

Introduction to GIT

jan.schulz@devugees.org

1. Agenda





1. What is GIT?
2. History
3. Collaboration
4. Feature Branches
5. Vocabulary
6. GitHub
7. Remote Repository Commands
8. Lets GIT our hands dirty ...

1. What is GIT?

- Version Control System (VCS)
- **GIT helps us manage our project's files**

1. What is GIT?

- Version Control System (VCS)
- **GIT helps us manage our project's files**

 index	15.10.2017 15:38	Chrome HTML Docu...	1 KB
 jquery-3.2.1.min	15.10.2017 15:35	JScript-Skriptdatei	85 KB
 main	15.10.2017 17:43	JScript-Skriptdatei	1 KB
 style	15.10.2017 15:35	Kaskadierendes Styl...	0 KB

1. What is GIT?

- What does GIT do, that makes managing our files easier?

1. History
2. Collaboration
3. Feature branches

2. History

- GIT keeps track on every change that we make on our files

1. History

Oct. 2017: We have a file **banners.css**

Dec.2017: We do some changes in **banners.css**

Jan.2018: Our page breaks for some reason!

We take a look at banners.css from Oct. 2017
and see, that we **removed** the lines

„ float: left;

padding: 2rem;

margin: 2rem; “

1. History

- GIT provides seeing the **history** of a file
- GIT provides **reverting** changes

1. History

- GIT provides seeing the **history** of a file
- GIT provides **reverting** changes

-> Nothing is ever lost

-> Nothing is ever final

2. Collaboration

- Creating something **alone**:
 - You
 - Your Files

2. Collaboration

- Creating something **alone**:
 - You
 - Your Files
- Creating something **in a team**:
 - You
 - Your Team members
 - Your Files
 - Your Team members' files

2. Collaboration

- Creating something **alone**:
—**EASY**
- Creating something **in a team**:
—**NOT SO EASY**

2. Collaboration

You want to write a book „mybook.docx“ about your home country with your friend.

2. Collaboration

You want to write a book „mybook.docx“ about your home country with your friend.

You: „Can you please check Chapter 4 and write something?“

Your buddy: „Okay, I need two days for that.“

2. Collaboration

Right after sending the Email to your friend ...

1. Now you see a few typos in your copy of the book and you fix them.
2. You have an idea of what images you would use for Chapter 1 and you insert them.

Now your friend does not work with the most updated version of your book anymore.

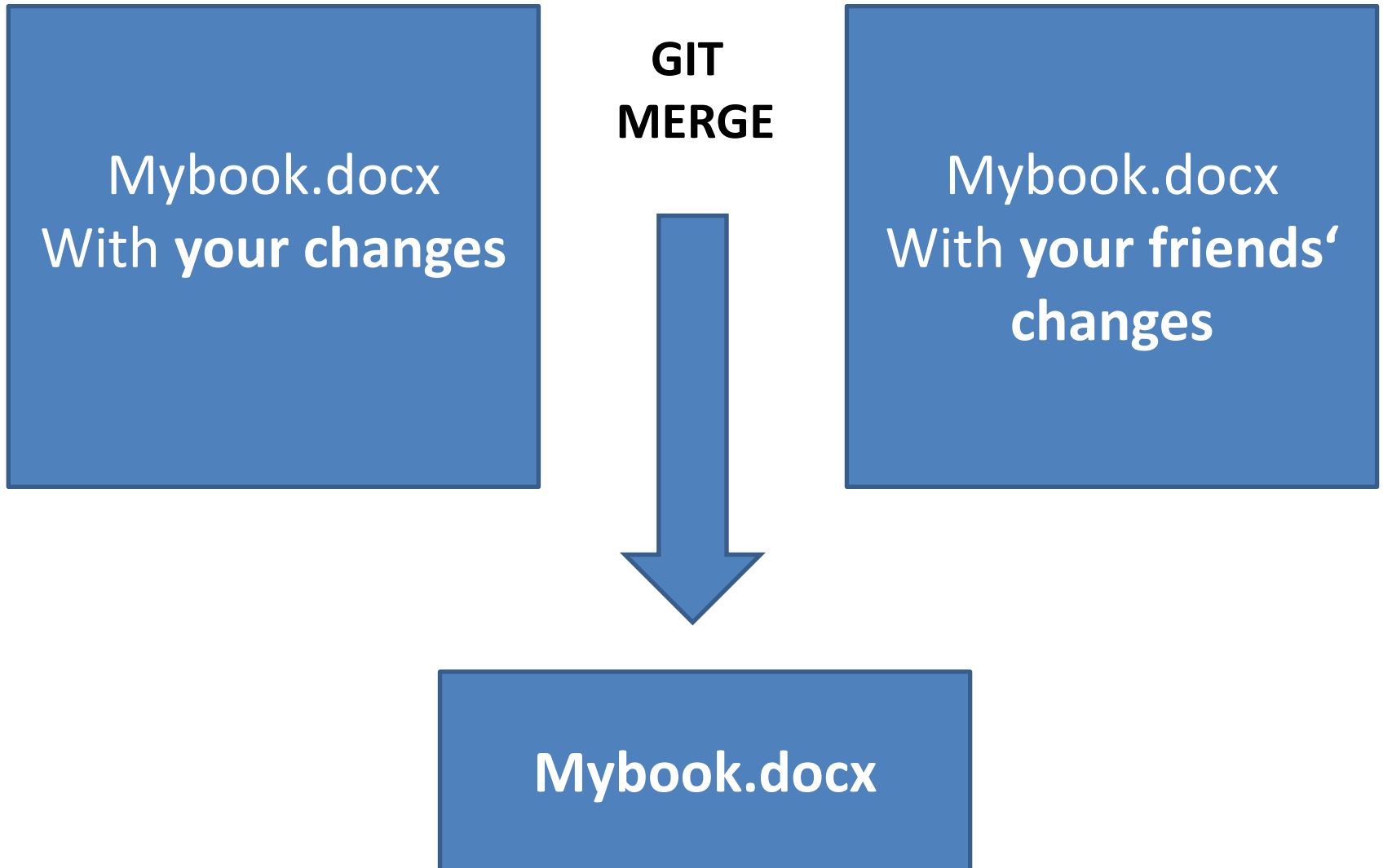
2. Collaboration

After two days your friend sends you back his copy.

