[Slide of title]
< Hello I was a computer science teacher mostly in high schools and mostly in New York City. Now I test software in Madison. This summer I started building small things in Elm. This is a talk for beginners. I can show what I have learned so far. >

[Argonne Forest]
Where are we going? This talk is in two parts. First, a normal talk, and then we can get up, move around, leave, while we install, clone in to the repo, and sew together a couple components.

[summary OUTLINE]
Elm is very fast and very small when compared with React, Angular, Ember, etc.
Elm is a programming language that compiles to JavaScript.
It is known for its friendly error messages, helping you find issues quickly and refactor large projects with confidence.
It has static types.
Elm is purely functional.

[Changelog slide]

Does anyone know Changelog podcast?
I remember I was riding my bike past the powerplant on the UW campus, listening to the Changelog podcast. There was a programmer talking about how to be well-rounded and how to get better at thinking outside of the box. He claimed the best thing you could do was to get three different types of languages and build something in each.
I pulled over and wrote the types down.
The three types were
Imperative
Functional
and declarative.
I put that in my 'someday' catagory.

(walk forward, counting on hands) Imperative is the type we all do anyway.

Functional and Declarative, according to Wikipedia sort of overlap.
So you kill two birds with one stone if you can build something in Declarative.
I looked up the languages.
They were pretty obscure, academic.
There wasn't an easy entry point.

Then a year later, same podcast: The Changelog, I was raking leaves and a guy with the same first name as me
was talking about a way to build web applications, including really large web applications with something that wasn't JS.
He was explaining a bunch of advantages to his approach. A lot of the advantages went over my head since they were solving problems that I didn't know were problems.
He said that browser, DOM, JS, all of that, should be built from the ground up with something rational. When he said that he had done it with a functional language, that sounded preeeety good.

To build a web page out of Elm, you write code, hit compile, and the compiler can spit out, HTML, CSS, and JS. And it is purely functional in its logic. Unlike Typescript, you can't cheat. You can't not have a type. You can't make anything that is not a function."

So twice in the space of a year, TheChangelog Podcast had suggested "Try Functional", and then "Use Functional to "
But I am not a student or professor, I need to learn to build stuff that people will pay me to build. And I need to know how to maintain legacy code.
And so I was thinking about figuring out React.
And I went to

[Elm is fast]
This is the REAL WORLD APP. Look it up on

github.
Because it is static typed, immutable,
There are a lot less eventualities to plan for.
Tree shaking, minification is different.
It's small.
Just loading the REACT base module is ~350

[compile Button example,] [terminal left]
[browser right]
<Elm uses a compiler. It is compiled before it runs.
Other languages are compiled before they run:
C++, C, C#, Pascal, Java
A compiler is a little like a roomate who looks at you before you leave the house and says: you have spinach on your teeth, you are missing your lunchbox, or, your new haircut looks great, you look good today>
The compiler for Java or C++ is basically giving you a certification that at the very least, the program will run.
Python, Perl, Javascript, they are like leaving the house with no roomate, or your roomate is asleep, or, in the case of Javascript, instead of having a helpful roomate, your roomate is a joker waiting to embarrass you in public."

[Undefined is not a function on movie marquee]
<actually compile a program>

[STATIC TYPED]
The lots of nice stuff of Elm is sort of overlapped with other languages.
This part is not. Its immitable.
Under the hood, the compiler runs Haskell.
You'll see that if you have to do a manual install: Haskell will install.

[functional]
The other Comp-Sci part is this is pURE functional.
Again, Haskell under the hood, compiles to

javascript, but
you cannot cheat.
And thats' the best way to do functional.
Maybe the only way, unless you are already a Haskell, Scala, or Clojure ? person.

[That Pumpkin]
this summer. And there was a talk from a React programmer. She works at U of Minnesota. And she writes React. But she also writes Elm. And she echoed the same thing that the Elm creator had said: You should build something in Elm. Elm is organized, disciplined, and not any harder than its peers. And by building in this organized envirnoment you could end up going 100% with it, or, just as fine, you will learn the patterns and style of good programming, and be a better React programmer after.

[That Sponsor]

Laud the free-ness of this.

[After last Functional slide]
So for me:
A functional language is so different that it will stretch your brain, like a boxer who learns ju-jitsu. I'm not powerful, physically or computationally, so I could dig knowing a little JuJitsu.
A way to program the front end with the logic of a back end person. React does some of this, but it is like building your house on sand.
And, your time spent with the strictly enforced MODEL - UPDATE - VIEW , every minute of learning that, makes you a better React coder.
<like how the classic rock drummers knew how to drum jazz. Black Sabbath, they may sound like troglodytes, but the drummer studied jazz, Zeppelin, Yes, Pink Floyd, they all knew the more musical styles>

[IBM headquarters]

So how has Elm been doing in the real world? IBM Decision Composer is a planning tool that allows teams to weigh multiple factors using graphical tools. The backend has existed for awhile and is Java. They rewrote the frontend this year in Elm. Here, the right side is a DOM tree written in Dojo. The left side is in Elm.

[IBM web page]
[IBM dom TREE]
Highlights:

Functional programming ! The Elm language is a good introduction to FP, especially for people with an OO background like us. Only problem :
Tooling isndebuggerloose

The build (elm make) is really simple. Compare this to your average WebPack **. And the UMD module can be consumed easily too : we include it into a dojo AMD layer build.**

Other cases
Prezi
No Red Ink
Strangeloop - Elm Conf EuropeanElmConf (Prezi legacy maybe)
FordMoCo

[DREAM CATCHER JS ECOSYSTEM]
Will Elm become something big?
React is big, useful. Node2 and Vue are good.

[HOUSE]
[HOUSE]
[HOUSE]

[YOUTUBE]
[MY LINKS]
[FAST]
[SPONSORS]
[PICK UP TRASH]