Chapter 6: When Things Break

> *"Bugs are not failures. They're conversations with your code about what it needs."*

The Most Important Lesson

Things will break. That's not a sign you're doing it wrong - it's a sign you're building.

Every developer, from beginners to 20-year veterans, spends significant time debugging. The difference is that experienced developers don't panic. They have a process.

As a vibe coder, your process is simpler: you describe the problem to AI, and AI fixes it. But knowing how to describe problems effectively is a skill worth developing.

The Debugging Mindset

What's Different for Vibe Coders

Traditional debugging:

1. Read error message
2. Find the line of code
3. Understand what the code is doing
4. Figure out why it's wrong
5. Fix the code

Vibe coding debugging:

1. Notice something's wrong
2. Describe what's happening vs what should happen
3. Copy any error messages
4. Ask AI what's wrong and how to fix it
5. Apply the fix

You don't need to understand the code. You just need to describe the symptoms accurately.

The Three Types of Breaks

1. Errors (something crashes) Example: App shows error message, nothing loads

2. Wrong behavior (works but does wrong thing) Example: Login button logs you out instead

3. Silent fails (should work but doesn't) Example: Form submits but data doesn't save

Each needs a slightly different approach, but the core process is the same.

Reading Error Messages

Error messages look scary. They're not.

Anatomy of an Error

```
Unhandled Runtime Error
Error: Invalid `prisma.user.findUnique()` invocation:

  at PrismaClient.user.findUnique
  at Object.handler (/app/api/user/route.ts:12:34)
  at async eval (webpack-internal:///(rsc)/./app/api/user/route.ts:12:34)
```

What this actually means:

- **First line:** Something crashed
- **Second line:** What went wrong (the important part)
- **Remaining lines:** Where it happened (less important for you)

What you care about: "Invalid prisma.user.findUnique() invocation"

Translation: The database query is wrong somehow.

How to Read Any Error

Look for these patterns:

"Cannot read property X of undefined" → Trying to access something that doesn't exist → Usually means missing data

"Failed to fetch" → Network request failed → Usually API endpoint wrong or server down

"[Something] is not a function" → Calling something that isn't callable → Usually wrong import or typo

"[Something] is required" → Missing required parameter → Usually form validation or API call

"Authentication required" → User not logged in → Usually session expired or redirect issue

Don't memorize these. Just know: error messages tell you what's wrong in plain English (mostly). Copy them to AI.

The Debug Process

Step 1: Notice the Problem

Be specific about what's broken:

❌ Bad: "It's not working" ✅ Good: "When I click the submit button, nothing happens"

❌ Bad: "The data is wrong" ✅ Good: "The dashboard shows 50 users but I only have 5"

❌ Bad: "Login is broken" ✅ Good: "After I enter my password and click login, the page refreshes but I'm not logged in"

What you're doing: Creating a clear description of the symptom.

Step 2: Check the Obvious

Before diving deep, check these:

For "nothing happens":

- Is the dev server running? (Check terminal)
- Did you save the file? (Cmd+S)
- Did you refresh the browser? (Cmd+R)

For "data not saving":

- Are you logged in?
- Check browser Network tab - did the request go out?
- Check Supabase dashboard - is data there but not showing?

For "page not loading":

- Is the URL correct?
- Check browser console for errors (F12)
- Try in incognito mode (rules out extension issues)

30% of bugs are solved by: Save, refresh, restart server.

Step 3: Gather Information

Open browser DevTools (F12 or Cmd+Option+I)

Check Console tab:

- Red errors? Copy them.
- Warnings? Note them.
- Nothing? That's information too.

Check Network tab:

- Click "XHR" or "Fetch" filter
- Look for red (failed) requests
- Click on them to see why they failed

Check Supabase dashboard (if using):

- Go to Table Editor
- Check if data actually saved
- Go to Logs to see database queries

What you're doing: Collecting evidence about what's happening.

Step 4: Describe to AI

Open Cursor (or ChatGPT/Claude)

Use this template:

```
I'm trying to: [what you were doing]

Expected: [what should happen]

Actually happening: [what is happening]

Error message: [copy exact error if there is one]
```

```
What I've checked:
- [thing 1]
- [thing 2]

Relevant code: [paste the file that's probably related]
```

Example:

```
I'm trying to: Save a new bookmark when user clicks the submit button

Expected: Bookmark appears in the list below the form

Actually happening: Form submits but bookmark doesn't appear. No error shown.

Error message: In browser console I see:
"Failed to fetch POST http://localhost:3000/api/bookmarks"

What I've checked:
- I'm logged in
- The form fields have values
- Other API calls work fine

Relevant code: [paste components/BookmarkForm.tsx]
```

What AI does: Analyzes the symptoms, checks the code, identifies the issue, suggests a fix.