Mybook.docx
With **your** changes

Mybook.docx
With **your friends'**
changes

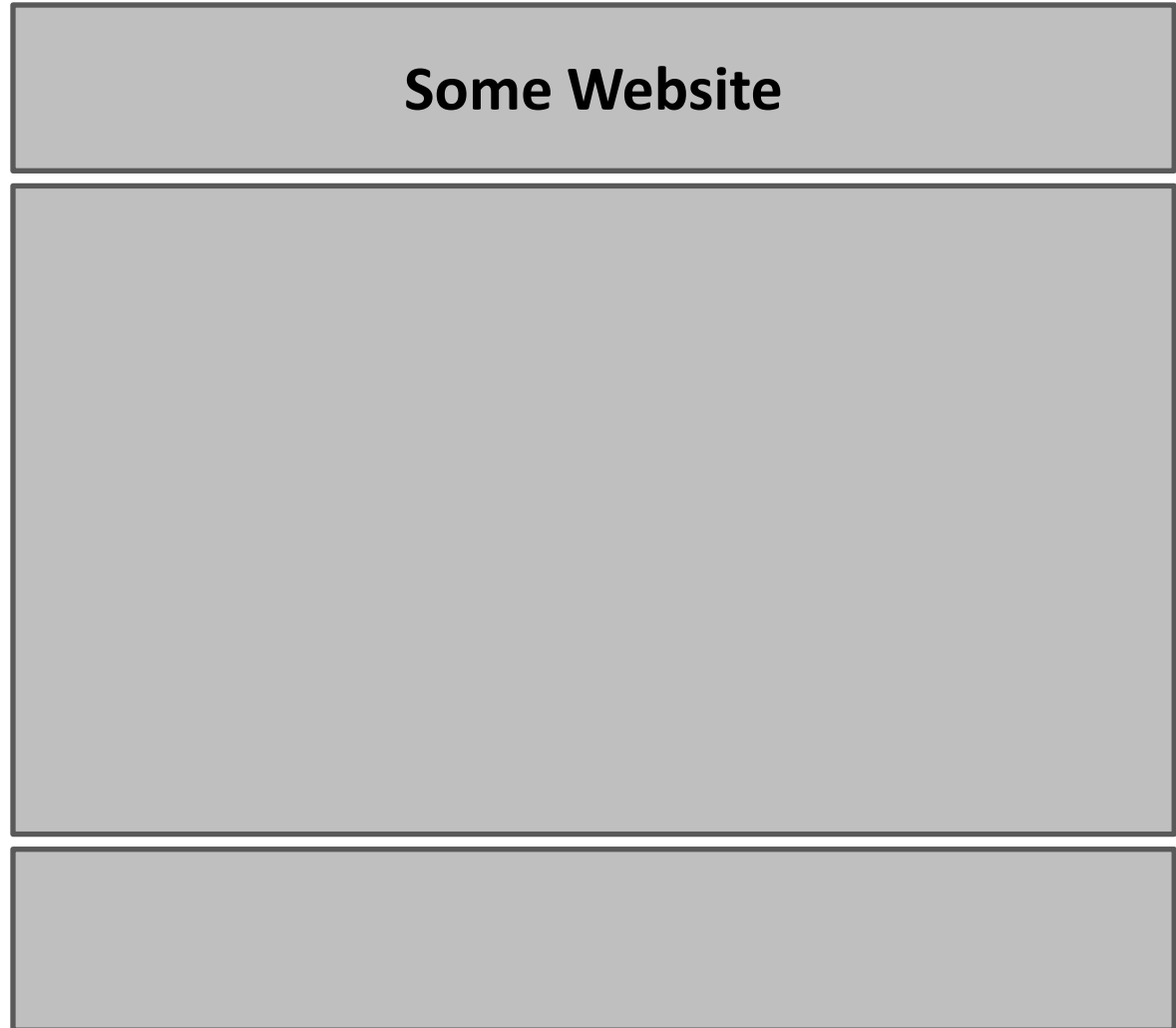
2. Collaboration



3. Feature Branches

Task #1

-> Redesign Header



3. Feature Branches

Task #1

-> Redesign Header

Some Website

Task #2

-> Redesign Footer

Day 1

Task #1

-> Redesign Header

UNDER CONSTRUCTION

Task #2

-> Redesign Footer

UNDER CONSTRUCTION

Day 2

Task #1

-> Redesign Header

UNDER CONSTRUCTION

Task #2

-> Redesign Footer

UNDER CONSTRUCTION

End of Day 2

Task #1

-> Redesign Header

UNDER CONSTRUCTION

Task #2

-> Redesign Footer

AMAZING FOOTER

End of Day 2

Task #1

-> Redesign Header

UNDER CONSTRUCTION

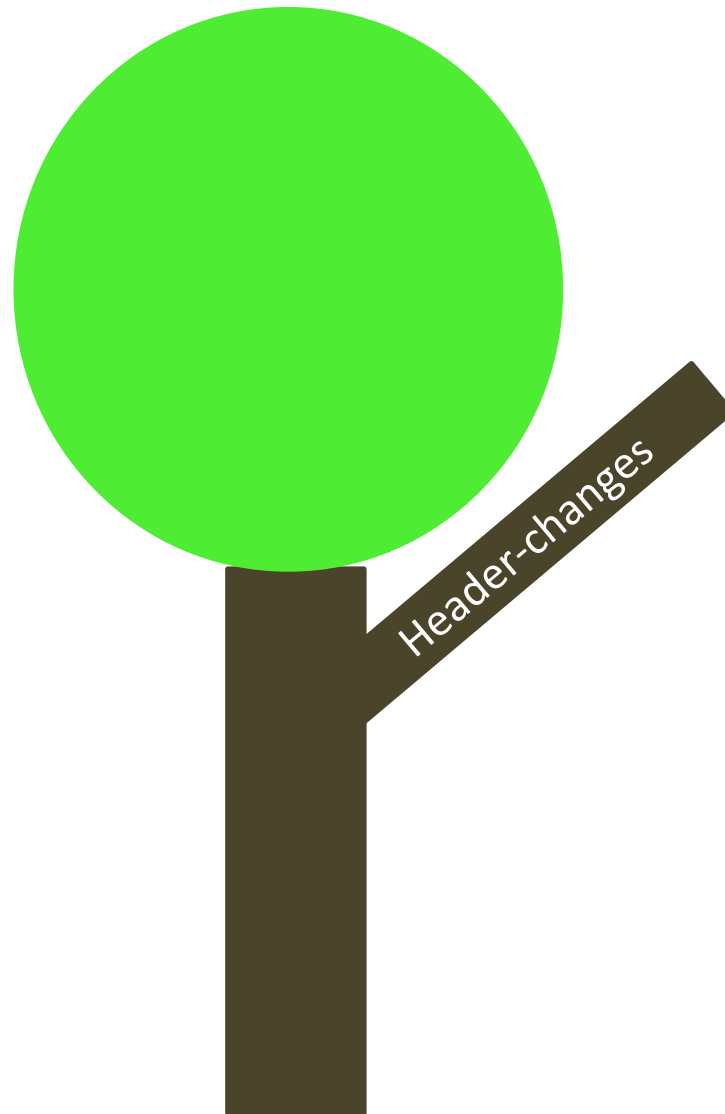
PROBLEM: UPLOADING WEBSITE WITH LUMPY HEADER CODE

