Phoenix LiveView

A Solution to Madness - Joshua Plicque

What We'll Cover

- What is Phoenix LiveView and what does it promise?
- How LiveView Works
- Phoenix LiveView Programming Model - Thinking in LiveView
- Using Phoenix Contexts to reuse code and quickly spin up API's
- Writing Tests - And why it's blissful
- Coding with alpha software

//= OR EQUALS

- I run a small web development shop
- Developing Software Professionally for 7 years
- Been using LiveView full-time for just under a year

“ Phoenix LiveView is an exciting new library which enables rich, real-time user experiences with server-rendered HTML. LiveView powered applications are stateful on the server with bidirectional communication via WebSockets, offering a vastly simplified programming model compared to JavaScript alternatives. ”

Chris McCord


high
low
Multi Page App
User Experience
Complexity

RESOURCES
About Me
discourse.org
stackexchange.com
Learn Markdown
Recommended Reading

Subscribe in a reader
Subscribe via email

18 Jul 2011

Building a PC, Part VII: Rebooting

I've had more or less the same PC, with various updates, since 2007. I've written about most of it here:

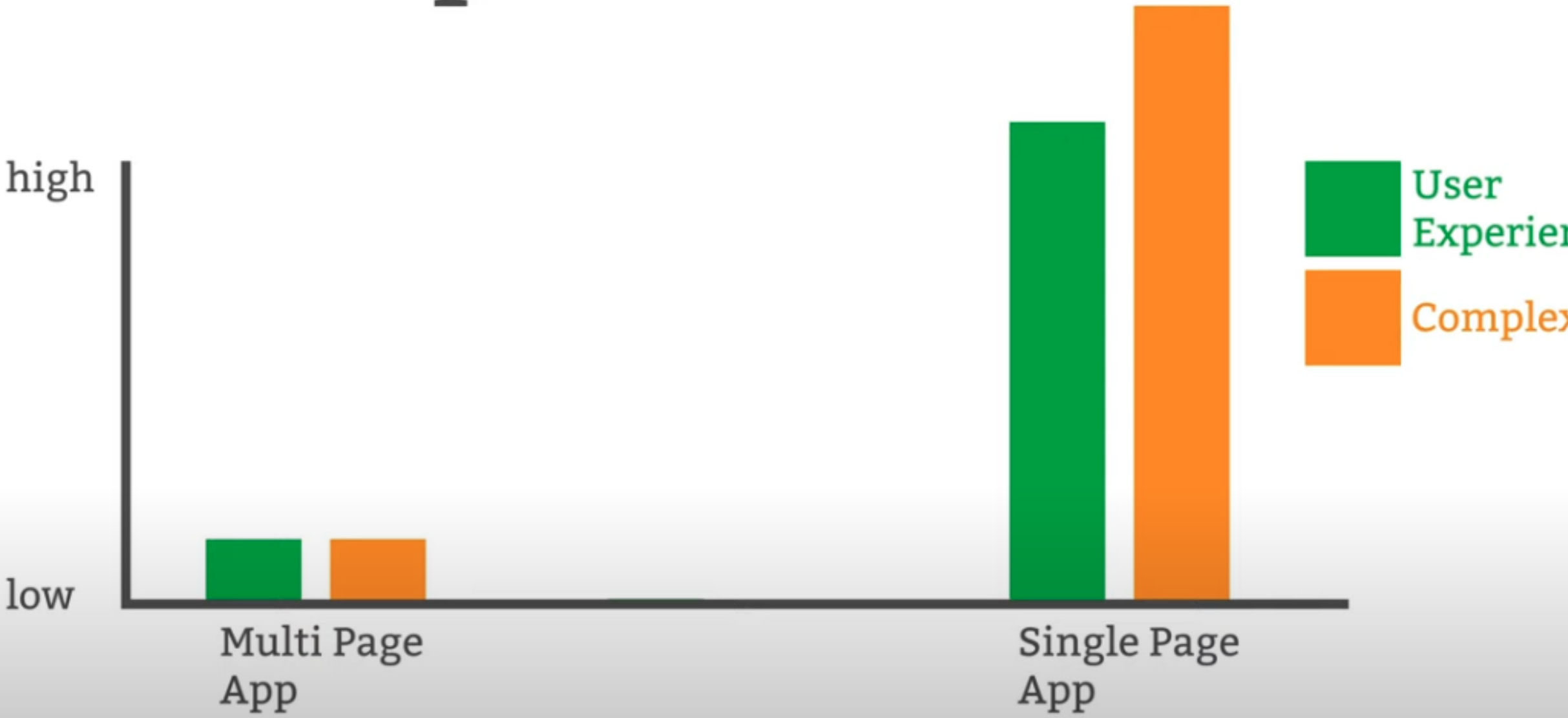
- Building a PC, Part I: Minimal boot
- Building a PC, Part II: Burn in
- Building a PC, Part III: Overclocking
- Building a PC, Part IV: Now It's Your Turn
- Building a PC, Part V: Upgrading
- Building a PC, Part VI: Rebuilding

