# Emergency Recovery Playbook

**Print this and keep it visible during presentation!**

## Quick Recovery Commands

```
# 1. DISCARD ALL CHANGES (most common fix)
git reset --hard

# 2. JUMP TO STAGE 1
git checkout demo/stage-1-naive-csharp

# 3. JUMP TO STAGE 2
git checkout demo/stage-2-first-du

# 4. JUMP TO STAGE 3
git checkout demo/stage-3-pattern-matching

# 5. CHECK WHERE YOU ARE
git branch --show-current

# 6. REBUILD SOLUTION
dotnet clean; dotnet build

# 7. RUN DEMO
dotnet run --project CustomerLiveDemo
```

## Scenario 1: Typo / Syntax Error

**Symptoms:** Red squiggles, won't build, compiler error

**Fix:**

1. Read error message out loud (buys time to think)
2. If obvious typo: Fix it and rebuild
3. If not obvious within 30 seconds:

   ```
   git reset --hard
   git checkout demo/stage-X  # Current stage you're on
   ```

4. Say: "Let me jump to the working version"
5. Continue from fallback branch

**Talking Point While Recovering:** "This is why we have version control! Let me grab the working version..."

## Scenario 2: Code Doesn't Compile

**Symptoms:** Build fails, errors in terminal

**Fix:**

1. Try once more with `dotnet clean; dotnet build`
2. If still fails:

```
git reset --hard
git checkout demo/stage-X  # Current stage fallback
```

3. Say: "Let me use the pre-tested version"
4. Continue presentation

**Talking Point:** "The important thing is the concept - let's look at the working code..."

---

## Scenario 3: Wrong Output / Logic Bug

**Symptoms:** Runs but output is wrong (e.g., wrong discount)

**Fix:**

1. Say: "Interesting! Let's debug this..."
2. Check the logic in CalculateTotal
3. If not obvious within 1 minute:

```
git checkout demo/stage-X -- CustomerLiveDemo/Program.cs
dotnet build
dotnet run --project CustomerLiveDemo
```

4. Say: "There we go - back to the correct version"

**Talking Point:** "This is actually a good reminder why type safety matters!"

---

## Scenario 4: Complete Disaster / Lost

**Symptoms:** Multiple errors, confused, not sure what went wrong

**Fix (NUCLEAR OPTION):**

1. Stay calm
2. Say: "Let me jump to the final version to show you the outcome"
3. Commands:

```
git reset --hard
git checkout demo/stage-3-pattern-matching
dotnet build
dotnet run --project CustomerLiveDemo
```

4. Show Stage 3 final code
5. Do VIP extension from there

**Talking Point:** "Rather than debug live, let's look at the working solution. The key concepts are..."

---

## Scenario 5: Running Out of Time

**Symptoms:** 5 minutes left, still on Stage 1 or 2

**Fix:**

1. Skip ahead:

```
git checkout demo/stage-3-pattern-matching
```

2. Say: "Let me show you the final result"
3. Show Stage 3 code
4. Highlight key improvements:
   - Domain language explicit (Standard, Registered, Guest)
   - No boolean checks needed
   - Simpler C# code
5. Do quick VIP extension if time

**Talking Point:** "The journey matters, but let's jump to the destination..."

---

## Scenario 6: Ahead of Schedule

**Symptoms:** 10 minutes left, already finished Stage 3

**Options:**

### Option A: VIP Extension (Recommended)

```
# Continue from Stage 3
# Live code VIP tier slowly with explanations
```

### Option B: Show Pattern Matching

```
// Uncomment pattern matching version in CalculateTotal
var discount = customer switch
{
    VIP _ when spend >= 100 => spend * 0.15m,
    Standard _ when spend >= 100 => spend * 0.1m,
    _ => 0m
};
```

### Option C: Show F# Pattern Matching

Add to Library.fs:

```
let calculateTotal customer spend =
    match customer with
    | VIP _ when spend >= 100m -> spend * 0.85m
    | Standard _ when spend >= 100m -> spend * 0.9m
    | _ -> spend
```

### Option D: Open Q&A

"Let's open it up for questions - what would you like to see?"

