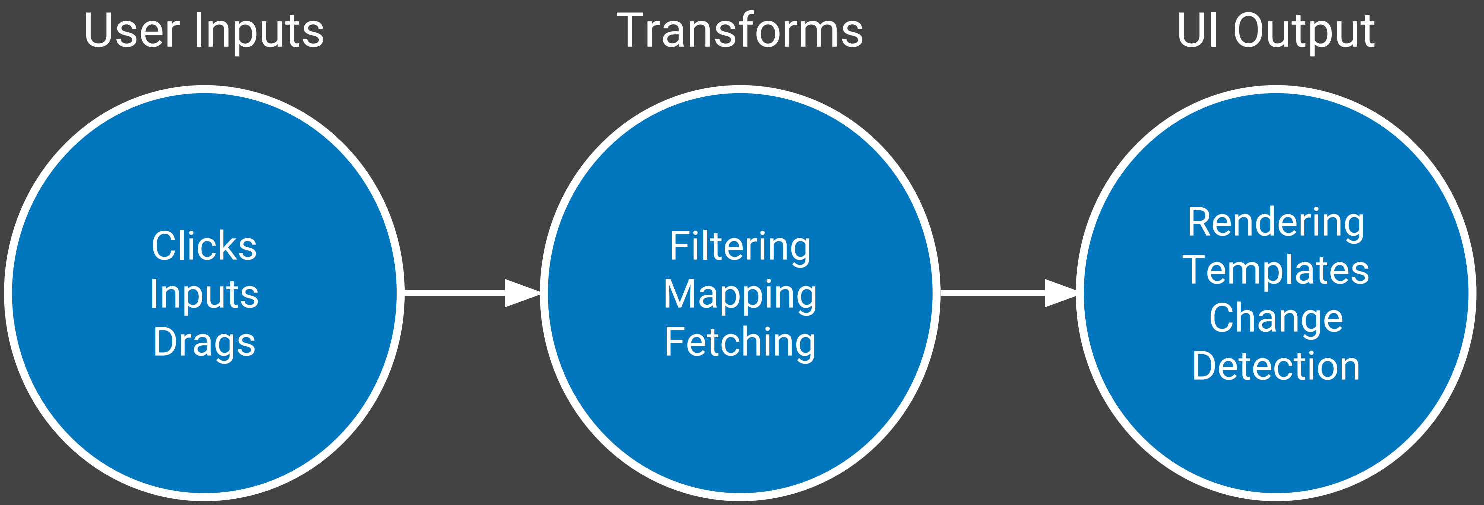
Angular Data Flow

How to build applications that stand the test of time.

Jeff Cross (@jeffbcross)
Rob Wormald (@robwormald)

Event Streams


User Inputs
Clicks
Inputs
Drags
Transforms
Filtering
Mapping
Fetching
UI Output
Rendering
Templates
Change
Detection

Think less

Reasonable Event Flow

Prevent bugs

State Containment

Love users

Minimal Runtime Overhead

RxJS
(Reactive Extensions for JS)

Observables

Observable: Like a Promise of Many Values

```
socketUpdates
  .subscribe((msg) => {
    this.latestMessage = msg.body;
  });
```

Observable Combinators

```
socketUpdates
  .map(msg => msg.body)
  .subscribe(body => {
    this.latestMessage = body;
  });
```