Task #2

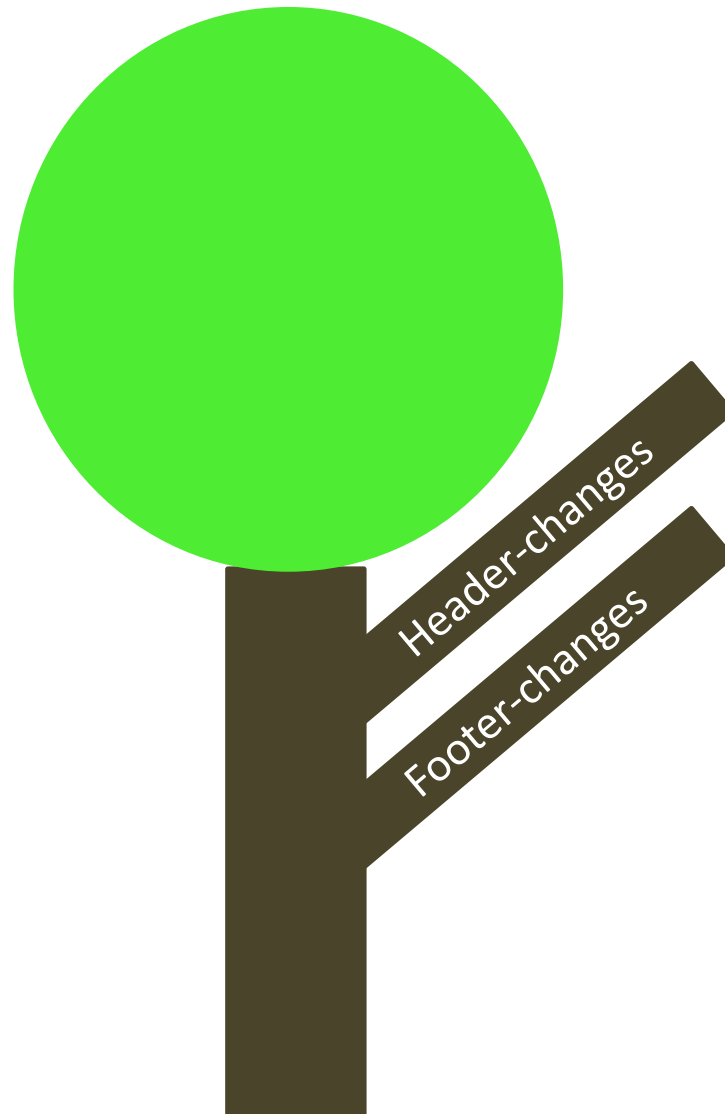
-> Redesign Footer

AMAZING FOOTER

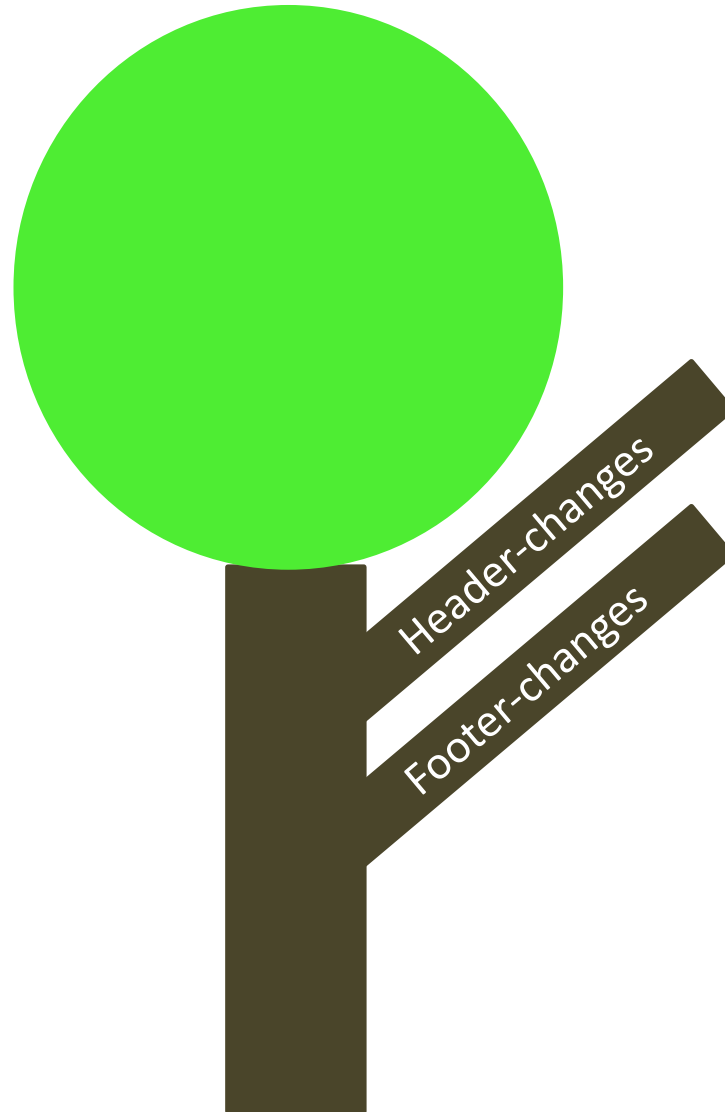
3. Feature Branches



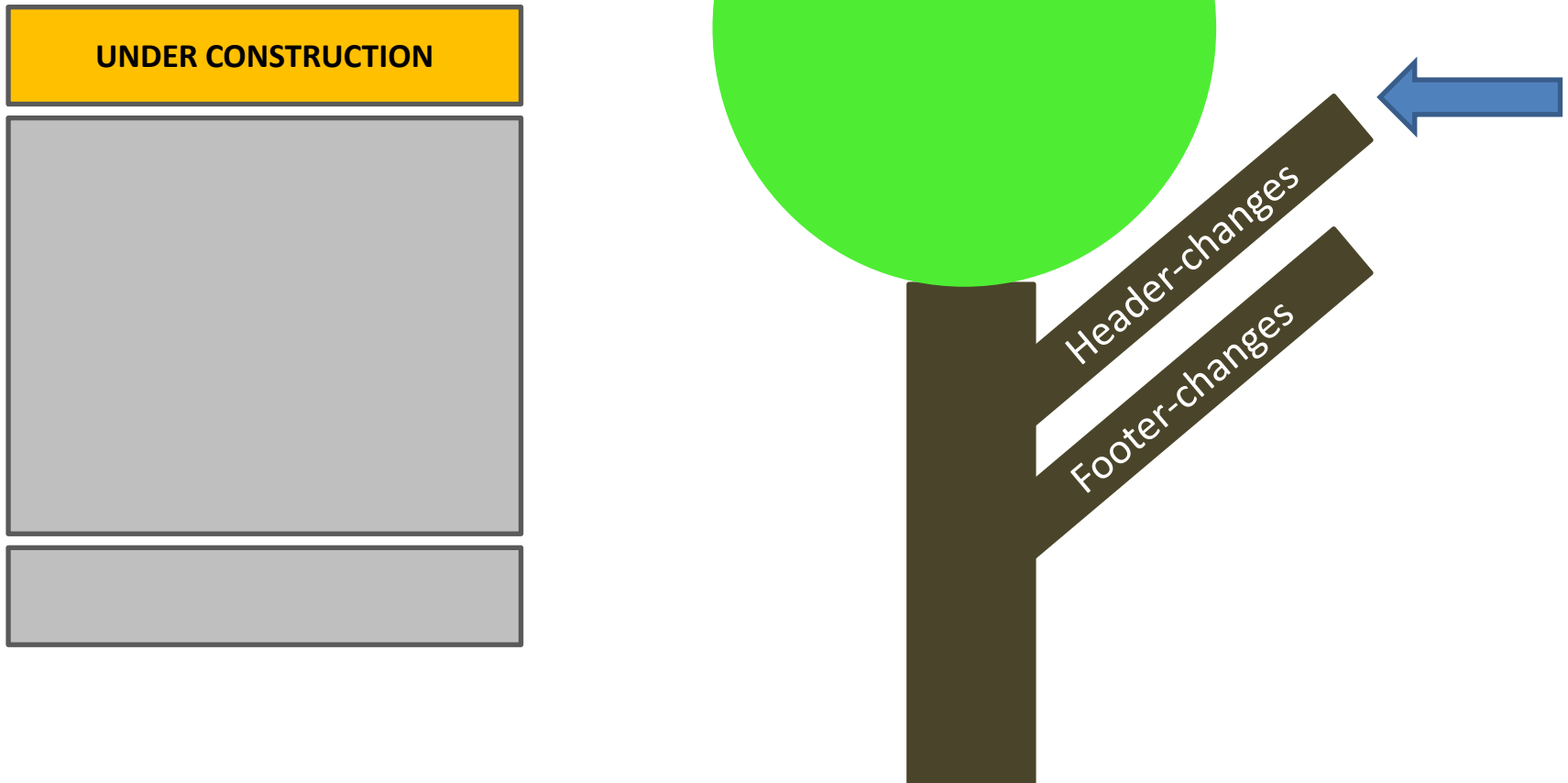
3. Feature Branches



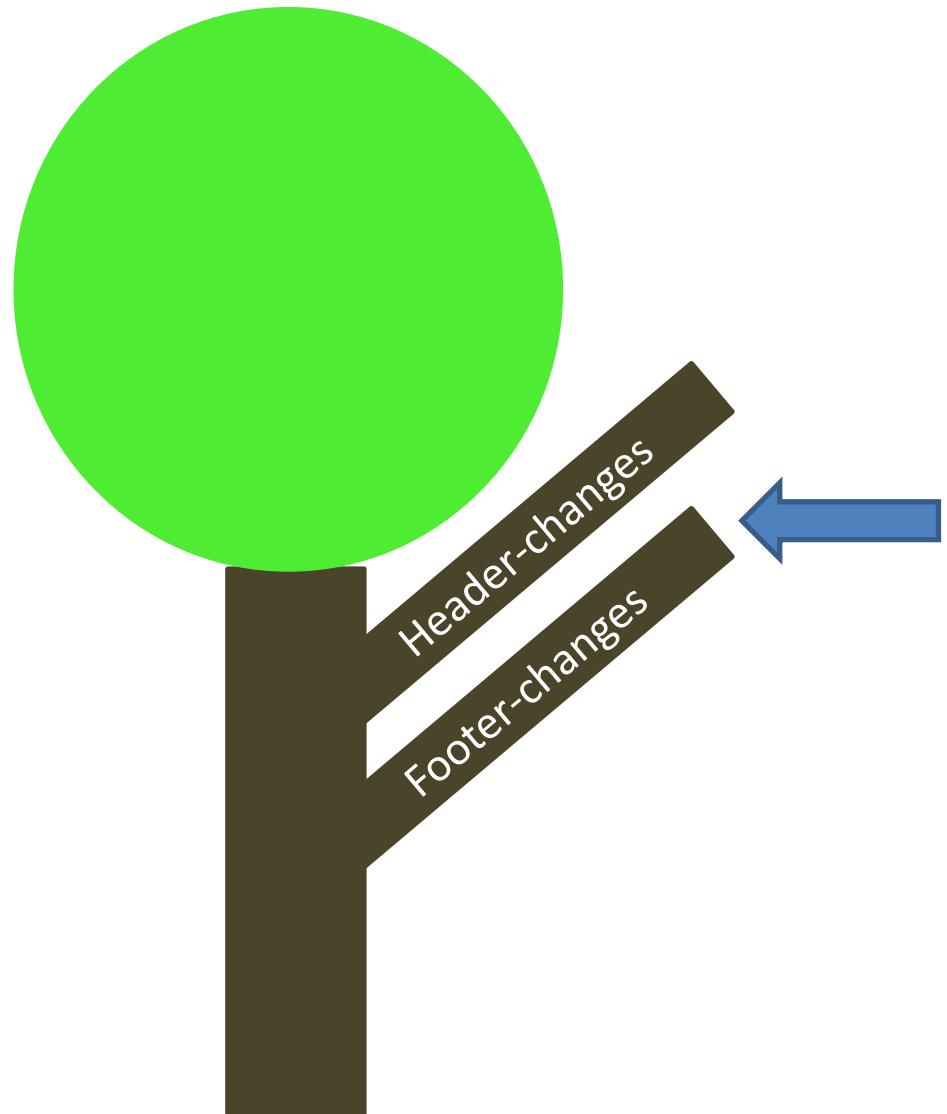
