Code Smells refer to *symptoms in the source code that may indicate deeper problems*. They are not bugs — your program will still compile and run — but they *indicate weaknesses in design or structure* that may lead to increased technical debt, reduced readability, and difficulty in future maintenance or testing.

Common Types of Code Smells

Code Smell	**Description**	**`Example (JavaScript)`**
Duplicated Code	Same code structure repeated in multiple places.	`js function calcTotal1() { return price * qty; } function calcTotal2() { return price * qty; }`
Long Function	A function that does too much or spans many lines.	`js function processOrder(order) { /* 100+ lines of code */ }`
Large Class	A class with too many responsibilities or too many methods/properties.	`js class OrderManager { createOrder() {...} cancelOrder() {...} printInvoice() {...} trackShipment() {...} }`
Long Parameter List	Too many parameters passed into a function.	`js function createUser(name, age, address, email, phone, isAdmin, preferences)`
Feature Envy	One class uses methods of another excessively.	`js class Invoice { print(order.getCustomer().getAddress()) }`
Shotgun Surgery	A small change requires updating many classes or methods.	`E.g., changing date format affects multiple files.`
God Object / God Class	One object/class does everything.	`A UI controller managing logic, data, and rendering.`

How to Overcome Code Smells

Strategy	**Description**	**Tools/Techniques**
Refactoring	Restructure code without changing its behavior.	Extract method, move method, rename variable
Apply SOLID Principles	Helps structure code in a clean and manageable way.	Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion
Use Design Patterns	Apply standard solutions to common problems.	Factory, Strategy, Observer, etc.

Strategy	Description	Tools/Techniques
Follow Clean Code Practices	Focus on readability, simplicity, and meaningful names.	Small functions, meaningful names, clear logic
Code Reviews	Regular peer reviews help detect smells early.	GitHub PR reviews, Pair Programming
Automated Linters/Analyzers	Tools to detect code smells and enforce standards.	ESLint (JavaScript/TS), SonarQube, PMD

Example: Fixing a Long Function (JS)

Before (Smell):

```
function registerUser(user) {
  // validate user
  if (!user.email.includes('@')) return false;
  // save user
  db.save(user);
  // send welcome email
  mailer.send(user.email, "Welcome!");
}
```

After (Refactored):

```
function registerUser(user) {
  if (!isValidUser(user)) return false;
  saveUser(user);
  sendWelcomeEmail(user.email);
}

function isValidUser(user) { return user.email.includes('@'); }
function saveUser(user) { db.save(user); }
function sendWelcomeEmail(email) { mailer.send(email, "Welcome!"); }
```

Summary

- **Code Smells** are *indicators* of possible design problems.
- They can be removed by **refactoring**, **applying principles**, and **tooling**.
- Clean code leads to **better maintainability**, **testability**, and **collaboration**.