HTML ,Show Different Images Depending on Browser Width

`<picture>` element allows you to define different images for different browser window sizes.

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_responsive_picture

coding.html

```
<picture>
    <source srcset="img_smallflower.jpg" media="(max-width: 600px)">
    <source srcset="img_flowers.jpg" media="(max-width: 1500px)">
    <source srcset="flowers.jpg">
    <img src="img_flowers.jpg" alt="Flowers" style="width:auto;">
</picture>
```

image sprites web app

An image sprite is a collection of images put into a single image.

A web page with many images can take a long time to load and generates multiple server requests.

Using image sprites will reduce the number of server requests and save bandwidth.

http://css.spritegen.com

---

css Grid fr

```
section {
    margin: 10vw auto;
    width: 50vw;
    display: grid;
    grid-template-rows: repeat(3, 10vw);

    grid-template-columns: repeat(5, 1fr);
      /*# 5*1fr = 5, so width/5=50/5=10vw, so here every 1fr=10vw */
    grid-column-gap: 1vw;
      /* firs-col + gab + col2 + gab + col3 + gab + col4 + gab + last-col = width=50vw*/

    grid-row-gap: 0.5vw;

    @for $i from 1 through 8 {
        &.div:nth-of-type(#{$i}) {
            @if $i == 1 { grid-area: 1 / 1 / span 3 / span 2 }
            @if $i == 1 { grid-area: 1 / 1 / span 3 / span 2 }
            @if null       { border: 3px double; }
        }
    }

}
```

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

All the commands start by git
Advantages :
- it is free
- create snapshot
- snapshot to snapshot
- local storing or online server (it is called “repository مستودع”, push and pull)
- update the code and save the old too

1- create the repository folder and make it as Home directory.

```
mhd@mhd-wahba:~$ cd Desktop/tester/
```

2- make it as local git repository.
git init

```
mhd@mhd-wahba:~/Desktop/tester$ git init
Initialized empty Git repository in /home/mhd/Desktop/tester/.git/
```

note: the git commands will apply only under this home directory (Git repository)
3- to be sure if it is Master
git status

```
mhd@mhd-wahba:~/Desktop/tester$ git status
On branch master
```

4-

```
mhd@mhd-wahba:~/Desktop/tester$ mkdir markup styles
```

5-

```
mhd@mhd-wahba:~/Desktop/tester$ cd markup/
mhd@mhd-wahba:~/Desktop/tester/markup$ touch index.html
```

6-

```
mhd@mhd-wahba:~/Desktop/tester$ git status
On branch master

No commits yet

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

        markup/

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

7-
git add --all
or
git add FileName

```
mhd@mhd-wahba:~/Desktop/tester$ git add --all
```

git add . adds all modified and new (untracked = not saved) files in the current directory and all subdirectories to the staging area (a.k.a. the index) that which files need to make the Snapshot , thus preparing them to be included in the next **git** commit .
8-

```
mhd@mhd-wahba:~/Desktop/tester$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   markup/index.html
```

9- create git acount

git config --global user.email best.pid@gmail.com

```
mhd@mhd-wahba:~/Desktop/tester$ git config --global user.email best.pid@gmail.com
```

10- create Snapshot
git commit -m 'initial commit from Mohammed'

```
mhd@mhd-wahba:~/Desktop/tester$ git commit -m 'initial commit from Mohammed'
[master (root-commit) ec70145] initial commit from Mohammed
 1 file changed, 12 insertions(+)
 create mode 100644 markup/index.html
```

```
mhd@mhd-wahba:~/Desktop/tester$ git status
On branch master
nothing to commit, working tree clean
```

11- to show all commits
git log ... more details
git log – -oneline -<number> .... short details.. *<number> is number of last commit*

12- to show what it is changed before to add
git diff

