let per = side \* 4;

console.log(`the perimeter of a square is ${per}cm`);

// 2. Calculate the perimeter of a triangle. Assume the length of the sides are 5cm, 6cm (base), 7cm.

let tside1 = 5;

let tside2 = 6;

let tside3 = 7;

let tper = tside1 + tside2 + tside3;

console.log(`the perimeter of the triangle is ${tper}cm`);

// 3. Calculate the area of a square. Each side is 5cm.

let qside = 5;

let qarea = Math.pow(5, 2);

console.log(`the area of the square is ${qarea}cm`);

// 4. Calculate the area of a triangle. Assume the length of the sides are 5cm, 6cm, 7cm.

// For a triangle with sides a, b and c, the half-perimeter s = 1/2(a + b + c). Once you know s, you calculate area using this formula:

// Area = square root [s (s - a) (s - b) (s - c)]

let tside4 = 5;

let tside5 = 6;

let tside6 = 7;

let s = (tside4 + tside5 + tside6) / 2;

let tarea = Math.sqrt(s \* (s - tside4) \* (s - tside5) \* (s - tside6));

console.log(`the area of the triangle is ${tarea.toFixed(2)}cm`);

// 5. Calculate the volume of a cube. Length of each side is 9cm.

let cside = 9;

let cvol = Math.pow(9, 3);

console.log(`the volume of the triangle is ${cvol}cm3`);

//6. Calculate the three bills including tips: €22.35 + 10% tip €26.67 + 15% tip €35.92 + 20% tipCalculate the perimeter of a square. Assume each side is 4.75cm.

let bill1 = 22.35;

let bill2 = 26.67;

let bill3 = 35.92;

let tip1 = (bill1 / 100) \* 10;

let tip2 = (bill2 / 100) \* 15;

let tip3 = (bill3 / 100) \* 20;

let totalbill = bill1 + bill2 + bill3 + tip1 + tip2 + tip3;

console.log(totalbill.toFixed(2));

console.log(`the three bill's total is ${totalbill.toFixed(2)}€`);

// Declare two variables: “a” with the value of true, and “b” with the value of false.

// Check the result of:

// a) a AND b.

// b) a OR b.

// c) NOT (a AND b).

const a = "true";

const b = "false";

console.log(a && b);

console.log(a || b);

console.log(!a && b);

// the second question is Declare three more variables “x”, “y”, “z”. Give these variables number values.

// Check the result of whether:

// a) x is greater than z AND x is greater than y.

// b) x is NOT equal to y.

// c) z is less than y OR z is greater than x.

// d) x is equal to z OR x is NOT equal to y.

// e) x is greater than or equal to 10 AND y is less than or equal to 10.

// f) x multiplied by z is less than 100 OR whether x multiplied by y is greater than 100.

// 16:33

let x = 6;

let y = 10;

let z = 7;

console.log(x > z && x > y); // false

console.log(x !== y); // true

console.log(z < y || z > x); // true

console.log(x == z || x !== y); // true

console.log(x >= 10 && y <= 10); // false

console.log(x \* z < 100 || x \* y > 100); // true

------------------------------------------------------------------------------------------------------------------------

**16.01.2020**

**JAVA SCRIPT**

**LOOPS**

→ in general if there is a method we are researching and following we write it in the notes. Like in the math challenges.

**// LOOPS**

**// there is a way to avoid to repeat the code**

**// there are 3 kinds of loops**

**// it will repeat what is written until the condition becomes falthy**

**// we use “break” whenever we wanna exit the loops**

let x = 3;

if (x <= 10) {

x += 1;

} else {

x -= 1;

}

while (x <= 10) {

console.log("Gabriela");

x++; // it will increase a number in every round and it will stop when the number gets higher than 10 // x = x + 1 increase 1 or x = x - 1 decreases 1

}

// it is not recommended to write a letter inside of the loop

let y = 1;

let result = 0; // this is declared but the value is not so important, just the declaration, to avoid problems better to declare 0

while (y <= 10) {

result = y \* 9;

console.log(`${y} \* 9 = ${result}`);

y++; // very important to not forget this in order to stop the forever repetition

} // this is a common way to use loops

let count = 10;

while (count >= 0) {

console.log(`this is number ${count}`);

count--;

}

let a = 3; // variable has to be declared 3 times --> declaring, checking, increment / decrement

do {

// for "do" you always need a "while"

// whatever the condition is it will do at least one round at a time and then check the condition, even though the condition is false it will print one time

console.log(a);

a++;

} while (a <= 100);

let zara = 100;

do {

console.log(`there is ${zara} on the line`);

zara--;

} while (zara >= 0); // with "do" the condition comes in between the do and the while, with "while" only the condition comes after

**// for : different syntax but same task**

**// in "for" the order is very important**

**// declaring, condition, increment/decrement**

**// mostly used in nested situations**

for (let i = 0; i <= 10; i++) {

console.log(i);

}

**// print out only the even number**

for (let e = 0; e <= 10; e++) {

if (e % 2 == 0) {

console.log(e);

}

}

for (let t = 0; t <= 10; ) {

console.log(t);

t += 2;

}

**// when the increment is 100, 1000 or higher numbers..**.

for (let w = 100; w <= 1000; ) {

console.log(w);

w += 100; // this can be inside the conditioner but better if it is here

}

**//for (..) {}**

for (let i = 0; i <= 100; i++) {

if (i == 30) {

break;

}

console.log(i);

}

for (let w = 0; w <= 100; w++) {

if (w <= 20) {

console.log(w);

break; // we use it to exit the loop after condition // stronger than any condition

}

}

let xx = false;

for (let w = 0; w <= 100; w++) {

if (xx) {

console.log("That is it");

break;

}

if (w == 20) {

xx = true;

}

}

**// control/Strg+D when selecting a variable and then select all the variables with the same name whose name we want to change**

