ConstraintJS

Programming Interactive Behaviors for the Web by Integrating Constraints and States

Stephen Oney, Brad Myers
Carnegie Mellon

Joel Brandt
Adobe

Constraints

- Relationships that are declared once & maintained automatically

Constraints

- Relationships that are declared once & maintained automatically
- “the toolbar is displayed above the workspace”

Constraints

- Relationships that are declared once & maintained automatically
- “the toolbar is displayed above the workspace”
- Can enable clearer, more concise code [Meyerovich, 2009; Myers, 1991]

Constraints

- Commercially: GUI layout & data binding

Constraints

- Commercially: GUI layout & data binding
- Constraint solvers can be unpredictable and difficult to control [Myers, 2000]

Constraints

- Commercially: GUI layout & data binding
- Constraint solvers can be unpredictable and difficult to control [Myers, 2000]
- “the toolbar is displayed above the workspace”

Constraints

- Commercially: GUI layout & data binding
- Constraint solvers can be unpredictable and difficult to control [Myers, 2000]
- “the toolbar is displayed above the workspace”
    - change the workspace or toolbar location

- “when the toolbar is docked, it is displayed above the workspace”
- “when the toolbar is being dragged, it follows the mouse”

- “when the toolbar is docked, it is displayed above the workspace”
- “when the toolbar is being dragged, it follows the mouse”

🟦 constraint

- “when the toolbar is docked, it is displayed above the workspace”
- “when the toolbar is being dragged, it follows the mouse”

■ state

- “when the toolbar is docked, it is displayed above the workspace”
- “when the toolbar is being dragged, it follows the mouse”

🟧 state 🟦 constraint

States & Constraints

- GUIs are state-oriented
    - Appearance & behavior

States & Constraints

- GUIs are state-oriented
    - Appearance & behavior
- Finite-state machines control when constraints are enabled/disabled

goal: reduce the complexity of programming interactive applications goal: reduce the complexity of programming interactive applications

hypothesis: interactive behaviors can be easier and more concisely expressed by combining states and constraints

Web development

- HTML (declarative) - content

Web development

- HTML (declarative) - content
- CSS (declarative) - style

Web development

- HTML (declarative) - content
- CSS (declarative) - style
- Javascript (imperative) - interactivity

ConstraintJS

- Integrates constraints & states on Web

ConstraintJS

- Integrates constraints & states on Web
- Integrates with HTML & CSS syntaxes

ConstraintJS

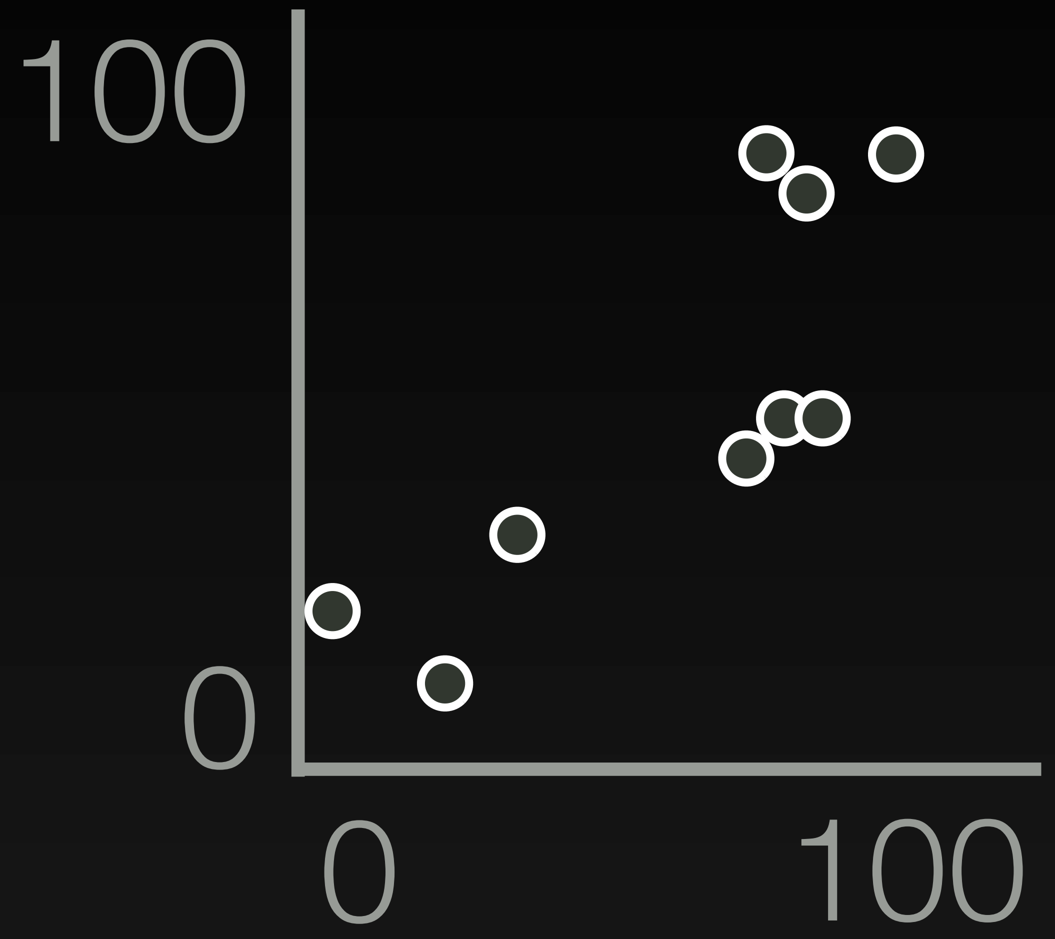
- Integrates constraints & states on Web
- Integrates with HTML & CSS syntaxes
- Efficient implementation

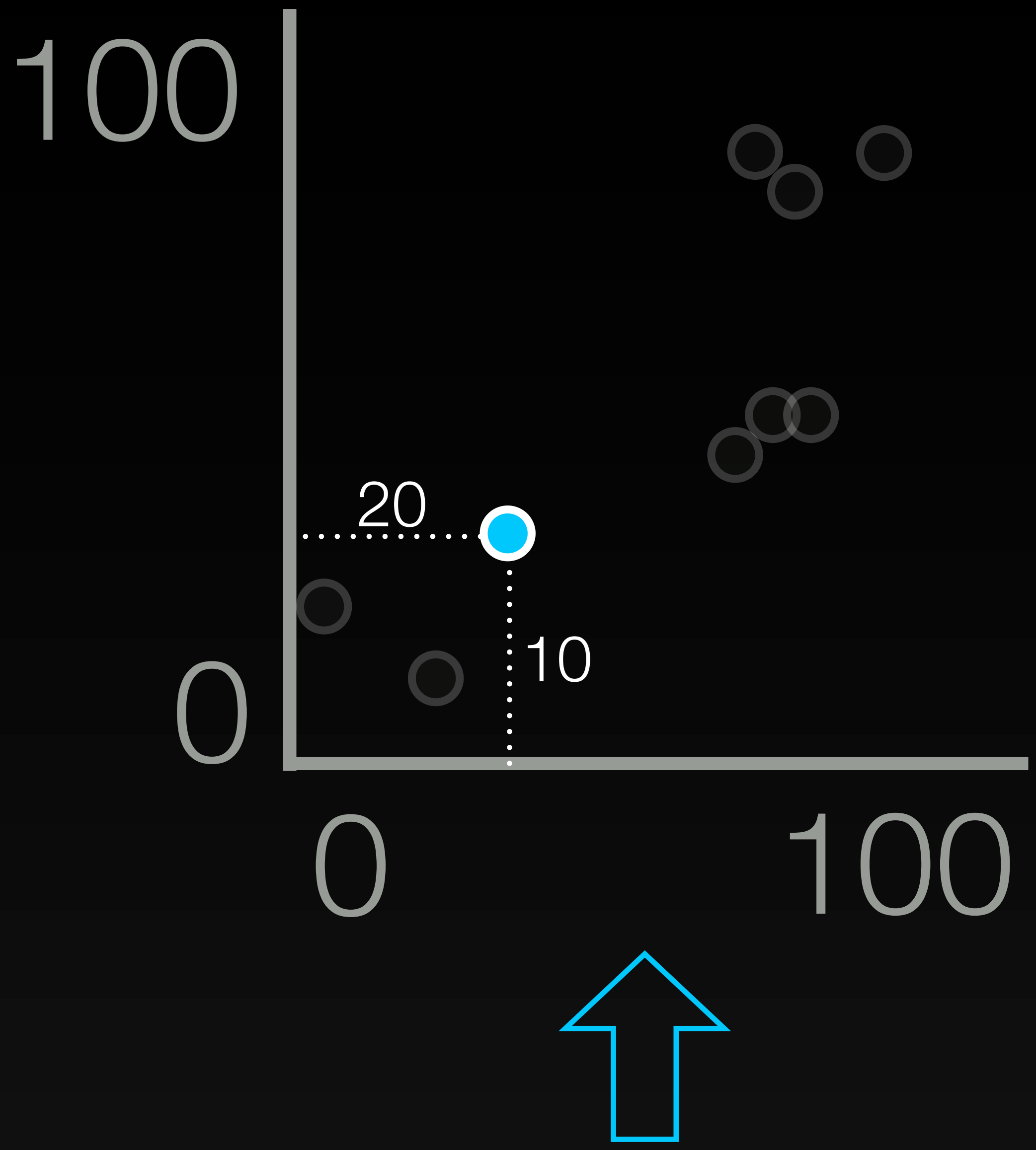
ConstraintJS

- Motivating example
  - Tying FSMs with states
- Fitting in with Web languages
  - Styles (CSS)
  - Templates (HTML)
  - Asynchronous values