```
mhd@mhd-wahba:~/Desktop/tester$ git diff
diff --git a/markup/index.html b/markup/index.html
index c435a9c..d600757 100644
--- a/markup/index.html
+++ b/markup/index.html
@@ -9,6 +9,6 @@
 </head>
 <body>
     <h1>First git 2</h1>
-    <p>myyyyyy</p>
+    <p>myyyyyrryy</p>
 </body>
 </html>
\ No newline at end of file
mhd@mhd-wahba:~/Desktop/tester$
```

13- to backup from last snapshot
git revert HEAD *... not on Master , not after push*

14- to clone(download) file from github.com
git clone https://github.com/blindthief10/course.git

```
mhd@mhd-wahba:~/Desktop/tester$ git clone https://github.com/blindthief10/course.git
Cloning into 'course'...
remote: Counting objects: 186, done.
remote: Compressing objects: 100% (156/156), done.
remote: Total 186 (delta 31), reused 181 (delta 27), pack-reused 0
Receiving objects: 100% (186/186), 7.09 MiB | 2.36 MiB/s, done.
Resolving deltas: 100% (31/31), done.
mhd@mhd-wahba:~/Desktop/tester$
```

to delete the cloned(downloaded) file
rm -rf <file name> *..... m -rf course*

- to backup to certain snapshot .... will delete the previous commits

```
mhd@mhd-wahba:~/Desktop/tester$ git log --oneline
50c7909 (HEAD -> master) css
1b636a1 html
ba2f647 from ali
f25bb1b initial commit from Mohammed
ec70145 initial commit from Mohammed
mhd@mhd-wahba:~/Desktop/tester$
```

git reset –-hard <ID> *.... hard will delete also the last stage area*

git reset –-soft <ID> *.... soft will not delete also the last stage area*

Stage area : if done ‘git add - -all’ ,and not done ‘git commit -m ‘mhd’’

```
mhd@mhd-wahba:~/Desktop/tester$ git reset --hard ba2f647
HEAD is now at ba2f647 from ali
```

note: we never done this on master, but on only branches.

```
mhd@mhd-wahba:~/Desktop/tester$ git log --oneline
ba2f647 (HEAD -> master) from ali
f25bb1b initial commit from Mohammed
ec70145 initial commit from Mohammed
```

Note that it was deleted was before of the last ID reset.

- to backup to certain snapshot .... but without delete the previous commits

git checkout <ID>
git branch -b <new branch name>

- to make ssh connection setup between local git repository and remote git repository

step 1 - **ssh-keygen generation**

1- ssh-keygen

