Learnbay

# Top 20 clean Code practices every Developer should follow

Here are the **top 20 clean code practices** that every developer should follow in 2025.

www.learnbay.co

Learnbay

Practice 1

# Use meaningful variable and function names

**Why it matters:**

Names should describe their purpose clearly so others can easily understand your code.

## Example:

**Bad**: let a = 10 ;  **Good:** let userAge = 10;

Learnbay

Practice 2

# Keep functions small and focused

- **Why it matters:** A function should do one thing and do it well. This ensures better readability and testability.
- **Rule:** Functions should ideally fit within 20-30 lines of code.

## Pro tip

If a function has "and" or "or" in its name, it likely does too much.

Learnbay

Practice 3

# Write self-documenting code

- **Why it matters:** Code should be written in a way that explains itself without excessive comments.

## Example

Instead of writing // Increment counter, use counter++ in a meaningful context.

Learnbay

Practice 4

# Avoid magic numbers

- **Why it matters:** Hardcoded numbers can be confusing and lead to bugs.
- **Solution:** Use constants or enums.

## Example

Bad: `if (score > 75)`
Good: `const PASSING_SCORE = 75; if (score > PASSING_SCORE)`

Learnbay

Practice 5

# Limit comments, focus on clarity

- **Why it matters:** Good code doesn't need excessive comments. Use comments only to explain why something is done, not what.

## Example

Bad: // Adding 10 to age
Good: Code itself explains the purpose: userAge += 10;

Learnbay

Practice 6

# Use consistent indentation and formatting

- **Why it matters:** Proper formatting makes your code easier to scan and maintain.
- **How to do it:**
  - Use tools like Prettier or Black to automate code formatting.
  - Follow your team's style guide (e.g., Google JavaScript Style Guide).

Learnbay

Practice 7

# Break down long methods

- **Why it matters:** Long methods make debugging difficult and are hard to read.
- **How to do it:**
  - Refactor methods into smaller, reusable pieces.
  - Extract common logic into helper functions.

Learnbay

Practice 8

# Use descriptive boolean variables

- **Why it matters:** Boolean variables like isActive, hasAccess, or shouldRetry improve readability.

## Example

Bad: if (flag)

Good: if (isUserLoggedIn)

Learnbay

Practice 9

# Avoid deep nesting

- **Why it matters:** Deeply nested code is hard to follow and debug.
- **Solution:** Use early returns or guard clauses to reduce nesting.

## Example

Bad:

```
if (user) {
  if (user.isActive) {
    if (user.role === "admin") {
      return true;
    }
  }
}
```

Learnbay

## Example

Good:

```
if (!user || !user.isActive || user.role !== "admin") return false;
return true;
```

## Practice 10

# Avoid deep nesting

- **Why it matters:** Tests ensure that your code works as expected and prevents future bugs.
- **How to do it:**
  - Use testing frameworks like Jest, JUnit, or PyTest.
  - Aim for 80-90% code coverage.

Learnbay

Practice 11

# Use meaningful class and file names

- **Why it matters:** Names like DataProcessor or UserService convey intent better than Misc or Helper.
- **Rule:** File names should reflect their contents.

Learnbay

Practice 12

# Adhere to the DRY principle (Don't Repeat Yourself)

- **Why it matters:** Duplicated code makes maintenance harder.
- **How to do it:**
  - Refactor common logic into reusable functions or components.
  - Example: Consolidate repeated API calls into a utility class.

Learnbay

Practice 13

# Follow the SOLID principles

- **Why it matters:** SOLID principles create scalable, maintainable, and robust code.
  - Single Responsibility: Each class should have one responsibility.
  - Open/Closed: Classes should be open for extension but closed for modification.
  - Liskov Substitution: Subtypes should be substitutable for their base types.
  - Interface Segregation: Keep interfaces small and specific.
  - Dependency Inversion: Rely on abstractions, not concretions.

Learnbay

Practice 14

# Handle errors gracefully

- **Why it matters:** Unhandled exceptions lead to crashes and poor user experiences.
- **How to do it:**
  - Use try-catch blocks to handle exceptions.
  - Log errors using tools like Sentry or Logstash.

## Example

```
try {
  const data = fetchData();
} catch (error) {
  console.error("Error fetching data:", error);
}
```

Learnbay

Practice 15

# Avoid side effects

- **Why it matters:** Side effects make code harder to predict and debug.
- **How to do it:** Write pure functions that only depend on their inputs and return predictable outputs.

Learnbay

Practice 16

# Avoid side effects

- **Why it matters:** Enums make code more readable and reduce hardcoding.

## Example

```
enum UserRole {
  Admin = "ADMIN",
  Editor = "EDITOR",
  Viewer = "VIEWER",
}
```

Learnbay

Practice 17

# Use dependency injection

- **Why it matters:** Dependency injection improves testability and modularity.

## Example

Pass required services into constructors instead of hardcoding dependencies.

Learnbay

Practice 18

# Write modular code

- **Why it matters:** Modular code is easier to test, debug, and scale.
- **How to do it:**
  - Split large files into smaller, focused modules.
  - Follow the Separation of Concerns (SoC) principle.

Learnbay

Practice 19

# Refactor regularly

- **Why it matters:** Refactoring improves code quality over time and removes technical debt.

## Pro Tip

Allocate time in every sprint for refactoring and cleaning up the codebase.

Learnbay

Practice 19

# Document your code

- **Why it matters:** Documentation helps your team (and your future self) understand how and why your code works.
- **How to do it:**
  - Use tools like JSDoc, Doxygen, or Sphinx for documentation.
  - Include inline comments for complex logic.

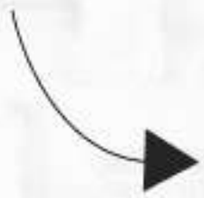
Learnbay

# Top benefits of clean code practices

| Benefit | Impact |
| --- | --- |
| Easier to debug | Reduces troubleshooting time |
| Improves collaboration | Enhances teamwork |
| Scales effectively | Supports future growth |
| Reduces technical debt | Saves costs in the long run |

## Which clean code practice do you swear by?

Share your favorite tips in the comments!



Upskill with Learnbay

# India's most trusted

## Program For **Working Professional**

## Data Structure Algorithms & System Design

With **Gen-AI** For Software Developers

Program electives:

- GenAI
- Product management
- DevOps
- FullStack (MERN)

**Get Certification from :**

IBM | IIT Guwahati | WOOLF UNIVERSITY | Microsoft