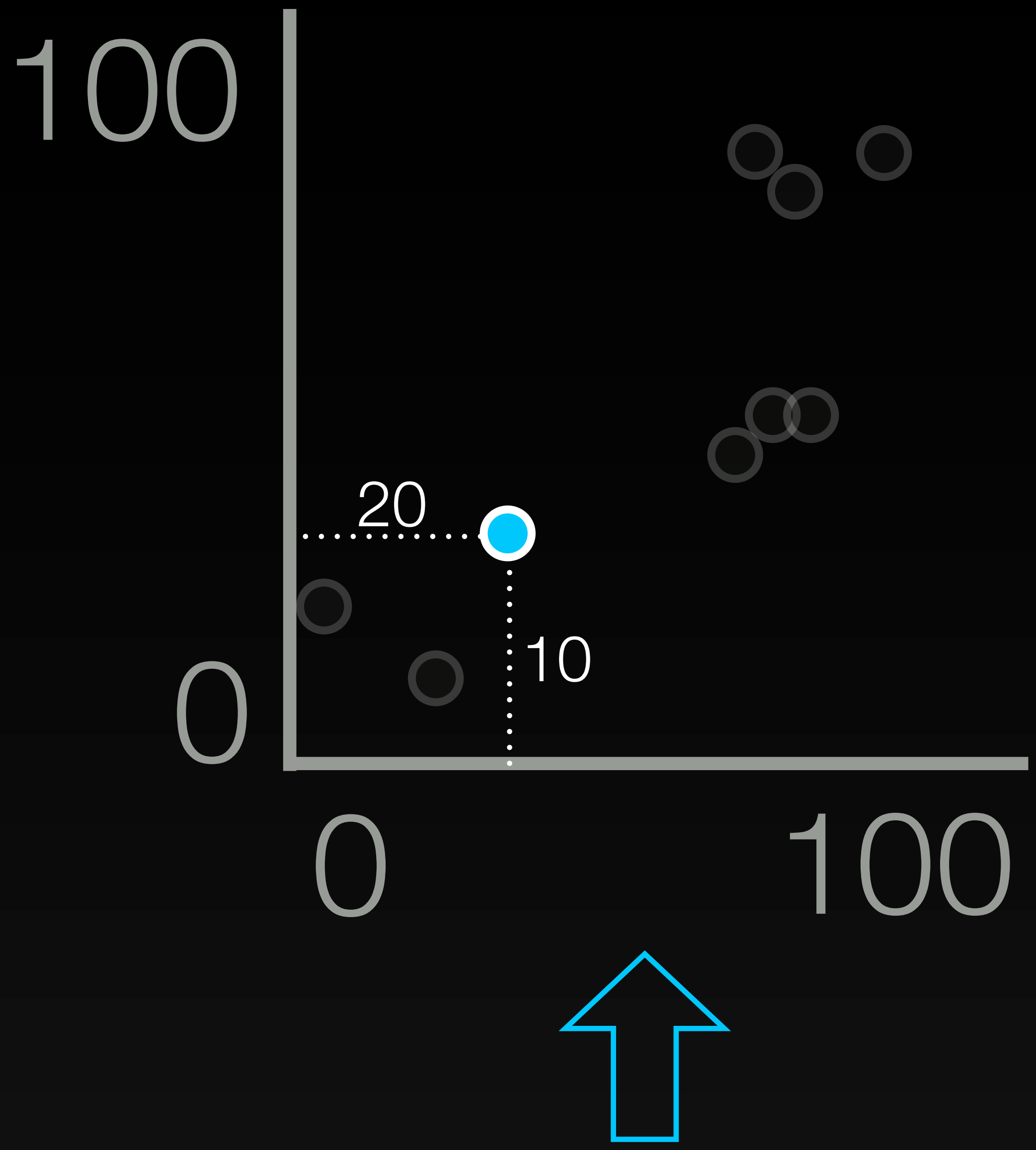
ConstraintJS

- Motivating example
  - Tying FSMs with states
- Fitting in with Web languages
  - Styles (CSS)
  - Templates (HTML)
  - Asynchronous values

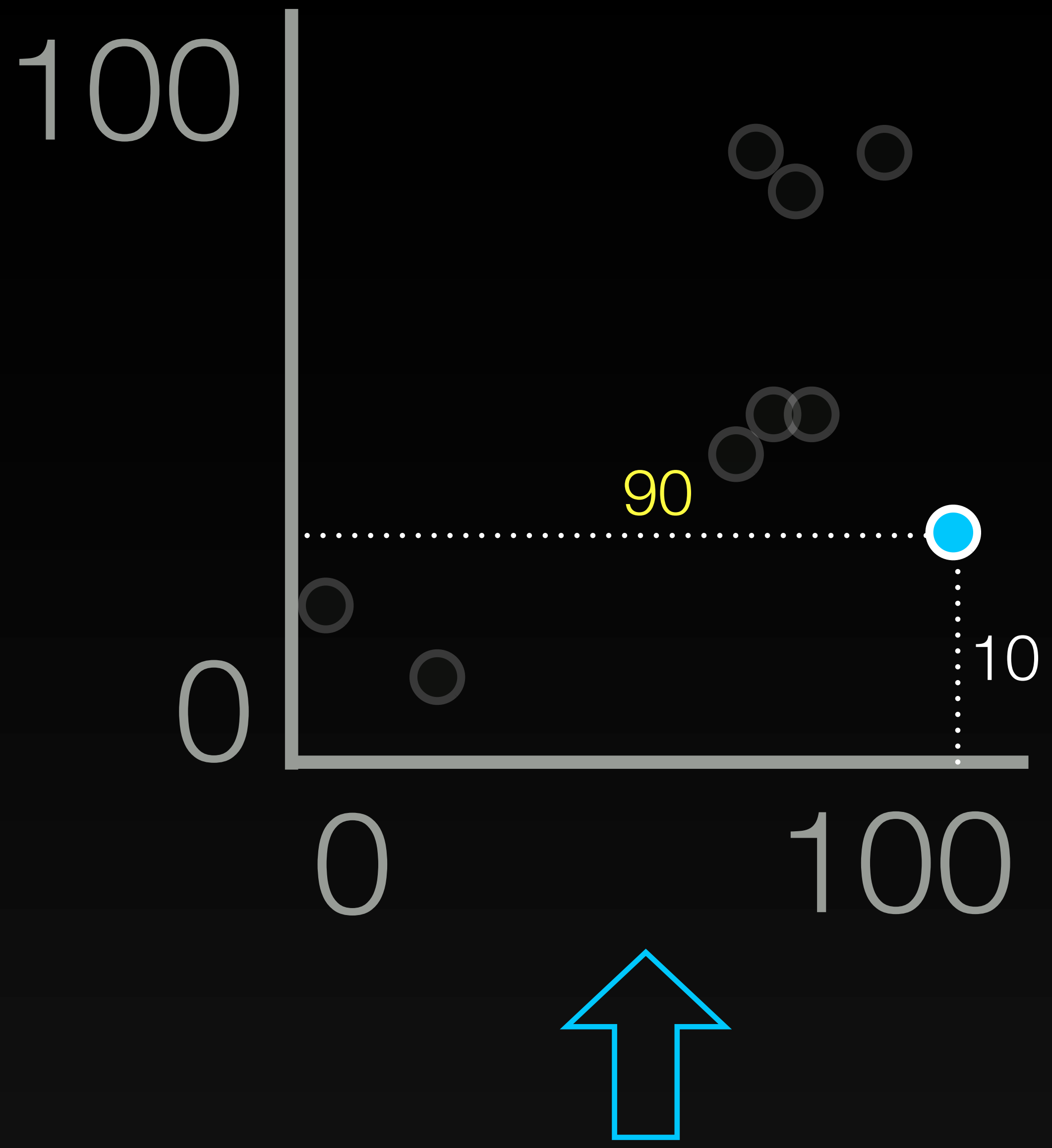

100
0
0
100



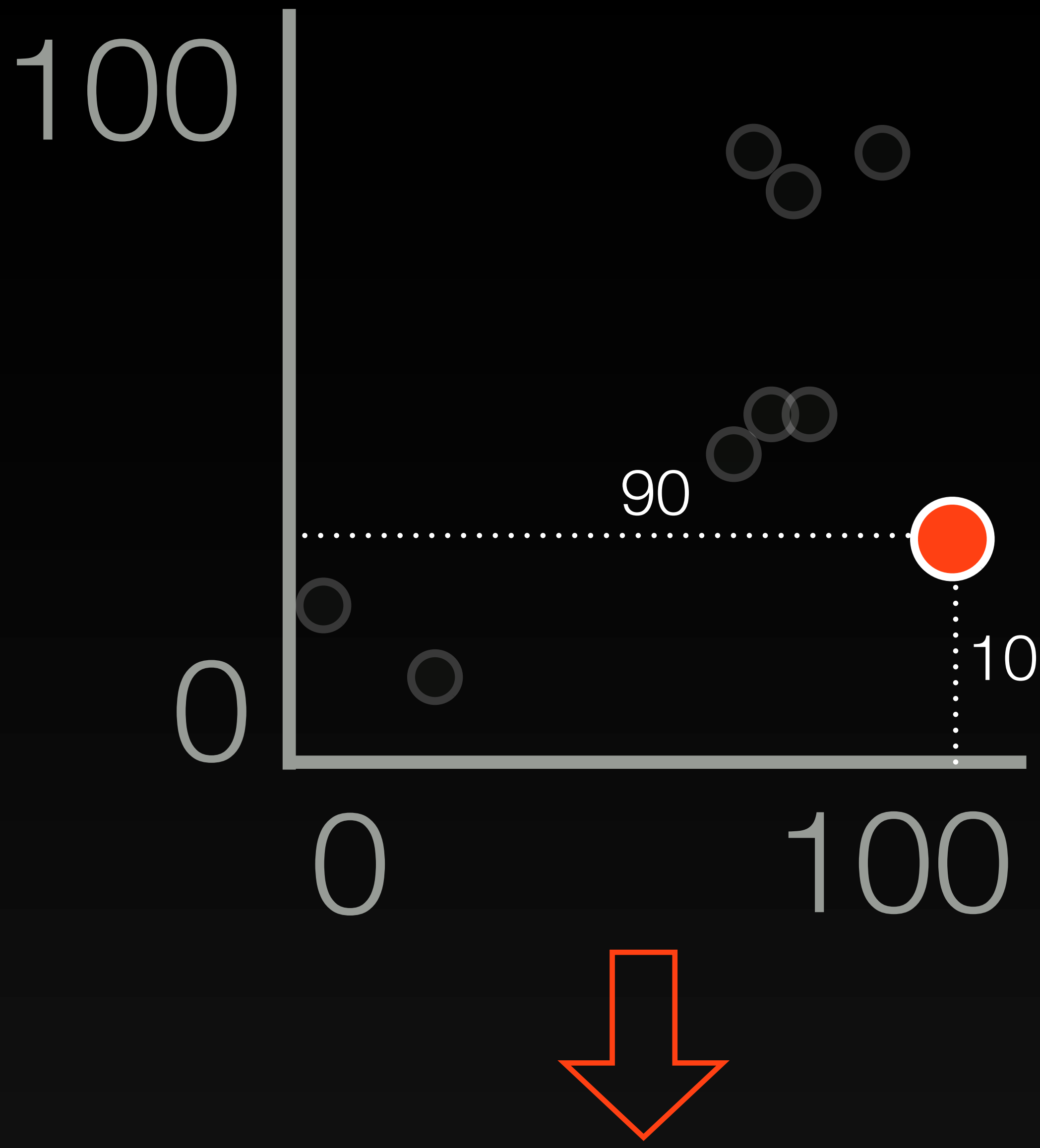
{x:5, y: 20}, {x: 20, y: 10}, {x:30, y:30},
{x:60, y:40}, {x: 65, y: 45}, {x: 70, y: 45},
{x: 63, y: 80}, {x: 68, y: 75}, {x: 80, y: 80}



