



What 15 Years of Web Development Taught Me

Ten lessons about the work, the people, and building a career you can sustain.

**Keep what helps.
Change what gets in the way.**

CRAFT

Understand the foundations



PEOPLE

Make the work clearer



CAREER

Learn, ship, and keep going

Learn what the framework is standing on.

Framework knowledge is easier to use when you understand the underlying system.

01 Follow the request

Know what HTTP sends, what the server does, and how the response reaches the browser.

02 Understand the data

SQL and data structures help you reason about what an abstraction is actually doing.

03 Build a mental model

When something breaks, explain the mechanism before searching for a matching snippet.

Reflection: which underlying technology do you use every day but understand least?

Spend novelty where it solves a real problem.

Every unfamiliar tool brings learning, operating, and maintenance costs.

01 Name the need

What does the project need that your current, familiar tools cannot provide?

02 Count the cost

Consider documentation, failure modes, and the people who will maintain it.

03 Separate work from play

Exploration is useful. A production dependency needs a reason beyond wanting to try it.

Reflection: can you explain your newest technology choice without mentioning excitement?

Use tests to shape the change.

The author's turning point was seeing a real application built test-first.

- | | | |
|----|-----------------------------------|---|
| 01 | Describe the behavior | Make the expected result clear before working on the implementation. |
| 02 | Let failure guide the work | A failing check gives a concrete target for the smallest useful change. |
| 03 | Keep the feedback loop | Rerun the relevant checks as the code changes. Choose assertions that prove behavior. |

Reflection: what important behavior would you hesitate to change without a test?

Write for the next person reading it.

Names and a clear flow help people understand the business work behind the code.

- | | | |
|----|------------------------------|--|
| 01 | Name the purpose | Choose words that explain what the value or operation represents. |
| 02 | Make the flow legible | A reader should follow the process without chasing unrelated implementation details. |
| 03 | Refactor for clarity | Prefer a change that makes intent easier to see over cleverness for its own sake. |

Article example: `CreateSubscription` names a specific operation, rather than a vague manager.

Make the next layer earn its place.

Start with ordinary framework patterns. Add structure when a concrete part becomes difficult.

- | | | |
|----|-------------------------------|--|
| 01 | Find the messy part | Point to the duplication, responsibility, or boundary that is causing trouble. |
| 02 | Extract with a purpose | A Service, Action, or DTO should make that specific problem easier to manage. |
| 03 | Check the result | Can a reader follow the work more easily?
Did the extra layer solve what you named? |

Reflection: which abstraction could you remove without losing a useful boundary?

Clear communication is part of the job.

Technical ability becomes more useful when teammates can work well with you.

- | | | |
|----|---------------------------------|--|
| 01 | Listen before solving | Understand the problem and expectations before proposing an answer. |
| 02 | Make uncertainty visible | Say what you know, what you do not know, and what would help you find out. |
| 03 | Practice the explanation | Write, present, and give feedback in words the other person can use. |

Reflection: what could you explain more clearly in your next update or review?

Build a career you can keep doing.

The article reflects on years of treating the body as a way to get the brain to the keyboard.

01 Notice the routine

Long stretches at a desk can become the default before you have chosen them.

02 Protect time outside work

Make room for movement, rest, and life alongside the work you care about.

03 Think beyond the deadline

A good week of output is only one part of a career that lasts.

Reflection: what part of your working routine needs more deliberate attention?

Choose who you will learn alongside.

A technology choice also shapes the people, examples, and support around your work.

- | | | |
|----|-----------------------------|--|
| 01 | Find useful teachers | Look for resources and people that help you understand, rather than just copy. |
| 02 | Ask and collaborate | Peers and maintainers can offer a different way to see a problem you are stuck on. |
| 03 | Make others better | Share what you learn and contribute where your experience can help. |

Reflection: who or what in your current community makes you a better developer?

Put the work in front of real users.

Feedback from a usable release can teach you something polishing alone cannot.

01 Define a useful first release

Choose a small result that people can actually try and respond to.

02 Keep the necessary checks

Shipping still includes testing and the approvals your project requires.

03 Learn from what happens

Use observed feedback to choose the next improvement.

Reflection: what is the smallest useful version of the project you keep postponing?

Keep learning without pretending to know it all.

Experience gives you a foundation. It does not remove the need to ask questions.

01 Stay interested

Follow a problem or feature that makes you want to understand more.

02 Leave room to be wrong

Treat a changed mind as part of learning, not something to conceal.

03 Keep perspective

Look for progress over time rather than an imaginary point where you know enough.

Reflection: what changed your mind recently, and what will you explore next?

Choose one change for your next week.

Ten lessons are a map. One concrete step is enough to start using it.

Keep

Name one habit or tool that is serving you well.

Why is it worth keeping?

Change

Name one source of friction in your code or routine.

What is the smallest useful fix?

Learn

Name one gap in your understanding or practice.

How will you explore it?

My next step: _____

[Read the full article: What 15 Years of Web Development Taught Me](#)