3. Feature Branches



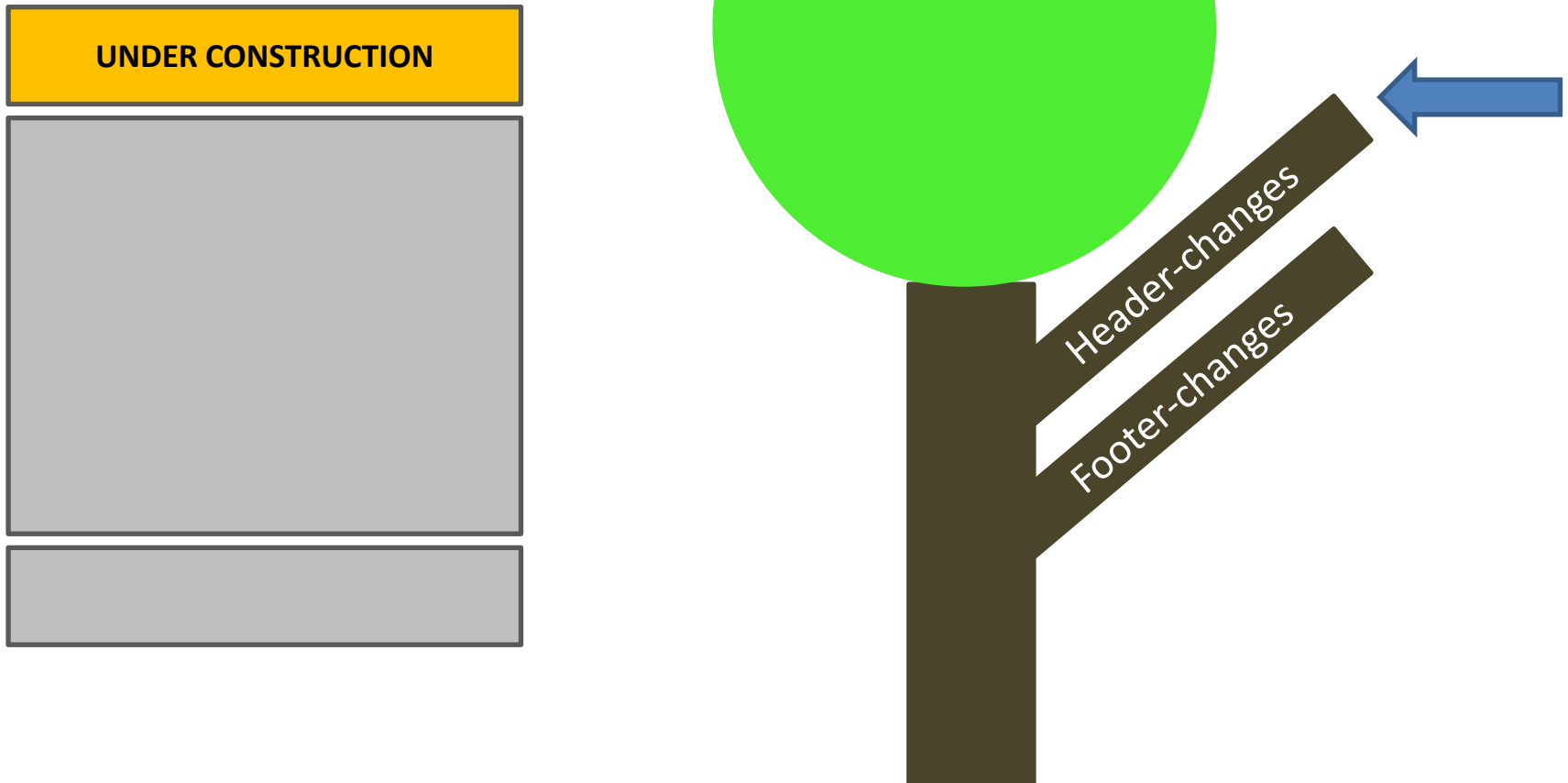
3. Feature Branches



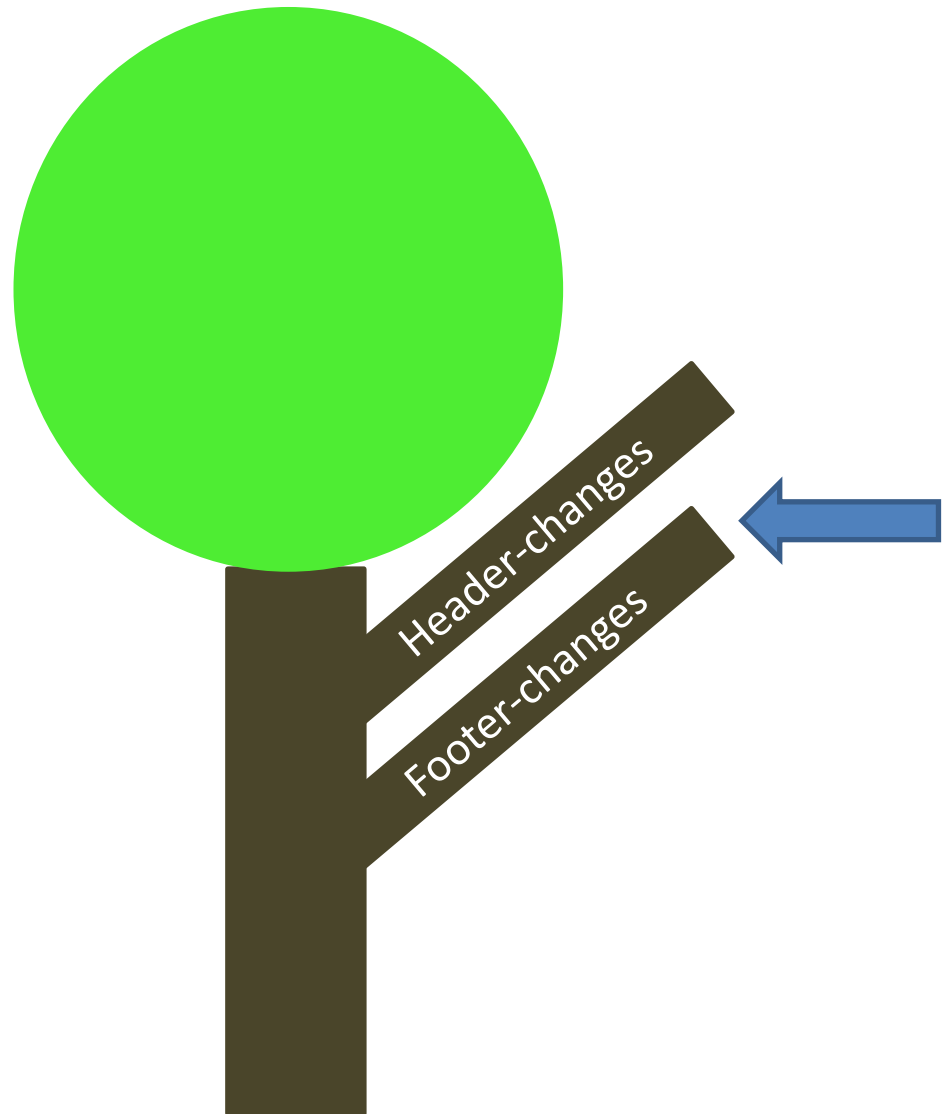
3. Feature Branches



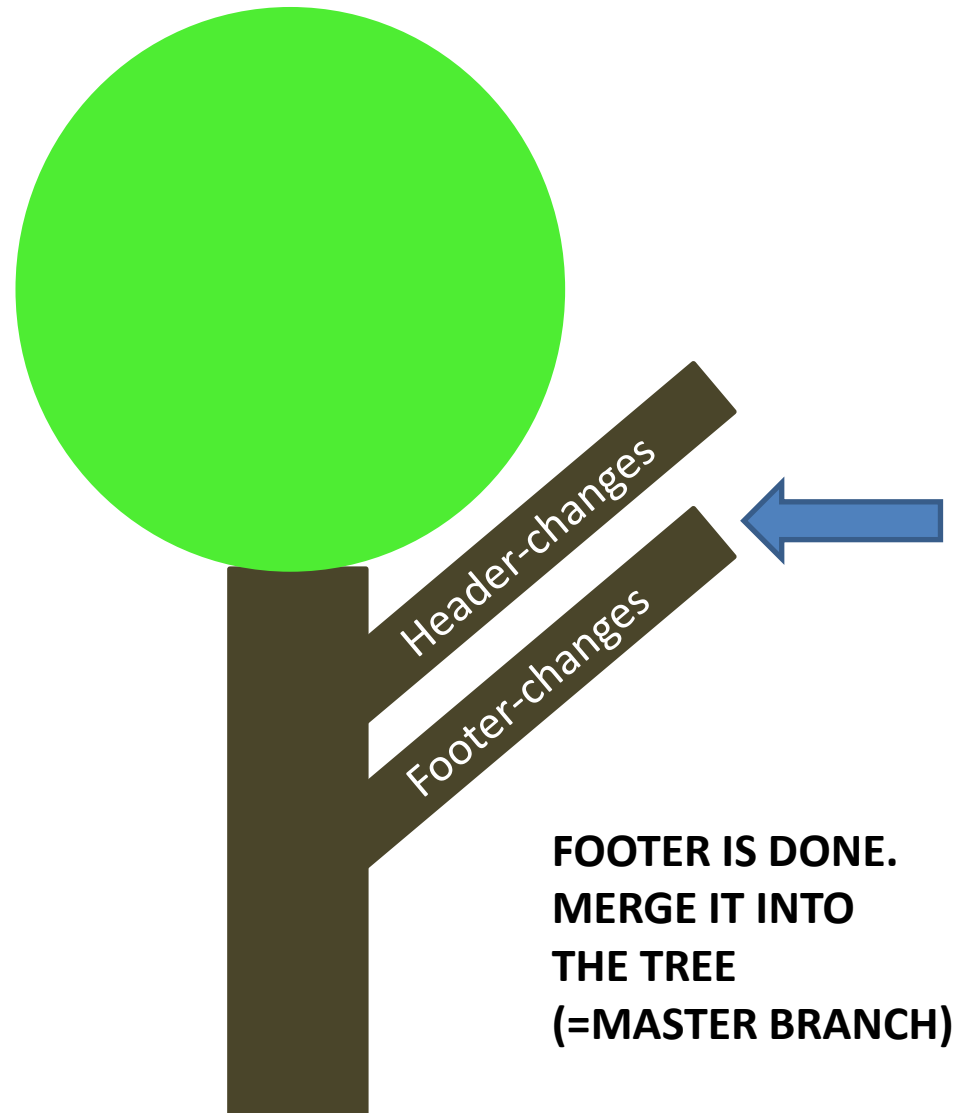
3. Feature Branches



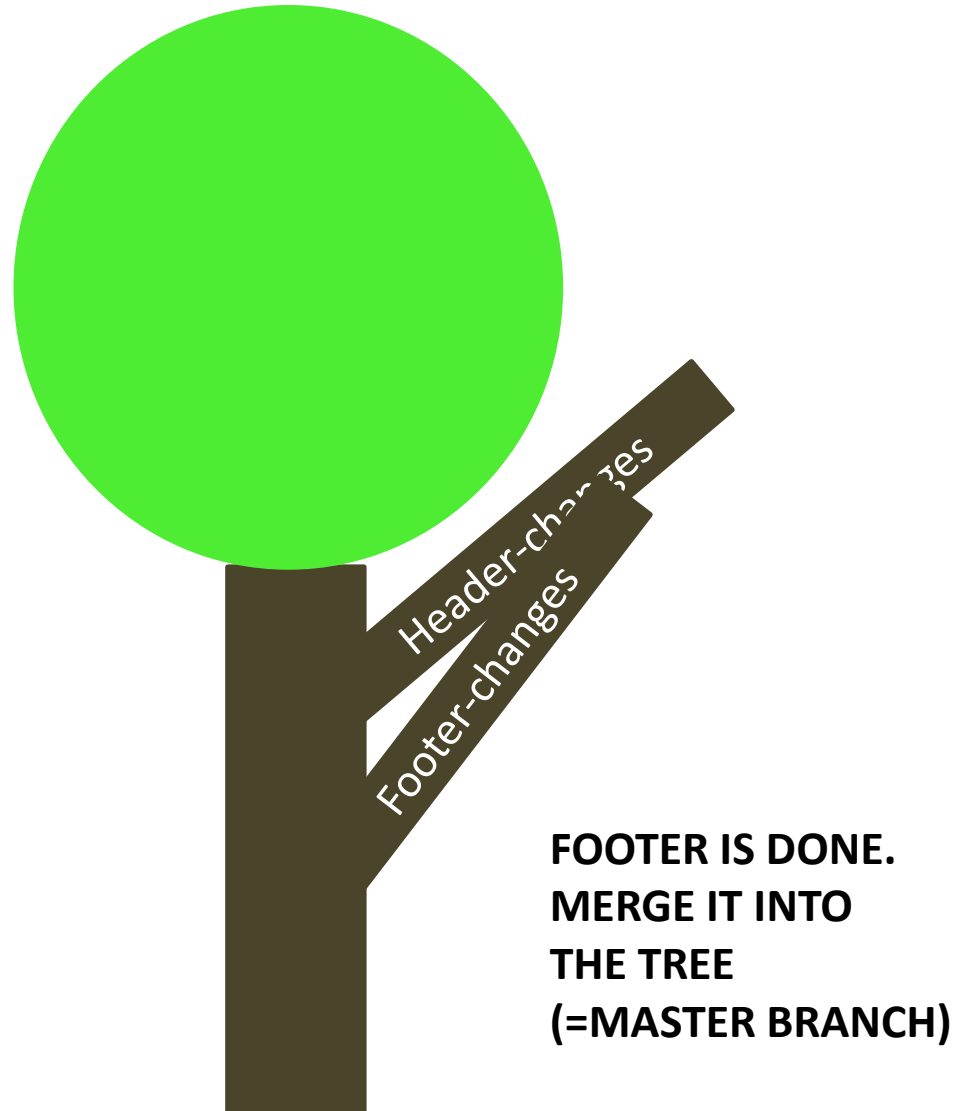
3. Feature Branches