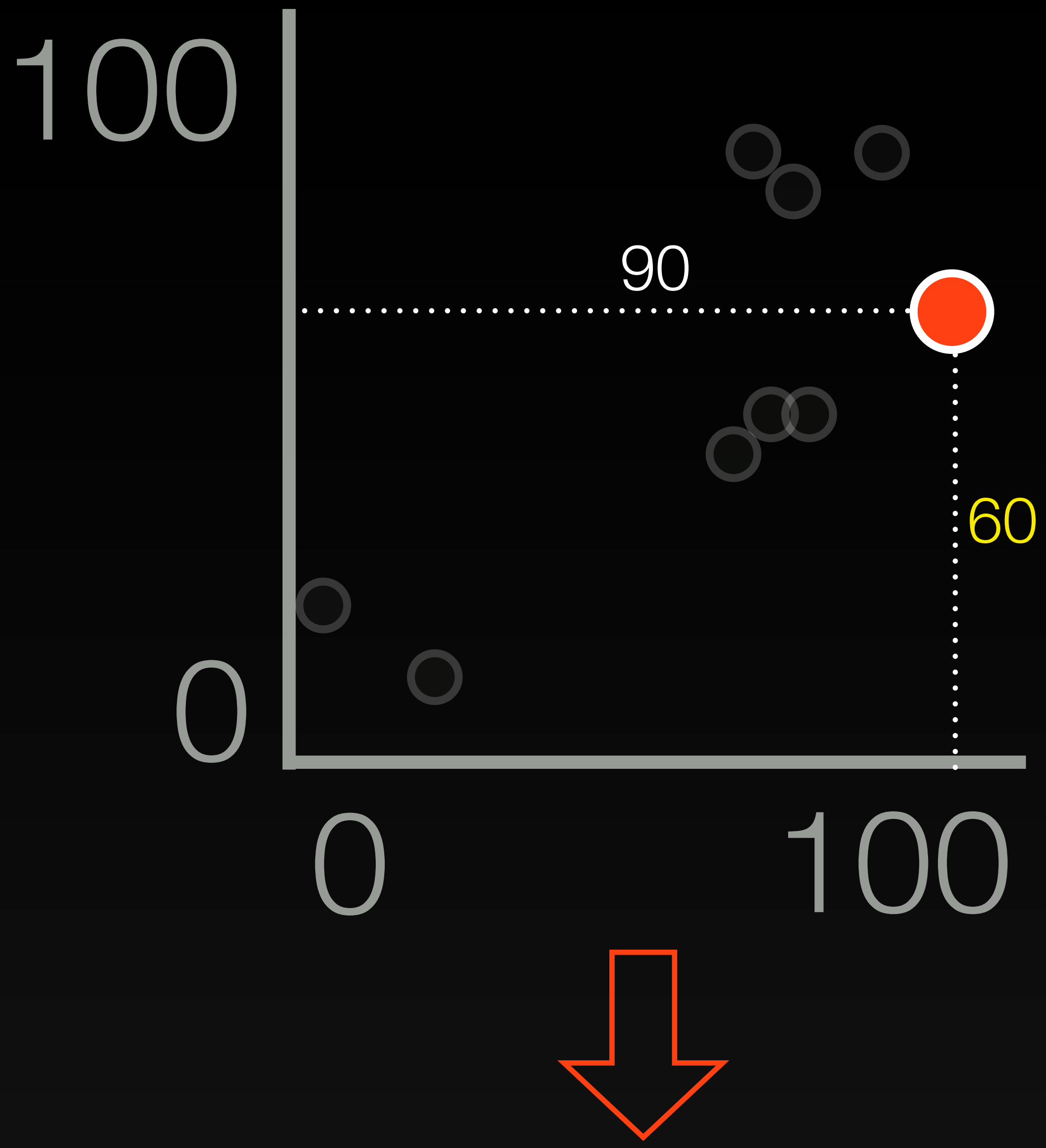
{x:5, y: 20}, {x: 90, y: 10}, {x:30, y:30},
{x:60, y:40}, {x: 65, y: 45}, {x: 70, y: 45},
{x: 63, y: 80}, {x: 68, y: 75}, {x: 80, y: 80}



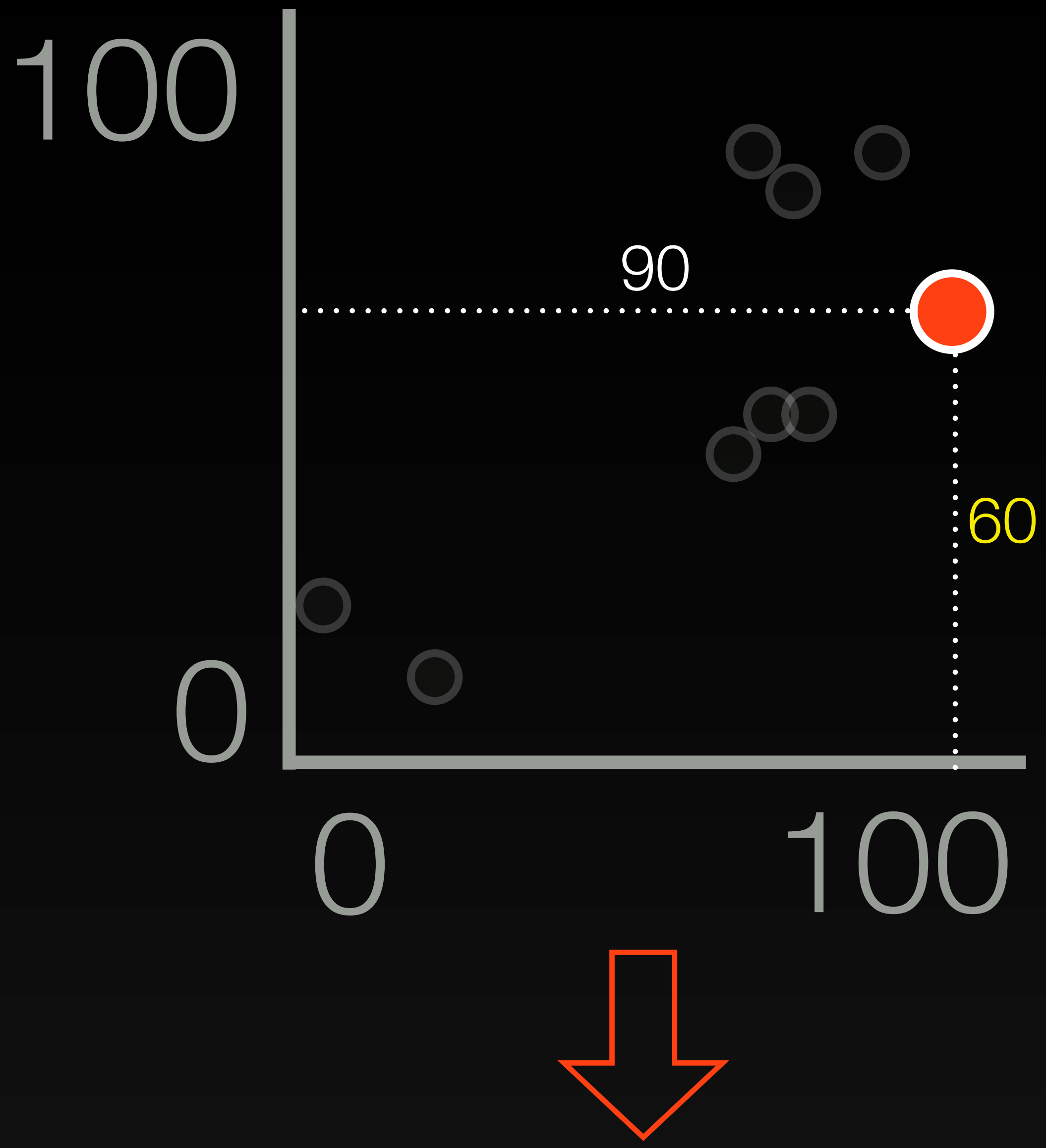
{x:5, y: 20}, {x: 90, y: 10}, {x:30, y:30},
{x:60, y:40}, {x: 65, y: 45}, {x: 70, y: 45},
{x: 63, y: 80}, {x: 68, y: 75}, {x: 80, y: 80}