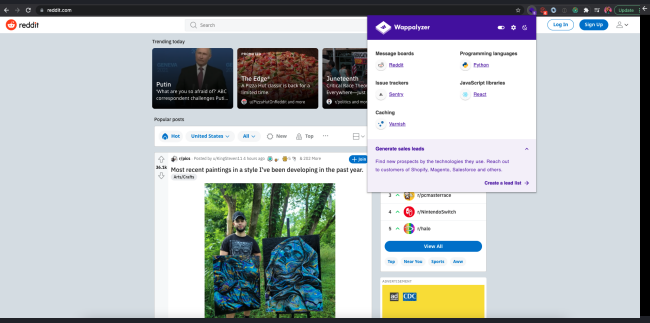
While the advice in those original articles is still quite sound, my old 2007 era case was feeling mighty creaky. I needed a new chassis. I also wanted a motherboard that supported native 6 Gbps SATA for the latest generation of SSDs that truly benefit from them. The buzz around the Sandy Bridge based Core i7-2600k was nearly deafening, and I've fallen *completely* in love with my last HTPC build based on the same technology. (Oh, and even if you already read that article, read it again because I added new PicoPSU and case information that takes it from awesome to sublime – on the order of 17 watts idle!)

So I decided it was time to build myself a nice Sandy Bridge system. What I ended up with is **easily the best case and motherboard combination I've ever laid hands on**. Read on!

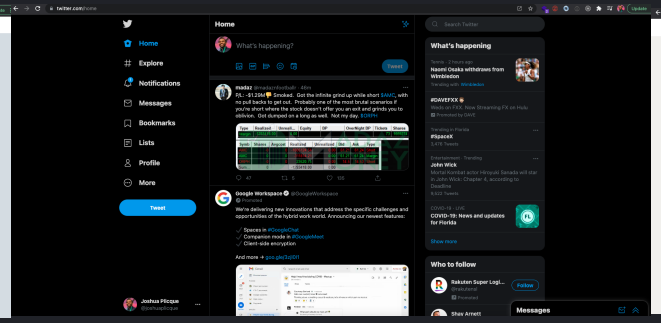
I cut out a lot of the initial research work by relying on my old, dear friends at Tech Report and their current workstation recommendations:

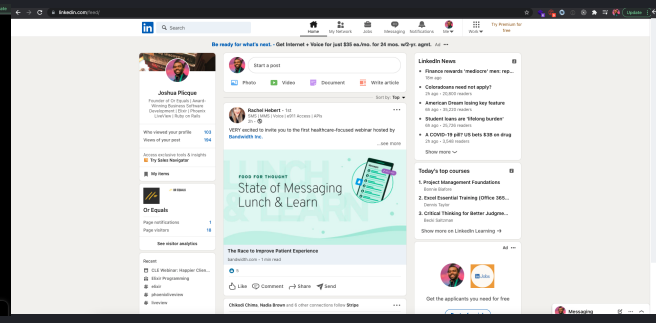
Compare & Contrast

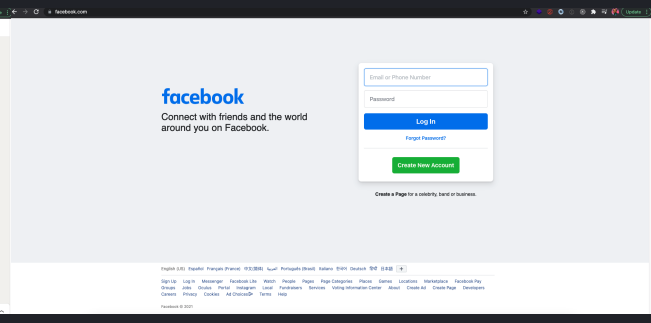



reddit.com
reddit
Search
Trending today
Popular posts
Most recent paintings in a style I've been developing in the past year.

Wappalyzer
Message boards
Reddit
Programming languages
Python
Issue trackers
Sentry
JavaScript libraries
React
Caching
Redis
Generate sales leads
Find new prospects by the technologies they use. Reach out to customers of Shopify, Magento, Salesforce and others.
Create a lead list
Log In
Sign Up

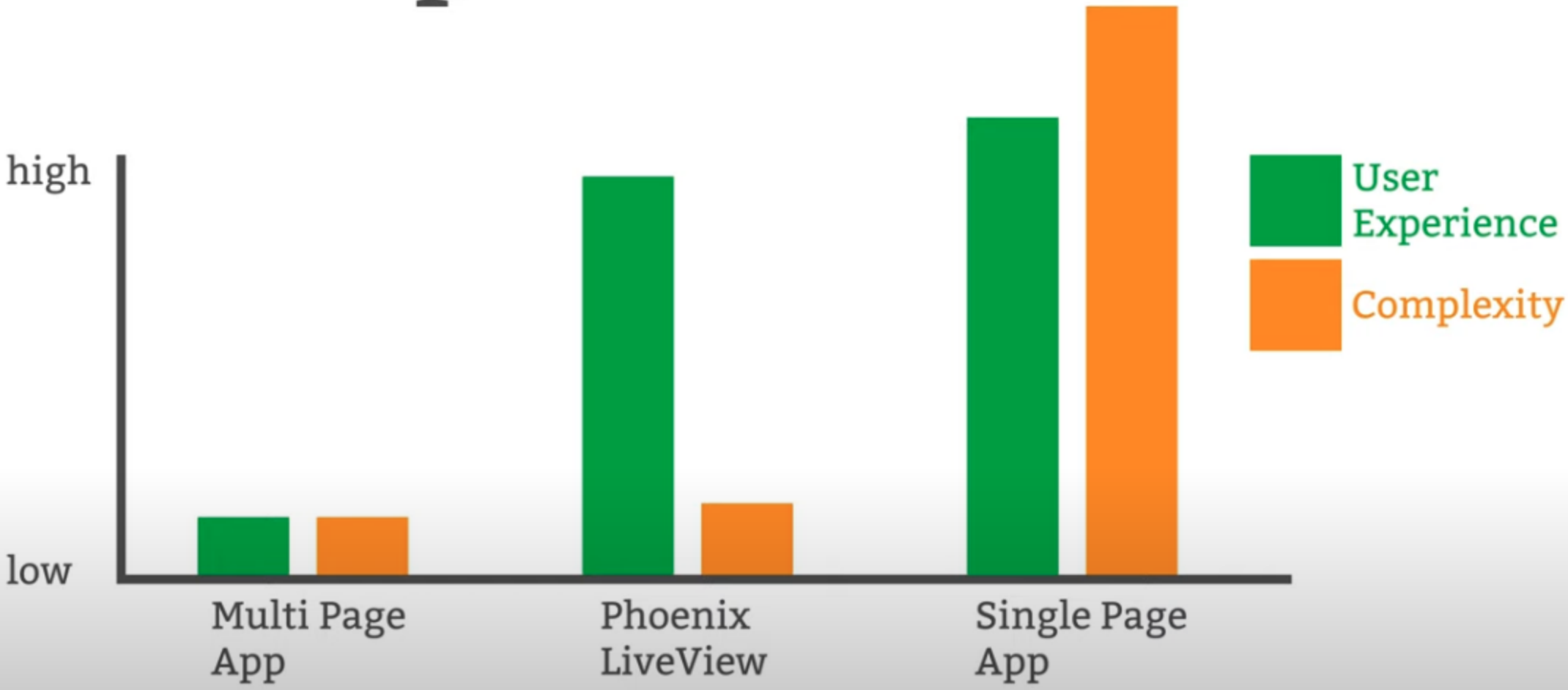

twitter.com/home
Home
Explore
Notifications
Messages
Bookmarks
Lists
Profile
More
Tweet
What's happening
Who to follow
Messages