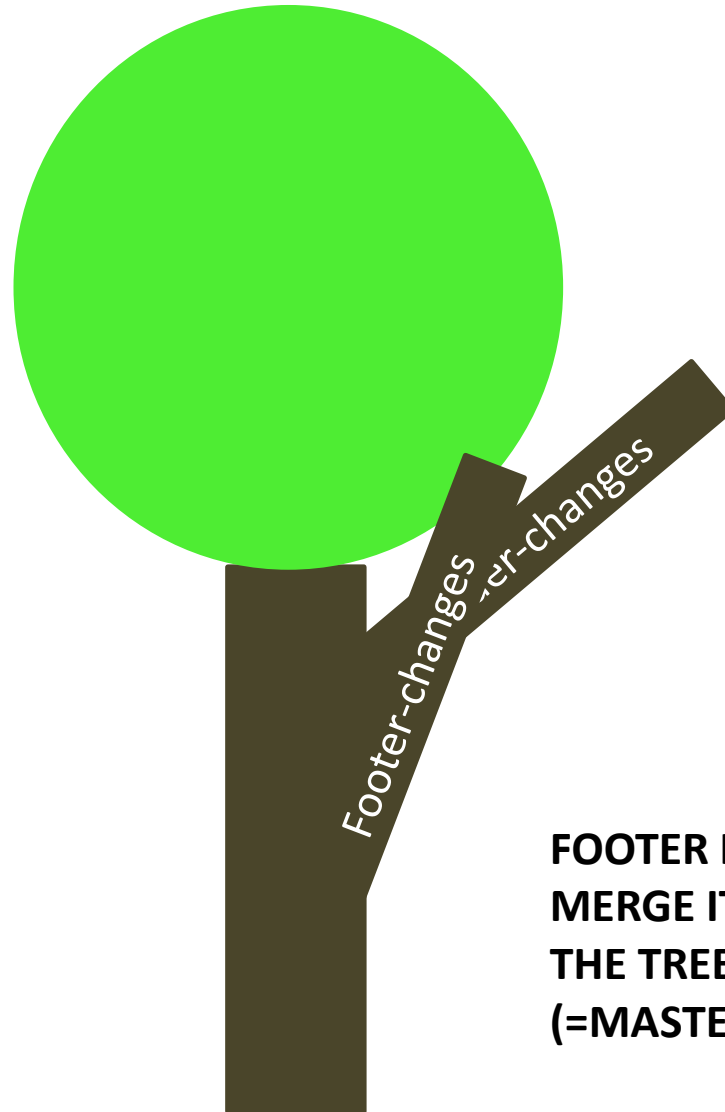
3. Feature Branches



3. Feature Branches

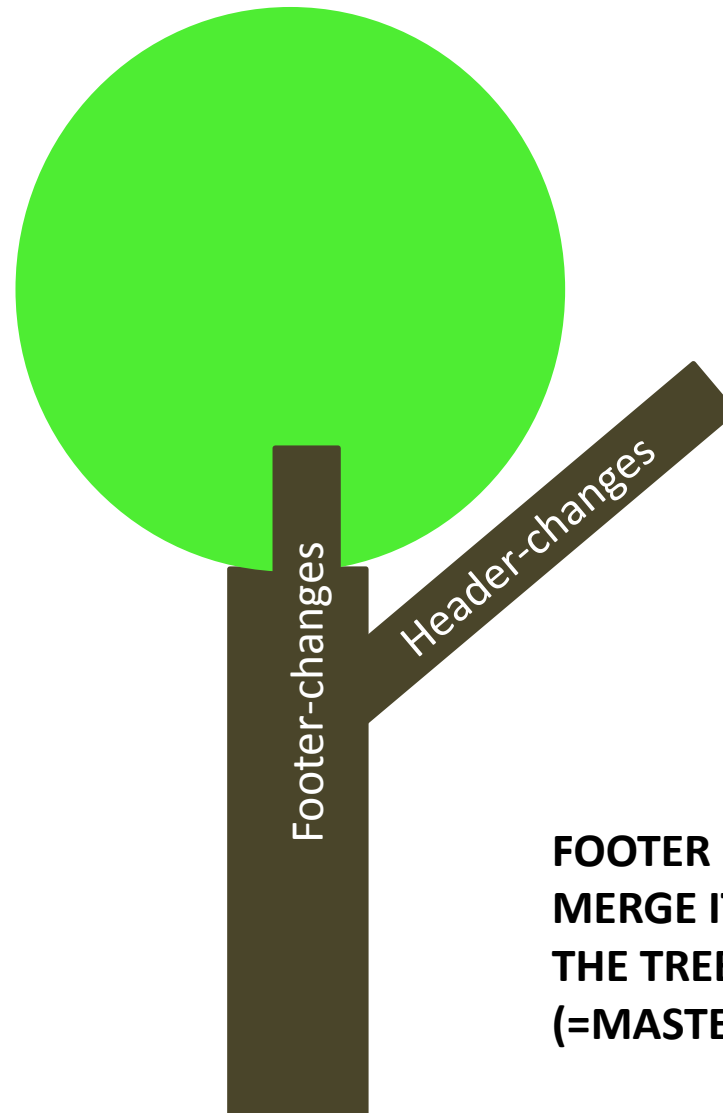


3. Feature Branches



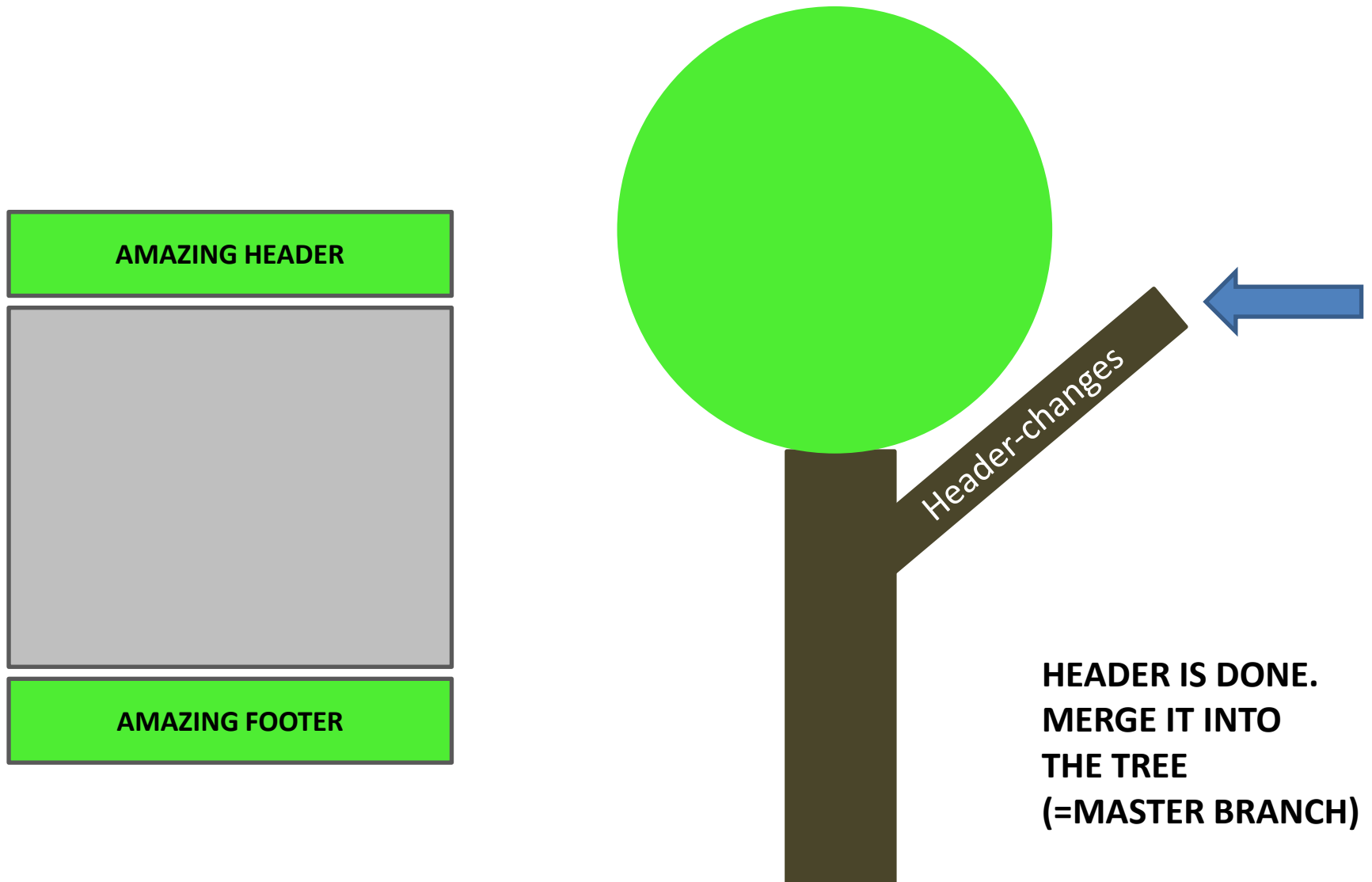
**FOOTER IS DONE.
MERGE IT INTO
THE TREE
(=MASTER BRANCH)**

