SproutCore and Performance
One should not pursue goals that are easily achieved. One must develop an instinct for what one can just barely achieve through one’s greatest efforts.

Albert Einstein
"Fast By Default"
Core Concepts
1. JS is cheaper than DOM
2. Keep Intermediate State in JS
(and out of DOM)
3. Events
Cannot Coalesce
(prefer cache clearing)
4. Materialize Objects When Used
(also, proxy, don't copy)
5. Follow Speed Guidelines
JS is
Cheaper
than DOM
change title
change name
change address
change DOM
Event Handling

- JS Code
- JS Code
- JS Code
- ...
- DOM Code
Keep Intermediate State in JS
Case Study
I want to have a view that displays the total number of items that are marked done.

I want to have a feature that can mark all remaining items done.
What do you want to do?

4 items remaining

- Mark All Done
- Wash dishes
- Take out garbage
- Make bed
- Relax
<table>
<thead>
<tr>
<th>Task</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wash dishes</td>
<td>✔️</td>
</tr>
<tr>
<td>Take out garbage</td>
<td></td>
</tr>
<tr>
<td>Make bed</td>
<td></td>
</tr>
<tr>
<td>Relax</td>
<td></td>
</tr>
</tbody>
</table>

What do you want to do?

Mark All Done

3 items remaining
What do you want to do?

3 items remaining

Mark All Done

- Wash dishes
- Take out garbage
- Make bed
- Relax
What do you want to do?

Mark All Done
Wash dishes
Take out garbage
Make bed
Relax

0 items remaining
"KVO"
item marked done

re-render stats view
window.AppView = Backbone.View.extend({
  initialize: function() {
    _.bindAll(this, 'addOne', 'addAll', 'render');
    this.input = this.$('#new-todo');
    Todos.bind('add', this.addOne);
    Todos.bind('refresh', this.addAll);
    Todos.bind('all', this.render);
    Todos.fetch();
  },

  render: function() {
    var done = Todos.done().length;
    this.$('#todo-stats').html(this.statsTemplate({
      total: Todos.length,
      done: Todos.done().length,
      remaining: Todos.remaining().length
    }));
  }
});
window.TodoList = Backbone.Collection.extend({
  done: function() {
    return this.filter(function(todo){
      return todo.get('done');
    });
  },

  remaining: function() {
    return this.without.apply(this, this.done());
  }
});

window.Todos = new TodoList;
Todos.forEach(function(todo) {
    todo.toggle();
});
item marked done

compute remaining

compute done

render stats view

x N
This is foundational
No notion of intermediate state
Prefer Coalescing Operations
ArrayController.create({
  content: [],

  remaining: function() {
    return this.filterProperty('isDone', false);
  }.property('@each.isDone')
});
Superficially Similar
- item marked done
- clear remaining cache
- compute remaining
- render stats view

**intermediate state**

run loop

x N
Easy to Overlook
ArrayController.create(
    content: [],
    remaining: function() {
        return this.filterProperty('isDone', false);
    }.property('@each.isDone')
});
Key: Declare Intent
"Coalesce"
Wrong
Right
run loop

A
"123"

B
"hello"
Not 3
Deferred Observers
Materialize Objects When Used
Large JSON Structure

{
    contacts: [
        { name: "Yehuda", ... },
        ... x 10,000
    ]
}

Acre, Julie
Appleseed, Johnny
Arrow, Bob
Astels, David
Atwood, Michael
Axelrod, Peter
Azeroth, Roy
Data Store
(JSON Hashes)
on demand

Contact
name
title
address
telephone
Ajax Response

Data Store (JSON Hashes)

RecordArray
Contacts
where company = "GOOGLE"
Contact data status

```json
{
    name: "yehuda",
    title: "Chief Technologist",
    address: "690 Spruce",
    telephone: "718.877.1325"
}
```
Again, Foundational
One More Thing
Page Speed
Packed Files
CDN: Versioned URLs
Stylesheets at the Top
Scripts at the Bottom
Simple Layout
External JS and CSS
Easier said than done
SproutCore
2.0
Overall File Size

Matters, but not as much as you think.
Modules w/ Declared Deps
Lazy Loading Modules
Thank you.
Questions