git branching

preserving version options

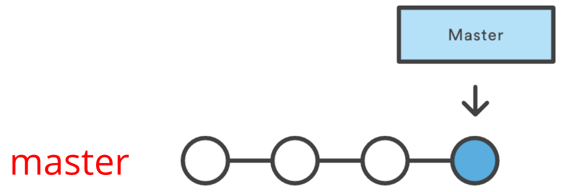
Learning Objectives

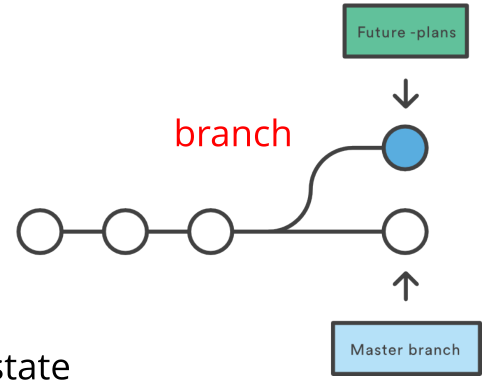
- Understand why branching is used
- Make and checkout your first branch
- Be able to resolve merge conflicts when they occur

diagrams source: https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud

branching

purpose of branching?

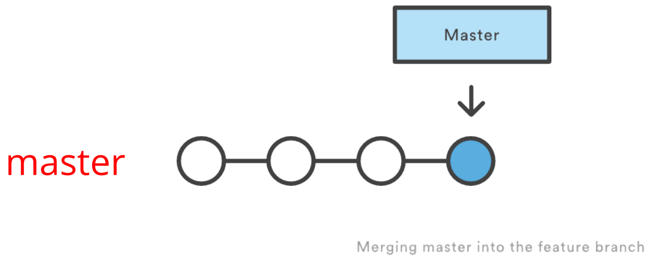


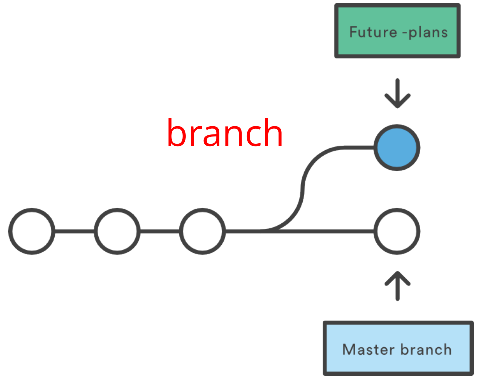


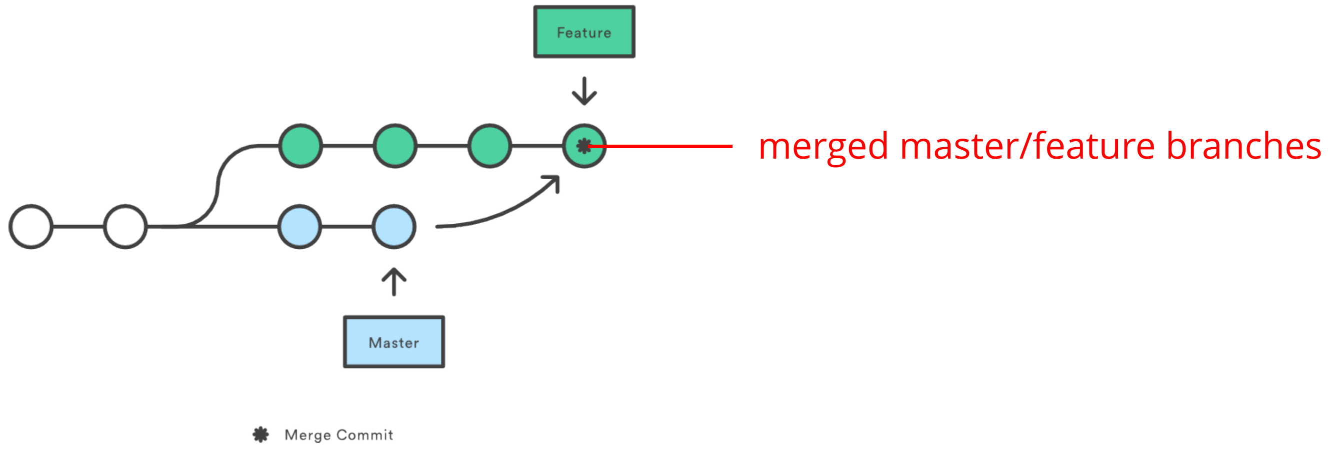
- branching creates a copy of the code in its current state
- allows working on new feature without disturbing main code base
- the branch operates independently from the original "master" branch
- branches can be merged with original code base (master branch) once working and complete