3. Feature Branches

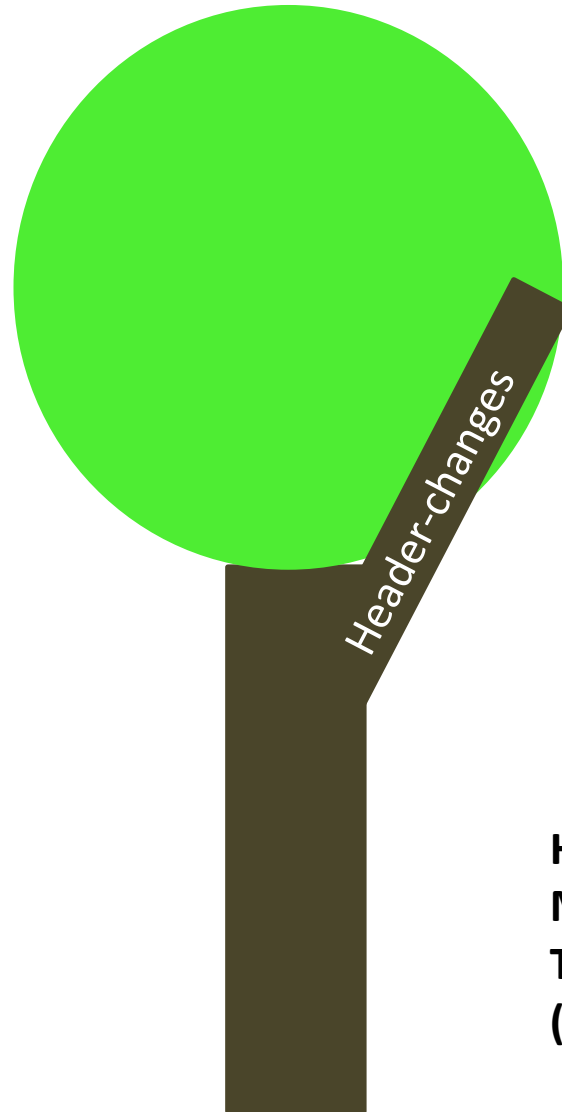


**FOOTER IS DONE.
MERGE IT INTO
THE TREE
(=MASTER BRANCH)**

3. Feature Branches

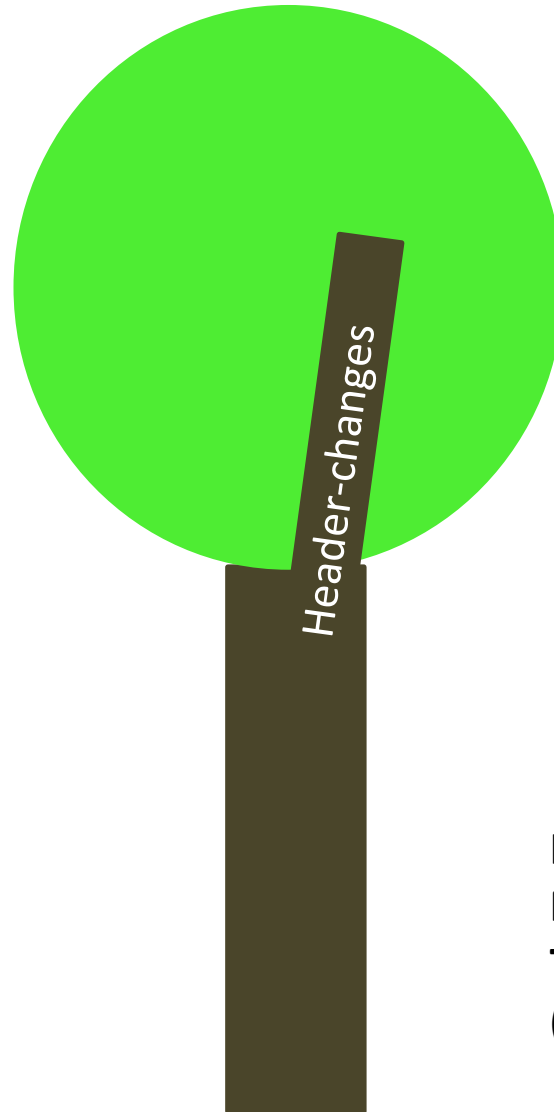


3. Feature Branches



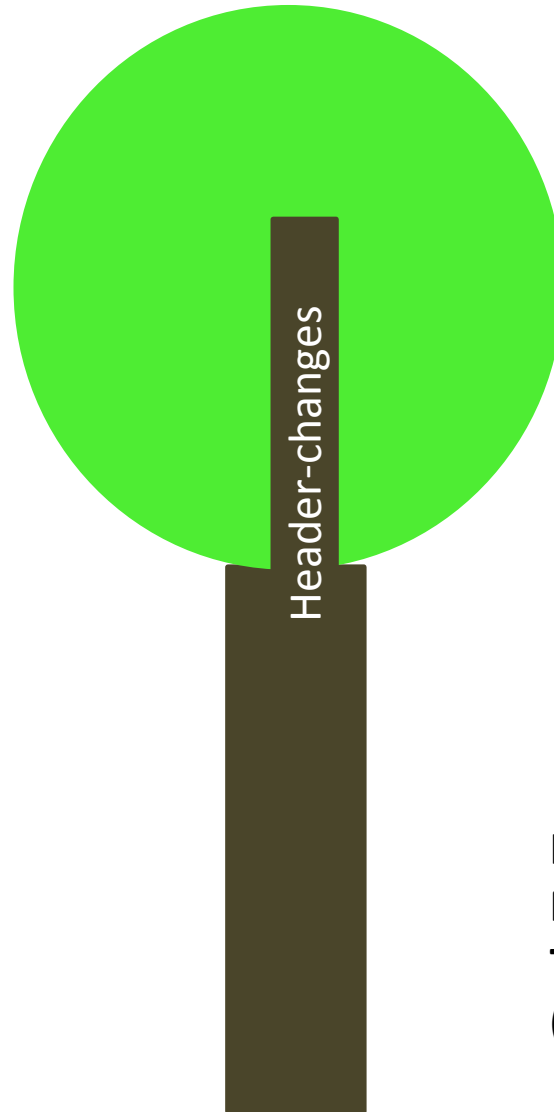
**HEADER IS DONE.
MERGE IT INTO
THE TREE
(=MASTER BRANCH)**

3. Feature Branches



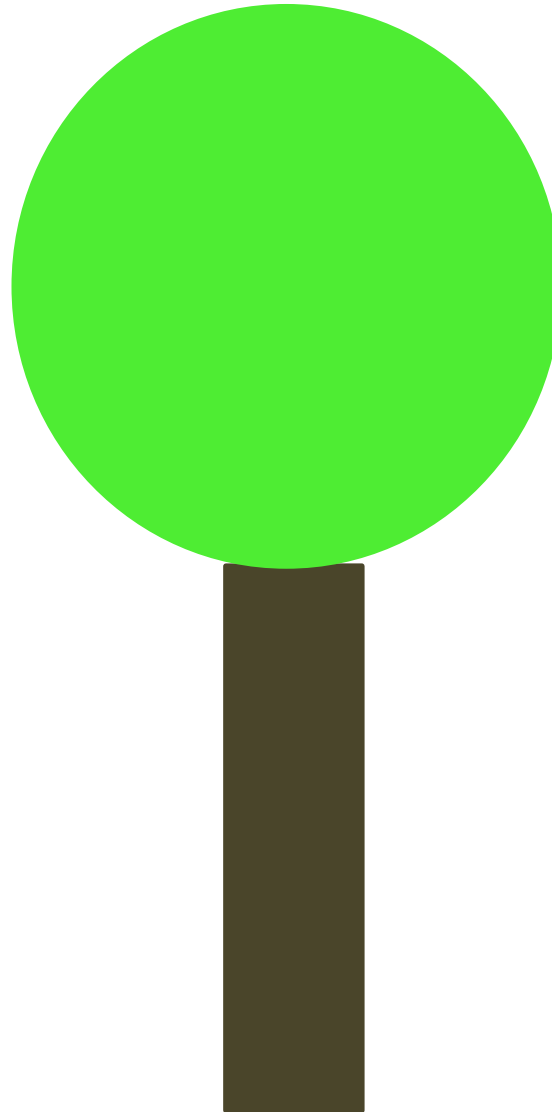
**HEADER IS DONE.
MERGE IT INTO
THE TREE
(=MASTER BRANCH)**

3. Feature Branches



**HEADER IS DONE.
MERGE IT INTO
THE TREE
(=MASTER BRANCH)**

3. Feature Branches



Review

- What are the three core functions of GIT?