**// DRY --> do not repeat yourself**

**// in javascript everything almost can be nested, also loops**

let result2 = 0;

for (let i = 0; i < 10; i++) {

// i = 1

for (let j = 1; j <= 10; j++) {

// j = 2

result2 = i \* j; // 2 \* 1

console.log(`${i} \* ${j} = ${result2}`);

}

}

let str = "This is a long text"; // the loop will run as far as there are characters

for (let i = 0; i < str.length; i++) {

console.log(`${str[i]}`);

}

let str2 = "Gabriela"; // the loop will run as far as there are characters

for (let i = 0; i < str2.length; i++) {

console.log(`${str2[i]}`);

}

**// EXERCISES**

**// 1.Addition. Write a program to add up the numbers 1 to 20.**

for (let a = 1; a <= 20; a++) {

console.log(a);

}

**// Hadis Answer**

let counterSum = 0;

for (let i = 1; i <= 20; i++) {

counterSum += i; // will add the previous numbers to the current number

}

console.log(counterSum);

**// 2.There are i bottles of beer on the wall. Write a program that will output,**

**// “There is one bottle of beer on the wall.” “There are two bottles of beer on the wall” up until there are five bottles.**

for (let a = 1; a <= 5; a++) {

if (a == 1) {

console.log(`There is ${a} bottle of beer on the wall`);

} else {

console.log(`There are ${a} bottles of beer on the wall`);

}

}

**// Hadis Answer**

let text = "of beer on the wall";

for (let i = 1; i <= 5; i++) {

if (i == 1) {

// you can also write i > 1 instead of ==

console.log(`There is ${i} bottle ${text}`);

} else {

console.log(`There is ${i} bottles ${text}`);

}

}

**// 3.The odd/even reporter. Write a program that will iterate from 0 to 20.**

**// For each iteration, it will check if the current number is even or odd, and report that to the screen (e.g. “2 is even”).**

for (let a = 0; a <= 20; a++) {

if (a % 2 == 0) {

console.log(`${a} is even`);

} else {

//you can also write (i % 2 == 1)...in this case you have to use "else if" cause there are two conditionings, not just "else"

console.log(`${a} is odd`);

}

}

// trick string.repeat(x) --> x = how many times

// for (let i = 1; i <= 10; i++);

**// EXCERCISE**

**// YYYY**

**// YYYY**

**// YYYY**

**// YYYY**

let box = "";

for (let i = 0; i <= 4; i++) {

box += "XXXX \n"; // it will add to the current element whatever you had before

}

console.log(box);

**// 1, 2, 4, 8... 128**

**// 0, 2, 4... 10**

**// 3, 6, 9 ... 15**

**// 9, 8, 7... 0**

**// 1 1 1 2 2 2 3 3 3 4 4 4**

**// 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4**

//---------------------------------------------------

**// Arrays, they**

let firstName = "Hadi";

let firstName1 = "Ali";

let firstName2 = "Olga";

let arrVar = [firstName, firstName1, firstName2];

let arr = ["Hadi", "Ali", "Olga", "Nancy"];

console.log(arr[2]);

let arrInt = [1, 2, 3, 4, 5, 6];

console.log(arrInt[3]); // index numbers always starts from 0

let arrNum = ["One", "Two", "Three", "Four"];

// console.log(arrNum.length); // .length will tell me how many indexes are in the array

for (let i = 0; i < arrNum.length; i++) {

console.log(`${i} is the ${arrNum[i]}`);

}

let arrSt = []; // we declare an empty array that we wanna fill with data

for (let i = 0; i <= 10; i++) {

arrSt[i] = 2 + i; // arrStr[0] = 2 + 0

}

console.log(arrSt);

let arrSt2 = []; // we declare an empty array that we wanna fill with data

for (let i = 0; i <= 10; i++) {

arrSt2[i] = 2 + i; // arrStr[0] = 2 + 0

console.log(arrSt2);

}

**// next week we will do functions and arrays**

---------------------------------------------------------------

**21.01.2020**

**JAVA SCRIPT**

**ARRAY**

**// array name <value 1>, <value2>,...**

let arrStr = ["hi", "text", "text"]; // 1st way of declaring arrays // this is the most common way

console.log(arrStr);

let arrInt = []; // 2nd way of declaring arrays // this is also usual because often you don't know what is the incoming data

console.log(arrInt);

arrInt[0] = 22; // assigning 22 to the very first element of the array

arrInt[1] = 22;

**// Array(num) // to create an array with num slots**

let arrString = new Array(4);

let arrStr2 = Array("Hadi", "NAme1", "NAme2");

console.log(arrStr2);

**// Array.of() creating array**

let strArr = Array.of(3);

console.log(strArr);

**// Accessing using square brackets**

**//Adding items**

arrInt[2] = 33;

arrInt[3] = 53;

arrInt.push(663); // same type of information but different placement this is why you change

arrInt.push(222); // pushing this array to the end of the line

// in order to push it to the front and not the end of the line, we use "unshift"

arrInt.unshift(322);

console.log(arrInt); // this method is not so practical if you don't know the outcome you want

console.log(typeof arrInt); // to check the type

var firstNameArr = ["Ali", "Olga", "HAdi"];

console.log(firstNameArr.indexOf("Olga")); // if the name is in the list it will give the corresponding nr. (1 in this case)

console.log(firstNameArr.indexOf("Nancy")); // if the consoled name is not in the list then you get result -1

// array and for go very often together

let arrStringName = ["Hello 1"];

for (let i = 0; i <= 10; i++) {

arrStringName[i] = `Hadi ${i}`;

}

console.log(arrStringName);

**// deleting items;**

**// array.pop()**

**// array.shift()**

let funArr = ["😎", "💪🏻", "😊"];

funArr.pop(); // in order to delete one item, it will be always the last one, brackets always empty, you don't enter a value in the brackets

funArr.shift(); // this will remove the very first element // no value inside the parenthesis

console.log(funArr);