branching

merging







branching commands

```
To find out what branch you're currently on:
$ git branch


To create a new branch:
$ git branch branch-name


To work on a specific branch:
$ git checkout branch-name


To push your branch up to GitHub:
$ git push origin branch-name


To pull down the latest from a specific branch into the branch you're working on:
$ git pull origin branch-name
```

git branch

git checkout

git push

git pull

branching commands

git branch

manage branches in repo

```
$ git branch
```

- see list of all branches

branching commands

git branch

manage branches in repo

```
$ git branch
```

- see list of all branches

```
$ git branch <new_branch_name>
```

- create a new branch

branching commands

git branch

manage branches in repo

`$ git branch`	• see list of all branches
`$ git branch <new_branch_name>`	• create a new branch
`$ git branch -b <new_branch_name>`	• create a new branch and switch to it

branching commands

git branch

manage branches in repo

`$ git branch`	• see list of all branches
`$ git branch <new_branch_name>`	• create a new branch
`$ git branch -b <new_branch_name>`	• create a new branch and switch to it

`$ git branch <new_branch_name>`
`$ git checkout <new_branch_name>`
	• long-hand equivalent commands

branching commands

git checkout

switch between branches in repo

```
$ git checkout
```

- switch working directory to different branch

working directory -- the active/current version

branching commands

git checkout

switch between branches in repo

```
$ git checkout
```

- switch working directory to different branch

working directory -- the active/current version

```
$ git checkout <branch_name>
```

- switch working directory to specified branch

branching commands

git checkout

switch between branches in repo

```
$ git checkout
```

- switch working directory to different branch

working directory -- the active/current version

```
$ git checkout <branch_name>
```

- switch working directory to specified branch

```
$ git checkout master
```

- return working directory to master branch

local/remote commands

git push

upload work to online storage

```
$ git push <remote> --all
```

- push all branches from LOCAL repo to REMOTE repo

local/remote commands

git push

upload work to online storage

```
$ git push <remote> --all
```

- push all branches from LOCAL repo to REMOTE repo

```
$ git push <remote> <branch_name>
```

- push specific LOCAL branch to REMOTE

local/remote commands

git push

upload work to online storage

`$ git push <remote> --all`	• push all branches from LOCAL repo to REMOTE repo
`$ git push <remote> <branch_name>`	• push specific LOCAL branch to REMOTE
`$ git push <remote> master`	• push LOCAL master to REMOTE master

git basics

Exercise

- make two new branches for your personal web page project

    new-menu-bar

    fix-responsiveness

- make commits on each branch
- push branches to github
- verify github upload by checking branch dropdown menu

local/remote commands

git fetch

retrieve work from online storage

```
$ git fetch <remote>
```

- imports REMOTE commits into your LOCAL repo

local/remote commands

git fetch

retrieve work from online storage

```
$ git fetch <remote>
```

- imports REMOTE commits into your LOCAL repo

```
$ git fetch <remote> <branch_name>
```

- fetches only specified branch

local/remote commands

git fetch

retrieve work from online storage

`$ git fetch <remote>`	• imports REMOTE commits into your LOCAL repo
`$ git fetch <remote> <branch_name>`	• fetches only specified branch
`$ git branch -r`	• see list of remote branches before fetching

local/remote commands

git fetch

retrieve work from online storage

```
$ git fetch <remote>
```

- imports REMOTE commits into your LOCAL repo

```
$ git fetch <remote> <branch_name>
```

- fetches only specified branch

```
$ git branch -r

  # origin/master
  # origin/develop
  # origin/some-feature
```

