
OmniBinder
Bind Angular models to anything

Why OmniBinder?

Why OmniBinder?

- Remote data binding should be as easy as in-memory data binding.
- Binding is easy for one user with one copy of data, not as easy when distributed.
- There should be a first-class API which data services, third-party APIs, and others can rely on.
- App developers should only have to know one library to bind to various protocols.

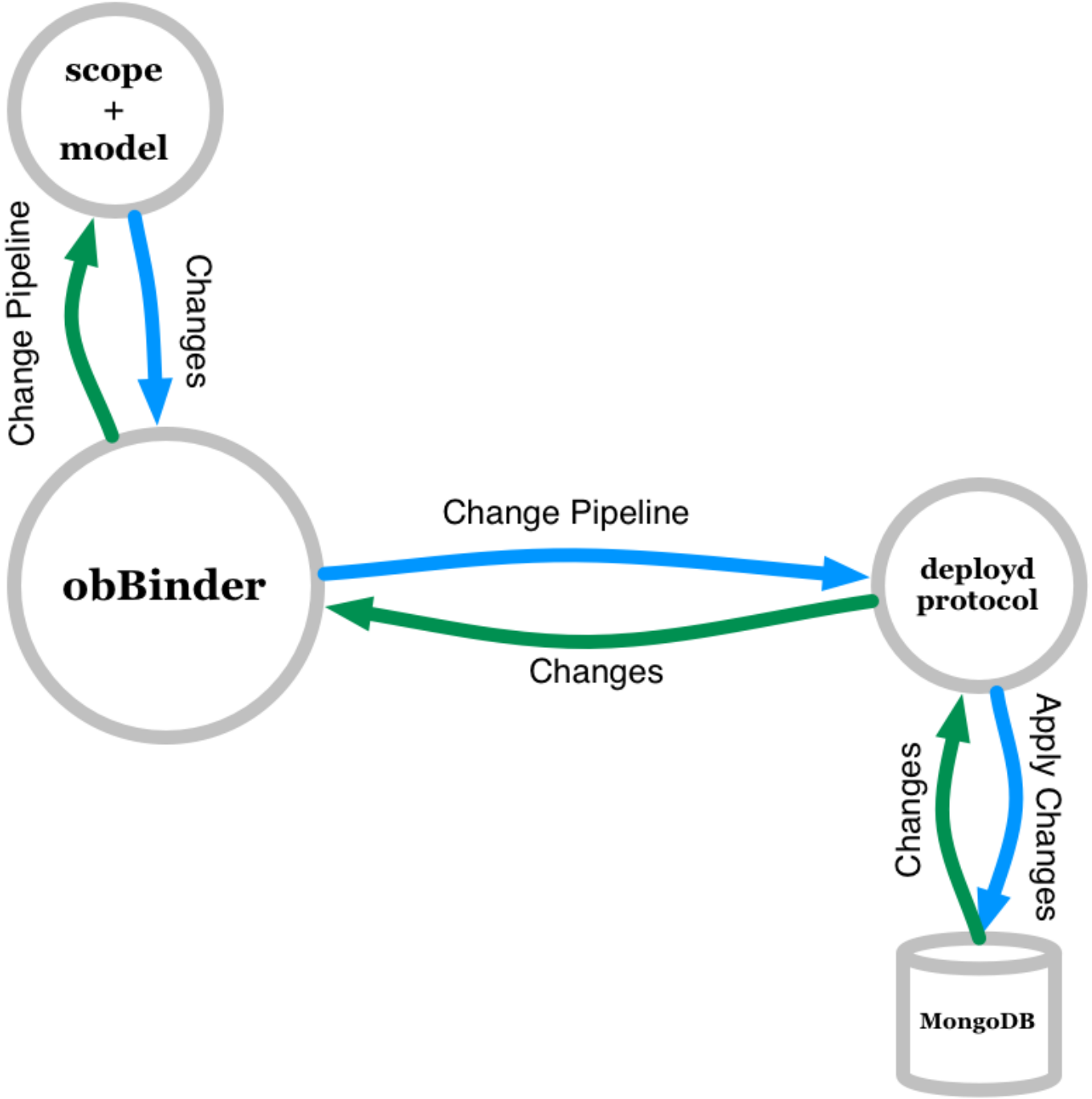
What is OmniBinder?

- A realtime micro-framework on top of Angular
- Makes it simple to bind angular models to any supported backend
- Automatically synchronizes data between client and data store
- A collaborative effort with members from Angular, Firebase, StrongLoop and others
- A work-in-progress

Demo - Todos with Deployd

Binding Logic

```
var app = angular.module('MyApp', ['OmniBinder']);
app.controller('TodosCtrl',
    function ($scope, obBinder, obBinderTypes, deployd) {
        var myBinder = obBinder($scope, 'items', deployd, {
            key: 'id',
            type: obBinderTypes.COLLECTION,
            query: {
                collection: 'todos'
            }
        });
    }
);
```


scope
+
model
Change Pipeline
Changes
obBinder
Change Pipeline
Changes
deployd
protocol
Changes
Apply Changes
MongoDB

What are OmniBinder's Goals?

- Provide a contract between "protocols" and angular models
- Solve common problems associated with syncing:
  - Recursive updating
  - Representing subsets of large collections of remote data
  - Throttling & batching updates (conserving network activity)

How is OmniBinder Accomplishing Its Goals?

How Is OmniBinder Accomplishing Its Goals?