// this will assign the value declared to the array // it will change the index, not the value

var lastName = ["Hello"];

lastName[0] = "Nsreeny";

lastName[1] = "Dany";

console.log(lastName);

// you can't change the array but you can change the index of the array

// const and var

let x = 0;

x = 2 + 2;

console.log(x);

var y = 0; // original value declaration // used for hoisting and in global situations

y = 2 + 2;

console.log(y);

const zz = 0; // i am not able to change its value anymore

// zz = 3 + 3;

console.log(zz);

let names = ["Hadi", "Ali", "Nancy", "Dany"];

let name = "jam"; // string

console.log(names.length); // index of array

console.log(name.length); // jam

**// Array.slice() // slice would cut**

console.log(names.slice(3));

console.log(names.splice(1, 3));

**// Array.split()**

let stringToArray = name.split("\n");

console.log(stringToArray); // it would split each character in the string within the array

// Manipulating arrays

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**// EXERCISE:**

**// 1. Declare a variable named “euroCities” and assign an array to the variable e.g. [“Paris”, “London”, “Valletta”, “Prague”, “Rome”].**

let euroCities = ["Paris", "London", "Valletta", "Prague", "Rome"];

**// 1b. Declare another variable and assign the second item of the array as a value.**

let euroCity2 = euroCities[1];

**// 2. Change the first item in the array to “Berlin”.**

euroCities[0] = "Berlin";

console.log(euroCities);

**// 3. Print the length of the array “euroCities”.**

console.log(euroCities.length);

**// 4. Remove the last item of the array “euroCities”.**

euroCities.pop();

console.log(euroCities);

**// 5. Use an array method to add “Budapest” to the euroCities array.**

let euroCity = "Budapest";

euroCities.push(euroCity);

console.log(euroCities);

**// 6. Create another variable named asianCities and assign an array of at least 5 cities to the variable.**

let asianCities = ["Tokio", "Shanghai", "Hanoi", "New Dehli", "Bangkok"];

**// 7. Use an array method to select items 2-4 from the array of asianCities and store this in another variable.**

console.log(asianCities.slice(1, 4)); // slice and splice work same

let newArray = asianCities.splice(1, 3); // always create new variable and assign the value

console.log(newArray);

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// Array.join() // erases spaces in between word

const arrayString = ["This", "is", "array", "was", "string"];

let stringToarr = arrayString.join("");

console.log(stringToarr);

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// EXERCISES:

// 1. sumOfNumbers: Create a program that adds up the numbers in an array (of at least 3 numbers).

// Bonus: Print to screen both the sum and the product of these numbers.

let sumOfNumbers = [3444, 7373, 54, 940, 3];

let sum = 0; // you just start with 0

let product = 1;

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

sum += sumOfNumbers[i]; // each time the

product \*= sumOfNumbers[i];

}

console.log(`The sum is ${sum} The product is ${product}`);

const sumOfNumbers3 = [3444, 7373, 54, 940, 3];

const add = (a, b) => a + b;

const sum3 = sumOfNumbers3.reduce(add);

console.log(sum3);

**// 2. Hello Friend: Create an array filled with your friends’ and family’s names.**

**// Loop over the array and greet each friend e.g. Hello Maria! Hello Mike! etc.**

**// Bonus: Print the indexes of each item in the array. Expected output example: Mike is at index 1 of my friends array.**

let friendsName = ["Samuel", "Maria", "Luke", "Mary"];

for (let i = 0; i < friendsName.length; i++) {

// or i keep the = and add -1 // the friends.length counts the amount of elements in the array, you write that if you don't know this amount

console.log(`Hello ${friendsName[i]} and he is index ${i}`);

}

// 3. City Names.: Create an array of city names.

// Loop through the array and add the city names to a string.

// Example of expected output: “Berlin, Paris, Prague, Rome”.

let cityNames = ["Berlin", "Paris", "Prague", "Rome"];

let cityNamesStr = "";

for (let i = 0; i < cityNames.length; i++) {

cityNamesStr += cityNames[i];

if (i == cityNames.length - 1) {

cityNamesStr += ".";

} else {

cityNamesStr += ", ";

}

}

console.log(cityNamesStr);

**// a. Odds and Evens. Create a program that changes a given array by adding 1 to each odd integer and subtracting 1 from each even integer.**

**// Examples:[3, 5, 2, 4] ➞ expected output: [4, 6, 1, 3]**

**// [6, 9, 10, 20] ➞ expected output: [5, 10, 9, 19]**

// let numArr = [2, 3, 1, 4, 5]; // expected result 1,4,2,3,6

// let int;

// for (let i = 0; i < numArr.length; i++) {

// int = numArr[];

// if (int % 2 == 0) {

// int--;

// numArr.push(int);

// } else {

// int++;

// numArr.push(int);

// }

// }

**// b. Capitalize. Create a program that capitalises the first letter of each element in an array of names. Examples:**

**// [“matt”, “sara”, “lara”] ➞ [“Matt”, “Sara”, “Lara”]**

**// [“samuel”, “MARIA”, “luke”, “mary”] ➞ [“Samuel”, “Maria”, “Luke”, “Mary”]**

**// [“Cynthia”, “Karen”, “Jane”, “Carrie”] ➞ [“Cynthia”, “Karen”, “Jane”, “Carrie”]**

const arrNames = ["sAra", "akI", "nancy"];

for (let i = 0; i < arrNames.ĺength; i++) {

arrNames[i] =

arrNames[i][0].toUpperCase() + arrNames[i].substr(1).toLowerCase(); // the first [i] value is a string // this kind of syntax is everywhere

}

console.log(arrNames);

------------------------------------------------------------------------

**21.01.2020**

**JAVA SCRIPT**

**ARRAY and FUNCTIONS**

**// BFF**

function display(str) {

// parameter is a var inside of the function but its value comes from outside

console.log(str);

}

display("Hadi"); // this is used in order to debug the code // it represents the above function and console.log

