

Preface

1. This is a big, messy topic.

The Web exists in continuous infancy, perpetual beta. Some standards and tools change quickly, others glacially. Ditto for user expectations.

2. You are learning a foreign language.

iHola! Bonjour! 喂! `print('hello')`

There's a unique vocabulary and grammar, with its own inconsistencies and frustrations, but is equally capable of creating poetry.

3. You will find your inner geek.

Forget WYSIWYG, embrace WYGIWYM.

What You Get Is What You Mean.

Dreamweaver, etc. only create the illusion of control and end up limiting you.

That Said...

1. This is ~~big~~, ~~messy~~ **revolutionary**.

We've lost the constraints of analog: space, trees, read-only, complex distribution, etc. We can interact with our readers immediately now.

2. You **are** learning a foreign language.
(*iHola! Bonjour! 喂!*)

Who complains about being bilingual, or trilingual? Plus, these languages are universal, spanning industries and borders (consider your next job)...

3. You **will** find your inner geek.

Yes, the learning curve can be daunting, but it's not like we haven't tackled other inanely complex tools (CCI, anyone?). Plus, there's a ton of collective wisdom out there.

GETTING SHIT DONE

- 1. Strategy*
- 2. Philosophy*
- 3. Tools*

Strategy

1. Identify the *right* projects.
Smaller, the better. Start low-risk to guarantee the most autonomy. Success reverberates.
2. Reduce. Reuse. Recycle.
Look for common interactive storytelling methods. Maybe you can adopt maps, or searchable lists, or user forms into multiple projects.
3. Build first. Ask for permission later.
Ideas are cheap. Execution matters. Find a project you can do with minimal time and resources, and go rogue. Bring a publish-ready solution to the table and you'll find that barriers disappear.

Strategy

4. Simple semantics.

Create a sandbox at projects.newspaper.com or beta.newspaper.com. Words change expectations.

5. Avoid premature optimization.

Don't be distracted by issues of scalability and maintenance. Launch it, see if it's successful, then sort out the logistics of sustaining/growing.

6. Experiment.

Responding to change is more valuable than following a plan.

7. Collaborate with the right people.

(And realize sometimes that means no one).

Getting Real is about skipping all the stuff that *represents* real (charts, graphs, boxes, arrows, schematics, wireframes, etc.) and *actually building the real thing*.

Getting real is less. Less mass, less software, less features, less paperwork, less of everything that's not essential (and most of what you think is essential actually isn't).

Getting Real is staying
small and being agile.

37signals “Getting Real”

Getting Real starts with the interface, the real screens that people are going to use. It begins with what the user actually experiences and builds backwards from there.

This lets you get the interface right before you get the software wrong.

Getting Real is about iterations and lowering the cost of change. Getting Real is all about launching, tweaking, and constantly improving.

Getting Real delivers better results because it forces you to deal with the actual problems you're trying to solve instead of your ideas about those problems.

Tools

1. Wordpress & Drupal

Open-source, flexible and you can tap into excellent documentation and community.

2. Host in the cloud.

Leapfrog your current servers and outdated software. Cheap hosting (Dreamhost, WebFaction) or enterprise-level Amazon EC2 (still cheap).

3. Don't reinvent Web 2.0.

Your audience is already using Flickr, Facebook, Google Docs, YouTube, etc. so don't waste time duplicating their functionality or communities. Instead, leverage every widget and API you can.