

**{ POWER.CODERS }**

# Intro to GitHub

# AGENDA

---

Today we will cover

- Git and GitHub: What's the difference?
- Setup a Github account
- Understanding repositories
- Working with files in the GitHub web UI
- Making your first pull request
- Finding and choosing projects and communities

# GIT AND GITHUB:

## WHAT'S THE DIFFERENCE?

# GIT AND GITHUB:

## WHAT'S THE DIFFERENCE?

**Git** is a version control system software that you can run on your own computer, or on a server.

# GIT AND GITHUB:

## WHAT'S THE DIFFERENCE?

**Git** is a version control system software that you can run on your own computer, or on a server.

**GitHub** is a cloud-based service for storing, sharing, and collaborating on projects managed with Git.

# TEAM COLLABORATION



# TEAM COLLABORATION

---

**GitHub** is the server, where your team projects are hosted.

# TEAM COLLABORATION

**GitHub** is the server, where your team projects are hosted.

**Git** allows you to communicate with Github and push/pull your files.



# ALTERNATIVES

---

We will use GitHub, but be aware that there are similar services, like GitLab or BitBucket.

[Read more on that](#)

# CREATE A GITHUB ACCOUNT

---

1. Go to <https://www.github.com/join>
2. Find yourself a username. It does not have to be your name.  
Mine - as an example - is **sardaykin**.
3. Use your powercoders email address.
4. Choose the free, public plan.
5. Verify your email address.

# CREATE A GITHUB ACCOUNT

---

1. Go to <https://www.github.com/join>
2. Find yourself a username. It does not have to be your name.  
Mine - as an example - is **sardaykin**.
3. Use your powercoders email address.
4. Choose the free, public plan.
5. Verify your email address.

Your GitHub Profile is like a social media profile for your code. All interested companies will look at your projects and code quality.

# SSH

---

- SSH is a protocol for secure communication between computers
- We'll use SSH to communicate between our computers and github servers
- To establish a secure connection we need to generate a public-private key pair.
- [\(Learn more about SSH here\)](#)

# CREATING AN SSH KEY FOR YOUR COMPUTER

---

- On the command line: `ssh-keygen`
- When asked for the path, just hit enter to save at the default location.
- When asked for the passphrase, leave empty.
- This generates two files, one ending in `.pub`. This is the *public* key that you can share with others.

# REGISTERING YOUR COMPUTER'S PUBLIC KEY ON **GIT**HUB

1. Run `cat ~/.ssh/id_rsa.pub` to display the public key.

Example:

```
seb@seb-mbp:~% cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDOM4TM74JdxVHSAg0A4bWaqhxTVM80HvKw6n3iF1MNT8LX0W4osjTZ6TLhuTaRG
lC8iDXG5BNdDh4uvkts41v/QBQZZA89MZPpuM4yRs8pu/QA7sg8vw+MsMHYJvElSMG0R/kErwhRAMuMPtcVnLhRjo5n06XHsDK0MM
cDCypd9UgTERJifS+/CmcXh24LsQPvMuzqjq0weW2ET0Yyef/LFUtoVTS4F2I5Qi68p3lW+G0/W3r2ETjNGkietINej4/HE3h44Gq
XeCS2yivvkAxYoUhy1NKqzpQMhFiIWPpnyKai3sVoLEMMmpeZ7hmHdn75LFZxNYyVlqd5T7z4IcTt seb@Windows-Phone
seb@seb-mbp:~%
```

2. Go to your [GitHub account settings](#) > SSH keys > add new
3. Name your key to remember on which computer it is, and paste the key in the appropriate form field.
4. Voilà, you'll be able to clone/push to GitHub from this computer for authorized repositories!

# CREATE A REPOSITORY AT GITHUB

---

- > <https://help.github.com/articles/create-a-repo/>
- > Create a public repository, called poco-yourname
- > Description: "Powercoders Switzerland: YOUR NAME"
- > Do **not** initialize with a README or LICENSE

# CONFIGURE LOCAL REPO TO SYNC

---

> Other copies of your repository are called **remotes**



# CONFIGURE LOCAL REPO TO SYNC

---

- > Other copies of your repository are called **remotes**
- > Find your GitHub repository URL, it looks like

```
git@github.com:username/poco-yourname.git
```

# CONFIGURE LOCAL REPO TO SYNC

- Other copies of your repository are called **remotes**
- Find your GitHub repository URL, it looks like

```
git@github.com:username/poco-yourname.git
```