**// Array.isArray(arr) // to check if it is an array**

**const names = ["Ali", "Hadi", "Olga", "Nancy"]; // with typrof we would bet that it is an object but we need to know that it is also an array**

let x = 1;

console.log(Array.isArray(names)); // i am asking if this is an array

**// Array.includes() // to ask which array i am using now // is this array including "hadi"?**

console.log(names.includes("Hadi"));

console.log(names.indexOf("Hadi")); // to show the index number

**// Array.reverse() // Reverse**

let rev = names.reverse(); // it exchanges the last and the one index at the beginning, its a shift

console.log(rev);

// Array.concat() // combines several arrays into one array

let arr1 = [1, 2, 3]; // array can also be text

let arr2 = [4, 5, 6];

let arrTotal = arr1.concat(arr2); // i call the first arry to apply a function that combines itself with the others arrays

console.log(arrTotal); // expected output --> [ 1, 2, 3, 4, 5, 6 ]

//Array.filter() // creates a new array // narrow down choices, will create a new array with the results

// its a function calling another function that we have to declare (callback is how it's called)

let result1 = names.filter(name => name.length == 4); // empty brackets so far was exec this function (no parameter)

// i am calling the array names to another function, in order to chose the name in the array that has 4 characters

// telling java script to work by name, to chose the individual names with the proposed length

// name can be replaced by x and any name (name => name.length == 4) --> this is a mini function inside the function

// you would do 2 at a time always

console.log(result1);

let result2 = names.filter(name => name.length == 4 || name.length == 5);

console.log(result2);

// Array.find() // return the value of the first element

// find will come up with the very first result it finds, as opposed to filter, which will chose several options

let result3 = names.find(name => name[0] == "A" && name[1] == "G"); // it will come only with the first option it finds

console.log(result3);

// Array.map() // creates a new array // it would replace "for" by this in order to alter the individual indexes

const intArray = [2, 3, 5, 6, 8];

let result4 = intArray.map(x => x + 1); // increase every individual index with 1

console.log(result4);

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// EXERCISES

// 1. Reverse the string: “JavaScript”

// expected outcome :

//[

// 'J', 'a', 'v', 'a',

// 'S', 'c', 'r', 'i',

// 'p', 't'

// ]

// tpircSavaJ

const str1 = "JavaScript";

let strToArr = str1.split("");

console.log(strToArr);

strToArr.reverse();

let revStr = strToArr.join("");

console.log(revStr);

// 1. Make an array of your siblings’ names or your favorite movie characters’ names.

let arrMov = ["Alma", "Elisabeth", "Dorothy"];

// 2. Make an array of your parents’ names.

let arrPar = ["Edurne", "Pablo"];

// 3. Combine these two arrays.

let arrNames = arrMov.concat(arrPar);

console.log(arrNames);

// 4. Add your pets’ names.

arrNames.push("Croquetas");

// 5. Reverse the order of the array.

let arrRev = arrNames.reverse();

console.log(arrRev);

// 6. Access one of your parents’ names.

let resultPar = arrNames.find(parName => parName == "Edurne");

console.log(resultPar);

console.log(arrNames[2]);

// 7. Update the name of one of your parents

let newName = (arrNames[2] = "Nieves");

console.log(arrNames);

// another solution

mom = "Nieves";

arrNames[2] = mom;

console.log(arrNames);

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// FUNCTIONS

// place where i exec this line multiple times whenever i call the function

// function sum (){

// }

// everything that happens in the function stays in the function

// for example:

function sum(x, y) {

let result = x + y;

return result; // you use return to link the console.log to the function, whenever it is located outside of it {}

}

// function declaration and function call

let fun = sum(2, 4); // this is function call

console.log(fun);

function div(x, y) {

return x / y;

}

function age(x) {

if (x == 30) {

return "you are cool";

} else {

return "you are nice";

}

}

console.log(age(20)); // the function call will come with a result and i am consoling.log the result

// better not to name local and global variables with the same name

const numbers = [1, 2, 3, 4, 5, 6, 10, 45, 66];

const numbers2 = [9, 289, 13, 214, 5654, 6, 130, 455, 66444];

function arrBig(array) {

// whatever the parameter is is connecting the function with the declared arrays

let result = array.filter(num => num <= 500);

return result; // it's pushing the function to the outside world, without return everything stays in the function

}

console.log(arrBig(numbers));

console.log(arrBig(numbers2));

**// create a function picking the names with 4 characters in an array**

function fourArrChar(array) {

let result = [];

for (let i = 0; i < array.length; i++) {

if (array[i].length == 4) {

result.push(array[i]);

}

}

return result;

}

const names2 = ["Ali", "Hadi", "Nancy", "Olga"];

console.log(fourArrChar(names2));

**// create a function grabbing the names that starts with the letter "o"**

function Ochr(array) {

let result = [];

for (let i = 0; i < array.length; i++) {

if (array[i][0] == "O") {

result.push(array[i]);

}

}

return result;

}

console.log(Ochr(names2));

const monthNames = [

"January",

"February",

"March",

"April",

"May",

"June",

"July",

"August",

"September",

"October",

"November",

"December"

];

function nameOfThatMonth(num) {

let result = "";

switch (num) {

case 1:

result = monthNames[0];

break;

case 2:

result = monthNames[1];

break;

case 3:

result = monthNames[2];

break;

case 4:

result = monthNames[3];

break;

case 5:

result = monthNames[4];

break;

case 6:

result = monthNames[5];

break;

case 7:

result = monthNames[6];

break;

case 8:

result = monthNames[7];

break;

case 9:

result = monthNames[8];

break;

case 10:

result = monthNames[9];

break;

case 11:

result = monthNames[10];

break;

case 12:

result = monthNames[11];

break;

default:

console.log("This is not a month");

}

return result;

}

console.log(nameOfThatMonth(2));

function monthOfMyBirthday(array) {

let result = [];

for (let i = 0; i < array.length; i++) {

if (array[i] == "J") {

result.push(array[i]);

}

}

return result;