---

## Scenario 7: Audience Question Derails Demo

**Symptoms:** Good question but takes you off track

**Response:**

1. Acknowledge: "Great question!"
2. Defer if needed: "Let me finish this stage and come back to that"
3. Or demo answer: `git stash` → show answer → `git stash pop`

**Commands for Tangent:**

```
# Save current work
git stash push -m "Demo in progress"

# Show something else, then restore
git stash pop
```

---

## Scenario 8: Git Command Fails

**Symptoms:** Git error message, checkout fails

**Common Causes:**

- Uncommitted changes
- Wrong branch name

**Fix:**

```
# Force checkout (discards changes)
git checkout -f demo/stage-X

# Or reset first
git reset --hard HEAD
git checkout demo/stage-X
```

---

## Scenario 9: Terminal Not Responding

**Symptoms:** Commands hang, no output

**Fix:**

1. Open new terminal window (Ctrl + Shift + `)
2. Navigate to directory:

   ```
   cd c:\git\NovuckFSharpPresentation
   ```

3. Continue from there

---

## Scenario 10: VS Code Crashes

**Symptoms:** Editor freezes or closes

**Fix:**

1. Stay calm: "Let me reopen VS Code"
2. Reopen: Click VS Code icon
3. Navigate to repo:

   ```
   cd c:\git\NovuckFSharpPresentation
   git status
   git checkout demo/stage-X
   ```

4. Continue

**Talking Point:** "Good thing we have version control - nothing is lost!"

---

## Universal Fallback: Show README

If everything is broken and nothing works:

```
git checkout main
```

Open `README.md` and walk through the concepts at a high level using the documentation as slides.

**Talking Point:** "Let me show you the concepts at a high level, and you can try the code yourself later from the repo."

---

## Calm Phrases to Use

- ☑ "Let me grab the working version"
- ☑ "This is why we have version control"
- ☑ "Rather than debug live, let's look at the solution"
- ☑ "The concept is what matters here"
- ☑ "Let me jump ahead to show you the final result"
- ☑ "Good question - let me show you"
- ☑ "Perfect opportunity to demonstrate git fallback!"

---

## What NOT to Say

- ❌ "This has never happened before"
- ❌ "I don't know what's wrong"
- ❌ "This worked this morning"
- ❌ "Why isn't this working?"
- ❌ Long silence while debugging

---

## Body Language During Recovery

- ☑ Smile / stay relaxed
- ☑ Keep talking (explain what you're doing)
- ☑ Maintain eye contact with audience
- ☑ Use recovery as teaching moment
- ☑ Act like it's part of the plan

- ❌ Panic or apologize excessively
- ❌ Stop talking during fix
- ❌ Turn back to audience
- ❌ Show frustration

---

## Remember

- ◇ **The audience is on your side** - they want you to succeed
- ◇ **Mistakes happen** - how you recover matters more

- **Content > perfection** - the concepts are what matter
- **You have backups** - all branches are pre-tested and pushed
- **Stay confident** - you know this material

---

## Emergency Contact Info

**GitHub Repo:** github.com/billkuck/NovuckFSharpPresentation
**Backup:** All branches pushed to origin
**Local Backup:** All stage branches exist locally

---

## Post-Recovery Actions

After recovering:

1. Breathe
2. Verify you're on correct branch: `git branch --show-current`
3. Verify code builds: `dotnet build`
4. Continue presentation from recovery point
5. Don't dwell on the error - move forward

---

## Practice These Commands!

Run through this recovery drill before presentation:

```
# Start on Stage 1
git checkout demo/stage-1-naive-csharp

# Simulate error: make a typo
# ... make breaking change to Program.cs ...

# Recover
git reset --hard

# Verify recovery
dotnet build
dotnet run --project CustomerLiveDemo

# Practice jumping stages
git checkout demo/stage-2-first-du
git checkout demo/stage-3-pattern-matching
git checkout demo/stage-1-naive-csharp
```

**Run this drill Tuesday and Wednesday!**