{x:5, y: 20}, {x: 90, y: 10}, {x:30, y:30},
{x:60, y:40}, {x: 65, y: 45}, {x: 70, y: 45},
{x: 63, y: 80}, {x: 68, y: 75}, {x: 80, y: 80}



{x:5, y: 20}, {x: 90, y: 10}, {x:30, y:30},
{x:60, y:40}, {x: 65, y: 45}, {x: 70, y: 45},
{x: 63, y: 80}, {x: 68, y: 75}, {x: 80, y: 80}

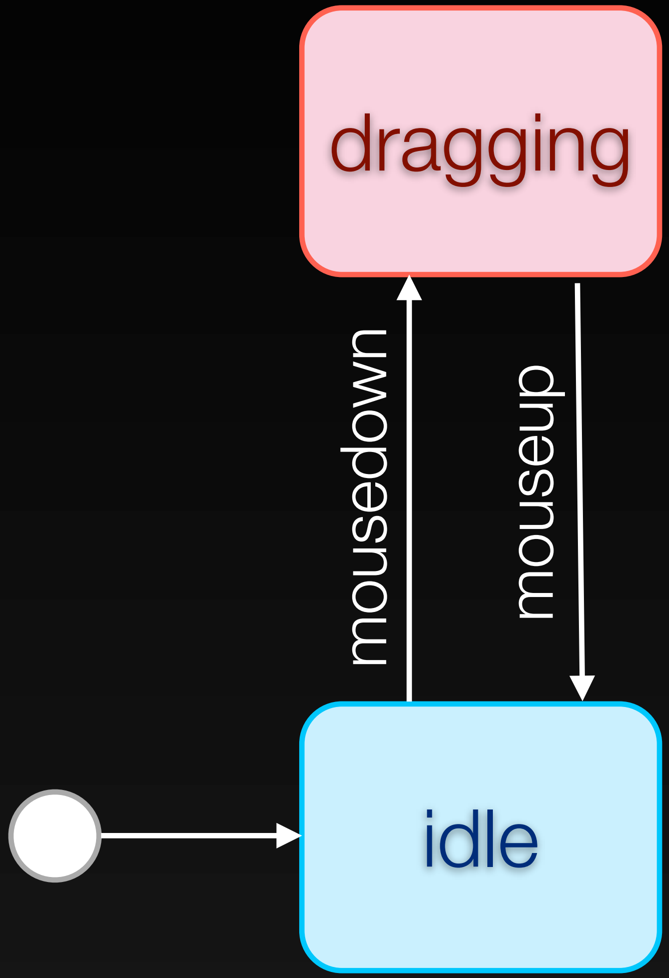


{x:5, y: 20}, {x: 90, y: 60}, {x:30, y:30},
{x:60, y:40}, {x: 65, y: 45}, {x: 70, y: 45},
{x: 63, y: 80}, {x: 68, y: 75}, {x: 80, y: 80}

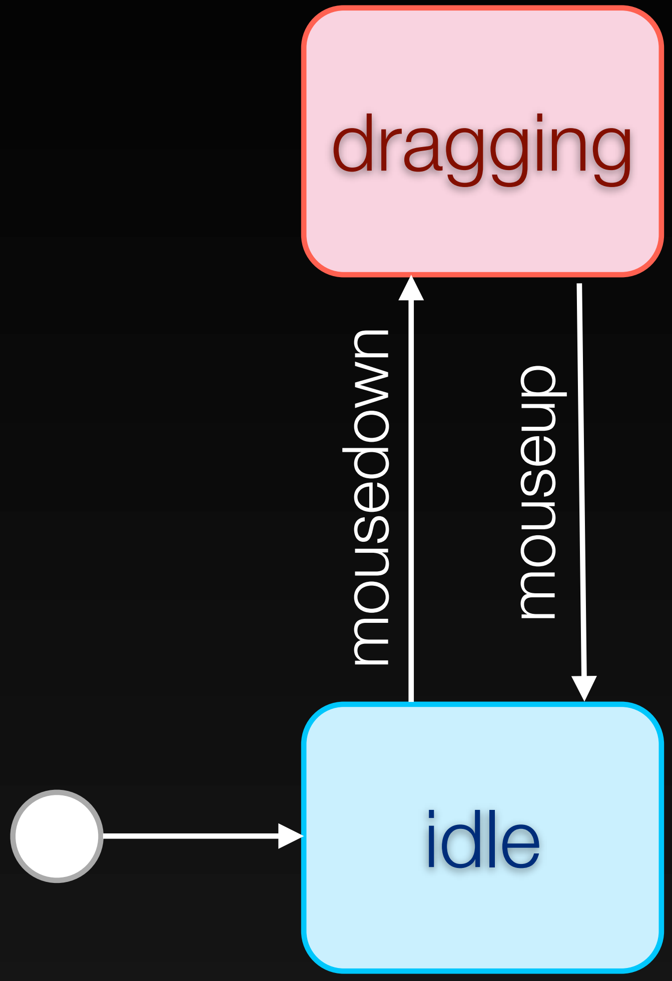
Multi-way Constraints

- Constraints where
  A depends on B & B depends on A
- Specify constraint hierarchy
- Difficult to control [Vander Zanden, 1994]

`fsm *for every point*`

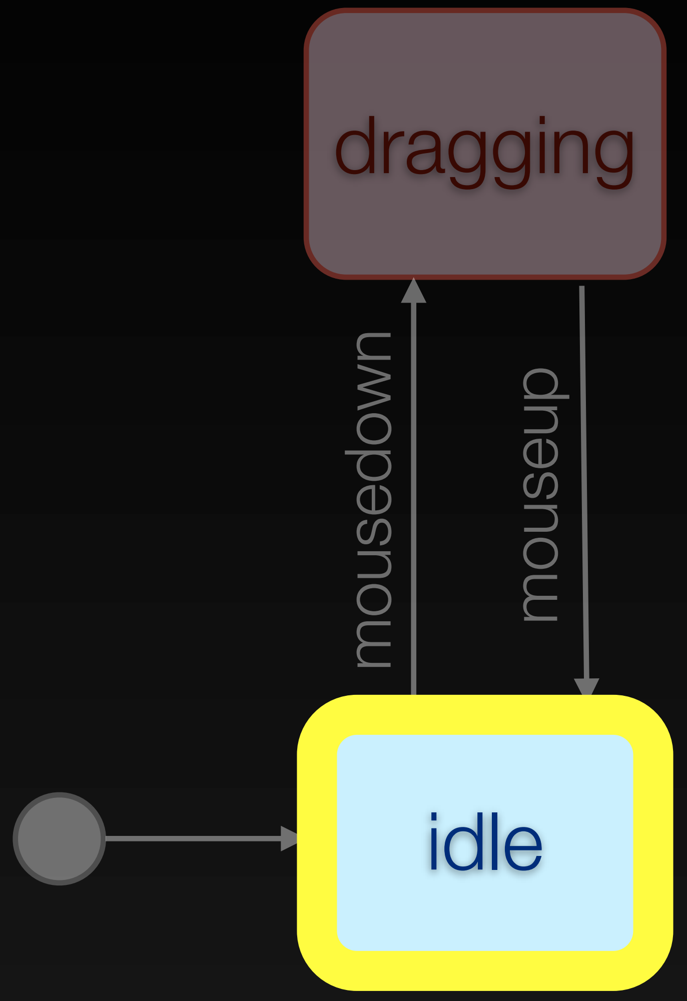


fsm:



```
view_x = cjs(fsm, {
   idle:      model_x,
   dragging: cjs.mouse.x
});


model_x = cjs(fsm, {
   init:      datum.x,
   dragging: view_x
});
```

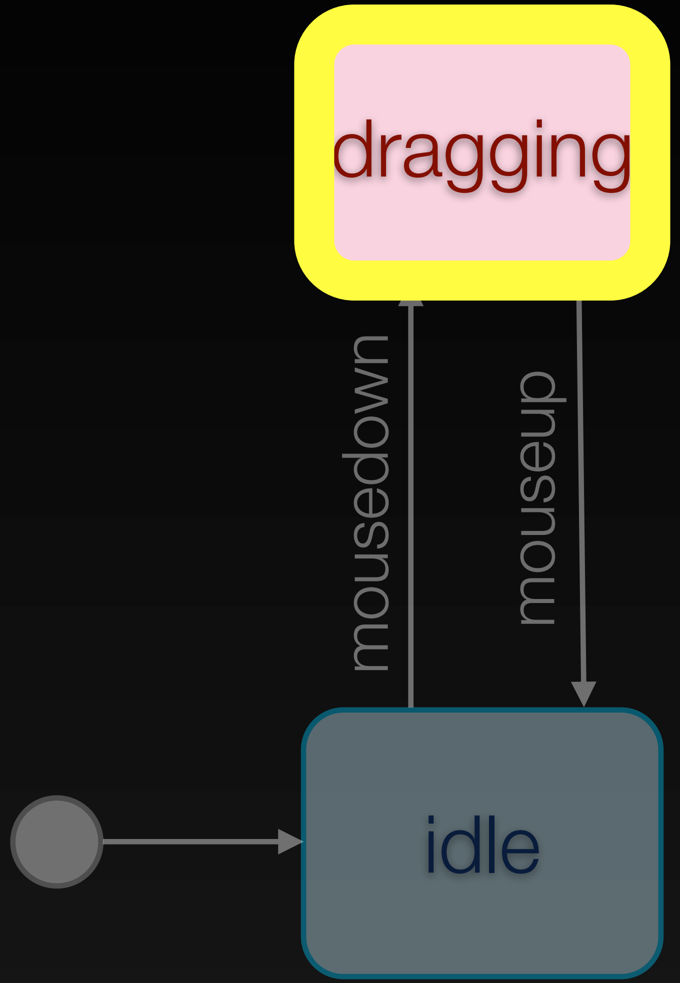
fsm:



```
view_x = cjs(fsm, {
  idle:      model_x,
  dragging: cjs.mouse.x
});
```