}

console.log(monthOfMyBirthday(monthNames));

let monthOfMyBirthday2 = monthNames.find(name => name.length == 4);

console.log(monthOfMyBirthday2);

// hadis

function nameOfThatMonthEasyVersion(num) {

let result = "";

let newNum = num - 1;

if (num <= 1 && num <= 12) {

result = monthNames[newNum];

} else {

result = "sorry it is not a month";

}

return result;

}

console.log(nameOfThatMonthEasyVersion(3));

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// 1. Reverse the a whole array

let arrRev1 = ["one", "two", "three", "four"];

// function strToArr2(array) {

// let revStr2 = arrRev[i].split("");

// revStr2.reverse("");

// let revStr3 = revStr1.join("");

// return result;

// }

// console.log(strToArr2(arrRev1));

// hadis solution

function reverseNames(arr) {

let newArray = [];

let strToArr = "";

let str = "";

let revStr = "";

for (let i = 0; i < arr.length; i++) {

str = arr[i];

strToArr = str.split(""); // "" split by letter is teh meaning

strToArr.reverse();

revStr = strToArr.join("");

newArray.push(revStr);

}

return newArray;

}

console.log(names);

console.log(reverseNames(names));

// hadis professional solution:

function reverseNames(arr) {

let newArray = []; // this will create a new array // therefore the empty array

let str = ""; //

for (let i = 0; i < arr.length; i++) {

str = arr[i]

.split("") // this will live inside of str

.reverse() // this will live inside of str // important to not write semi-colons until the end

.join(""); // this will live inside of str

newArray.push(str);

}

return newArray;

}

console.log(arrRev1);

console.log(reverseNames(arrRev1));

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

resources to read:

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

reducer is a function calling another function

const array1 = [1, 2, 3, 4];

const reducer = (accumulator, currentValue) => accumulator + currentValue;

// 1 + 2 + 3 + 4

console.log(array1.reduce(reducer));

// expected output: 10

// 5 + 1 + 2 + 3 + 4

console.log(array1.reduce(reducer, 5));

// expected output: 15

it is fine to have the function in side of the variable

------------------------------------------------------------

**22.01.2020**

**JAVA SCRIPT**

**FUNCTIONS**

// ES6 system uses => instead of the function word. it is assigning the function result to a variable. var sum = function(z,s){}

const sum = (x, y) => {

return x + y;

};

sum(5, 4);

// Array.map(value, index =>{} // creates a new array

// its not about naming system but about the order

const names = ["Ali", "Olga", "Nancy", "Hadi"];

const newArr = names.map((name, index) => name[0] == "0");

console.log(newArr);

// below the updated ES6 system:

// its a call back situation

names.forEach((name, index) => console.log(name + " index num = " + index));

//old system below:

for (let i = 0; i < names.length; i++) {

console.log(names[i] + " index num = " + i);

}

// Array.reduce()

// const reducer = (acc, cur) => acc + value;

// its also a call back function

const numArr = [1, 3, 4, 6, 7];

const reducer = (acc, cur) => acc + cur; // acc (accumulator) + cur (current value) // acc = 0 and cur = 1 in the first round

console.log(numArr.reduce(reducer));

console.log(numArr.reduce(reducer, 100)); // syntax of ES6, its a number to start in the accumulator

// call stack

// 6000 max. of functions inside functions

function first(callback) {

return callback;

}

function callback(callTwo) {

let x = 9;

return callTwo;

}

function callTwo() {

return "Hi";

}

console.log(first(callback(callTwo()))); // up until 6000 functions

// our bff

function display(str) {

console.log(str);

}

//turns into...

const print = str => console.log(str);

print("hadi");

// function <name>(<parameter>){...}

function yellow(val, val2) {

const x = 100;

const y = 200;

console.log(x / y + val + val);

}

yellow(2, 3); // you call the function follow by the executing situation

// parameters are values used inside of the function, value comes when i apply my function call

// when i have two parameters i separate them with a comma

function pink(name, age) {

console.log(`Hi ${name}, you are young ${age}`);

}

pink("Ali", 20);

yellow(2, 3);

const AliPets = ["Dyson", "Sofi"];

function pink(name, array, age = 10) {

// in case someone will not enter age the default value will be 10, default value at the end

console.log(`Hi ${name}, you are young ${age}, you hace nice pets ${array}`);

}

pink("Ali", AliPets, 20);

pink("Ali", 20, ["Dyson", "Sofi"]); // very important to follow the order and not skip any value

// ES6 system would be

const pink2 = (name, array, age = 10) => {

console.log(`Hi ${name}, you are young ${age}, you hace nice pets ${array}`);

};

pink2("Ali", AliPets, 20);

pink2("Ali", 20, ["Dyson", "Sofi"]); // very important to follow the order and not skip any value

const greeting = (nam, add, age, pla) => {

// not able to change the variable anymore

age += 20; // this is implemented into the initial value, it would to the existing value, in this case 20 + 10add up

nam += ",";

pla += " DE";

add += "";

return `Hey ${nam} it is very nice to live in ${add} you are getting older too fast ${age}, in your old ${pla}`; // return will give me the result

};

console.log(greeting("Ali", "xx str", 10, "Berlin")); // this is the initial value, the function call

// create a function without parameter and without return:

const add = () => {

let x = 1120 \* 3;

console.log(x);

};

add(); // this is the function call, without it is not executing at all // functions, array and objects always needs const

// variables in function

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//EXERCISES

// 1. Amplify the Multiples of 4. Create a function that takes an integer and returns an array of integers ascending from 1 to the given number, where:

// --> For each number in the array that can be evenly divided by 4, that number is amplified by 10 (i.e. return 10 times the number).

// --> If the number cannot be divided evenly by 4, simply return the number.

// --> The given integer will always be equal to or greater than 1.

// --> Include the given number (the number in the parameters).