Review

- What are the three core functions of GIT?
 - History
 - Collaboration
 - Feature Branches

4. Vocabulary Time-Out

- **Repository?**

4. Vocabulary Time-Out

- **Repository**
 - Working directory, your project files folder
 - GIT's job is to keep track of any changes here

4. Vocabulary Time-Out

- **Repository**
 - Working directory, your project files folder
 - GIT's job is to keep track of any changes here
- **Commit?**

4. Vocabulary Time-Out

- **Repository**

- Working directory, your project files folder
- GIT's job is to keep track of any changes here

- **Commit**

- GIT does not save changes in its history, until we actively commit those changes
= "**GIT's way of saving**"
- In a text-editor, we hit "Save" or CTRL+S and then save it.
- In GIT, nothing gets saved into history until we hit **COMMIT**

4. Vocabulary Time-Out

- **Before we commit ... we STAGE!**
- **STAGING** = we prepare something,
 - like if you want to sell a house,
 - first you have to prepare it,
 - make it nice and clean then you sell it.

Index.html – before staging



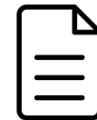
Index.html – staged and ready to commit



Git States

**Working
Directory**

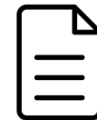
**Modified
Files**



Git States

**Working
Directory**

**Modified
Files**

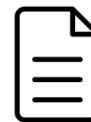
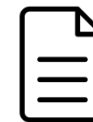
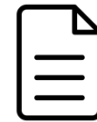


Git States

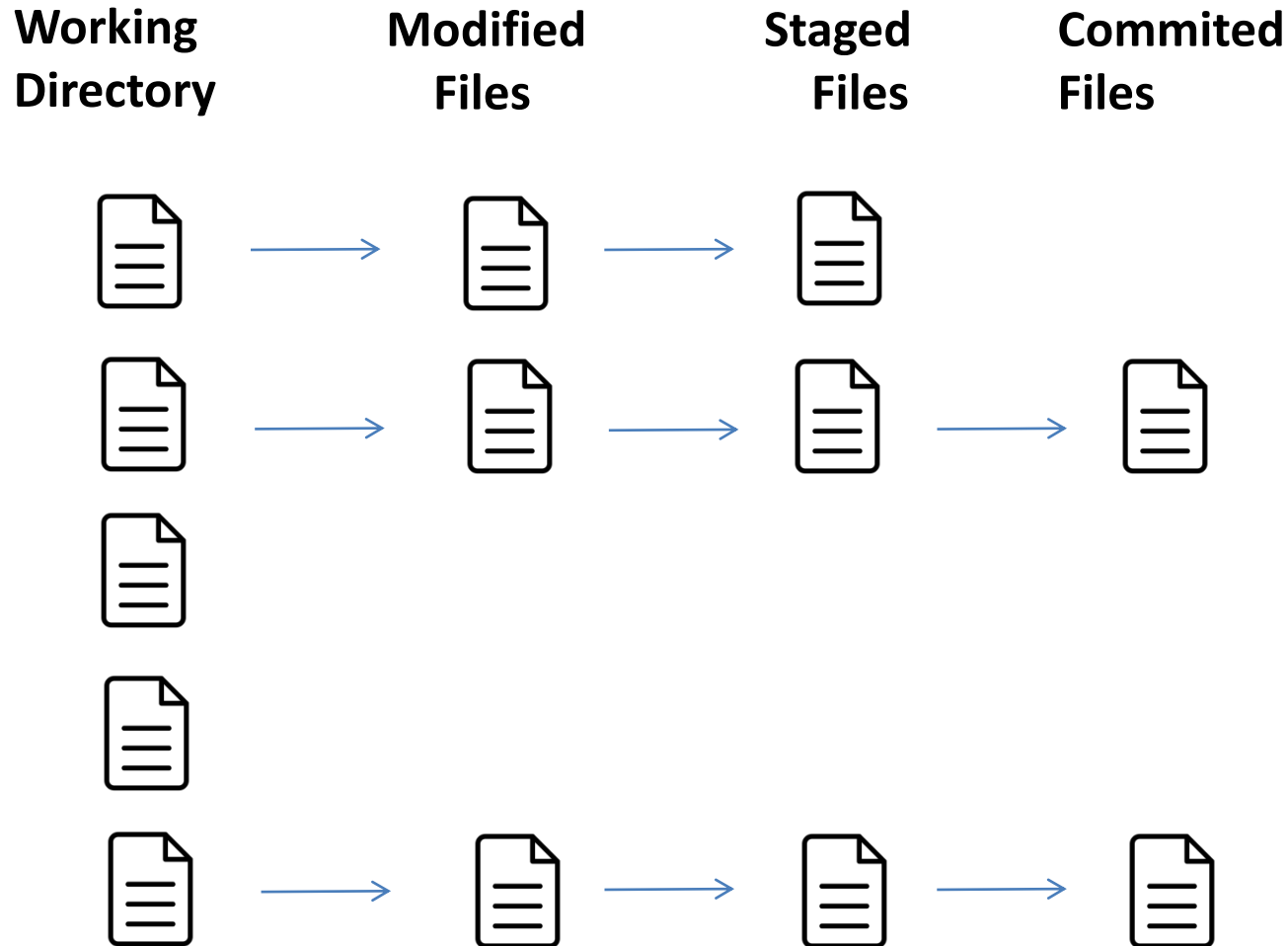
**Working
Directory**

**Modified
Files**

**Staged
Files**



Git States



Git States



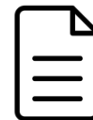
**Working
Directory**

**Modified
Files**

**Staged
Files**

**Committed
Files**

**Remote
Repository**



Git States

LOCAL

REMOTE

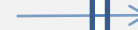
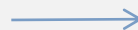
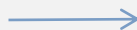
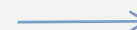
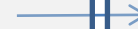
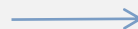
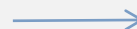
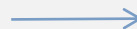
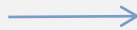
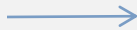
Working
Directory

Modified
Files

Staged
Files

Committed
Files

Remote
Repository



5. GitHub

- = Your Remote Repository on the web
- Free: Public Repositories
- Premium: Public + Private Repositories
- Public: Everybody can see your code
- Private: You decide who can see your code
- BitBucket.com: Free Private Repositories

6. Remote Repository Commands

- Clone
 - Download an entire remote repository as a local repository
- Fork
 - Download an entire remote repository as another remote repository
- Push
 - Uploading/Pushing our local repository to the remote server
- Pull
 - Downloading/Pulling the remote repository's latest changes into our local repository

7. Remote and Local Repository Commands

LOCAL

COMMIT
RESET
CHECKOUT
ADD
RM
STATUS
DIFF
MERGE

REMOTE

CLONE
FORK
PUSH
PULL

Ok ...

8. Lets **GIT** our hands dirty

Git States



**Working
Directory**

**Staged
Files**

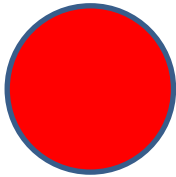
**Committed
Files**

**Remote
Repository**

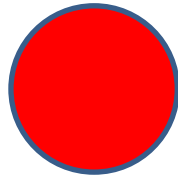
git diff



**Working
Directory**



**Staged
Files**



**Committed
Files**

**Remote
Repository**

git diff --staged

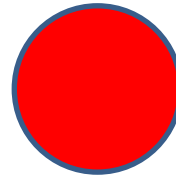
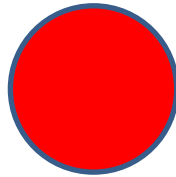


**Working
Directory**

**Staged
Files**

**Committed
Files**

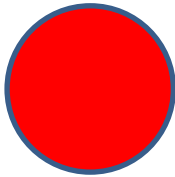
**Remote
Repository**



git diff HEAD

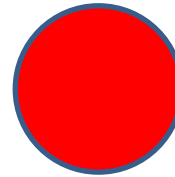


**Working
Directory**



**Staged
Files**

**Committed
Files**



**Remote
Repository**

git diff master origin/master

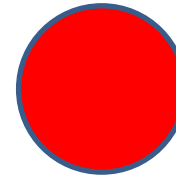
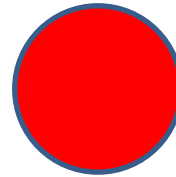