Object.observe-style change sets

Object.observe:

```
{
  type: 'update',
  name: 'foo',
  oldValue: 'baz',
  object: {
    foo: 'bar'
  }
}
```

Array.observe:

```
{
  addedCount: 1,
  index: 0,
  removed: [{
     foo: ‘bar’
  }]
}
```

How Is OmniBinder Accomplishing Its Goals?

Simple contract for **protocols** to implement

```
app.service('myProtocol', function () {
    this.processChanges = function (binder, delta) {
        delta.changes.forEach(function (change) {});
    };
    this.subscribe = function (binder) {};
});
```

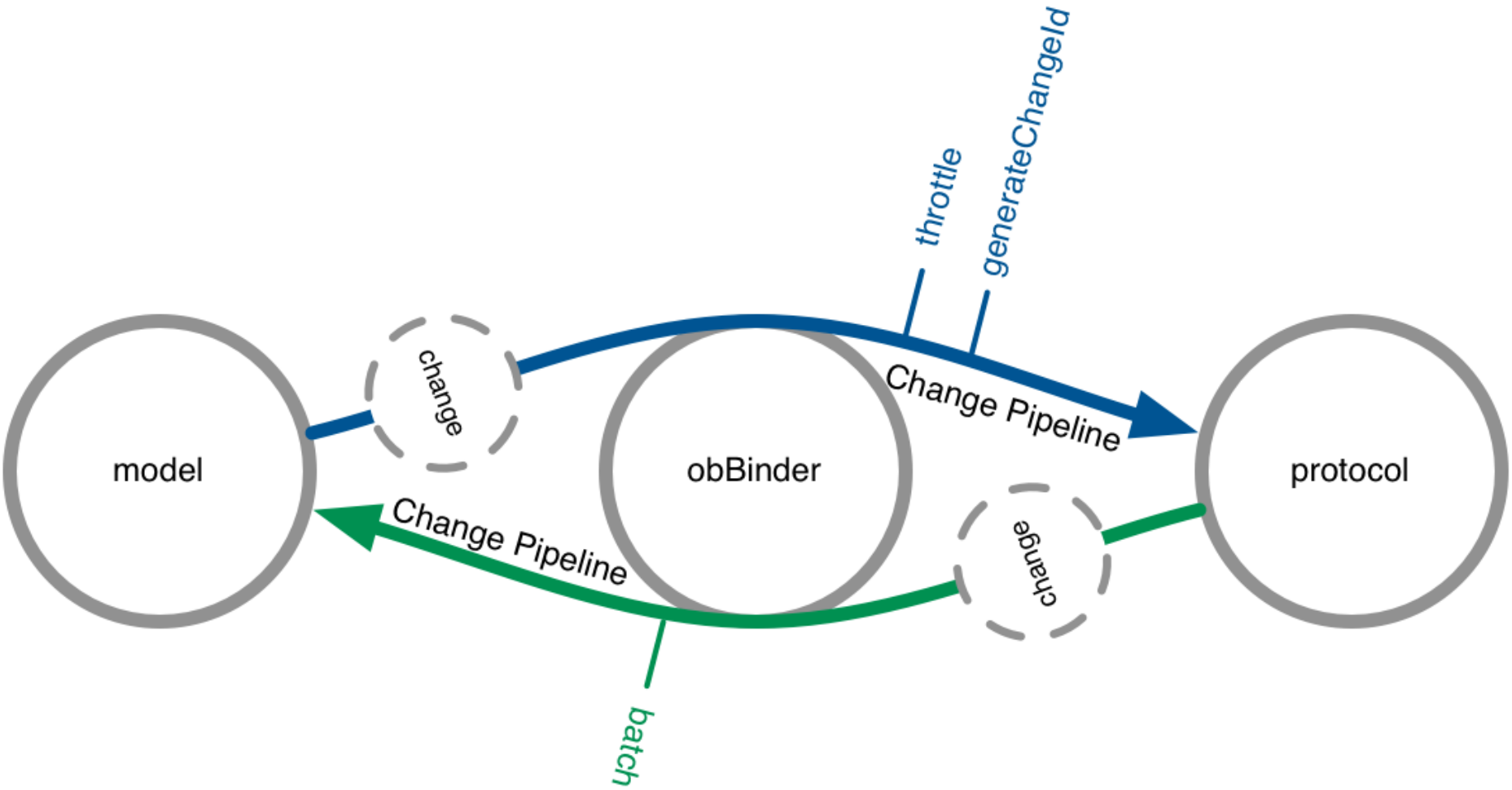
How Is OmniBinder Accomplishing Its Goals?

Under the Covers:

- obObserver watches local arrays and their child objects.
- obBinder observes both directions, syndicates changes to model or protocol.
- obModelWriter applies changes from protocol to local model.

How…?

Change Pipeline



function (binder, delta, next) {}

See [Change Pipeline proposal]

Current state of OmniBinder

- Working prototype, focusing on binding arrays of objects
- Requires Object.observe enabled
- Evolving design doc on Github

What's next for OmniBinder?

- Using with real world applications
- Change pipeline implementation
- More protocols
- Supporting more types of data
- Implementing Object.observe more deeply in Angular
- Standardizing a protocol for Realtime web apps

Help!

- Provide feedback on the docs via issues or pull requests: github/jeffbcross/syncResource
- Create a protocol
- Try it in an app, provide feedback


Thanks
github.com/jeffbcross/syncResource

Future Goals: Stretch Protocol

Stretch: A protocol for the realtime web

- A collaborative effort
- Focuses on change processing instead of stateless data transmission
- Supports HTTP & WebSockets as primary transports
- Moves the line from client > server to program > persistence
- Supports client-side storage

See Stretch Protocol doc on Github