linkedin.com/feed/
Start a post
LinkedIn News
State of Messaging
Lunch & Learn


facebook.com
facebook
Connect with friends and the world around you on Facebook.
Email or Phone Number
Password
Log In
Forgot Password?
Create New Account

The people demand better

Compare & Contrast

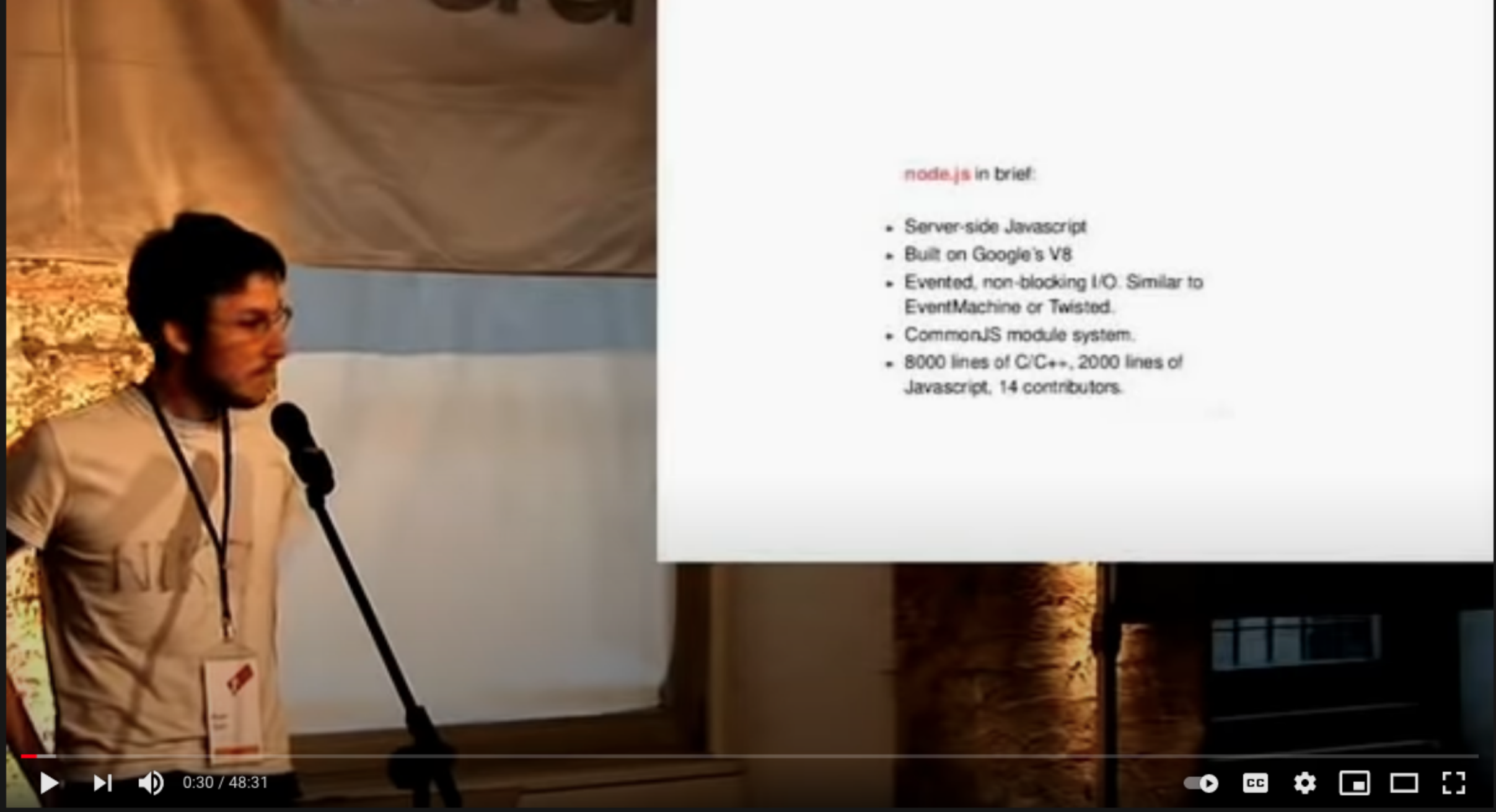


The LiveView Promise

Get blackout drunk with no hangover.

Mac Siri • Nov 3 '17

Definitely DHH's "How to build a blog in 15 minutes with Rails" from 2005. Saw this before I fully understood Rails(not that I understand it 100% today). Still wow me to this day.



Ryan Dahl: Original Node.js presentation

196,679 views • Jun 8, 2012

3.2K 22 SHARE SAVE

Phoenix Framework
Get Started
LiveDashboard

Timeline

@chris_mccord
Another!
0 0

@chris_mccord
how much data?
0 0

@chris_mccord
Right?
0 0

@chris_mccord
New Post!
0 0

New Post

16:49 / 17:54

Phoenix Framework
Get Started
LiveDashboard

Timeline

@chris_mccord
Another!
0 0

@chris_mccord
how much data?
0 0

@chris_mccord
Right?
0 0

@chris_mccord
New Post!
0 0

New Post

Build a real-time Twitter clone in 15 minutes with LiveView and Phoenix 1.5

76,345 views • Apr 22, 2020

1.6K 20 SHARE SAVE ...

