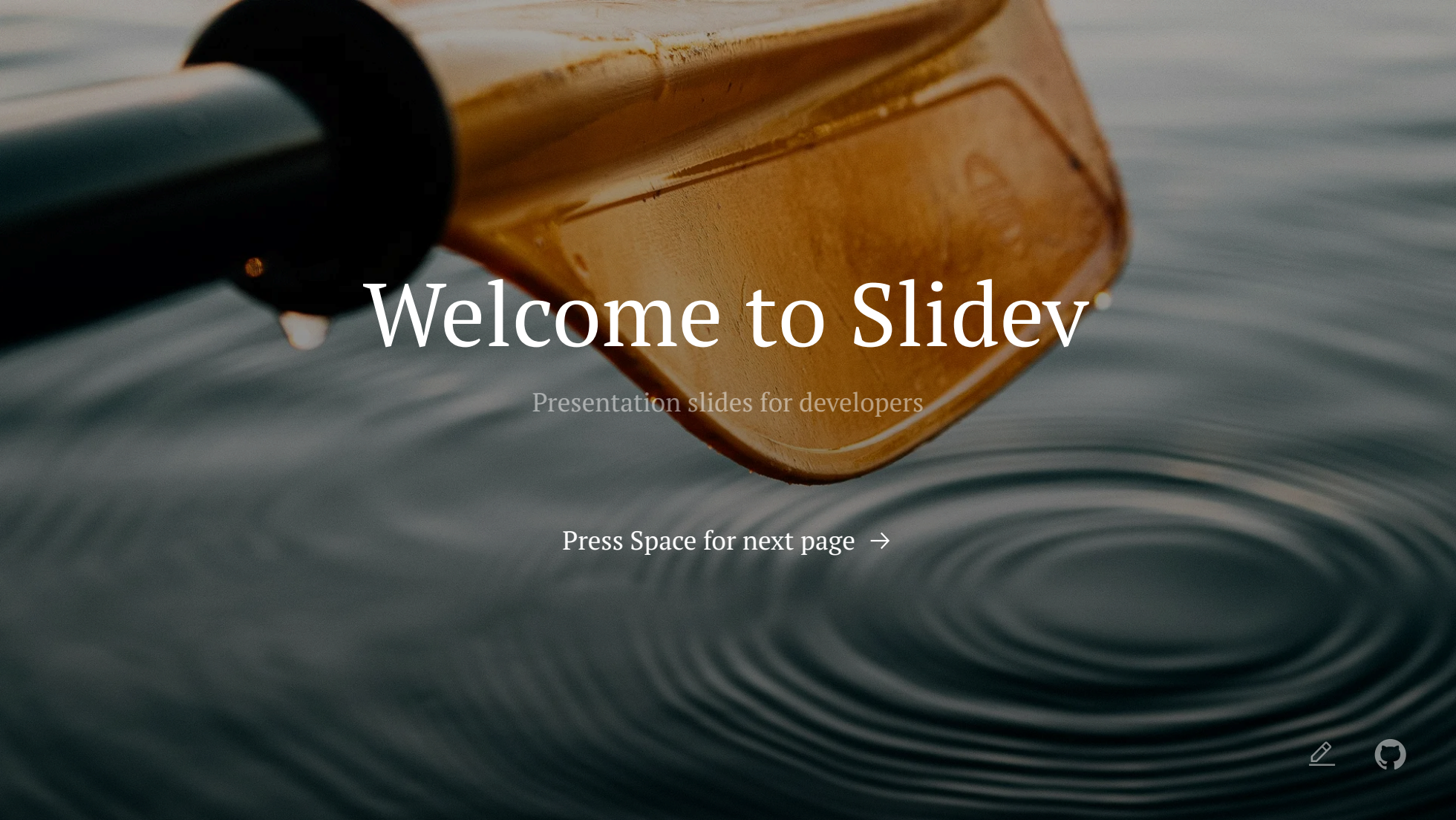


Welcome to Slidev

Presentation slides for developers

Press Space for next page →

What is Slidev?

Slidev is a slide maker and accompanying presentation tool designed for developers. It consists of the following features:

- 📝 **Text-based** - focus on the content with Markdown, and apply styles later
- 🎨 **Themable** - themes can be shared and used as npm packages
- 👩‍💻 **Developer Friendly** - code highlighting, live coding with autocompletion
- 🤹 **Interactive** - embedding Vue components to enhance your slides
- 🎥 **Recording** - built-in recording and camera view
- 📤 **Portable** - export to PDF, PPTX, PNGs, or even a hostable SPA
- 🛠️ **Hackable** - virtually anything that's possible on a webpage is possible in Slidev

Read more about Slidev in Why Slidev?