ES2016 Standards Track

RxJS

A powerful implementation of Observable

Hundreds of Contributors

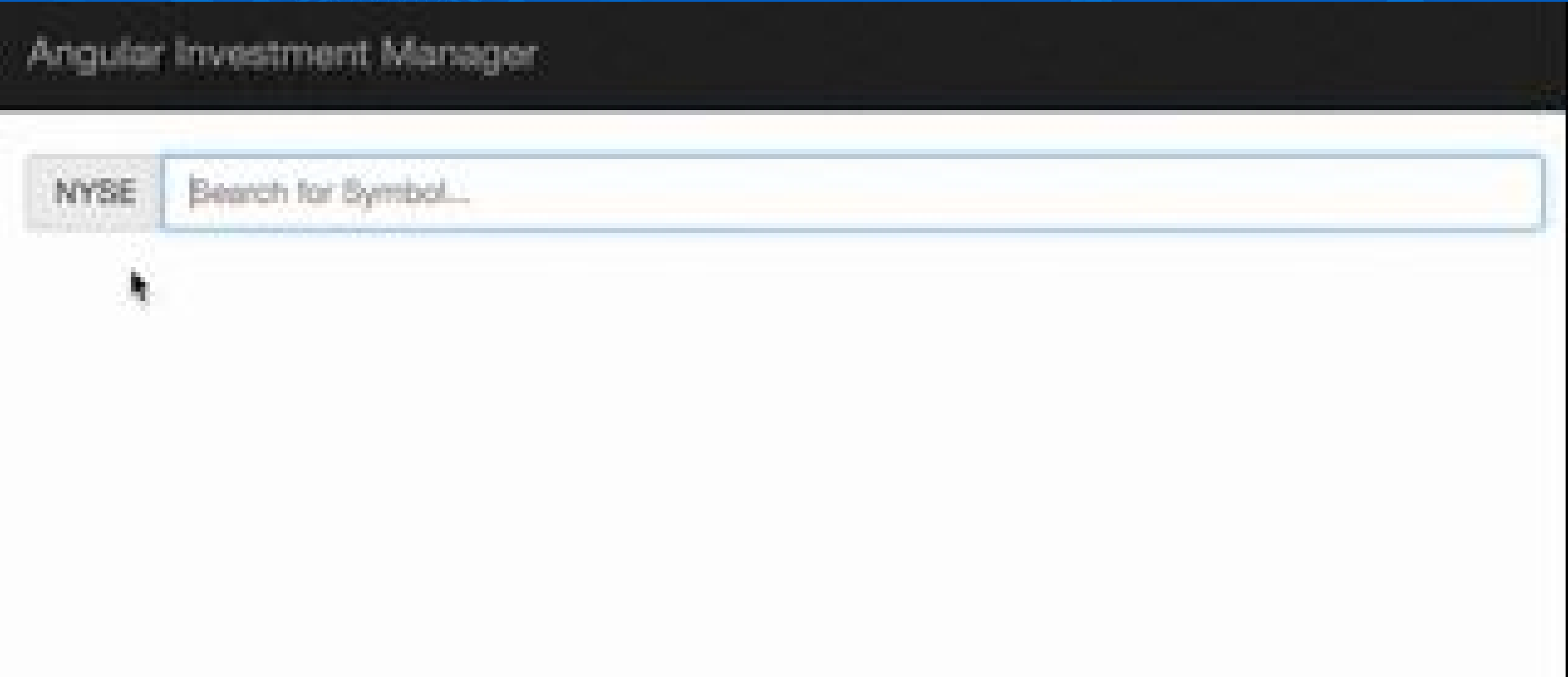
Largely developed and maintained by **@mattpodwysocki** (Microsoft) and **@benlesh** (Netflix)

All the combinators you need

Thanks, Ben Lesh!

Angular Investment Manager

Typeahead - Search for Ticker Symbols



Typeahead in Angular2: Classical Style

Typeahead Classical Style

Template

```
<input
  type="text"
  [(ng-model)]="searchText"
  (keyup)="searchChanged($event)">
```

Component

```
doSearch(){
  let searchText = this.searchText;
  this.currentRequest =
    fetch(`/stocks?symbol=${searchText}`)
      .then(res => res.json())
      .then(tickers => this.tickers = tickers);
}

searchChanged(){
  this.doSearch(this.searchText);
}
```

Typeahead Classical Style - Throttle

Template

```
<input
  type="text"
  [(ng-model)]="searchText"
  (keyup)="searchChanged($event)">
```