- see list of remote branches before fetching
- remote branch names are preceded by remote name

local/remote commands

git fetch

update local version to match remote

```
$ git fetch <remote>
```

• imports REMOTE commits into your LOCAL repo

```
$ git fetch <remote> <branch_name>
```

• fetches only specified branch

git merge

```
$ git checkout master
```

• switch to master branch

local/remote commands

git fetch

update local version to match remote

```
$ git fetch <remote>
```

- imports REMOTE commits into your LOCAL repo

```
$ git fetch <remote> <branch_name>
```

- fetches only specified branch

git merge

```
$ git checkout master
```

- switch to master branch

```
$ git merge origin/master
```

- merge fetched REMOTE branches into LOCAL master branch

local/remote commands

merge conflicts

definition

- git can't decide how to merge two separate versions of a project
- the same two files have been modified in a different way

resolving

- open the affected file(s)
- look for merge conflict indicators

conflict start

```
<<<<<<< HEAD
<title>I like this title</title>
=======
<title>Lets do this one instead</title>
>>>>>>> Another commit.
```

local/remote commands

merge conflicts

definition

- git can't decide how to merge two separate versions of a project
- the same two files have been modified in a different way

resolving

- open the affected file(s)
- look for merge conflict indicators

```
<<<<<<< HEAD
<title>I like this title</title>
=======
<title>Lets do this one instead</title>
>>>>>>> Another commit.
```

conflict end

local/remote commands

merge conflicts

definition

- git can't decide how to merge two separate versions of a project
- the same two files have been modified in a different way

resolving

- open the affected file(s)
- look for merge conflict indicators

```
<<<<<<< HEAD
<title>I like this title</title>
=======
<title>Lets do this one instead</title>
>>>>>>> Another commit.
```

conflict separator

local/remote commands

merge conflicts

definition

- git can't decide how to merge two separate versions of a project
- the same two files have been modified in a different way

resolving

- open the affected file(s)
- look for merge conflict indicators
- delete "rejected" option/demarcation lines

```
<<<<<<< HEAD
<title>I like this title</title>
=======
<title>Lets do this one instead</title>
>>>>>>> Another commit.
```

```
<title>I like this title</title>
```

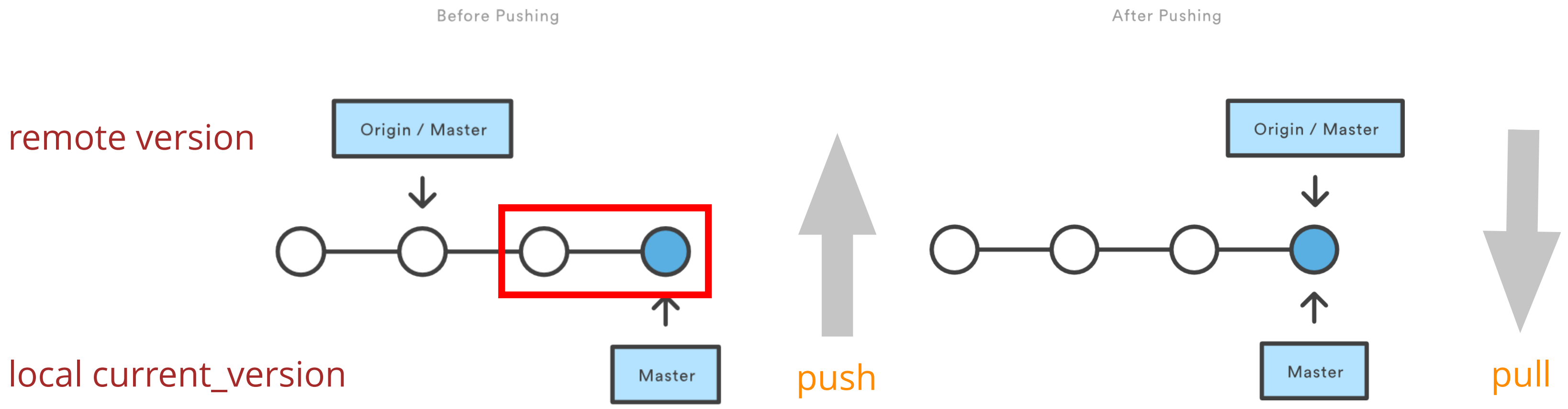
local/remote commands

git pull

update local version to match remote (same as fetch/merge)