- Open terminal in VSC and enter

```
$ git remote add origin  
git@github.com:user/poco-yourname.git
```

# CONFIGURE LOCAL REPO TO SYNC

> Other copies of your repository are called **remotes**

> Find your GitHub repository URL, it looks like

```
git@github.com:username/poco-yourname.git
```

> Open terminal in VSC and enter

```
$ git remote add origin  
git@github.com:user/poco-yourname.git
```

> \$ **git** remote -v to view configured remotes

# PUSH COMMITS TO GITHUB

> `git push -u origin main` - first time

> `git push` - after that

# PUSH COMMITS TO GITHUB

```
> git push -u origin main - first time
```

```
> git push - after that
```

The very first time we push a commit we have to tell Git exactly how these two repositories (local and GitHub) are linked, including how they branch.

# PUSH COMMITS TO GITHUB

```
> git push -u origin main - first time
```

```
> git push - after that
```

The very first time we push a commit we have to tell Git exactly how these two repositories (local and GitHub) are linked, including how they branch.

The `-u` flag is used to set origin as the **upstream** remote in your git config. You only have to do that the first time and then can use `push` and `pull` without arguments.

# PUSH COMMITS TO GITHUB

```
> git push -u origin main - first time
```

```
> git push - after that
```

The very first time we push a commit we have to tell Git exactly how these two repositories (local and GitHub) are linked, including how they branch.

The `-u` flag is used to set origin as the **upstream** remote in your git config. You only have to do that the first time and then can use `push` and `pull` without arguments.

You will probably be prompted for credentials or an authentication token.

# NEW WORKFLOW

---

- > `git add [...]` to add changes
- > `git commit [...]` to commit the changes locally
- > `git push` to send the changes to GitHub



# NEW WORKFLOW

---

- > `git add [...]` to add changes
- > `git commit [...]` to commit the changes locally
- > `git push` to send the changes to GitHub

We had `add` and `commit` before, new is the command `push`

# WORKING IN A TEAM

---

- > Make sure you have the latest changes to avoid merge conflicts
- > `git pull origin main` before you start working on a repo that day

# GITHUB FROM NOW ON

- Commit all your work to your Git repository
- `git` push regularly
- Expect comments! Programmers **communicate**

# COMMENTS ON GITHUB

---

Anyone can see the comments and comment on public repositories.

Check it out yourself and go to the **conversation** tab.

Comment on your buddy's repository. Request a change for example.

# PULL REQUESTS



Github has a mechanism that allows you to contribute to someone else's repo, called "pull requests".

# PULL REQUESTS

---

Github has a mechanism that allows you to contribute to someone else's repo, called "pull requests".

You do your change on a personal branch, then submit a pull request to the repo owner.

# PULL REQUESTS

---

Github has a mechanism that allows you to contribute to someone else's repo, called "pull requests".

You do your change on a personal branch, then submit a pull request to the repo owner.

The owner (or reviewer) can then accept or dismiss the pull request, or ask for changes.

# PULL REQUESTS

---

Github has a mechanism that allows you to contribute to someone else's repo, called "pull requests".

You do your change on a personal branch, then submit a pull request to the repo owner.

The owner (or reviewer) can then accept or dismiss the pull request, or ask for changes.

When working in a small team you do not *have* to use pull requests, it's mostly used for outside contributions or big teams.



# ACCEPTING FEEDBACK



It's totally normal for a pull request to require changes.

# ACCEPTING FEEDBACK



It's totally normal for a pull request to require changes.

Don't be afraid to ask questions for clarification.

# ACCEPTING FEEDBACK

---

It's totally normal for a pull request to require changes.

Don't be afraid to ask questions for clarification.

Most maintainers are friendly and helpful. If you encounter one who isn't, consider finding a different project worthy of your contributions.

# WE'VE GOT ISSUES



GitHub **issues** are where collaborators track bugs, propose features, and discuss changes. They are for conversation, and do not actually change code.

# WE'VE GOT ISSUES



GitHub **issues** are where collaborators track bugs, propose features, and discuss changes. They are for conversation, and do not actually change code.

Anyone can open or comment on an issue.

# WE'VE GOT ISSUES

GitHub **issues** are where collaborators track bugs, propose features, and discuss changes. They are for conversation, and do not actually change code.

Anyone can open or comment on an issue.

**Try it yourself. Propose a feature to the participants sitting next to you.**

# CLAIMING ISSUES

---

On Open Source projects anyone can claim issue to solve them for the community.

# CLAIMING ISSUES

---

On Open Source projects anyone can claim issue to solve them for the community.