Component

```
if(typeof this.searchTimeout !== 'number'){
  clearTimeout(this.searchTimeout);
  this.searchTimeout = null;
}
this.searchTimeout = setTimeout(() => {
  this.doSearch(this.searchText);
  this.searchTimeout = null;
}, 500);
```

Typeahead Classical Style - Response Order

Template

```
<input
  type="text"
  [(ng-model)]="searchText"
  (keyup)="searchChanged($event)">
```

Component

```
var searchText = this.searchText;
...
  .then(tickers => {
    if (this.searchText === searchText) {
      this.tickers = tickers;
    }
  });
```

Classical Style Pitfalls

Out-of-Band Logic and Side Effects

```
// template
[(ng-model)]="searchText"
(keyup)="searchChanged($event)"

// searchChanged()
this.searchTimeout = setTimeout(() => {...}, 500);

// doSearch()
this.currentRequest = fetch(`/stocks?symbol=${this.searchText}`)
```

Inefficiency

```
// doSearch()
this.currentRequest = fetch(`/stocks?symbol=${this.searchText}`)
```

Typeahead: now with streams

Angular 2 Forms

Angular 2 Forms: Control

Template

```
<input
  type="text"
  #symbol
  [ng-form-control]="searchText"
  placeholder="ticker symbol">
```

Component

```
export class TypeAhead {
  searchText = new Control();
}
```

Angular 2 Forms: Control valueChanges

Template

```
<input
  type="text"
  #symbol
  [ng-form-control]="searchText"
  placeholder="ticker symbol">
```

Component

```
export class TypeAhead {
  ticker = new Control();
  constructor() {
    this.searchText.valueChanges
      .subscribe(...);
  }
}
```

Event Flow Step 1: Throttling

```
this.searchText.valueChanges
    .debounceTime(200)
    .subscribe(...);
```

Angular 2 Http

Event Flow Step 2: TickerLoader

TickerLoader

```
load(val:string):Observable<any[]> {
  return this._http
    .request(`/stocks?symbol=${val}`)
    .map(res => res.json());
}
```

Event Flow Step 3: SwitchMap

TypeAhead

```
this.searchText.valueChanges
   .debounceTime(200)
   .switchMap(val => tickerLoader.load(val))
   .subscribe(...);
```

Event Flow 4: View Binding

Template

```
<li *ng-for="#tick of tickers">
  ...
</li>
```

Component

```
this.searchText
   .valueChanges
   .debounceTime(200)
   .switchMap(val => {
     return tickerLoader.load(val);
   })
   .subscribe(tickers => {
     this.tickers = tickers;
   });
```

Angular 2 Pipes

Pipes

Template

```
The current date is
{{ today | async | date }}
```

View

The current date is Oct 20, 2015

Component

```
export class Today {
  today:Promise<Date>;
  constructor(ts:TimeService) {
    this.today = ts.getServerDate();
  }
}
```

Event Flow 4: View Binding with Pipes

Template

```
<li *ng-for="#tick of tickers | async">
  ...
</li>
```

Component

```
this.tickers = this.searchText.valueChanges
  .debounceTime(200)
  .switchMap(val => {
    return tickerLoader.load(val);
  });
```

Before and After

Classical Style Typeahead (Component - 26 LOC)

```
export class TypeAhead {
  searchText: string;
  searchTimeout: any;
  currentRequest: any;
  constructor() {}
  doSearch(text){
    var searchText = this.searchText;
    this.currentRequest = null;
    this.currentRequest = fetch(`server?symbol=${this.searchText}`)
    this.currentRequest
      .then(res => res.json())
      .then(tickers => {
        if (this.searchText === searchText) {
          this.tickers = tickers;
        }
      });
  }

  searchChanged(){
    if(typeof this.searchTimeout !== 'number'){
      clearTimeout(this.searchTimeout);
      this.searchTimeout = null;
    }
    this.searchTimeout = setTimeout(() => {
      this.doSearch(this.searchText);
      this.searchTimeout = null;
    }, 500);
  }
}
```

