AI Coding Cheat Sheet Collection

Quick Reference • Shortcuts • Commands • Patterns

END OF CODING
Build Apps with AI • No Code Experience Needed

1. Cursor & Windsurf Shortcuts

Master these keyboard shortcuts to code at lightning speed.

Action	Cursor	Windsurf	Description
AI Chat	Cmd/Ctrl+L	Cmd/Ctrl+L	Open AI chat panel
Inline Edit	Cmd/Ctrl+K	Cmd/Ctrl+K	Edit selected code with AI
Generate Code	Cmd/Ctrl+I	Cmd/Ctrl+I	Generate code at cursor
Accept Suggestion	Tab	Tab	Accept AI suggestion
Reject Suggestion	Esc	Esc	Dismiss AI suggestion
Next Suggestion	Alt+]	Alt+]	Cycle to next suggestion
Prev Suggestion	Alt+[	Alt+[	Cycle to previous suggestion
Chat with File	Cmd/Ctrl+Shift+L	@filename	Reference specific file in chat
Chat with Docs	N/A	@docs	Search framework documentation
Multi-file Edit	Cmd/Ctrl+Shift+K	Cascade	Edit across multiple files
Debug Mode	Cmd/Ctrl+Shift+D	Cmd/Ctrl+Shift+D	Debug with AI assistance
Terminal Chat	Cmd/Ctrl+Shift+P	Cmd/Ctrl+`	AI help in terminal

2. Best Prompting Patterns

Use these patterns to get better results from AI code assistants.

Context First: Always provide context before asking: 'I'm building [X] with [tech]. I need [Y] that does [Z].'

Specify Format: Tell AI exactly what you want: 'Give me a TypeScript function with JSDoc comments and unit tests.'

Include Constraints: Mention limitations upfront: 'Must work in Node 18, bundle size under 5KB, no external dependencies.'

Show Examples: Provide an example: 'Like this pattern [example], but for [use case].'

Request Explanation: Ask for reasoning: 'Explain why this approach is better than [alternative].'

Iterate Explicitly: Build on previous: 'Now add [feature] while keeping [existing feature].'

Error Context: When debugging: 'Error: [full error]. Code: [code]. Expected: [behavior]. Actual: [behavior].'

Request Alternatives: Get options: 'Show me 3 approaches for [problem], with pros/cons for each.'

3. Supabase Quick Reference

Essential Supabase commands for auth, database, and storage.

Authentication

```
// Sign up
const { data, error } = await supabase.auth.signUp({
email: 'user@email.com', password: 'password'
})

// Sign in
await supabase.auth.signInWithPassword({
email: 'user@email.com', password: 'password'
})

// OAuth (Google, GitHub, etc.)
await supabase.auth.signInWithOAuth({ provider: 'google' })

// Sign out
await supabase.auth.signOut()

// Get current user
const { data: { user } } = await supabase.auth.getUser()
```

Database Operations

```
// Select
const { data } = await supabase.from('table').select('*')
const { data } = await supabase.from('table').select('id, name').eq('status', 'active')

// Insert
const { data } = await supabase.from('table').insert({ name: 'value' })

// Update
const { data } = await supabase.from('table').update({ name: 'new' }).eq('id', 1)

// Delete
const { data } = await supabase.from('table').delete().eq('id', 1)

// Realtime
supabase.channel('table').on('postgres_changes',
{ event: '*', schema: 'public', table: 'table' },
(payload) => console.log(payload)
```

```
).subscribe()
```

Storage

```
// Upload file
const { data } = await supabase.storage
.from('bucket')
.upload('path/file.jpg', file)

// Download file
const { data } = await supabase.storage
.from('bucket')
.download('path/file.jpg')

// Get public URL
const { data } = supabase.storage
.from('bucket')
.getPublicUrl('path/file.jpg')

// Delete file
await supabase.storage.from('bucket').remove(['path/file.jpg'])
```

4. Vercel Deployment Commands

Deploy your apps to production with these commands.

`vercel` - Deploy to preview (generates preview URL)

`vercel --prod` - Deploy to production

`vercel dev` - Run local development server with Vercel functions

`vercel env pull` - Pull environment variables from Vercel

`vercel env add` - Add environment variable

`vercel link` - Link local project to Vercel project

`vercel logs` - View deployment logs

`vercel inspect` - Inspect deployment details

`vercel rollback` - Rollback to previous deployment

`vercel domains add` - Add custom domain

5. Git Commands for AI Workflows

Essential Git commands when working with AI-generated code.

`git init` - Initialize new repository

`git add .` - Stage all changes

`git commit -m 'message'` - Commit with message

`git status` - Check working tree status

`git diff` - See uncommitted changes

`git diff --staged` - See staged changes

`git branch feature-name` - Create new branch

`git checkout -b feature-name` - Create and switch to branch

`git merge branch-name` - Merge branch into current

`git push origin main` - Push to remote main

`git pull origin main` - Pull latest from remote

`git reset --hard HEAD` - Discard all local changes (dangerous!)

`git stash` - Save changes for later

`git stash pop` - Restore stashed changes

6. Common Debugging Patterns with AI

How to effectively debug with AI assistance.

React Hydration Mismatch

```
Prompt: Getting hydration error in Next.js. Check for: client-only code running on server, random values, Date.now(), mismatched HTML structure.
```

TypeScript Errors

```
Prompt: Show me the type error and suggest fix with proper typing. Include the full type definition.
```

API Route Failing

```
Prompt: API returning 500. Check: error handling, try-catch blocks, database connection, environment variables, CORS.
```

Build Failing

```
Prompt: Build error: [error]. Check: missing dependencies, TypeScript errors, environment variables, import paths.
```

Slow Performance

```
Prompt: Page is slow. Profile: unnecessary re-renders, missing memoization, large bundles, unoptimized images, N+1 queries.
```

Authentication Not Working

```
Prompt: Auth failing. Verify: cookies set correctly, session stored, middleware protecting routes, token refresh logic.
```

Quick Tips

• Use Cmd/Ctrl+K for quick inline edits instead of explaining in chat

• Reference files with @filename in Cursor chat for better context

• Always review AI-generated code before committing

• Save your best prompts in a snippets file for reuse

• Use .cursorrules or .windsurfrules file to set project-wide AI instructions

• When stuck, try rephrasing your prompt with more specific requirements

• Leverage AI for boilerplate, but write critical business logic yourself

• Use AI to generate tests - it's great at covering edge cases