`$ git pull`	• pull all REMOTE branches to LOCAL repo
`$ git pull <remote> <branch_name>`	• pull specific REMOTE branch to LOCAL repo
`$ git pull <remote> master`	• pull REMOTE master to LOCAL master

local/remote commands



- common message: "local branch is 2 commits ahead of remote"

local/remote commands

matching local version to remote version

typical collaborative project workflow

- you created a new branch and did work on that branch
- other coders have modified their branches (including master) and pushed them to the remote
- your local master branch is now out of date with remote master branch
- rebase your local master branch (update it to newest remote version)
- make commits to your local branch and push to remote with pull request

before opening pull request

- make sure that branch is up to date with master (rebase or merge)

local/remote commands

matching local version to remote version

rebase your branch with master

- commit any local changes to your branch
- use git pull to download current branch versions from remote

```
git pull
```

- git pull is the equivalent of these two commands:

```
git fetch
git merge
```

- by default git pull fetches new commits from remote, then merges in local changes
- use git pull --rebase fetches remote commits/rebases local commits on top of them

```
git pull --rebase origin master
```

collaboration

pull requests

goal: seek to merge local branch work into remote master branch

- make sure that local master branch is up to date with remote master
- open a pull request on GitHub so your peers can review your work
- verify that project manager has merged your branch into the master
- if merge could not be done, read notes and revise code to enable merge

collaboration

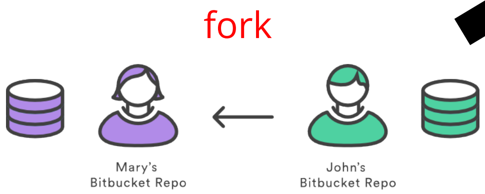
pull requests

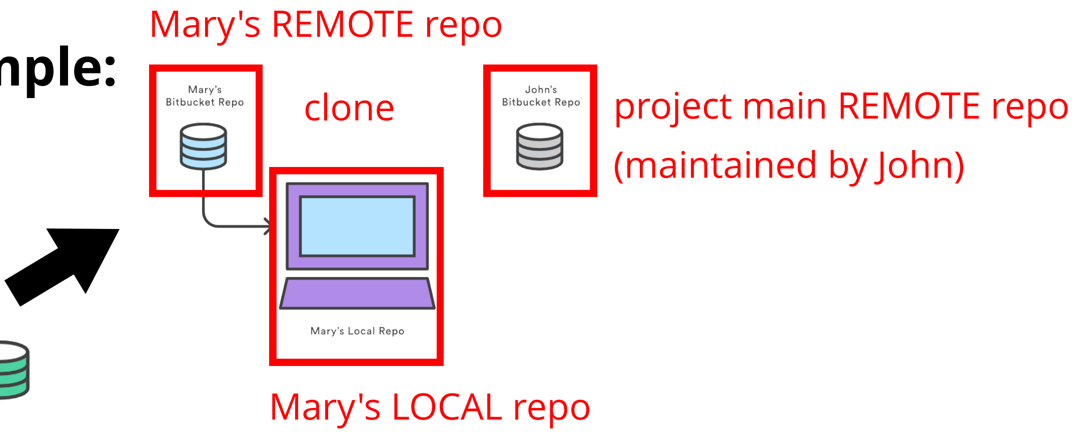
workflow

- info required to make pull request
  - source repository
  - source branch
  - destination repository
  - destination branch