Reactive Style Typeahead (Component - 11 LOC)

```
export class TypeAhead {
  ticker = new Control();
  tickers: Observable<any[]>;
  constructor(http:Http, tickerLoader:TickerLoader) {
    this.tickers = this.searchText.valueChanges
      .debounceTime(200)
      .switchMap((val:string) => {
        return tickerLoader.load(val);
      });
  }
}
```

In Conclusion: Start Small

Angular Data Roadmap

Before we get started

Motivation

- Web apps are growing
- The web itself is evolving
- Users are expecting more

Goals

- Reduce boilerplate
- Improve testability
- Enable high performance

Roadmap

- Template Transforms
- Tactical

Template Transforms

Template Transforms

- Plugin to transform Angular templates
- Happens during *compilation*, on application load

```
<div>
  {{model.get("firstName")}}
  {{model.get("lastName")}}
</div>
```

←

```
<div>
  {{model.firstName}}
  {{model.lastName}}
</div>
```

What can it do?

- For app developers:
  - Better sugar from third party libraries
  - Create domain specific languages (DSLs) in templates
  - Optimize queries with view metadata
- For library developers:
  - Ship a template transformer to reduce boilerplate

Example: query DSL

```
db
  .select('users')
  .sortBy('firstName')
  .limit(10)
  .exec()
  .then((response) => {
    // Do something cool with the data!
  });
```

Example: query DSL

```
<li *ng-for="#user of
  db.query('users').sortBy('firstName').limit(10).exec() | async">
  {{user.firstName}} {{user.lastName}}
</li>
```

Example: query DSL

```
<li *ng-for="#user of db.users | sort: firstName | limit: 10">
  {{user.firstName}} {{user.lastName}}
</li>
```

Example: query DSL

```
<li *ng-for="#user of db.users | sort: firstName | limit: 10">
  {{user.firstName}} {{user.lastName}}
</li>
```

↓

```
<li *ng-for="#user of
  db.query('users').sortBy('firstName').limit(10).exec() | async">
  {{user.firstName}} {{user.lastName}}
</li>
```

Example: view metadata

```
db
  .select('users', ['firstName', 'lastName'])
  .sortBy('firstName')
  .limit(10)
  .exec()
  .then((response) => ...);
```

Example: view metadata

```
<li *ng-for="#user of db.users | sort: firstName | limit: 10">
  {{user.firstName}} {{user.lastName}}
</li>
```

Example: view metadata

```
<li *ng-for="#user of db.users | sort: firstName | limit: 10">
  {{user.firstName}} {{user.lastName}}
</li>
```