If you find an issue you'd like to work on, avoid duplicating work by commenting to say you'll volunteer.



# CLAIMING ISSUES

On Open Source projects anyone can claim issue to solve them for the community.

If you find an issue you'd like to work on, avoid duplicating work by commenting to say you'll volunteer.

The screenshot shows a GitHub issue page for the repository '[ajax] update READ.me #575'. The issue is marked as 'Open' and was opened by 'Karia-Isabel-Sandoval' on April 17. The issue title is '[ajax] update READ.me #575'. Below the title, there are two comments. The first comment, from 'Karia-Isabel-Sand...', states 'The materials and slide link are broken.' The second comment, from 'julliberk', states 'This is related to issue #466 and I'm going to work on it!'. On the right side of the issue, there are sections for 'Assignees' (No one—assign yourself), 'Labels' (None yet), and 'Projects' (None yet).

<> Code ① Issues 125 Pull requests 18 Projects 0 Wiki Insights Settings

[ajax] update READ.me #575 Edit New issue

🔓 Open Karia-Isabel-Sandoval opened this issue on Apr 17 · 1 comment

Karia-Isabel-Sand... commented on Apr 17

The materials and slide link are broken.

julliberk commented on Apr 18

This is related to issue #466 and I'm going to work on it!

Assignees  
No one—assign yourself

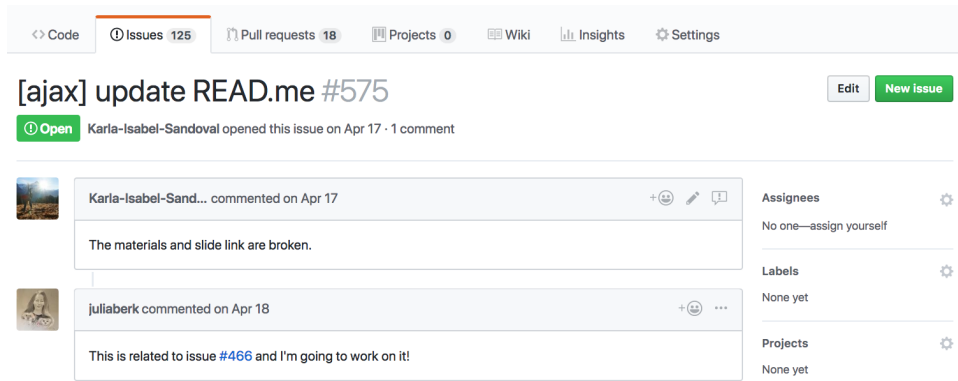
Labels  
None yet

Projects  
None yet

# CLAIMING ISSUES

On Open Source projects anyone can claim issue to solve them for the community.

If you find an issue you'd like to work on, avoid duplicating work by commenting to say you'll volunteer.



The screenshot shows a GitHub issue page for the repository '[ajax] update READ.me #575'. The issue is marked as 'Open' and was opened by 'Karia-Isabel-Sandoval' on April 17. The issue title is '[ajax] update READ.me #575'. Below the title, there are two comments. The first comment, from 'Karia-Isabel-Sand...', states 'The materials and slide link are broken.' The second comment, from 'juliaberk', states 'This is related to issue #466 and I'm going to work on it!'. On the right side of the issue, there are sections for 'Assignees' (No one—assign yourself), 'Labels' (None yet), and 'Projects' (None yet).

Feel free to ask questions if you need clarification before starting!

# TIPS AND RESOURCES

---

- [GitHub Guides](#) (by GitHub)
- [Youtube Playlist for GitHub beginners](#)
- [Kent C. Dodds's GitHub video series](#)
- [Shubeksha Jalan's insightful story](#) of her journey into open source
- [A list of awesome beginners-friendly projects](#). Check it out.

# GOT SOME TIME LEFT?

Play a game: clmystery

<https://github.com/veltman/clmystery/>

# AFTERNOON



# CLI GROUP CHALLENGE

Prepare a game that can be played exclusively with the command line, using `cd`, `ls`, `cat` and friends.