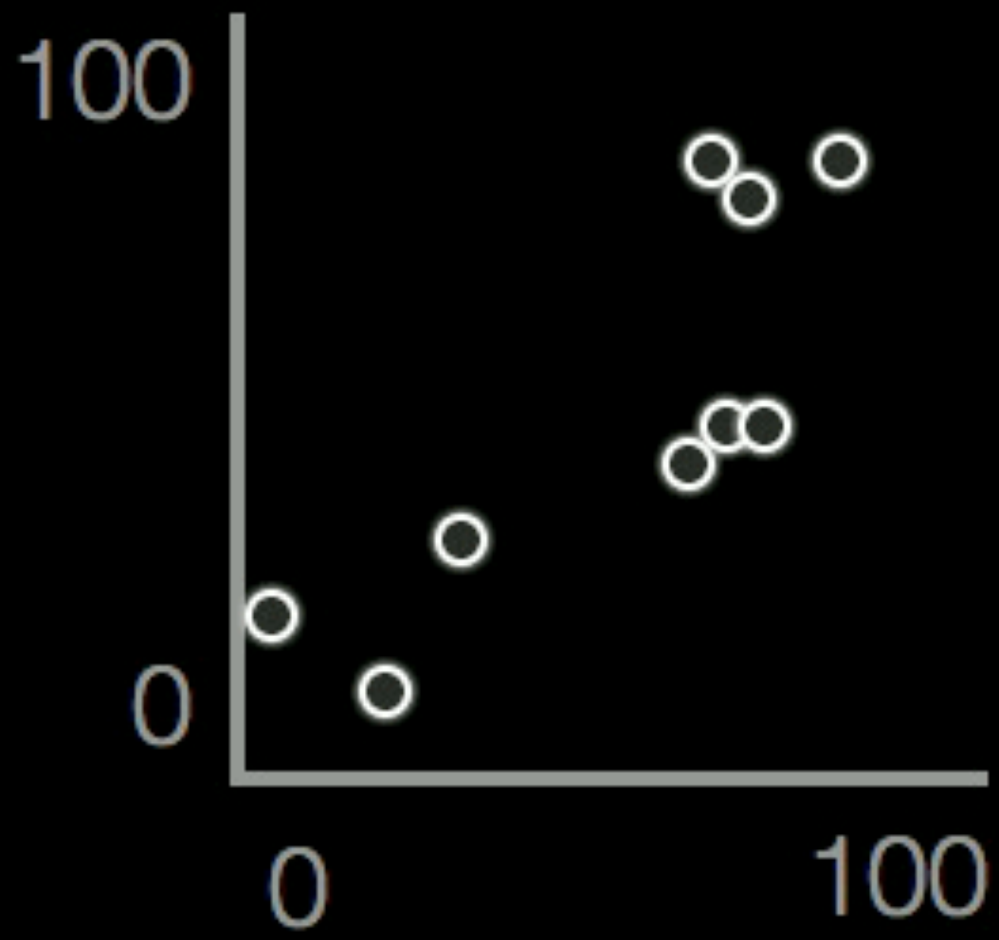
```
model_x = cjs(fsm, {
  init:      datum.x,
  dragging: view_x
});
```

fsm:



```
view_x = cjs(fsm, {
   idle:      model_x,
   dragging: cjs.mouse.x
});
```

```
model_x = cjs(fsm, {
   init:      datum.x,
   dragging: view_x
});
```


100
0
0
100

ConstraintJS

- Motivating example
  - Tying FSMs with states
- Fitting in with Web languages
  - Styles (CSS)
  - Templates (HTML)
  - Asynchronous values

Styles

```
cjs.css(element,
“background-color”,
$selected_color);
```

Styles

```
cjs.css(element,
“background-color”,
$selected_color);
```

Styles

```
cjs.css(element,
“background-color”,
$selected_color);
```

Styles

```
cjs.css(element,
“background-color”,
$selected_color);
```

ConstraintJS

- Motivating example
  - Tying FSMs with states
- Fitting in with Web languages
  - Styles (CSS)
  - Templates (HTML)
  - Asynchronous values

Templates

Templates

```
{{#if form_complete}}
<button>Submit</button>
{{#else}}
<div>Incomplete form...</div>
{{/if}}
```

ConstraintJS

- Motivating example
  - Tying FSMs with states
- Fitting in with Web languages
  - Styles (CSS)
  - Templates (HTML)
  - Asynchronous values