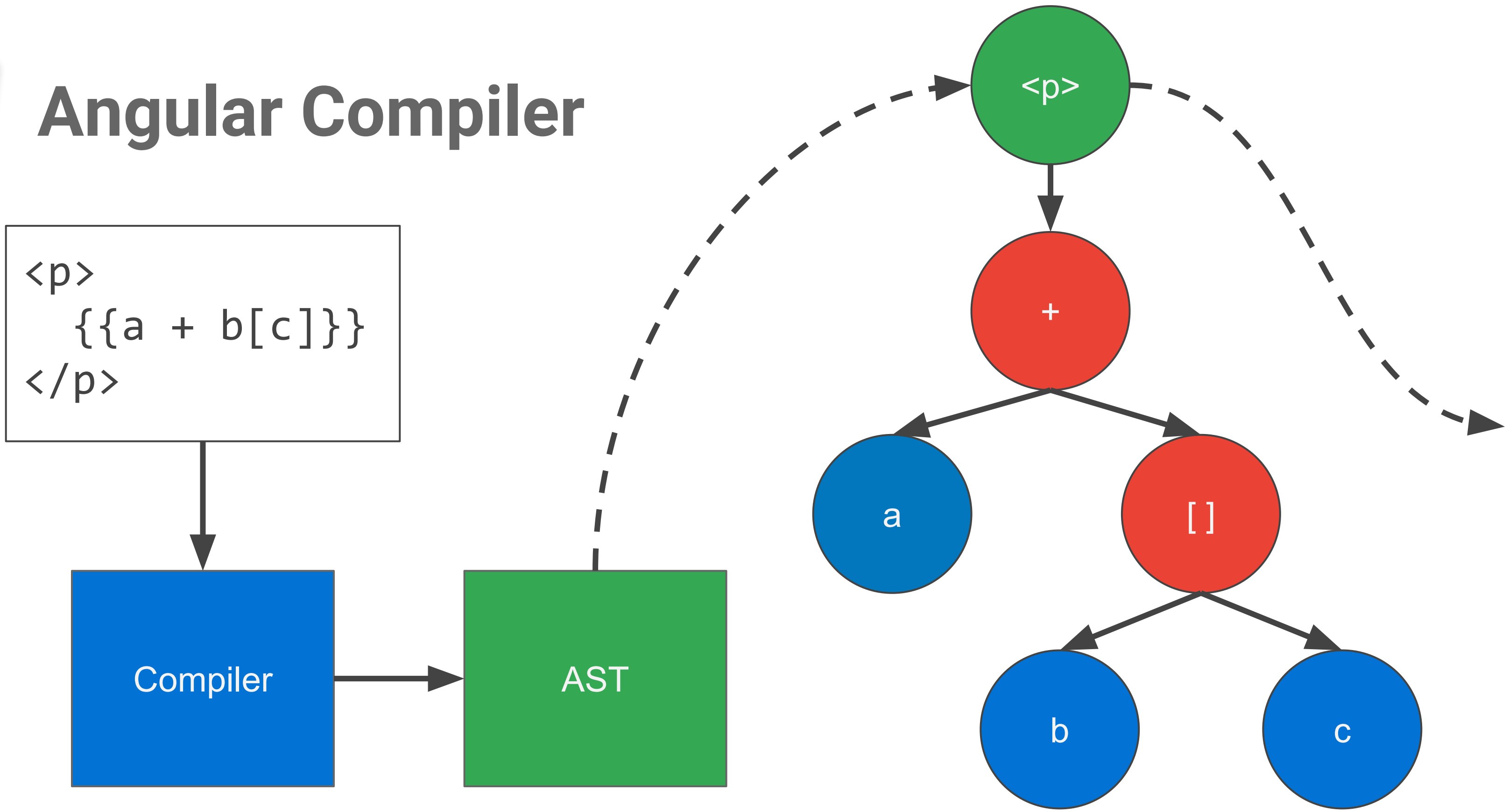
```
<li *ng-for="#user of db.query('users', ['firstName', 'lastName'])
             .sortBy('firstName').limit(10).exec() | async">
  {{user.firstName}} {{user.lastName}}
</li>
```

How does it work?