collaboration

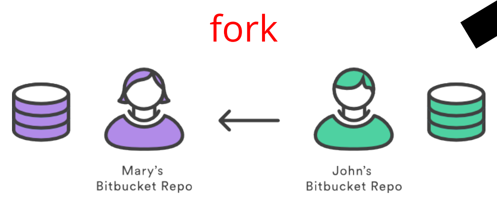
pull request example:
forking workflow

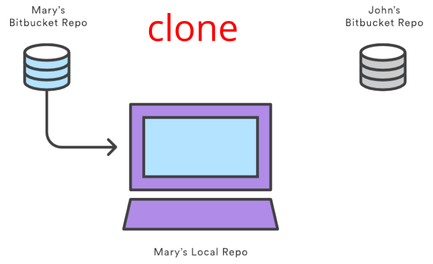


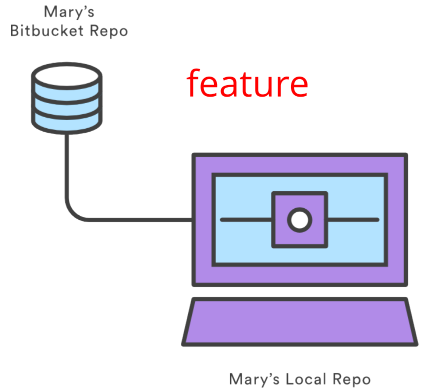


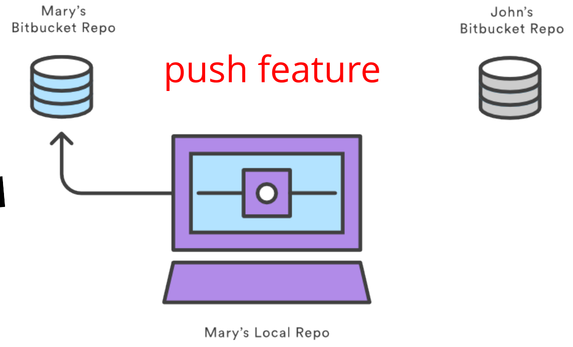
collaboration

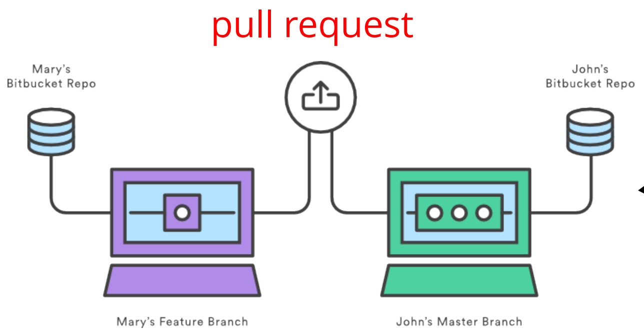
pull request example:
forking workflow







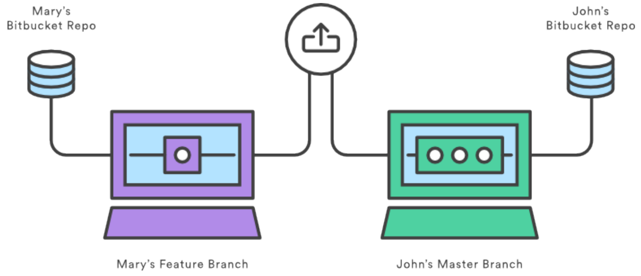


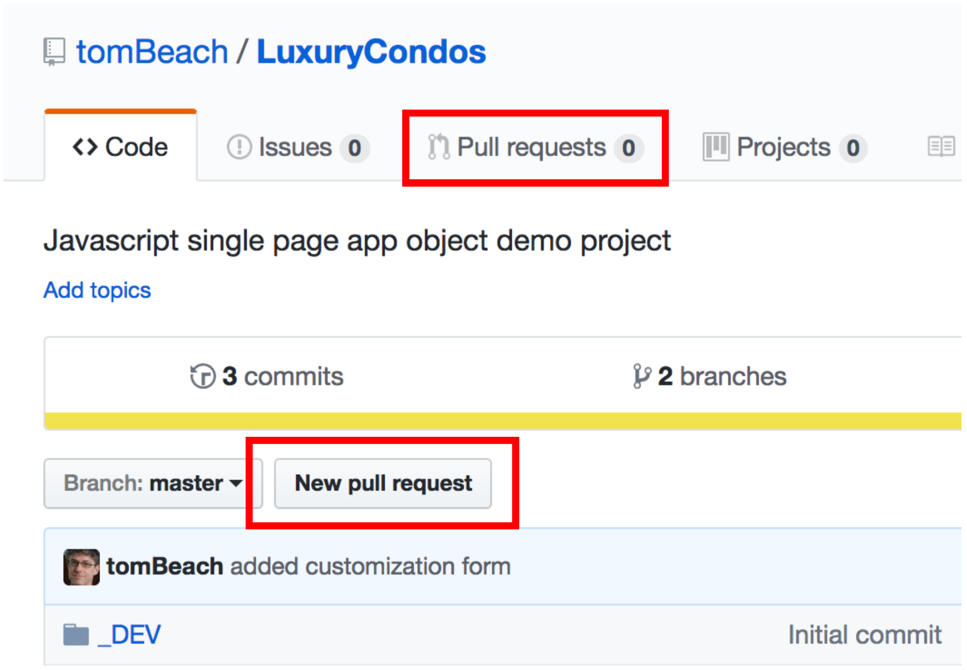


collaboration

pull request example:
forking workflow

pull request




tomBeach / LuxuryCondos

Javascript single page app object demo project

Add topics

3 commits
2 branches