Asynchronous Values

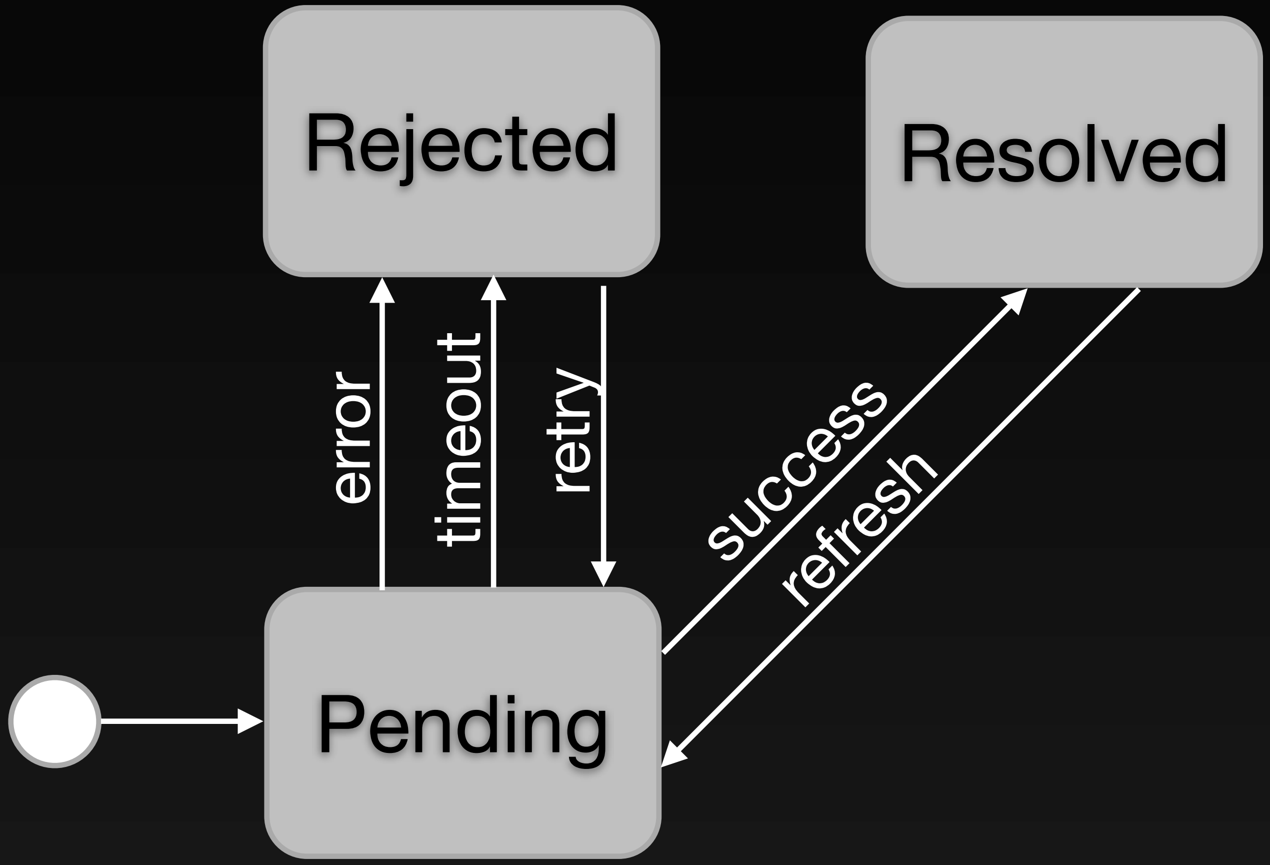
Asynchronous Values

- Indeterminate wait before return

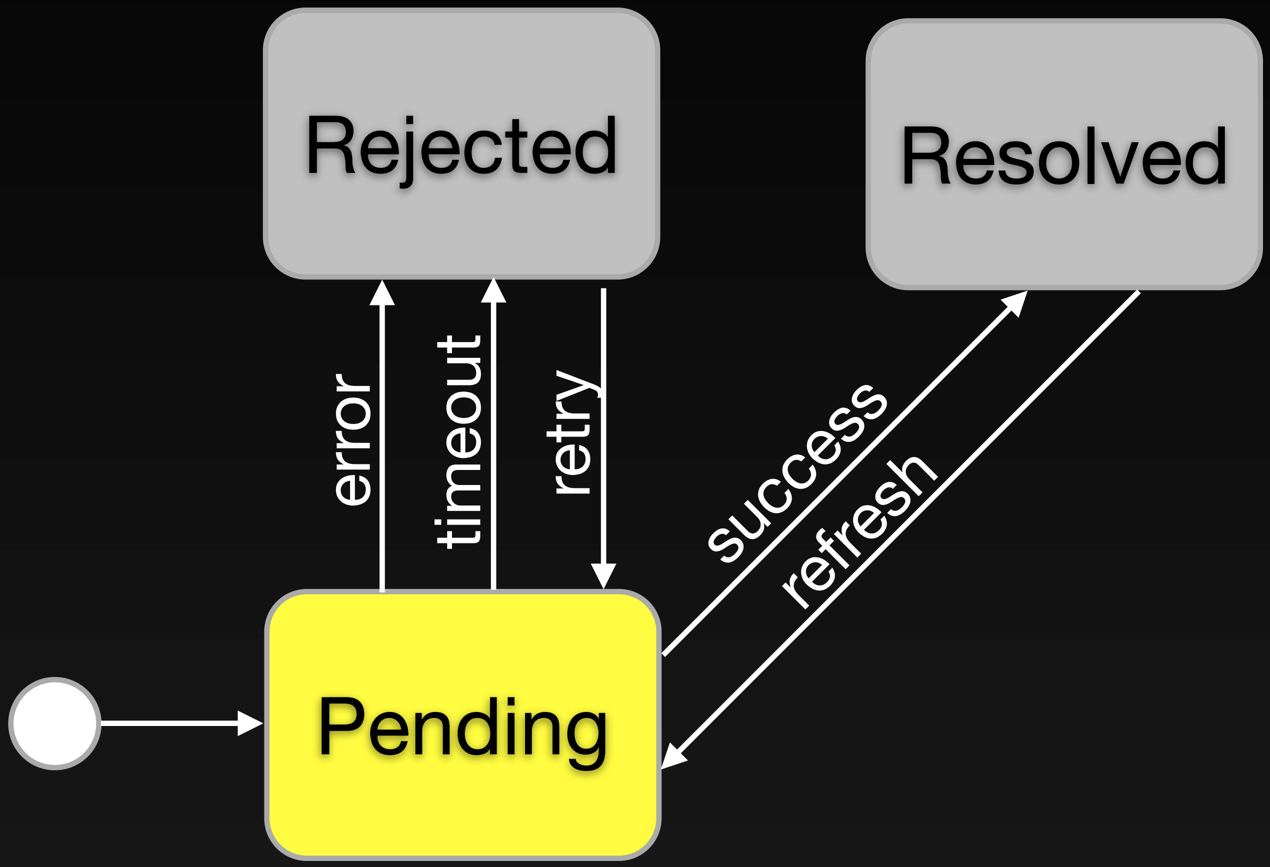
Asynchronous Values

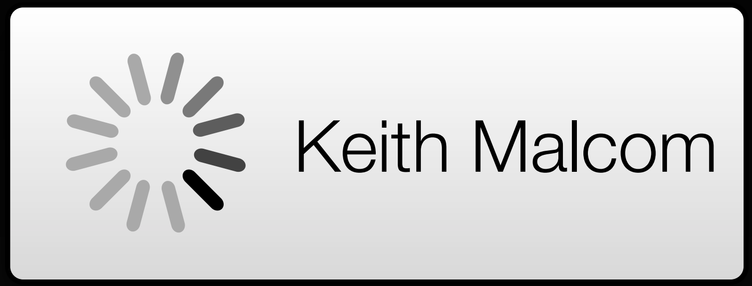
- Indeterminate wait before return
- Can greatly increase complexity of imperative code
    - Control timing
    - Propagation of values

Asynchronous Values



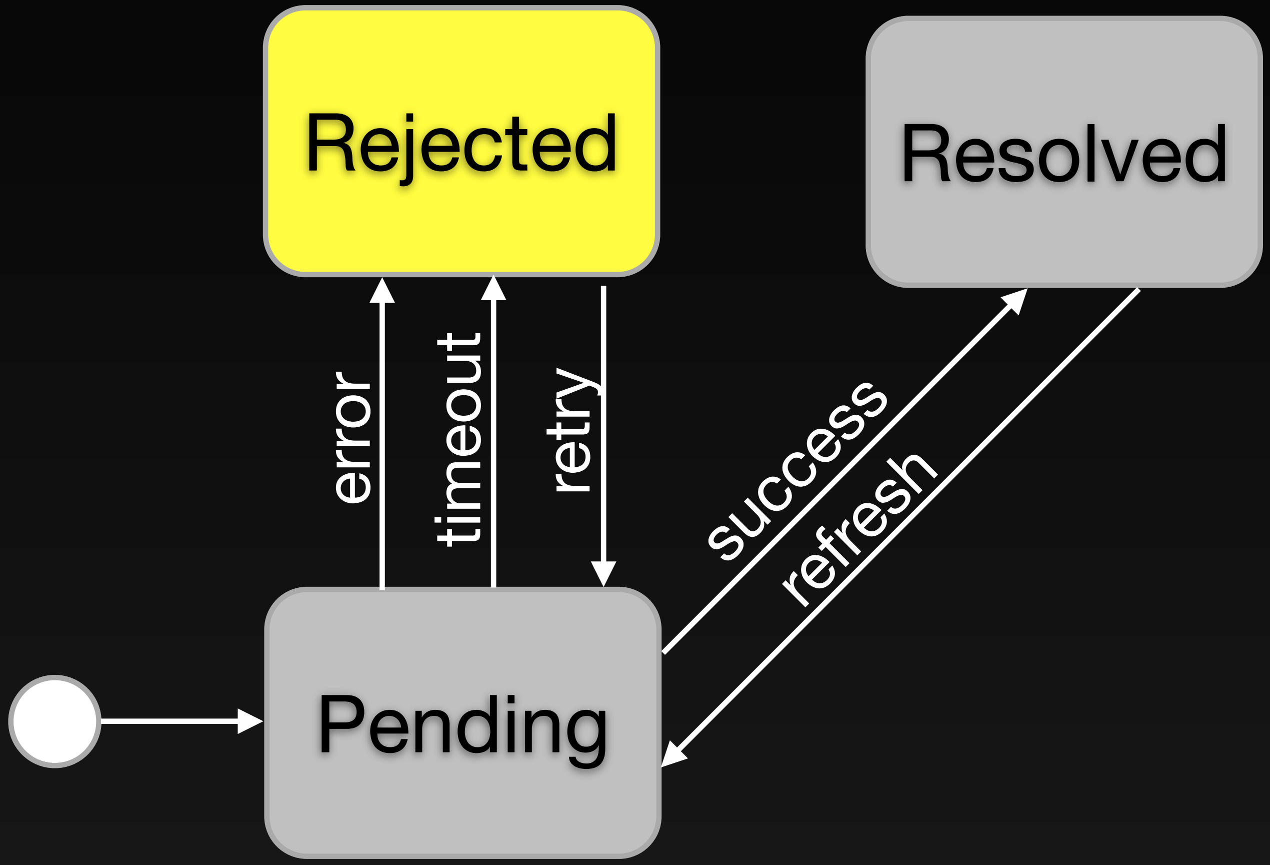
Asynchronous Values

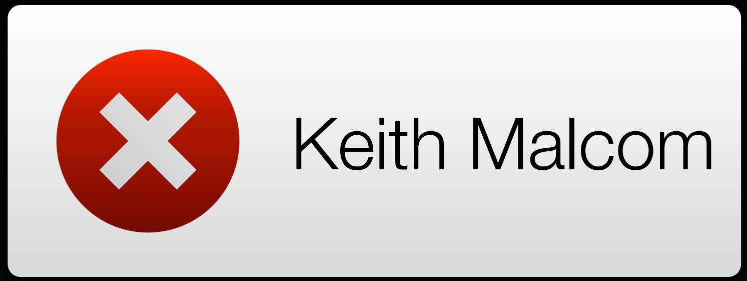




```
pending:  "loading.gif"
```

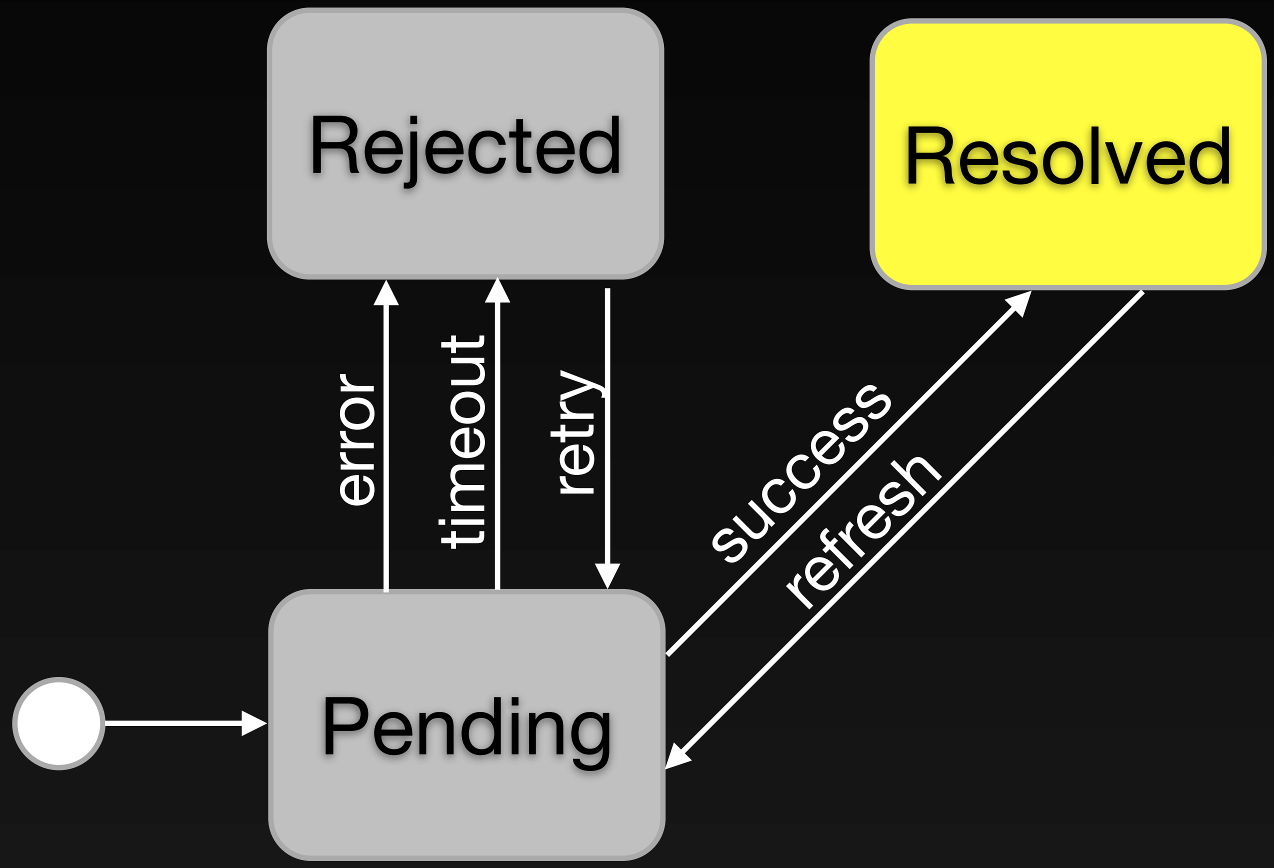
Asynchronous Values





```
pending:  "loading.gif",
rejected: "error.gif"
```