- Transformers plug into the Angular compiler

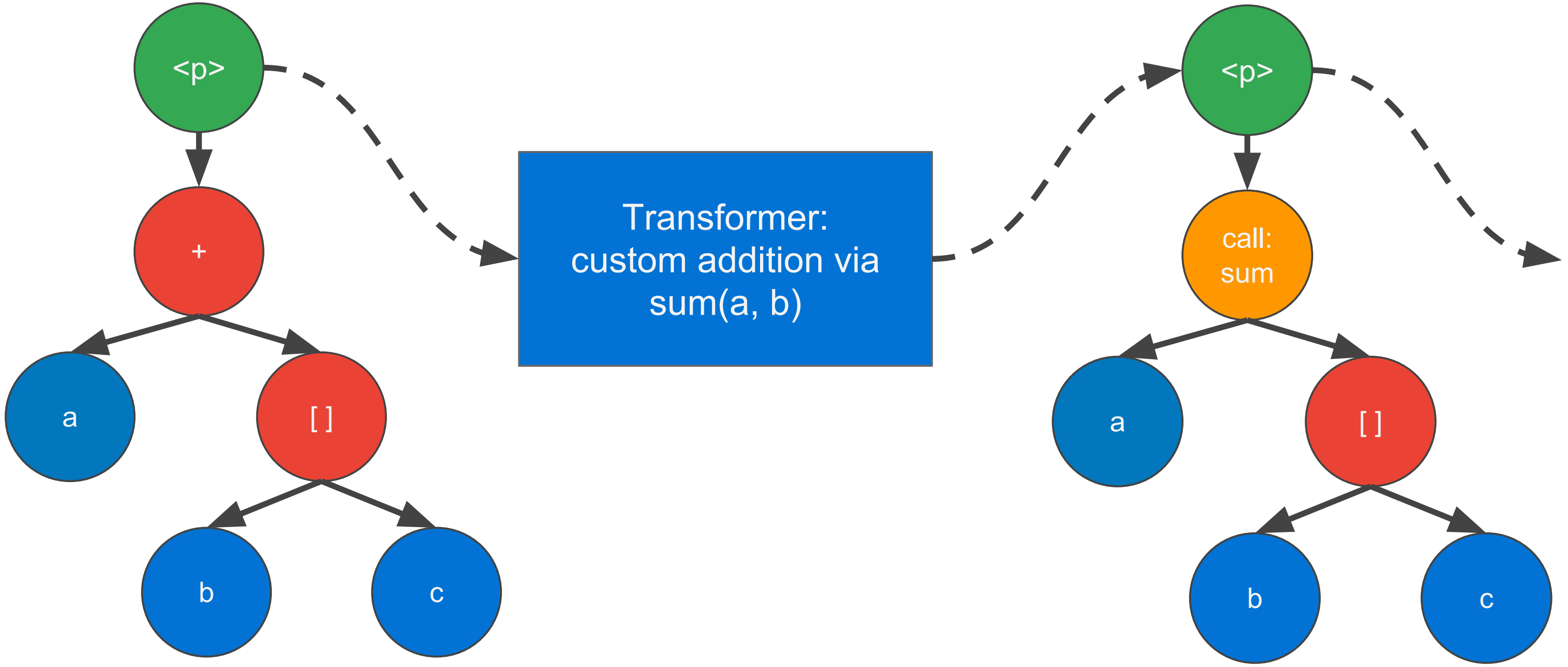
- Operate on abstract syntax, not strings

Angular Compiler

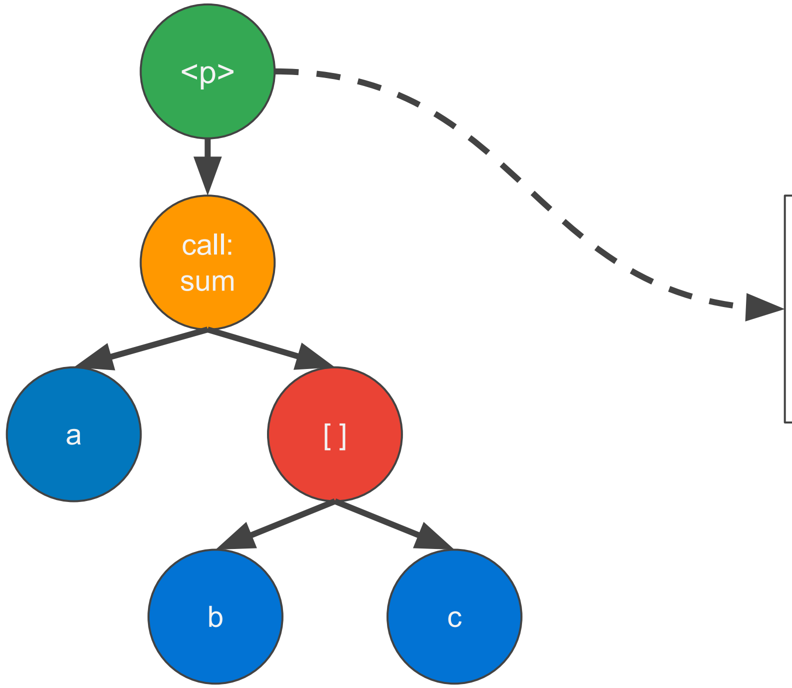


```
<p>
  {{a + b[c]}}
</p>
```

Transforms



Transforms



```
<p>
  {{sum(a, b[c])}}
</p>
```

Real world example: Falcor

- Data access library built by Netflix
- Pretend all data is in one JSON object (the "graph")

```
graph.users[123].firstName
```