Step 5: Apply the Fix

AI will give you a solution. Usually it's one of:

"Replace this code with this new code" → Copy the new code, paste it in the file, save

"Add this new file" → Create the file, paste the content, save

"Change this environment variable" → Update `.env.local`, restart server

"Run this command" → Copy to terminal, press Enter

Then test it. Did it work?

- ✅ Yes → Great! Move on.
- ❌ No → Reply to AI with what happened, get next fix.

Common Issues & Quick Fixes

"Port already in use"

Symptom: Can't start dev server, says port 3000 in use

Fix in terminal:

```
# Kill whatever's using port 3000
kill $(lsof -t -i:3000)

# Then start your server again
npm run dev
```

"Module not found"

Symptom: Error says "Cannot find module '@/components/Something'"

Fix:

1. Check if file exists at that path
2. Check spelling (case-sensitive)
3. If it's a package: `npm install [package-name]`

"Environment variable undefined"

Symptom: Says "NEXT_PUBLIC_SUPABASE_URL is undefined"

Fix:

1. Check `.env.local` has the variable
2. Variable name matches exactly
3. Restart dev server (Ctrl+C, then `npm run dev`)
4. No quotes around values in `.env.local`

"Authentication error" / "Session expired"

Symptom: Logged out unexpectedly, can't access protected pages

Fix:

1. Clear cookies: DevTools → Application → Cookies → Delete all
2. Log out and log in again
3. Check Supabase dashboard → Authentication → Users

"Data not appearing"

Symptom: Save works but data doesn't show in list

Possibilities:

1. Data saved but fetch query is wrong → Check Supabase Table Editor
2. Data didn't save → Check browser Network tab for failed request
3. UI not updating → Check React state management in component

Debug: Add `console.log()` statements to see what data you're getting

"CSS not applying"

Symptom: Tailwind classes not working

Fix:

1. Check `tailwind.config.js` includes your files
2. Make sure dev server is running
3. Try a different class to verify Tailwind works
4. Check for typos in class names

Advanced Debugging Techniques

Using console.log()

This is your best friend for understanding what's happening.

Where to put them:

```
// Before database call
console.log("About to fetch bookmarks for user:", userId)

// After database call
console.log("Fetched bookmarks:", bookmarks)

// In event handler
console.log("Button clicked, form values:", formData)
```

What this does: Shows you what data exists at each step. Helps you find where things go wrong.

Remove them before deploying (or AI will do it for you).

Network Tab Deep Dive

Why it's useful: Shows all requests your app makes

How to read it:

1. Open DevTools → Network tab
2. Perform the action that's broken
3. Look for red (failed) requests

Click on a request to see:

- Request URL (is it correct?)
- Status code (200 = success, 404 = not found, 500 = server error)
- Response body (what error message came back?)

Common status codes:

- 200: Success
- 400: Bad request (you sent wrong data)
- 401: Not authenticated
- 403: Not authorized
- 404: Not found
- 500: Server error

Supabase Logs

When to check: Database-related issues

How:

1. Go to Supabase dashboard
2. Click "Logs" in sidebar
3. Filter to "Errors only"

Look for:

- Row Level Security blocks
- Invalid queries
- Missing columns
- Foreign key violations

When to Start Over vs When to Fix

Keep Fixing When:

✅ The error makes sense and AI can fix it ✅ You're learning something from the debugging ✅ It's been less than 30 minutes on this issue ✅ The fix is localized (one file, one feature)

Start Over When:

❌ Multiple cascading errors that keep appearing ❌ You've lost track of what changes you made ❌ AI gives contradictory advice ❌ It's been 2+ hours with no progress ❌ You've broken something fundamental

Starting over isn't failure. Sometimes the fastest path forward is:

1. Create new project
2. Copy over only the working parts
3. Rebuild the broken feature from scratch with better prompts

You'll know more the second time. Many experienced developers do this regularly - it's faster than untangling a mess.

Prevention Is Better Than Cure

How to Avoid Common Issues

Commit often:

```
git add .
git commit -m "Working bookmark save"
git push
```

Why: You can always go back to a working version.

Test incrementally: Don't build five features then test. Build one, test, commit. Build next, test, commit.

Use TypeScript: Catches many errors before you even run the code. AI writes TypeScript as easily as JavaScript.

Be specific with AI: "Add a bookmark save feature with validation" is better than "make bookmarks work"

Read AI's code before applying: Not the whole thing - just scan for anything obviously wrong. AI makes mistakes too.