## Example for inspiration

- 14:00 to 14:15 - Intro & Groups
- 14:15 to 14:30 - Tips for a basic game
- 14:30 to 16:30 - Work on your game
  
- 16:30 to 17:00 - Present your game (5 minutes)
  - Goal of the game
  - The rules
  - What challenges did you face?
  - Quick Demo

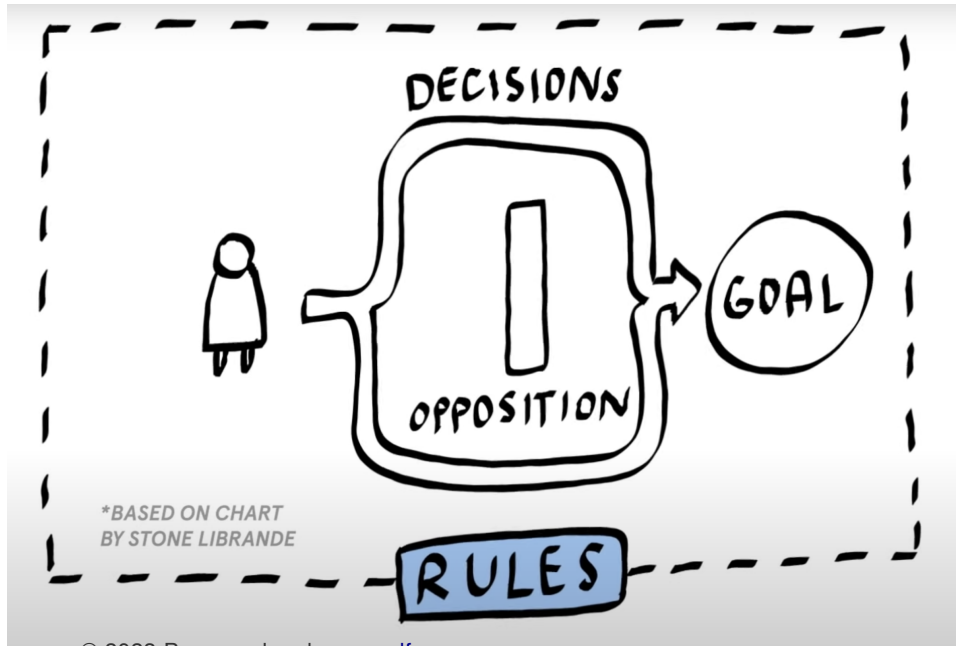
# INTRODUCTION TO GAME DESIGN



# WHAT IS A GAME?

A game has 4 key characteristics:

- > Goal
- > Opposition
- > Decisions
- > Rules





# KEY ELEMENTS OF A GAME

---

Some common elements are

- **Mechanics:** how the game works
- **Dynamics:** how the player interacts with the game
- **Aesthetics:** how the game looks and sounds
- **Narrative:** how the game tells a story

# GAME MECHANICS

---

Game mechanics are at the heart of gameplay design.

Per definition it is how players and rest of the fundamental interlocking pieces of a game such as

- > rules,
- > challenges,
- > goals,
- > actions,
- > strategies,
- > game states

interact with each other in a meaningful way.

# EXAMPLES



Can you think of any?

# EXAMPLES

---

Can you think of any?

- > **Jump** in Super Mario
- > **Shield** in Halo
- > **Skydiving** and **the storm** in Fortnite
- > **Rotate** in Tetris

# THE CORE OF GAMEPLAY

... are the "Core Actions" (game mechanics = actions = verbs) and the "Core Gameplay Loop".

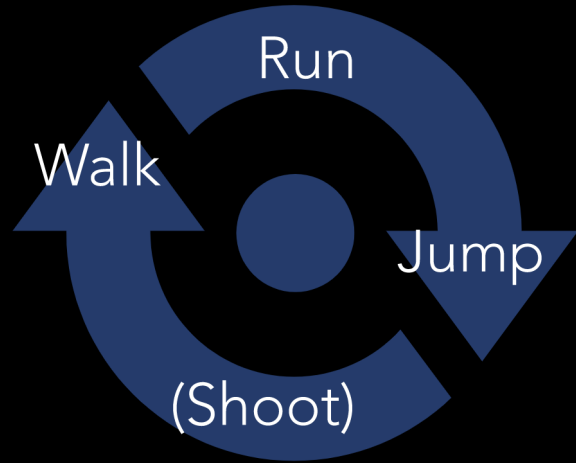
# CORE GAMEPLAY LOOP

---

A gameplay loop is the collective set of actions (game mechanics) that the player will be doing over a specific time frame. At the most basic, we're talking about actions that usually happen within seconds of each other.



# SUPER MARIO BROS.



Further information



# SO WHAT IS IMPORTANT FOR YOUR CLI GAME?

---

- Have a clear goal
- Give the player choices
- Have a clear ruleset
- Keep the rules and the mechanics as simple as possible