Big

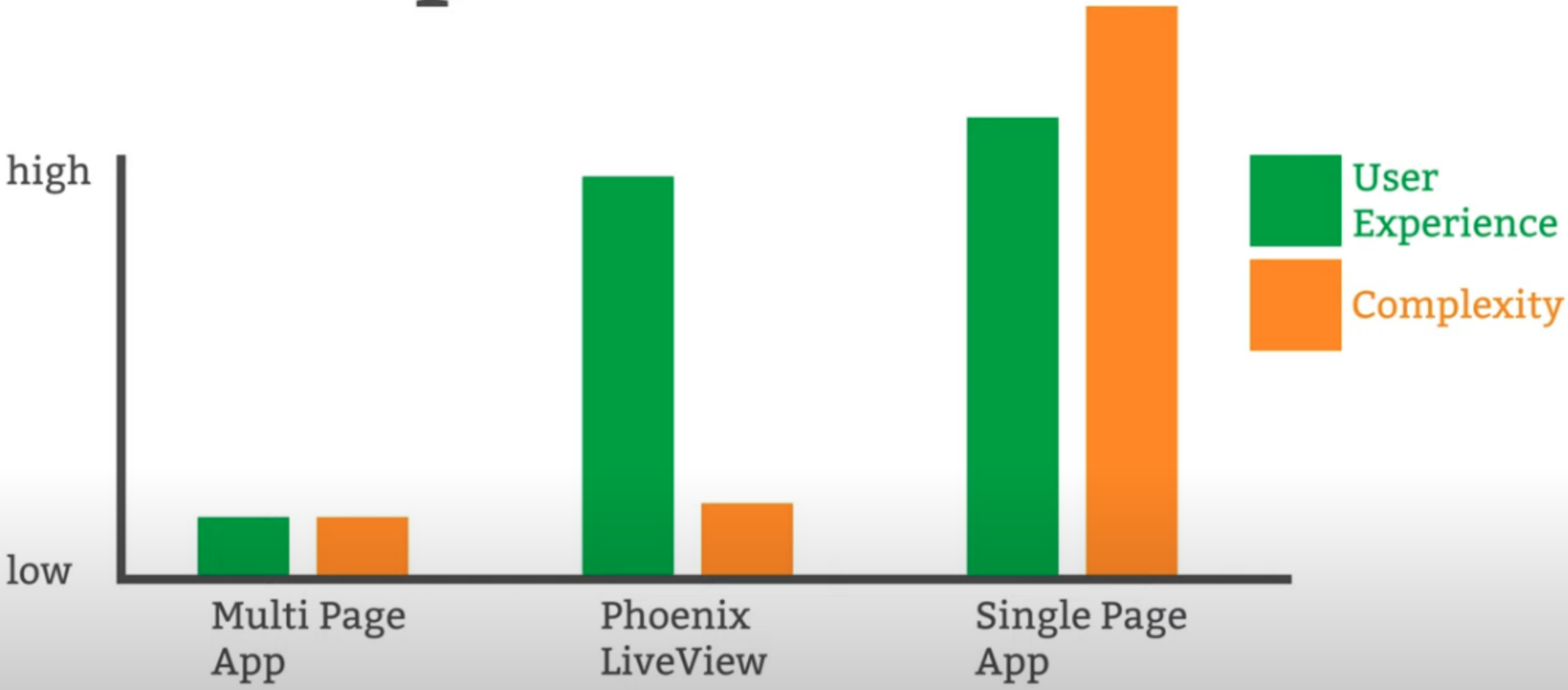
Cojones

Phoenix LiveView: Interactive, Real-Time Apps. No Need to Write JavaScript.