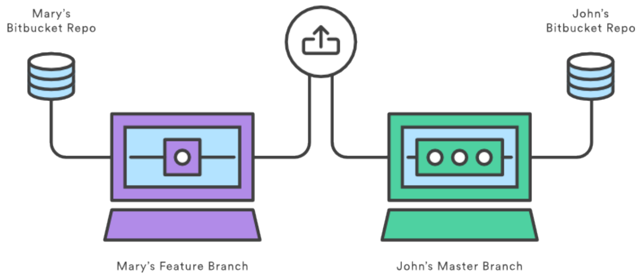
tomBeach added customization form

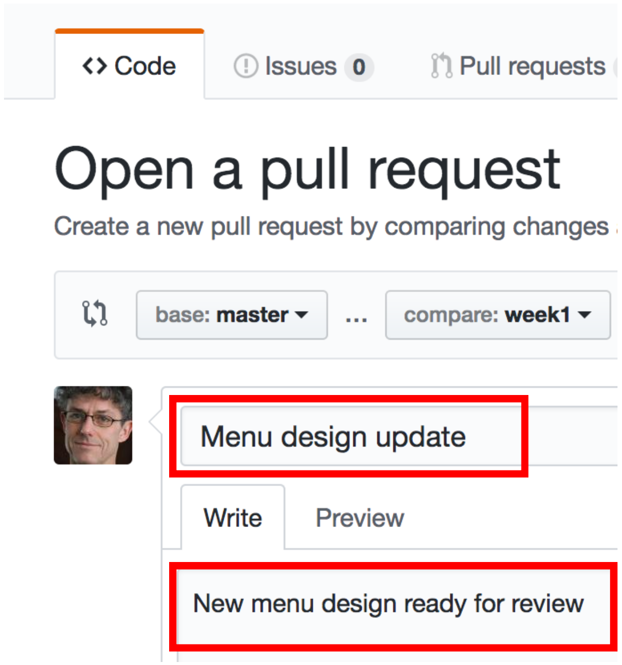
_DEV Initial commit

collaboration

pull request example:
forking workflow

pull request





Open a pull request

Create a new pull request by comparing changes

Menu design update

New menu design ready for review

Pull request label

description

collaboration

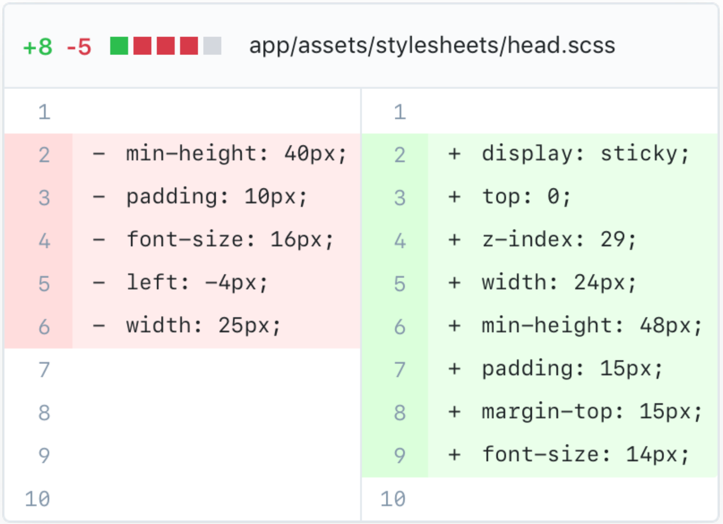
issues

- issues help identify, assign, and keep track of team tasks
- use cases
  - track a bug
  - discuss an idea with a team member (e.g. @mention)
  - start distributing work