Getting Unstuck

If AI Isn't Helping

Try a different prompt: Instead of "fix this error", try:

- "Explain what this error means in simple terms"
- "What are the 3 most likely causes of this?"
- "Show me how to debug this step by step"

Break it down: Instead of "why isn't authentication working", try:

- "Is the Supabase client configured correctly?"
- "Is the session being stored?"
- "Is the protected route checking auth?"

Start a new conversation: Sometimes AI gets confused by long threads. Start fresh with a clear problem statement.

If You're Completely Stuck

Options:

1. **Take a break** (seriously, step away for an hour)
2. **Google the exact error message** (someone else has had it)
3. **Check the library docs** (Supabase docs, Next.js docs)
4. **Ask in communities:**
    - r/webdev on Reddit
    - Supabase Discord
    - Next.js Discord

When asking:

- Be specific about what's wrong
- Include code snippet (use pastebin.com for longer code)
- Say what you've tried already
- Say what error message you're getting

People are helpful if you show you've tried to solve it yourself.

Learning from Breaks

Every bug teaches you something:

"I forgot to restart the server after env changes" → Now you know env changes need restart

"I was missing await on the database call" → Now you understand async operations matter

"Row Level Security was blocking my query" → Now you understand security policies

"I was using user.id instead of user.uid" → Now you know to check exact property names

Keep a "lessons learned" list. When you hit an issue twice, it's time to remember it.

Debugging Checklist

When something breaks, run through this:

- [ ] Is the dev server running?
- [ ] Did I save all files?
- [ ] Did I refresh the browser?
- [ ] Any red errors in console?
- [ ] Any failed requests in Network tab?
- [ ] Am I logged in (if needed)?
- [ ] Are environment variables set?
- [ ] Did I commit before breaking it?

If all checked:

- [ ] Copy error message
- [ ] Check what changed recently
- [ ] Ask AI with full context

If still stuck:

- [ ] Try incognito mode
- [ ] Restart dev server
- [ ] Clear browser cache
- [ ] Start fresh conversation with AI

Real Examples

Example 1: Silent Form Failure

Problem: "Form submits but bookmark doesn't save"

Process:

1. Open Network tab
2. Submit form
3. See 401 error on POST request
4. Check response: "User not authenticated"
5. Test: Am I logged in? No.
6. Fix: Redirect to login page before showing form

Lesson: Silent fails often mean auth issues. Check that first.

Example 2: Data Shows Wrong

Problem: "Bookmarks showing but they're all the same"

Process:

1. Check database - data is correct

2. Add console.log to see what's being fetched
3. Console shows: correct data fetched
4. Add console.log to see what's being rendered
5. Console shows: Only using first item in loop
6. Tell AI: "I'm mapping over bookmarks array but all cards show the same data"
7. AI: "You're using bookmark[0] instead of bookmark in your map function"
8. Fix: Use correct variable

Lesson: When data exists but displays wrong, problem is in the UI logic.

Example 3: Mysterious Crashes

Problem: "App crashes when I click edit"

Process:

1. Console shows: "Cannot read property 'title' of undefined"
2. Tell AI: "Getting undefined error when editing bookmark"
3. AI asks: "Can you show me the edit function?"
4. Paste code
5. AI: "You're accessing bookmark before checking if it exists"
6. Fix: Add null check

Lesson: "Undefined" errors mean you're accessing data that doesn't exist yet.

The Psychology of Debugging

Frustration is normal. Even after 20 years, developers get frustrated debugging. You're not alone.

Breaks help. If you've been stuck for 30+ minutes, walk away. Solutions often appear when you stop thinking about them.

Document your wins. When you solve something tricky, write down what was wrong and how you fixed it. Future you will thank you.

Celebrate the fix. Debugging is problem-solving. Each fix is progress. Don't discount it just because "it should have worked the first time."

Remember: The goal isn't perfect code. The goal is shipped code. Bugs are temporary obstacles, not permanent failures.

When Something Works, Don't Question It

Developer instinct: "Wait, why did that work?" **Vibe coder instinct:** "Cool, it works! Moving on."

Both are valid. You don't need to understand every fix. If AI fixed it and it works, that's a win. You can always ask "why did that fix it?" later if you're curious.

Shipping beats understanding. Perfect understanding prevents shipping. Shipped products with some mystery are better than unshipped products you fully understand.

Now let's talk about what to do once you've shipped your first project.

Connect & Share

💌 **Newsletter**: Build to Launch - Weekly AI building tips, templates, and real builder stories

🦋 **Bluesky**: @jenny-ouyang - Daily insights

💼 **LinkedIn**: Jenny Ouyang - Professional network