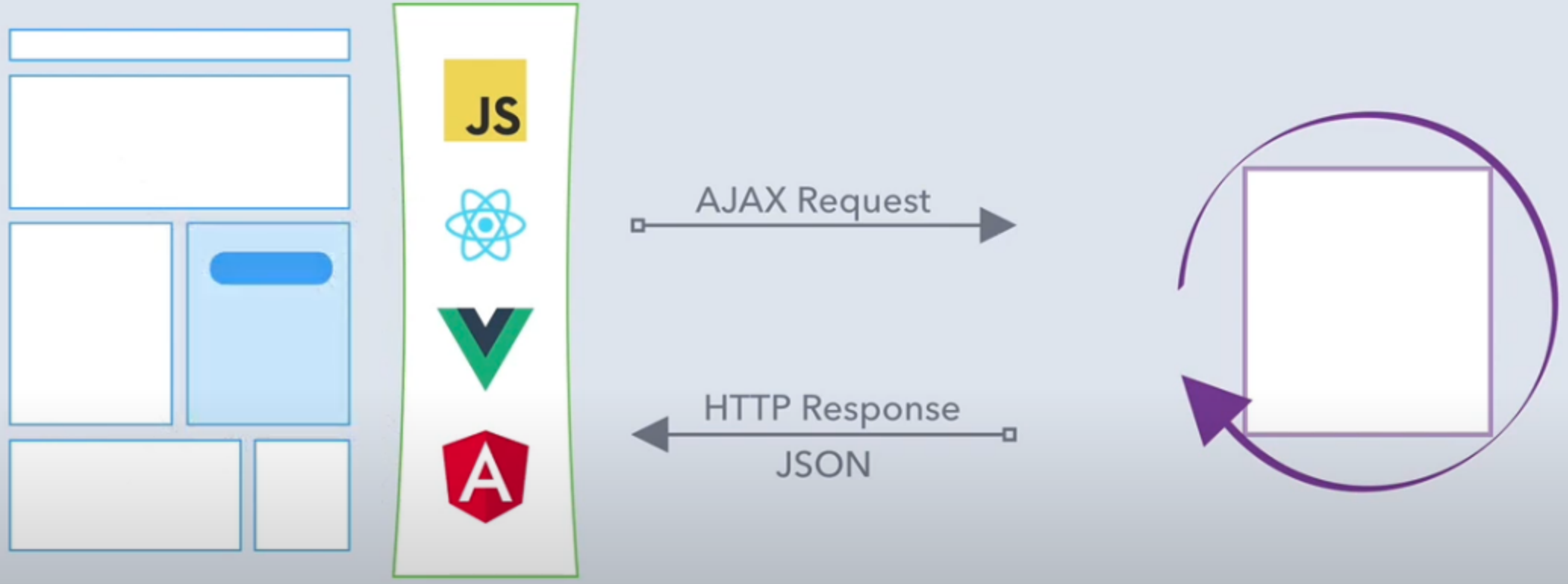
By: Chris McCord • December 12, 2018

Elixir | Web Development | Engineering

Compare & Contrast



How does LiveView work?