```
mhd@mhd-wahba:~/Desktop/tester$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/mhd/.ssh/id_rsa):
/home/mhd/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mhd/.ssh/id_rsa.
Your public key has been saved in /home/mhd/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:pHI0waTcG3Tt+V5qk/iL4aimVNiSaqc/WdjZ2vLzmMA mhd@mhd-wahba
The key's randomart image is:
+---[RSA 2048]----+
|     o+ ..       |
|   . +.o  .      |
|    o = .. .     |
|     = *  o      |
|    =o*oS  .     |
|   ..*+ .   . .  |
|  o ooEo  .o +   |
| . +o +.o=.o*    |
|  ..o+.+=o+ooo   |
+----[SHA256]-----+
```

three times : enter

2- cd ~/.ssh
ls -l

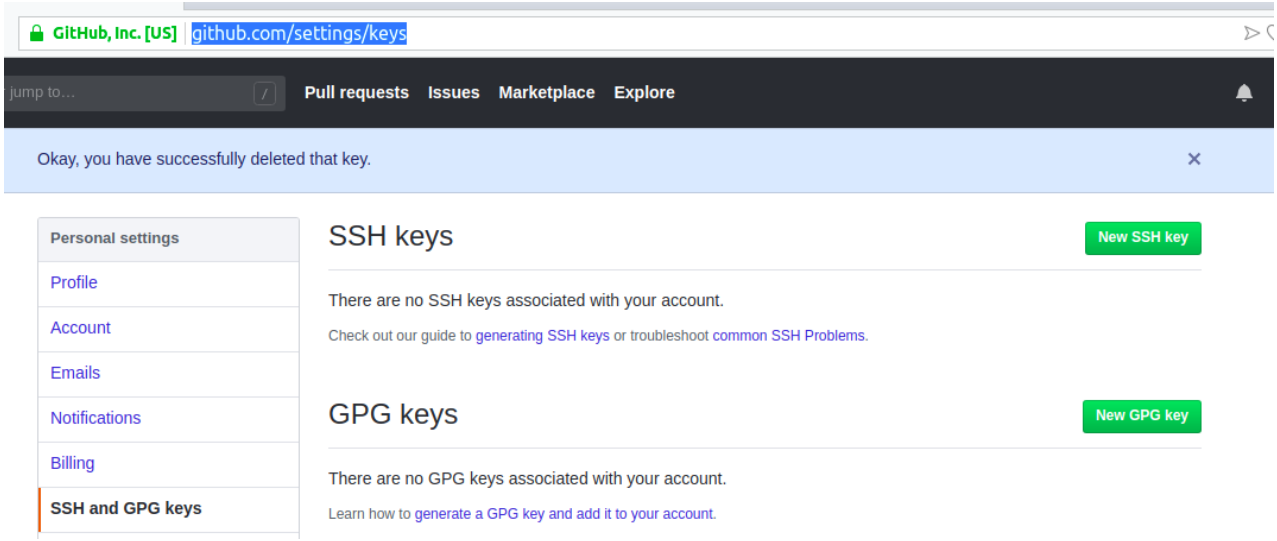
```
mhd@mhd-wahba:~/Desktop/tester$ cd ~/.ssh
mhd@mhd-wahba:~/.ssh$ ls -l
total 12
-rw------- 1 mhd mhd 1675 Aug 18 18:09 id_rsa
-rw-r--r-- 1 mhd mhd  395 Aug 18 18:09 id_rsa.pub
```

3- cat id_rsa.pub

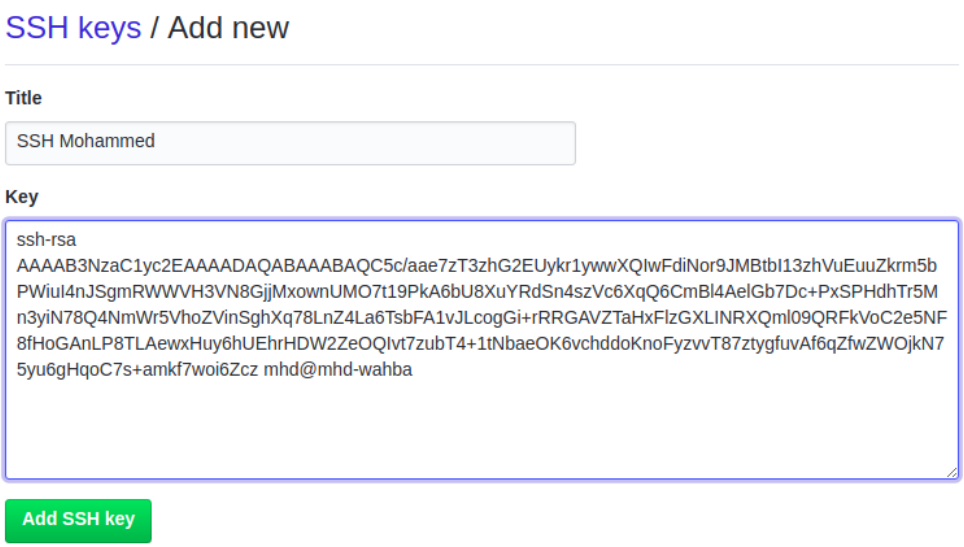
and then the copy this public key