collaboration

Exercise (cont.) https://github.com/features

- read github instructions for issues, pull requests and diffs



```
+8 -5 🟩🟥🟥🟥⬜ app/assets/stylesheets/head.scss

                              |
- min-height: 40px;           | + display: sticky;
- padding: 10px;              | + top: 0;
- font-size: 16px;            | + z-index: 29;
- left: -4px;                 | + width: 24px;
- width: 25px;                | + min-height: 48px;
                              | + padding: 15px;
                              | + margin-top: 15px;
                              | + font-size: 14px;
                              |
```

diffs

git basics

Exercise (cont.)

https://github.com/features

- open a new issue on your github project repo
- note issue number (probably #1)
- modify your project (locally)
- push to github after commit with message that includes issue number

```
-m "closes issue #1"
```

- note result on github after push (issue #1 should be closed)
- github automatically closes issues with key words and issue #...

closes, closed, fix, fixes, fixed, resolve, resolves, resolved (e.g. "fixes #1)

- ...if you are on the default branch (usually master)

other git commands

git remote

```
To get url of remote repo:
  $ git remote show origin

To add a remote repo:
  $ git remote add <remote_name>
  $ git remote add origin

  $ git remote add <remote_url>
  $ git remote add https://github.com/<yourRepo>.git

To verify new remote
  $ git remote -v
```

other git commands

git stash

```
Temporarily stashes recent changes to working copy:
$ git stash

== example ==
$ git status
  On branch master
  Changes to be committed:
    new file: style.css

  Changes not staged for commit:
    modified: index.html

$ git stash
  Saved working directory and index state WIP on master: 5002d47 our new homepage
  HEAD is now at 5002d47 our new homepage

$ git status
  On branch master
  nothing to commit, working tree clean
```

other git commands

git stash

```
Temporarily stashes recent changes to working copy:
$ git stash

== example ==
$ git status
  On branch master
  nothing to commit, working tree clean

$ git stash pop
  On branch master
  Changes to be committed:
    new file: style.css

  Changes not staged for commit:
    modified: index.html

  Dropped refs/stash@{0} (32b3aa1d185dfe6d57b3c3cc3b32cbf3e380cc6a)
```

removes changes from stash, reapplies to working copy

git basics

Resources

- Code Academy: https://www.codecademy.com/en/courses/learn-git
- TeamTreeHouse: https://teamtreehouse.com/library/git-basics
- Roger Dudler: http://rogerdudler.github.io/git-guide/

code left

current_topic

```
def def_name
    puts "******* def_name *******"
end

def def_name
    puts "******* def_name *******"
end
```

main point

- sub point

code right

current_topic

main point

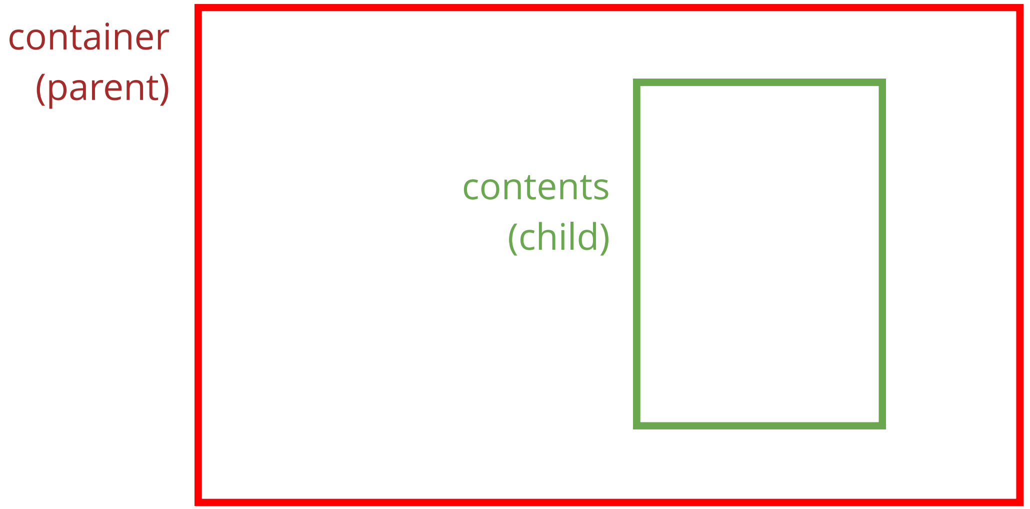
- sub point

```
def def_name
    puts "******* def_name *******"
end

def def_name
    puts "******* def_name *******"
end
```