JS
AJAX Request
HTTP Response
JSON

How does LiveView work?

1. Endpoint
2. Router
3. Initial Mount
   1. Mount (HTML and Websocket)
   2. `handle_params` ()
   3. `render()`
4. Responding to Events
   1. `handle_event()`
   2. `render()`

Grossly Overengineered Church Website Demo

Thinking in Model-View-Controller

1. Endpoint
2. Router
3. Controller
    1. Loads data from schema
    2. Supplies data to the view
4. View
5. Template

Thinking in LiveView

LiveView

1. Endpoint (This turns an HTTP request into Elixir code)
2. Router - Decides what LiveView to send the request
    1. Lots of security responsibilities
    2. Authentication responsibilities
    3. Passes your requests to the correct plug
3. Requests lands on a live view
    1. HTML mount, Mount function → handle_params → render
    2. Websocket connection is made, mount function → handle_params → render
4. User started editing a form on the live view
    1. handle_event → very often interacts with the database
    2. Very often we leave the web stack and enter the pure Elixir stack
5. Context layer
    1. Responsible for business logic
    2. Responsible for saving data to the database
6. Schema layer
    1. Responsible for data validation
    2. Database logic

Thinking Demo - Banner Uploads

1. Endpoint

Thinking Demo - Banner Uploads

2. Router

Thinking Demo - Banner Uploads

3. Initial Load

Thinking Demo - Banner Uploads

4. Responding to Events

Thinking Demo - Banner Uploads

5. The Context Layer

Thinking Demo - Banner Uploads

6. The Schema Layer

Pretty sweet right?

Sounds cool. But you're just a lowly CRUD app developer

BUT WAIT, THERE'S MORE

Quickly Exposing APIs

You can leverage Phoenix's Context layer standard architecture to quickly bolt on API's

GUIDES
MODULES
MIX TASKS

Views and templates
Ecto
Contexts

Top
Thinking about design
Starting With Generat...
In-context Relationships
Adding Account functi...
Cross-context depend...
Cross-context data
Adding CMS functions
FAQ

Mix Tasks
Telemetry

REAL-TIME COMPONENTS
Channels
Presence

TESTING
Introduction to Testing
Testing Contexts
Testing Controllers
Testing Channels

DEPLOYMENT

Contexts

> *Requirement: This guide expects that you have gone through the introductory guides and got a Phoenix application up and running.*

> *Requirement: This guide expects that you have gone through the Request life-cycle guide.*

> *Requirement: This guide expects that you have gone through the Ecto guide.*

So far, we've built pages, wired up controller actions through our routers, and learned how Ecto allows data to be validated and persisted. Now it's time to tie it all together by writing web-facing features that interact with our greater Elixir application.

When building a Phoenix project, we are first and foremost building an Elixir application. Phoenix's job is to provide a web interface into our Elixir application. Naturally, we compose our applications with modules and functions, but simply defining a module with a few functions isn't enough when designing an application. It's vital to think about your application design when writing code. Let's find out how.

> *How to read this guide: Using the context generators is a great way for beginners and intermediate Elixir programmers alike to get up and running quickly while thoughtfully designing their applications. This guide focuses on those readers. On the other hand, experienced developers may get more mileage from nuanced discussions around application design. For those readers, we include a frequently asked questions (FAQ) section at the end of the guide which brings different perspectives to some design decisions made throughout the guide. Beginners can safely skip the FAQ sections and return later when they're ready to dig deeper.*

Thinking about design

Contexts are dedicated modules that expose and group related functionality. For example, anytime you call Elixir's standard library, be it `Logger.info/1` or `Stream.map/2`, you are accessing different contexts. Internally, Elixir's logger is made of multiple modules, but we never interact with those modules directly

Let's go back to church. API goodness

Also, testing is lightning fast

LiveView Version 0.15.x????

The documentation is best-in-class. Premium. Free. Just read the docs. It's ALL THERE!!!

Shill

???