Asynchronous Values



Rejected

Resolved

Pending

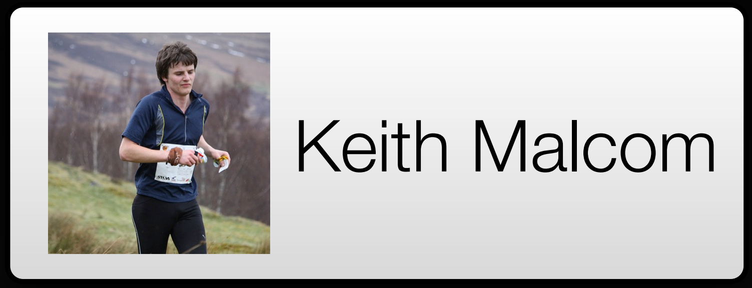
error

timeout

retry

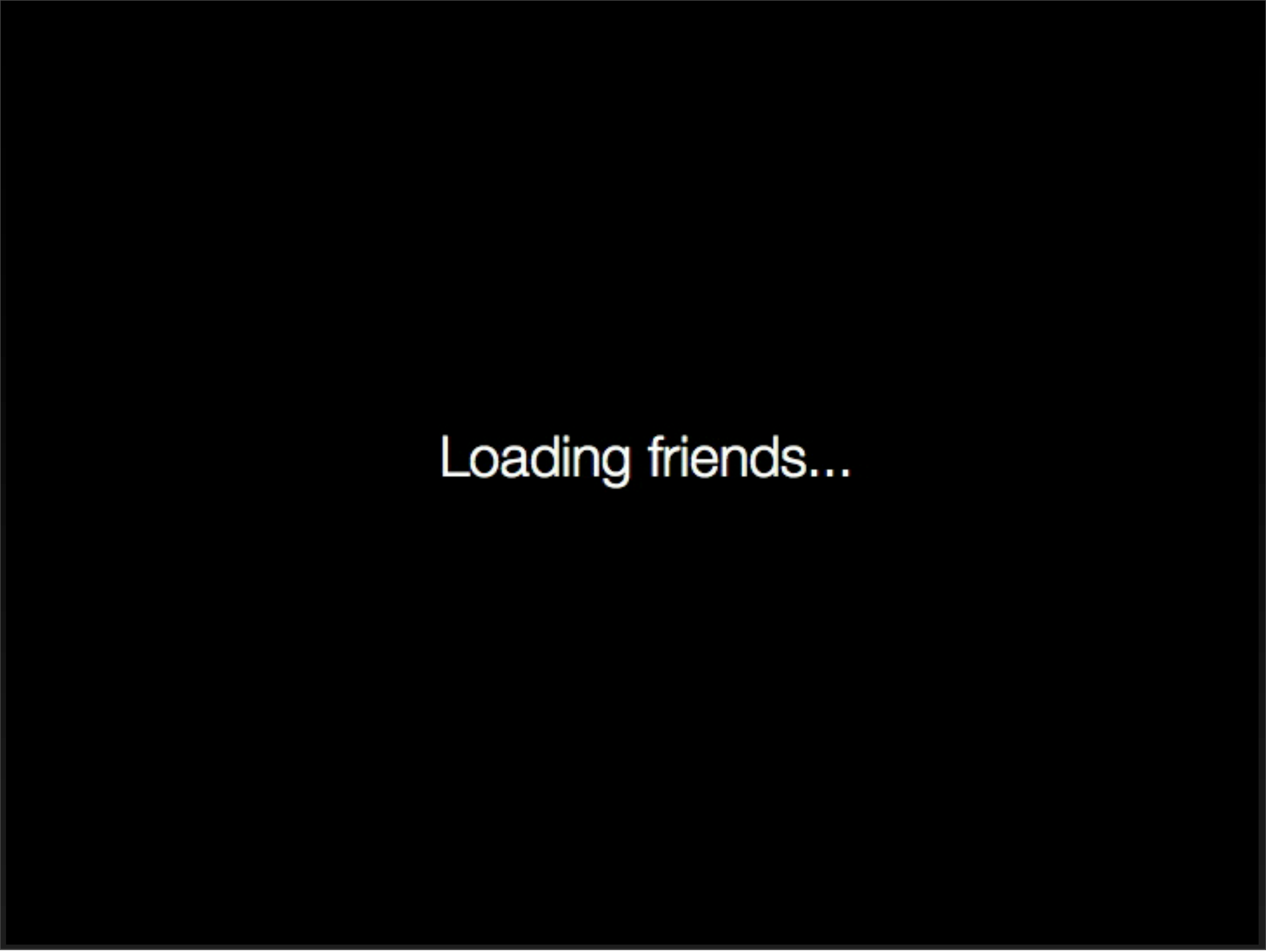
success

refresh



Keith Malcom

```
pending:  "loading.gif",
rejected: "error.gif",
resolved: {{picture}}
```



Loading friends...

```
friends = cjs.async(fb_request("/me/friends"));
pics    = friends.map(function(friend) {
              return cjs.async(fb_request( "/" + friend.id
                                               + "/picture"));
          });

//...

{{#diagram friends.state}}
   {{#state pending }} Loading friends...
   {{#state rejected}} Error
   {{#state resolved}}
      {{#each friends friend i}}
         {{#diagram pics[i].state}}
            {{#state pending }} <img src = "loading.gif" />
            {{#state resolved}} <img src = "{{pics[i]}}" />
            {{#state rejected}} <img src = "error.gif"   />
         {{/diagram}}
         {{friend.name}}
      {{/each}}
{{/diagram}}
```

```
friends = cjs.async(fb_request("/me/friends"));
pics    = friends.map(function(friend) {
              return cjs.async(fb_request( "/" + friend.id
                                             + "/picture"));
          });

//...

{{#diagram friends.state}}
   {{#state pending }} Loading friends...
   {{#state rejected}} Error
   {{#state resolved}}
      {{#each friends friend i}}
         {{#diagram pics[i].state}}
            {{#state pending }} <img src = "loading.gif" />
            {{#state resolved}} <img src = "{{pics[i]}}" />
            {{#state rejected}} <img src = "error.gif"   />
         {{/diagram}}
         {{friend.name}}
      {{/each}}
{{/diagram}}
```

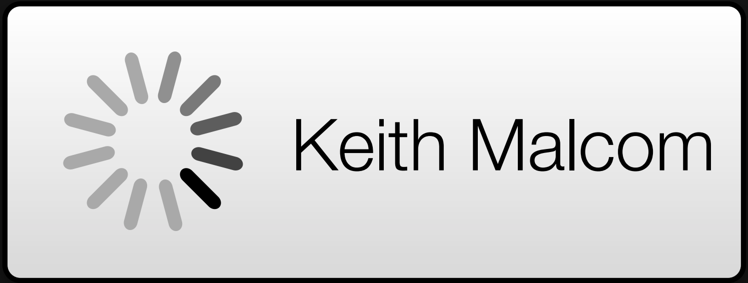
```
friends = cjs.async(fb_request("/me/friends"));
pics    = friends.map(function(friend) {
              return cjs.async(fb_request( "/" + friend.id
                                               + "/picture"));
          });

//...

{{#diagram friends.state}}
   {{#state pending }} Loading friends...
   {{#state rejected}} Error
   {{#state resolved}}
      {{#each friends friend i}}
         {{#diagram pics[i].state}}
            {{#state pending }} <img src = "loading.gif" />
            {{#state resolved}} <img src = "{{pics[i]}}" />
            {{#state rejected}} <img src = "error.gif"   />
         {{/diagram}}
         {{friend.name}}
      {{/each}}
{{/diagram}}
```

```
friends = cjs.async(fb_request("/me/friends"));
pics    = friends.map(function(friend) {
              return cjs.async(fb_request( "/" + friend.id
                                               + "/picture"));
          });

//...

{{#diagram friends.state}}
   {{#state pending }} Loading friends...
   {{#state rejected}} Error
   {{#state resolved}}
      {{#each friends friend i}}
         {{#diagram pics[i].state}}
            {{#state pending }} <img src = "loading.gif" />
            {{#state resolved}} <img src = "{{pics[i]}}" />
            {{#state rejected}} <img src = "error.gif"   />
         {{/diagram}}
         {{friend.name}}
      {{/each}}
{{/diagram}}
```

