Prompt Cheat-Sheet for Developers

---

1. Universal Prompt Template (Best Practice)

Use this template for **most developer tasks**:

```
Role:

Task:

Context:

Constraints:

Input:

Output Format:
```

---

2. Role Definitions (Set Behavior First)

Use Case	Role Prompt
Backend dev	You are a senior backend developer
Frontend dev	You are a senior frontend engineer
Architect	You are a software architect
QA	You are a QA automation engineer
DevOps	You are a DevOps engineer
Data	You are a data engineer

Example

Role: You are a senior Java backend developer.

---

3. Code Generation Prompts

3.1 Generate Code (Clean & Deterministic)

```
Generate a Java method to validate email addresses.

Constraints:
- Java 17
- No regex
- Return boolean only

Output Format:
Return code only.
```

3.2 REST API Generation

```
Generate a Spring Boot REST controller.

Constraints:
- Java 17
- Spring Boot 3
- Follow REST best practices
- No explanation

Output Format:
Code only.
```

---

4. Structured Output (JSON / APIs)

4.1 JSON-Only Output

```
Return ONLY valid JSON.

Do not add explanations.


Schema:

{

 id: string,

 name: string,

 totalAmount: number

}
```

4.2 Function-Call Style Output

```
Return ONLY JSON arguments for function:


createOrder(customerId, product, quantity)


User input:

"Create order for C001 with 2 laptops"
```

---

5. Code Explanation Prompts

5.1 Simple Explanation

```
Explain the following code in simple terms.

Use bullet points.

Max 5 points.
```

5.2 Step-by-Step Explanation

```
Explain the logic step by step.

Use numbered steps.
```

---

6. Refactoring & Optimization Prompts

6.1 Refactor Code

```
Refactor the following code to:

- Improve readability

- Reduce complexity

- Follow best practices

Do not change behavior.
```

6.2 Performance Optimization

```
Optimize the following code for performance.

Explain changes briefly.
```

---

7. Testing Prompts

7.1 Unit Test Generation

```
Generate JUnit 5 test cases.


Constraints:

- Cover edge cases

- No mocks

- Code only
```

7.2 Test Data Generation

```
Generate sample test data.


Constraints:

- Valid values only

- Return JSON array
```

---

8. Debugging Prompts

8.1 Error Explanation

```
Explain this error and suggest a fix.

Be concise.
```

8.2 Log Analysis

```
Analyze the following logs.

Identify root cause and solution.
```

---

9. Documentation Prompts

9.1 JavaDoc / Code Comments

```
Generate JavaDoc for the following method.

Keep it concise.
```

9.2 README Generation

```
Generate a README.md for this project.


Include:

- Overview

- Setup steps

- Usage
```

---

10. Prompting with Constraints (Critical for Developers)

Common Constraints to Add

```
- Language version

- Framework version

- Security rules

- Performance limits

- Output length
```

Example

Constraints:

- O(n) time complexity

- No external libraries

- Max 50 lines of code

---

11. Output Control Keywords (Power Words)

Keyword	Effect
ONLY	Strict enforcement
MUST	Hard requirement
DO NOT	Prevents unwanted output
MAX	Limits size
EXACT	Avoids variations

---

12. Prompt Patterns (Quick Reference)

Task	Prompt Pattern
Code gen	Generate ... Constraints ... Code only
JSON	Return ONLY JSON
Explain	Explain step by step
Refactor	Refactor without changing behavior
Test	Generate unit tests
Debug	Analyze error and suggest fix

---

13. Common Mistakes to Avoid

- Vague prompts
- Multiple tasks in one prompt
- Missing output format
- High creativity for code
- No constraints

---

14. Developer Productivity Tips

- Save reusable prompts
- Standardize prompts across team
- Use low temperature for code
- Validate outputs
- Treat prompts like APIs

---

15. One-Line Prompts (Quick Use)

Explain this code in 5 bullet points.

Refactor this code for readability.

Generate JSON only.

Write unit tests for this method.

Fix this error.

---

16. Final Takeaway

Good prompts = **better code, less rework, faster delivery**.

Treat prompts as:

- Reusable assets
- Development tools
- Part of your SDLC