// Examples:

// --> amplify(4) ➞ [1, 2, 3, 40]

// --> amplify(3) ➞ [1, 2, 3]

// --> amplify(25) ➞ [1, 2, 3, 40, 5, 6, 7, 80, 9, 10, 11, 120, 13, 14, 15, 160, 17, 18, 19, 200, 21, 22, 23, 240, 25]

function amp(arr) {

let result2 = [];

let mulBy4 = result2 \* 10;

for (let i = 0; i < arr.length; i++) {

if (amp[i] % 4 == 0) {

result2 = mulBy4;

} else {

result2 = [i];

}

}

return result2;

}

console.log();

//---Hadi's solution-----------

function amp(num) {

let result = [];

let ampTen = 0;

for (let i = 1; i <= num; i++) {

if (i % 4 == 0) {

ampTen = i \* 10;

result.push(ampTen);

} else {

result.push(i);

}

return result;

}

}

console.log(amp(4)); // the value is given in the function call and this will generate the new array

// ----ES6 version-------------

const amp2 = num => {let result = [];

let ampTen = 0;

for (let i = 1; i <= num; i++){

if (i % 4 == 0) {

ampTen = i \* 10;

result.push(ampTen);

}else {

result.push(i);

}

return result;

}

console.log(amp2(4));

//---------------------------

// 2. One is not like the others... Create a function that takes an array of numbers and return the number that’s unique.

// Examples:

// --> unique([3, 3, 3, 7, 3, 3]) ➞ 7

// --> unique([0, 0, 0.77, 0, 0]) ➞ 0.77

// --> unique([0, 1, 1, 1, 1, 1, 1, 1]) ➞ 0

// let uniNum = [3, 4, 5, 3, 4, 5, 6, 7, 7, 7, 8, 8, 8, 9, 9, 9];

// let unlike = uniNum.map(uniNUm.find(num, index => num ^ uniNUm));

// console.log(unlike);

//------hadis solution--------------

function unique(arr){

let currentValue = 0;

let current = []; // we have 2 arrays at first ( current and remainder)

let remainder = []; // we have 2 arrays at first (current and remainder)

for (let i = 0; i < arr.length; i++) {

if(current.length == 0 || current[0] == arr[i]){

current.push(arr[i]);

currentValue ++; // currentValue = currentValue + 1;

} else if (current[0] !== arr[i]){

remainder.push(arr[i]);

}

}

if (currentValue === 1) {

return currentValue[0]

} else {

return remainder[0];

}

}

let arr = [1,1,1,1,1,4,1,1,1,1,1,1];

console.log(unique(arr));

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

let p = 0;

p++ // or ++p the result would be 1

let c = p;

console.log(c);

------------------------------------------------------------------------

**23.01.2020**

**JAVA SCRIPT**

**FUNCTIONS**

hah

//===============Recap of important javascript topics================

//===============var, let, const=====================================

var x = 0; // var is global and can be used with hoisting

let y = "3";

y = 3; // i can change the value of the variable after the semicolon

const q = 12; // is fixed, I can't change the variable after the semicolon

// sumInt; // we use camel case // no use special characters in naming the variables

//=================increment==========================================

let a = 0;

a++; // every time i write this, there will be 1 added a = a+1 or a += 1, both are exactly the same

let str = "Hi"; // it also works with strings

str += " how are you";

console.log(str);

//=============decrement===============================================

//let b--; // a will be decreased by 1 --> same as a= a -1 or a=-1

//===============functions <name>(<parameters>){...}=============================

function sum(x, y) {

// function - name - parameter // you can have no parameter // parameter is a value that is outside the function and is called by the function

let z = x + y;

return z; // you want to send the result outside of the local scope

}

console.log(sum(2, 3)); // without console.log i won't see anything // this is the function call, this will make the operation been executed

//====variables in function===============================================

function fun() {

const num = 10;

const num2 = 20;

console.log(num + num2);

}

fun(); // the function cannot call itself, you always need a function call

//------------------------------------

function fun2() {

const num3 = 10;

const num4 = 20;

return num3 + num4; // if you use return you have to return the values (not the function), or a backtick or something else, but not the function name

}

console.log(fun2()); // the function cannot call itself, you always need a function call

//------------------------------------

let string = "this is a long test";

console.log(string[0]);

console.log(string.substring(0, 4));

let newStr = string[0].toUpperCase() + string.substring(1).toLowerCase();

let newStr2 =

string.substring(0, 2).toUpperCase() + string.substring(1).toLowerCase();

console.log(newStr);

console.log(newStr2);

//-------------------------------------

const arr = [1, 2, 3, 4, 5, 6, 7];

console.log(typeof arr); // we will be able to see what type of element we are dealing with

//-------------------------------------

let blue = true;

console.log(blue ? "Nice" : "Nooo"); // ternary operator, true or false

//-------------------------------------

if (1 < 10) {

arr.push(1);

} else if (1 == 20) {

arr.unshift(1);

} else {

arr.push(2);

a++;

}

//-------------------------------------

let color = "Black";

switch (color) {

case "pink":

arr.push(1);

break;

case "black":

console.log("hi");

break;

default:

// whatever other cases that were not mentioned

a++;

console.log("THis is not OK");

}

//=========== const <function name = (<parameters>) => {...}

const div = () => {

const you = () => {

return 3;

};

return you();

};

console.log(div());

//===================Arrays====================================

const names = ["Ali", "Olga", "Nancy"];

//---------Array.includes()-------------------------

console.log(names.includes("Ali")); // it wil check the entire array and say if its there or not (true or false)

//----------Array.indexOf(<item>)------------------

console.log(names.indexOf("Ali"));

//-------Add items------------------------------

//----Arr.unshift --> it would add an index to the beginning of the array

console.log(names.unshift("Zara"));

//-----arr.push --> it would add an index at the end of the array

console.log(names.push("Zara"));

