FUNCTIONAL
REACTIVE
JAVASCRIPT
FUNCTIONAL REACTIVE PROGRAMMING
There are only two hard things in Computer Science: cache invalidation and naming things.

- Phil Karlton
There are only two hard things in Computer Science: cache invalidation, naming things, and off-by-one-errors

- anon
COMPOSITIONAL EVENT SYSTEMS
a different way of thinking about data and how it flows through our programs
RxJS 5 / Observable
Observable
time

space
UI elements emit streams of values.

Functions combine and transform those streams into new streams.

Streams can be assigned to UI attributes to effect UI output.
related artists

Taylor S  go!
Taylor Swift

related artists:
Miley Cyrus
Katie Perry
etc...
Spotify Search

“Taylor S”

[{}, {}, {}]

Spotify Related Artists
"Taylor S"

[ {...}, {...}, {...} ]

"Z41T71G27"

[ {...}, {...}, {...} ]

Spotify
Search

Spotify
Related Artists
searchForArtist("Taylor")

```json
{ id: "0vnm2817Yf1P", name: "James Taylor", ...
}

{ id: "Shc2QfaQcjApz0", name: "Taylor Swift", ...
}
```
searchForArtist("Taylor")

{ id: "@vnmm2817Yf1P", name: "James Taylor", ... }

{ id: "Shc2QfoQcjApz0", name: "Taylor Swift", ... }

.take(1)
.take(1)

{ id: "5hc2QfaQcjApz0", name: "Taylor Swift", ... }
.take(1)

{ id: "5hc2QfaQcjApz0", name: "Taylor Swift", ... }

.map( relatedArtists )
.take(1)

{ id: "5hc2QfaQcjApz0", name: "Taylor Swift", ...
}

.map( relatedArtists )
.take(1)

{ id: "5hc2QfaQcjApzO", name: "Taylor Swift", ...
}

.flatMap( relatedArtists )

{...

{...}
relatedArtists

{  
id: "0vmn2817Yf1P",
  name: "Miley Cyrus",
  ...
}

{  
id: "5hc2QfQcjaApz0",
  name: "Kelly Clarkson",
  ...
}
relatedArtists

{ 
    id: "0vnm2817Yf1P",
    name: "Miley Cyrus",
    ...
}

{ 
    id: "5hc2QfoQcjApz0",
    name: "Kelly Clarkson",
    ...
}

.map( renderFn )

<li>
    Miley Cyrus
</li>

<li>
    Kelly Clarkson
</li>
HTTP

node streams

node streams

HTTP
reduce (aka *fold*)

*summing an array*
reduce (aka fold)

summing an array

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array}
\]

\[
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 = 36
\]
reduce (aka fold)

\[(a,b) \Rightarrow a+b\]
reduce (aka fold)

(a, b) => a + b
reduce (aka fold)

\[(a,b) \Rightarrow a + b\]
reduce (aka fold)

\[(a,b) \Rightarrow a+b\]
reduce (aka fold)

\[(a, b) \Rightarrow a + b\]
reduce (aka *fold*)

\[(a, b) \Rightarrow a + b\]
reduce (aka *fold*)

\[(a, b) \Rightarrow a + b\]

*numbers*
reduce (aka *fold*)

\[(a, b) \Rightarrow a + b\]

*numbers*
reduce (aka \textit{fold})

\[(a,b) \mapsto a+b\]
reduce (aka fold)

\[(a, b) \Rightarrow a + b\]
reduce (aka *fold*)

\[(a, b) \Rightarrow a + b\]
reduce (aka fold)

$(a,b) \Rightarrow a+b$

numbers
scan

numbers

summed numbers
scan

numbers

summed numbers
scan

numbers

summed numbers
scan

numbers

summed numbers
scan

numbers

summed numbers
scan

numbers

1 2

summed numbers

1 3
scan

numbers

summed numbers
scan

numbers

1 2 3 4 5 6 7 8

summed numbers

1 3 6 10 15 21 28 36
scan

(a, b) => Math.max(a, b)
what is FRP?
what is FRP?

a powerful, framework-agnostic way to work with event streams

a unifying universal abstraction

a declarative way to model relationships between values in our programs
shared mutable state
thanks!