**Working
Directory**

**Staged
Files**

**Committed
Files**

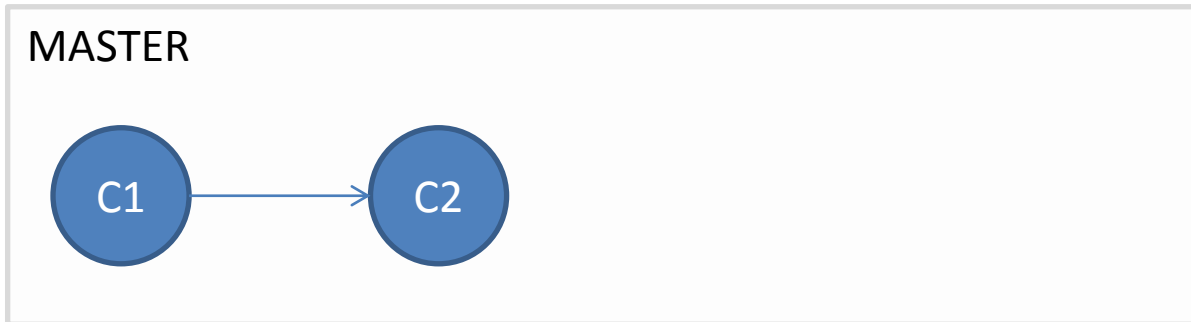
**Remote
Repository**



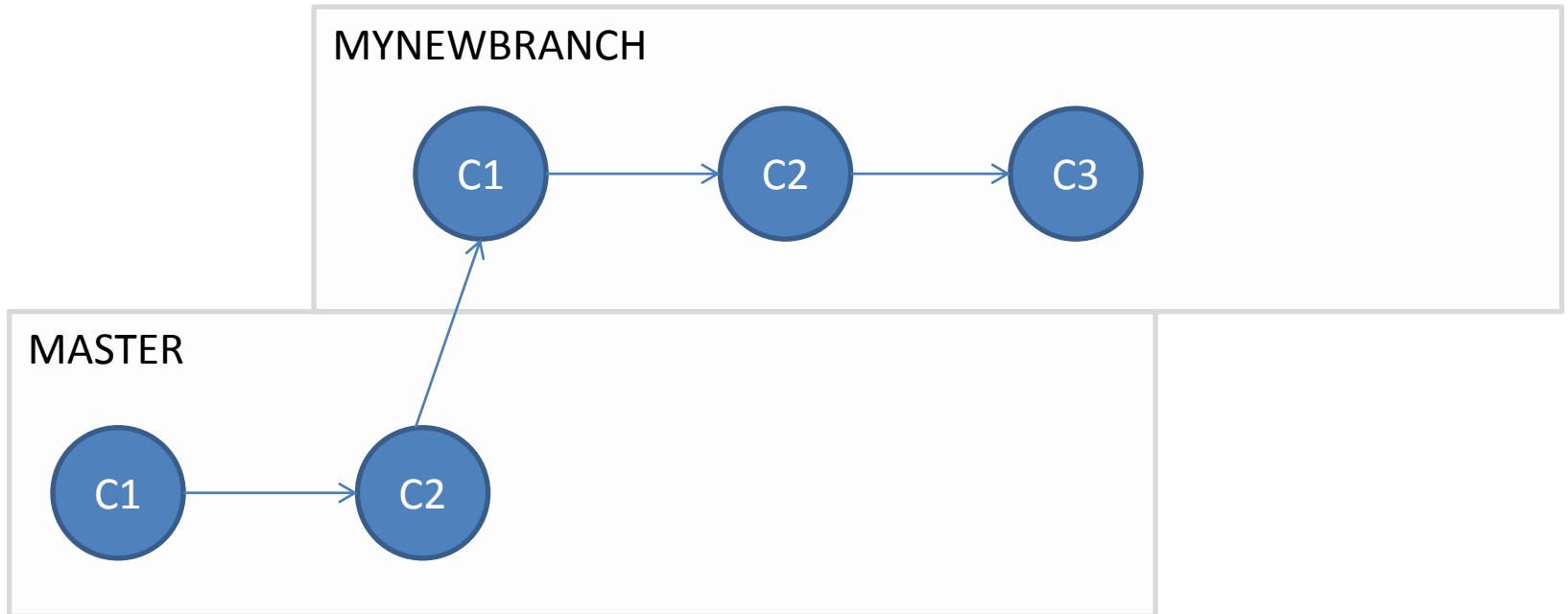
Our branches so far ...



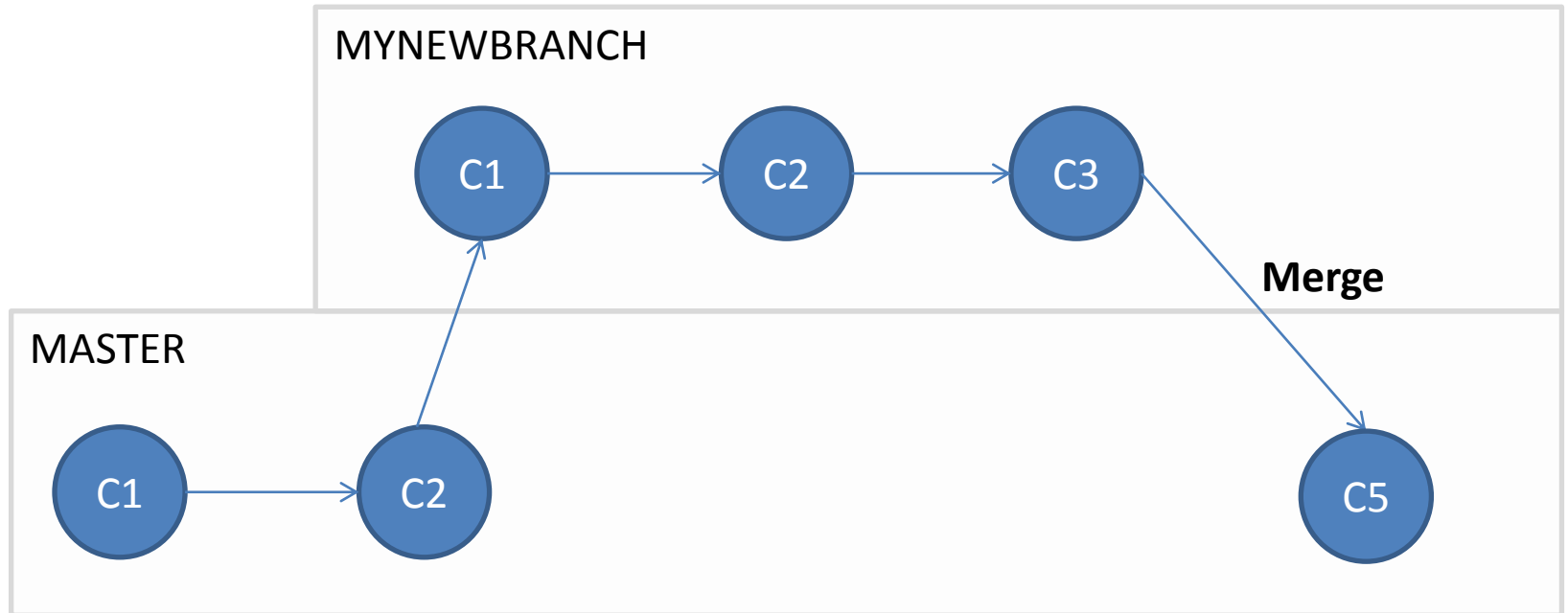
Our branches so far ...



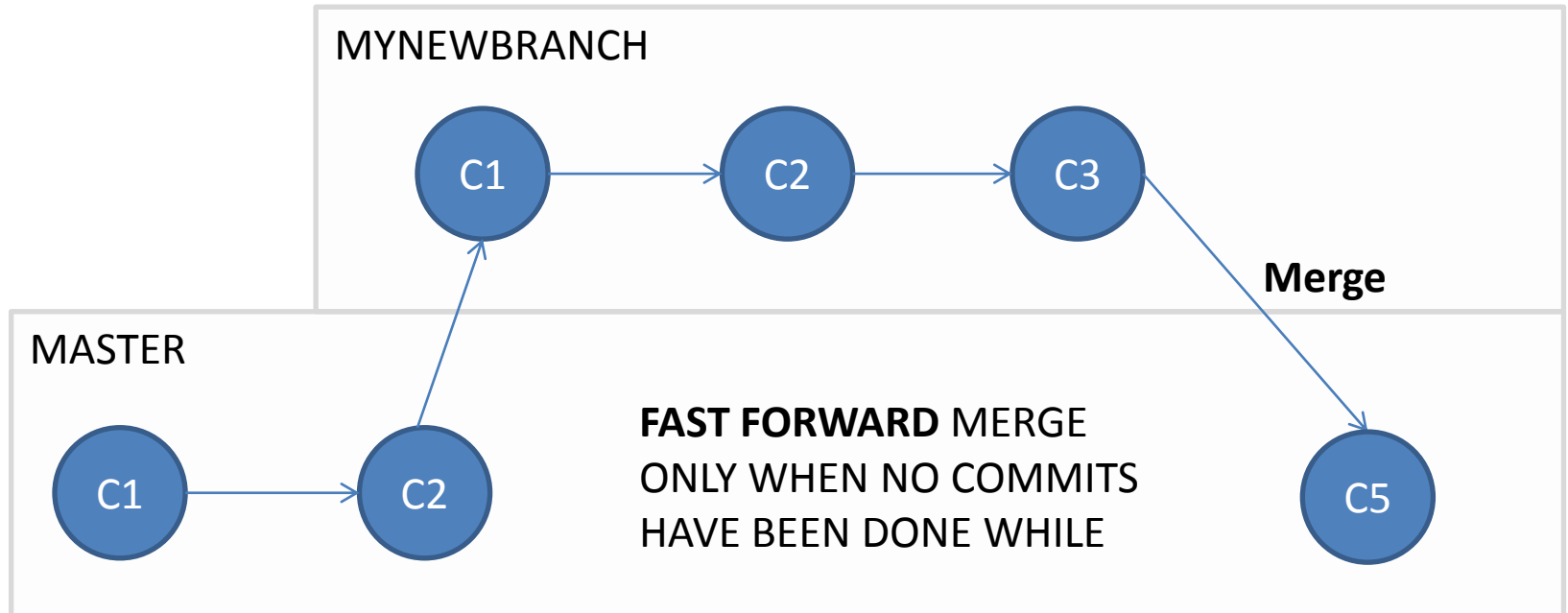
We need a new branch



We need a new branch



We need a new branch



We need a new branch