```
mhd@mhd-wahba:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC5c/aae7zT
n4szVc6XqQ6CmBl4AelGb7Dc+PxSPHdhTr5Mn3yiN78Q4NmW
y6hUEhrHDW2ZeOQIvt7zubT4+1tNbaeOK6vchddoKnoFyzvv
```

4- go to *https://github.com/settings/keys*



then copy the public key

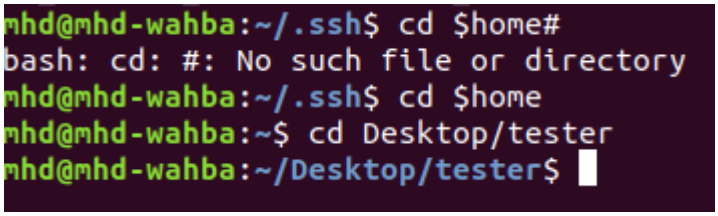


then enter your account's password

5- change the home directory to git repository

cd $home

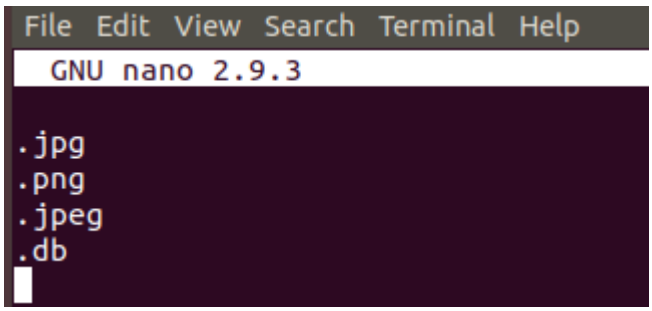
cd Desktop/tester



6- create .gitignore file to ignore file types (no push to online git repository)

touch .gitignore

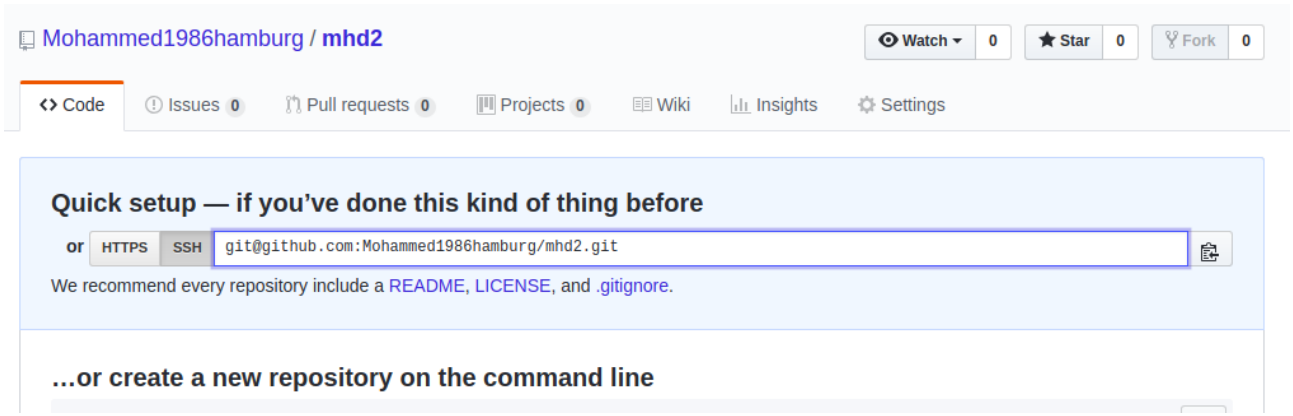
nano .gitignore



7- add and commit .gitignore file to local git repository

git add --all
git commit -m 'add .gitignore file'

8- copy the SSH the url of new online github repository



step 2- git remote add origin <url of new remote git repository>

- to show remote repository
git remote -v
- to remove remote repository
git remote remove <name>

step 3- git push --set-upstream origin master *.....* to create file and connection between remote origin repository and local mgit aster branch

- to push from local master to remote repository
git push

the summary to push:

git add –-all
git commit -m 'my message'
git push

- to create branch
git branch *...* to see the branches ,* means which branch I am using
git branch <name> *....* to add new branche
git checkout name *.....* swich to the name branch
git push --set-upstream origin name ..... to create file and connection between remote origin repository and local name branch

the summary to push:

git add –-all
git commit -m 'my message'
git push *.... or git push - –force* (never used in Master)

- to delete branch
git branch -d name

- to merge branch to master
- git checkout master
- git diff master name
-git merge name
-git push

- to Pull from online repository (Fetch+Merge with local)
git pull

or pull from different remote branch name
git pull origin <remote branch name>

* `git pull` is shorthand for `git fetch` followed by `git merge <branch>`

- to go back one step (redo)
git reset HEAD *.... ignore last change on repository*
git checkout - - <file name.__> *.....ignore last change on this file*

- to Marge a current commit with last commit
git commit - -amend *.... but it will generate new ID*

- to create new branch and switch to it
git checkout -b <new branch>

- to merge all remote branches together , to create the final file project

step 1 on my branch -to decide which the right code-
..... wait for anther branch to push.....
-git pull origin <my partner branch> *..... to download the remote files of <my partner branch> .. this local*
-git checkout <my name branch>
-git merge <my partner branch>
... check the codes and accept which it...
-git add - -all
-git commit -m 'message'
-git push

then every branch have to do the above

step 2 on my master- to merge to master
-git checkout master *....*
-git merge < *my name branch>*
-git add - -all
-git commit -m 'message'
-git push

- to see all the actions and commands

git reflog

- to undo the file if not done ‘git add - -all’ ,and not done ‘git commit -m ‘mhd’’

git checkout - - <file name>

- to undo the file if done ‘git add - -all’ ,and not done ‘git commit -m ‘mhd’’

git reset HEAD <file name>

git checkout - - <file name>