Falcor API

```
graph.deref(['users', 123]).subscribe((model) => {

  model.get(['firstName']).subscribe((firstName) => {
    // ...
  });

}
```

Falcor API

```
<div #user="graph.deref(['users', 123]) | async">
  {{user.get(['firstName']) | async}}
  {{user.get(['lastName']) | async}}
</div>
```

Falcor API

```
<div #user="graph.users[123]">
  {{user.firstName}} {{user.lastName}}
</div>
```

Falcor API

```
<div #user="graph.users[123]">
  {{user.firstName}} {{user.lastName}}
</div>
```

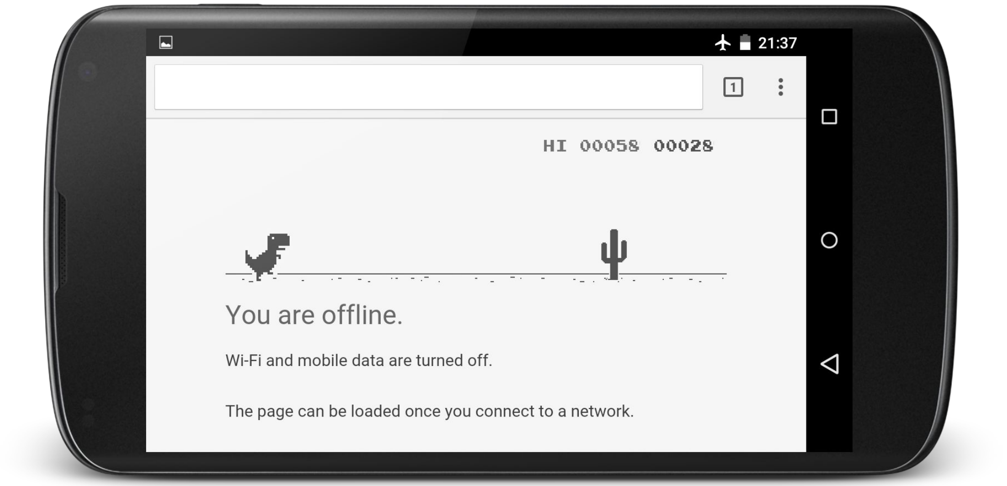
```
<div #user="graph.deref(['users', 123]) | async">
  {{user.get(['firstName']) | async}}
  {{user.get(['lastName']) | async}}
</div>
```

Tactical

Tactical

- Data library outside of Angular core
- Offline functionality against online API
- Work in progress

Why should you care?



You are offline.

Wi-Fi and mobile data are turned off.

The page can be loaded once you connect to a network.

Why don't more apps work offline?

- Backend support for synchronization
- Consistency is hard
- Other features take priority

Tactical's Approach

- "Offline on a budget"
- Works against most APIs
- Focus on the User Experience, not perfect consistency
  - Cache reads
  - Eventual consistency for writes
  - Client-side conflict resolution

What do you get?

- Freshest available data
- Offline mutations with background sync
- First write wins guarantee
- Server-side push, if backend supports it

Edge Cases

- List vs Get request
- Arbitrary searches
- Prefetching

Tactics

- Application extensions to the model
- Add context to improve UX
    - Offline search by filtering available data
    - Prefetch important data when the app loads

State of Tactical

- In development
- Offline reads/writes working
- Some synchronization support
- Follow us: http://github.com/angular/tactical

Thanks!

Slides: g.co/ng/ac-dataflow

AIM: github.com/jeffbcross/aim

Tactical: github.com/angular/tactical

@jeffbcross

@robwormald

@synalx

@ttowncompiled