Table of contents

```
<Toc minDepth="1" maxDepth="1"></Toc>
```

Real-World Git Scenarios

A Practical Guide for Developers

Scenario 1: The Quick Fix During Feature Development

The Situation

- Working on a big feature
- Boss needs urgent bug fix in main branch

The Solution

```
git stash save "feature-x-in-progress"
git checkout main
git checkout -b hotfix-urgent-bug
# Fix bug, then return to feature
git checkout feature-branch
git stash pop
```

Remember

Think of `git stash` as a magical pocket for your unfinished work - like hitting pause on Netflix!

Scenario 2: The Messy Main Branch

The Problem

- Main branch has bad commits
- Need to return to previous version

The Solution

```
git log --oneline
git checkout -b recovery-branch <good-commit-hash>
git checkout main
git reset --hard <good-commit-hash>
```

Key Point

It's like a time machine for your code - you can always go back to when things worked!

Scenario 3: Juggling Multiple Tasks

The Challenge

- Multiple features in development
- Need to switch between them quickly

The Approach

```
# Feature 1
git checkout -b feature1
git stash save "feature1-progress"

# Feature 2
git checkout -b feature2
git stash save "feature2-progress"

# View stashed work
git stash list
```

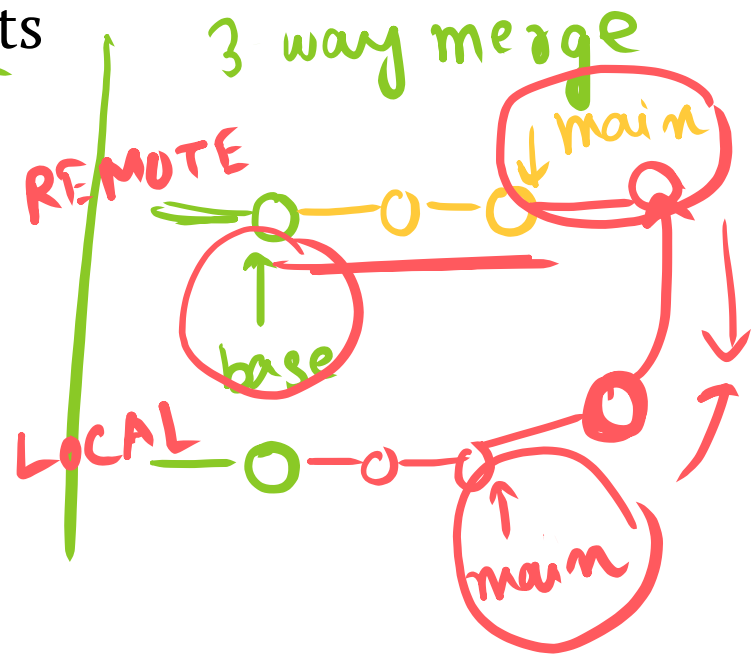
Pro Tip

Think of it as having multiple save slots in a video game!

Scenario 4: Handling Merge Conflicts

Steps to Resolution

1. Stay calm - conflicts are normal!
2. Use `git status` to identify conflicts
3. Look for conflict markers:
   - `<<<<<<<`
   - `=======`
   - `>>>>>>>`
4. Choose which changes to keep
5. `git add` resolved files
6. Complete merge with `git commit`



Remember

It's just like choosing between two lunch options - you can pick one or combine both!

Scenario 5: Branch Cleanup

Commands

```
# View merged branches
git branch --merged

# Remove local branch
git branch -d old-feature-branch

# Remove remote branch
git push origin --delete old-feature-branch

# Clean up tracking branches
git remote prune origin
```

Think of it as...

Spring cleaning for your repository!

Quick Reference: Common "Help!" Moments

Wrong Branch Commit

```
git stash
git checkout correct-branch
git stash pop
```

Undo Last Commit

```
# Keep changes
git reset --soft HEAD~1

# Remove changes
git reset --hard HEAD~1
```

Practice Makes Perfect

Exercise 1: Time Machine

- Make and commit changes
- Practice using `git log`
- Try `git checkout` with commit hashes

Exercise 2: Branch Juggler

- Create multiple branches
- Make different changes
- Practice branch switching
- Use stash for incomplete work

More Practice Exercises

Exercise 3: Conflict Creator

- Create a merge conflict on purpose
- Practice resolution
- Try different merge tools

Exercise 4: Clean-up Crew

- Create several branches
- Merge some branches
- Practice branch cleanup
- Learn repository maintenance

Final Thoughts

Remember:

- Git is your safety net
- Don’t fear experimentation
- You can always go back
- Practice regularly

Happy coding! 🚀