- *rebase* **to merge but keeps the commits of your current branch on top**

git rebase jake ..... rebase is merges , but Rebase brings all commits from the specified branch but keeps the commits of your branch on top of them.

- to go back to the commit with hash eb49uf2 safely without deleting all the in-between commits

Git checkout <ID>

- *cherry-pick* **To bring a specific commit from different branch. This commit will go on top of your commits.**

git cherry-pick <ID>

- *stash* **to Save temporarily untracked files without commit**

git stash .... Save temporarily untracked files without commit

git stash pop ..... Bring the stashed files back to life

-to unstage a file that i have accidentally staged

git reset HEAD <file name>

ex:

git reset HEAD jake.html ..... Unstage jake.html

You created a branch from master and you have been working on it tirelessly. You have already made some changes but you havn't staged them.

Suddenly a colleague of yours says to you that he has updated the master branch on remote and you must downloaded the latest changes.

How are you going to save your file changes on your current branch without performing a commit, in order to fetch the new code from master?

correct

POINTS: 1

You answered

Git stash

Incorrect answers

Git add --all

Git commit -m 'some changes'

Git push origin branch

Explanation

Git stash command gives you the opportunity to save unstaged and/or uncommited files temporarily and recover them later. Thus you can do something like checkout the master branch and fetch new code without having to lose or commit your changes. If you want to recover your changes back in your branch you type *git stash pop* and then you have your changes back to life!

POINTS: 1

You answered

Fill the previous commit and add new changes there by typing *Git commit --amend*

Correct answer

After the changes, make a new commit, then push.

Incorrect answer

Fill the previous commit and add new changes there by typing *Git commit --amend*

Explanation

Git commit --amend should never been used when the previous commit has been already pushed to master. That is because git commit --amend deleted the first of the two commits (which has been already pushed and become public) and created a newone (which nobody has now).

Question 7 of 16

A colleague of your has updated the master branch while you were working on your branch locally. You have already brought these changes to your local master and you want to have them to your local branch as well regardless of commit order.

What seems correct? Your branch's name is jake.

incorrect

POINTS: 1

You answered

1. Git checkout master 2. Git rebase jake

Correct answer

1. Git checkout jake
2. Git merge master

Incorrect answers

1. Git checkout master
2. Git merge jake

1. Git checkout master
2. Git rebase jake

1. Git checkout jake
2. Git rebase master

Explanation

Since we don't care about the outer order of commits after the merge we don't really need to do a rebase at this phase. And since we want to have master's latest changes INTO our branch we go to our branch and merge master into it.

ss Example

Css Example

css Example
Css Example

css Example
Css Example

css Example
Css Example 6 7 11 13 14 15

css Example
Css Example