//-------Remove items------------------------

//Array.pop --> it would remove the last index

console.log(names.pop()); // I don't have to add anything

//Array.shift --> it would remove the first item

console.log(names.shift()); // i have to add the index that I wanna add

//---------Array.slice()-------------------------

//----------Array.split()------------------------

const arr1 = ["Hi", "this", "is"];

const arr2 = ["me", "writing", "you"];

//---------Array.concat()-------------------------

const arrTotal = arr1.concat(arr2);

arrTotal.push("mothafucka");

console.log(arrTotal.join(""));

//-----------Array.filter()---------------------

// creates new array

console.log(names.filter(name => name.length == 3 || name.length == 4)); // this is function without name, anonymous function, I am only calling it inside the function

//-----------Array.find()----------------------

const numbers = [1, 2, 3, 4, 5, 6];

let arrCall = numbers.find(num => num <= 1);

console.log(arrCall);

//----------Reverse()--------------------------

const str3 = "javascript";

let newRevStr = str3 // you don't need a semicolon here

.split("")

.reverse()

.join("");

console.log(newRevStr);

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//==================EXERCISE=============================================

//------------Exercise 1-----------------------------------------

// c4n y0u r34d th15? Create a function that takes a string as an argument and returns a coded (h4ck3r 5p34k) version of the string.

// NB: for your program to work properly, the function should replace all ’a’s with 4, ’e’s with 3, ’i’s with 1, ’o’s with 0, and ’s’s with 5.

// Examples:

// hackerSpeak(“javascript is cool”) ➞ “j4v45cr1pt 15 c00l”

// hackerSpeak(“programming is fun”) ➞ “pr0gr4mm1ng 15 fun”

// hackerSpeak(“become a coder”) ➞ “b3c0m3 4 c0d3r”

//-----------solution--------------------------------------------

// 1. check that there are no upper cases

// 2. create empty array and empty value

// 3. then first split

// 4. create a loop

// 5. letters[i] = 4; --> condition after if (if a letter is == A, then the letter will be replaced by 4)

// 6. hackedArr.push(letters[i]); --> this is to push it into the new string

// 7. the do the same for every replacement

// 8. let result = hackedArr.join("") outside the brackets --> this will bring the letters together again

// 9. let codeStr = "Hi i am nice aaaa"; --> this is to implement the string that we wanna have replaced

// 10. console.log(hacking(codeStr)); --> this will print the first declared variable (hacking) including the declared string (codeStr)

const hacking = str => {

let hackedArr = []; // first start with empty array,

let toSmall = str.toLowerCase();

let letters = toSmall.split(""); // empty double quotes because i wanna separate every word with a space

for (let i = 0; i < letters.length; i++) {

//it will loop until the condition is falthy

if (letters[i] == "a") {

letters[i] = 4;

hackedArr.push(letters[i]);

} else if (letters[i] == "e") {

letters[i] = 3;

hackedArr.push(letters[i]);

} else if (letters[i] == "i") {

letters[i] = 1;

hackedArr.push(letters[i]);

} else if (letters[i] == "o") {

letters[i] = 0;

hackedArr.push(letters[i]);

} else if (letters[i] == "s") {

letters[i] = 5;

hackedArr.push(letters[i]);

} else {

hackedArr.push(letters[i]);

}

}

let result = hackedArr.join(""); // default setting, we will replace later

return result;

};

let codeStr = "Hi i am nice aaaa";

let codeStr2 = "javascript is cool";

let codeStr3 = "programming is fun";

console.log(hacking(codeStr));

console.log(hacking(codeStr2));

console.log(hacking(codeStr3));

//------------Exercise 2-----------------------------------------

// Is it Symmetrical? Create a function that takes a number as an argument and returns true or false

// depending on whether the number is symmetrical or not.

// NB(Nota Bene: used to mark something as particularly important): A number is symmetrical when it is the same as its reverse.

// Examples:

// isSymmetrical(7227) ➞ true

// isSymmetrical(12567) ➞ false

// isSymmetrical(44444444) ➞ true

// isSymmetrical(9939) ➞ false

// isSymmetrical(1112111) ➞ true

//------------------------solution--------------------------

// 1. turn the variable into a string

// 2. turn it into a string

function isSymmetrical(num) {

let numToStr = num.toString();

let result = numToStr

.split("")

.reverse()

.join("");

if (numToStr == result) {

return true;

} else {

return false;

}

}

console.log(isSymmetrical(1122332211));

//------------Exercise 3-----------------------------------------

// snake\_case ➞ camelCase Create a function toCamelCase() that takes a single string in snake\_case and converts it into camelCase.

// Examples:

// toCamelCase(“hello\_world”) ➞ “helloWorld”

// toCamelCase(“javascript\_is\_fun”) ➞ “javaScriptIsFun”

//-------------solution------------------------------------------

const toCamelCase = str => {

let splitStr = str.split("\_");

let camelArr = [];

for (let i = 0; i < splitStr.length; i++) {

let firstChr, restStr;

if (splitStr[i] == splitStr[0]) {

camelArr.push(splitStr[0]);

continue; // jumps to the loop again but skipping what is after the continue

} else {

firstChr = splitStr[i].slice(0, 1);

restStr = splitStr[i].slice(1, splitStr[i].length);

}

let newWord = firstChr.toUpperCase() + restStr;

camelArr.push(newWord);

}

let result = camelArr.join("");

return result;

};

console.log(toCamelCase("snake\_log\_hi"));

//-----------------Exercise 4------------------------------

// Pig Latin Translation. Create a function that takes a string of words and moves the first letter of each word to the end of it,

// then adds ‘ay’ to the end of the word. This is a basic form of “Pig Latin”.

// Move the first letter of each word to the end of the word.

// Add “ay” to the end of the word.

// Words starting with a vowel (A, E, I, O, U) append “way” to the end instead.

// Extra Practice

// Preserve proper capitalization as in the examples below.

// Examples:

// pigLatin(“Cats are great pets.“) ➞ “Atscay areway reatgay etspay.” // 1st letter goes to the end of the word and then + "ay" or "way" if word starts with vowel

// pigLatin(“Tom got a small piece of pie.“) ➞ “Omtay otgay away allsmay iecepay ofway iepay.”

