Gemini CLI Cheatsheet

Basics

```
gemini chat "fix bug"          # One-shot prompt
gemini interactive             # Enter interactive mode
gemini -i                      # Short form for interactive
gemini "analyze this file" @   # Include current directory files
gemini --help                  # Show all commands
```

Flags & Options

```
--model=<id>                   # Choose model (gemini-1.5-pro, claude-code)
--input=file.txt               # Pipe file content
--output=result.md             # Save output to file
--non-interactive              # Run without user interaction
--verbose                      # Detailed logging
--config=path                  # Custom config file
```

Interactive Mode Commands

```
/help                          # Show all commands
/clear                         # Reset conversation context
/save                          # Export conversation
/load                          # Import conversation
/exit                          # Gracefully exit
Ctrl+C                         # Force quit
```

Error Fix Macros

```
gemini fix "429 rate limit exceeded"
gemini fix "npm ERR! peer dep missing"
gemini fix "git merge conflict"
gemini fix "typescript type error"
gemini fix "docker build failed"
```

File Operations

```
gemini "refactor this" @file.js        # Target specific file
gemini "add tests" @src/               # Target directory
```

```
gemini "explain code" @*.py             # Target pattern
gemini --input=log.txt "analyze errors" # Pipe file content
```

Model Selection

```
--model=gemini-1.5-pro          # Default, balanced performance
--model=gemini-1.5-flash        # Faster responses
--model=claude-code             # Code-focused model
--model=gpt-4                   # OpenAI model
```

Configuration

```
gemini config set api_key YOUR_KEY
gemini config set model gemini-1.5-pro
gemini config set output_format markdown
gemini config show              # Display current config
```

Advanced Workflows

```
# Code analysis pipeline
gemini "audit security" @src/ --output=security-report.md

# Documentation generation
gemini "generate docs" @*.js --model=claude-code

# Error diagnosis
gemini --input=error.log "diagnose and fix"

# Multi-file refactoring
gemini interactive
> "I want to refactor authentication across these files"
> @auth/ @middleware/ @routes/auth.js
```

Common Patterns

```
# Quick debugging
gemini "debug this error" @logs/

# Code review
gemini "review for best practices" @pull-request-files/

# Performance optimization
gemini "optimize performance" @slow-component.js

# Test generation
gemini "generate unit tests" @src/utils/
```

Troubleshooting

```
# Check installation
gemini --version

# Verify API key
gemini config show

# Clear cache
gemini cache clear

# Reset configuration
gemini config reset

# Enable verbose logging
gemini --verbose "test command"
```