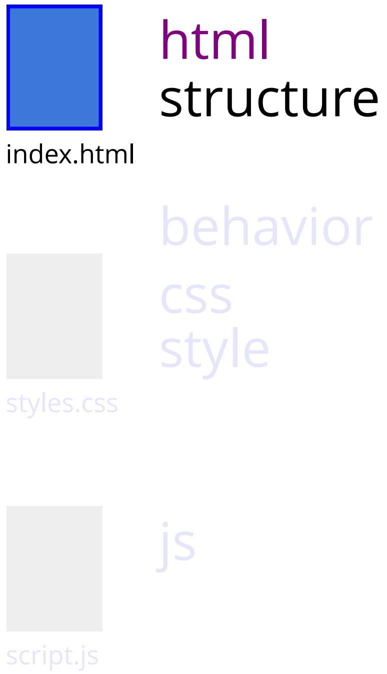
boxes and labels

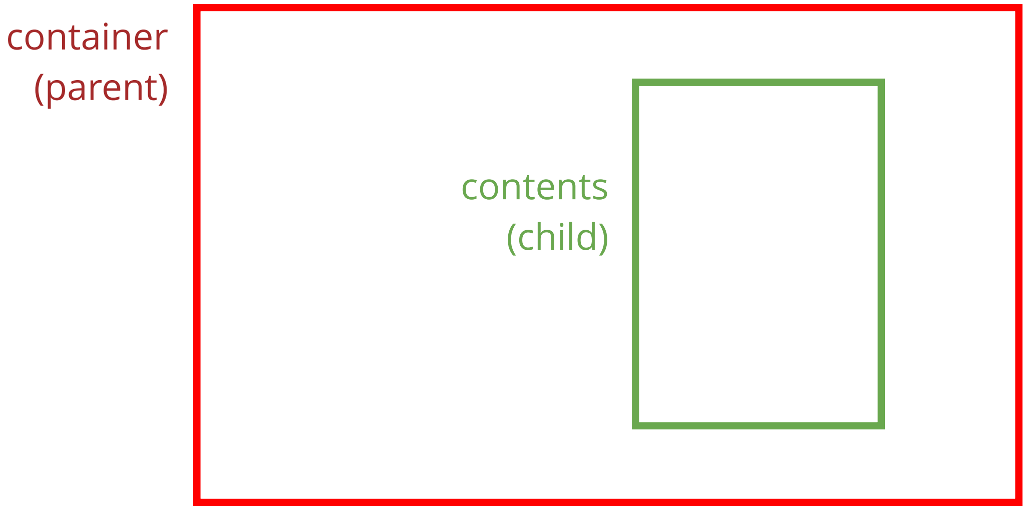
current_topic



topic

current_topic





lists and lines

host

server

backend

client

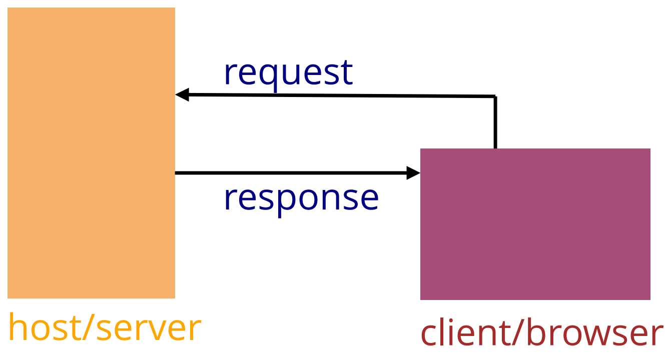
browser

frontend

request

response

how the web works


request
response
host/server
client/browser