// pigLatin(“He told us a very exciting tale.“) ➞ “Ehay oldtay usway away eryvay excitingway aletay.”

//-----------------Solution GA------------------------------

// const pigLat = str => {

// let firstStr = pigLat.split(" ");

// let secondStr = firstStr[i][0].slice;

// let vowStr = firstStr.join(" ");

// let consStr = secondStr.join(" ");

// let pigArr = [];

// let endWay = "way";

// let endAy = "ay";

// for (let i = 0; i < vowStr.length || constStr.length; i++) {

// if (firstStr[i][0] == "A") {

// vowStr.concat(endWay);

// } else if (firstStr[i][0] == "E") {

// vowStr.concat(endWay);

// } else if (firstStr[i][0] == "I") {

// vowStr.concat(endWay);

// } else if (firstStr[i][0] == "O") {

// vowStr.concat(endWay);

// } else if (firstStr[i][0] == "U") {

// vowStr.concat(endWay);

// } else {

// firstStr.push(0);

// consStr.concat(endAy);

// }

// }

// let pigWord = vowStr.concat(endWay) || consStr.push(0).thirdStr.concat(endAy);

// let result = pigArr.push(pigWord);

// return result;

// };

// console.log(pigArr.push(pigWord));

//---------------------------solution 2 GA-------------------------------------------

// array with vowel

// const pigLat2 = str => {

// let firstStr2 = pigLat.split(" ");

// let secondStr2 = firstStr[i][0].slice;

// let vowStr2 = firstStr.join(" ");

// let consStr2 = secondStr.join(" ");

// let pigArr2 = [];

// let vowArr = ["a", "e", "i", "o", "u"];

// let endWay2 = "way";

// let endAy2 = "ay";

// for (let i = 0; i < vowStr2.length || constStr2.length; i++) {

// if (firstStr2[i][0] == vowArr) {

// vowStr2.concat(endWay2);

// } else {

// firstStr2.push(0);

// consStr2.concat(endAy2);

// }

// }

// let pigWord2 =

// vowStr2.concat(endWay2) || consStr2.push(0).thirdStr2.concat(endAy);

// let result2 = pigArr2.push(pigWord2);

// return result2;

// };

// console.log(result2);

//-----------------Solution Hadi------------------------------

function pigLatin(str) {

let translation = [];

let lowerCase = str.toLowerCase();

let splitStr = lowerCase.split(" "); // empty space as a rule

const vowels = ["a", "e", "o", "i", "u"];

for (let i = 0; i < splitStr.length; i++) {

let firstChr = splitStr[i][0];

let remainder = splitStr[i].slice(1, splitStr[i].length); // this is splitting the first character

if (vowels.includes(firstChr.toLowerCase)) {

let firstChrVal =

(i ? firstChr : firstChr.toUpperCase()) + remainder + "way";

} else {

if (i == 0) {

let firstChrRan = remainder[0];

let wordRem = remainder.slice(1, remainder.length); //its taking the whole word except the 1st character

remainder = firstChrRan.toUpperCase() + firstChrRan;

}

const firstChrMain = remainder + firstChr + "ay";

translation.push(firstChrMain);

}

}

let result = translation.join(" ");

return result;

}

const strPig = "are is the way";

console.log(pigLatin(strPig));

-------------------------------------------------------------------------

270120

JAVA SCRIPT FUNCTIONS

→ always declare our steps when solving exercises

→ moreover if we dont understand them well-

--------------------------------------------------------------------------

06.02.2020

JAVA SCRIPT | ARRAYS

**// searching for an array method**

→ **indexOf(**element); // it will show me in which position is the element that i am looking for // if the element does not exist in the array it will return -1 (it means it couldn’t find it/falthy)

→ **find**(predicate) // define a test that determines ehat the right elemnt is / i will find the elements values

[3,4,5].find(e => e === 4)/ specifying what you are looking for (i want to find the element that is equal to 4). when it returns true then js knows that’s the element

→ **findIndex**(predicate)

[3,4,5].findIndex(e => e === 4) / it will give you the index nr, not the value of the element.

→ it will give you the first answer and stop looking

**// adding and removing elements**

→ **beginning of the array**

→ **shift** -> removes first element of the array

let arr = [3,4,5];

arr.shift() // outcome is 3

→ **unshift() →**

arr.unshift(1,2) // outcome 5 // it would add the new elements to the array / outcome is the amount of elements in the array

→ **end of array**

→ **pop()** → remove last element

arr.pop() // outcome 5 (the element that was removed

→ **push(e1,e2,...)** - add to end

arr.push(5,6,7) // outcome 7

new elements get added at the end

**// changing the entire array**

→ **reverse()**

arr.reverse()--> it will change the order of the array

**// iterating over the array**

→ **forEach**(callback)

→ callback arguments: value, index, array

→ arr.forEach(e => console.log(e))

→ the function decides what you want to do with the elements in the array. this function is calling a console.log with every one of the elements

→ [“bird”, “dog”, “cat”].forEach(animal => console.log(“My pet is:” animal)

→ **map**(callback) → returns new array

→ call back arguments: value, index, array

// if you want to transform the array into something else then you use map

→ arr.map(e=> e + 1) // this is not changing the array, it is generating a new one

→ **filter**()

→ // only if the test returns true it returns the element

→ arr.filter(e => e < 4)

→ arr.filter((value, index) => {console.log(value, “is an index”, index); return true;})

**// iterating over the array(2)**

→ **reduce((a,b) => a + b, 0);**

→ arr.reduce((result,current) => result + current, 0;

// it reduces the array to one single thing

// 0 is the initial value, it can be another number, not just 0

→ callback arguments:

→ accumulator (from previous iteration)

→ current value

→ index

→ array

→ arr.reduce((result,current) => {console.log(“sum so far”, result); return result + current;}, 0)

// exercises