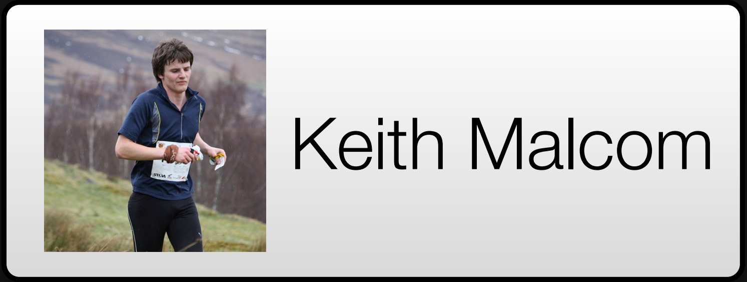
```
friends = cjs.async(fb_request("/me/friends"));
pics    = friends.map(function(friend) {
              return cjs.async(fb_request( "/" + friend.id
                                               + "/picture"));
          });

//...

{{#diagram friends.state}}
   {{#state pending }} Loading friends...
   {{#state rejected}} Error
   {{#state resolved}}
      {{#each friends friend i}}
         {{#diagram pics[i].state}}
            {{#state pending }} <img src = "loading.gif" />
            {{#state resolved}} <img src = "{{pics[i]}}" />
            {{#state rejected}} <img src = "error.gif"   />
         {{/diagram}}
         {{friend.name}}
      {{/each}}
{{/diagram}}
```

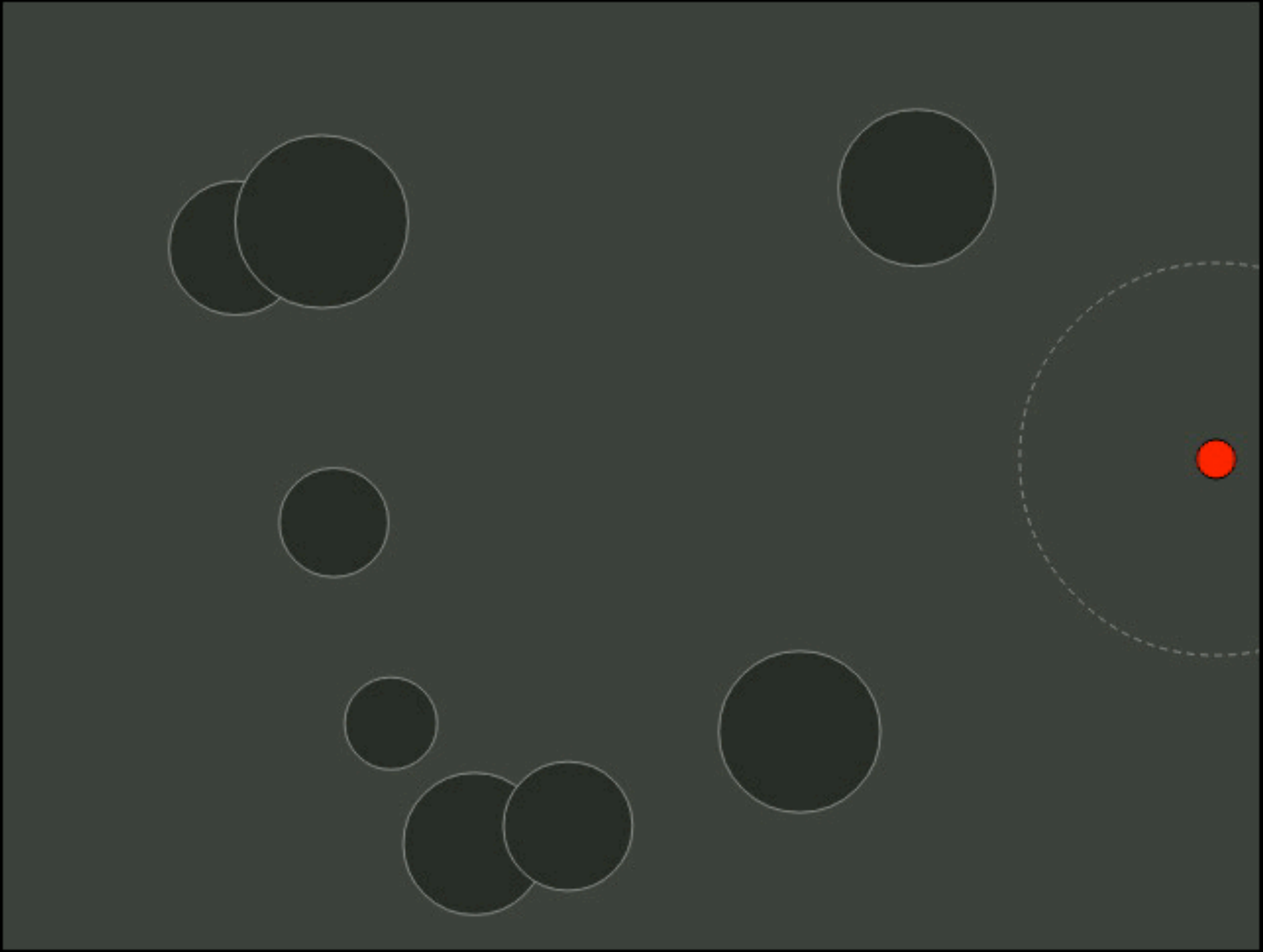
```
friends = cjs.async(fb_request("/me/friends"));
pics    = friends.map(function(friend) {
              return cjs.async(fb_request( "/" + friend.id
                                               + "/picture"));
          });

//...

{{#diagram friends.state}}
   {{#state pending }} Loading friends...
   {{#state rejected}} Error
   {{#state resolved}}
      {{#each friends friend i}}
         {{#diagram pics[i].state}}
            {{#state pending }} <img src = "loading.gif" />
            {{#state resolved}} <img src = "{{pics[i]}}" />
            {{#state rejected}} <img src = "error.gif"   />
         {{/diagram}}
         {{friend.name}}
      {{/each}}
{{/diagram}}
```

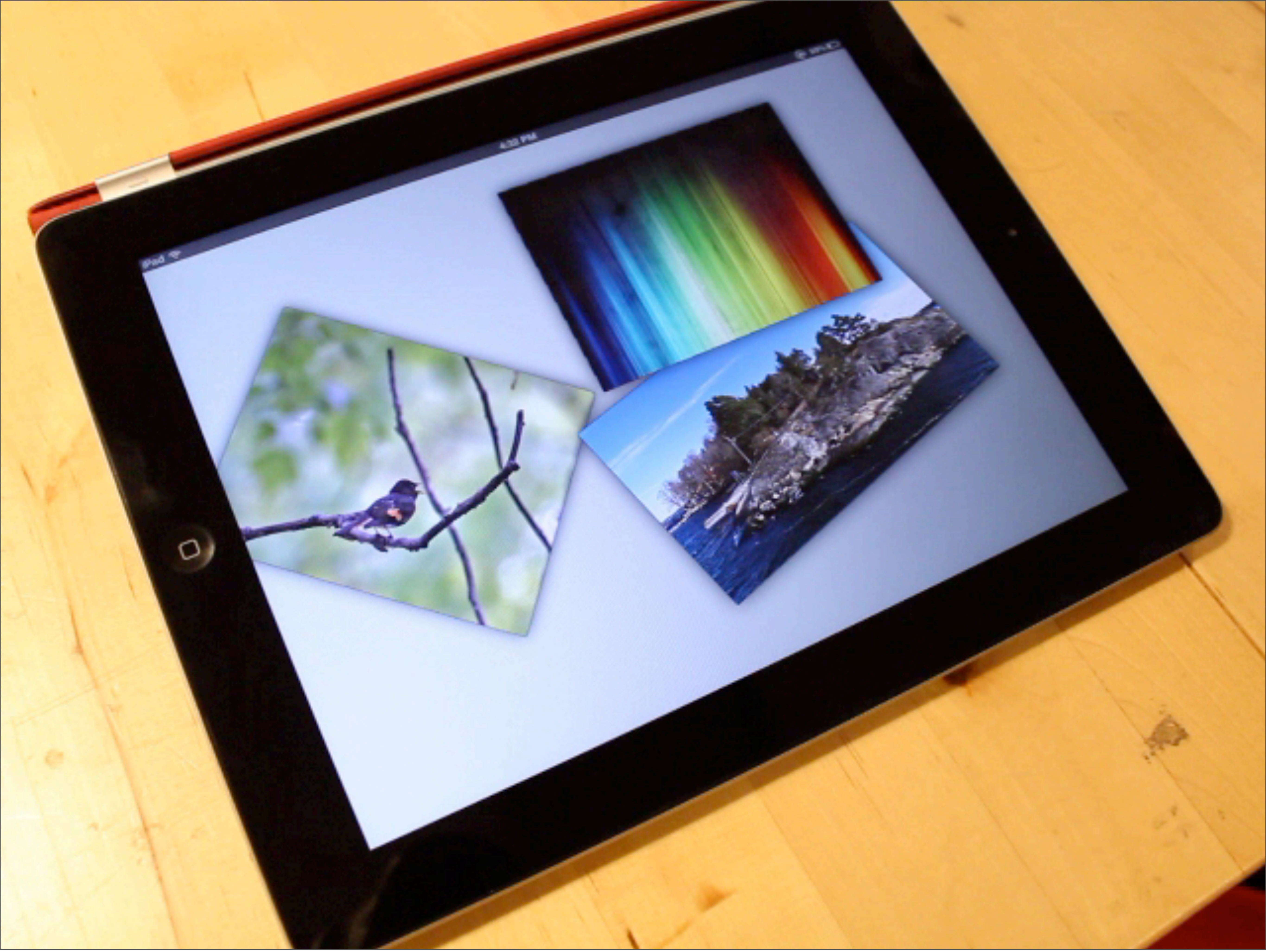
```
friends = cjs.async(fb_request("/me/friends"));
pics    = friends.map(function(friend) {
              return cjs.async(fb_request( "/" + friend.id
                                             + "/picture"));
          });

//...

{{#diagram friends.state}}
   {{#state pending }} Loading friends...
   {{#state rejected}} Error
   {{#state resolved}}
      {{#each friends friend i}}
         {{#diagram pics[i].state}}
            {{#state pending }} <img src = "loading.gif" />
            {{#state resolved}} <img src = "{{pics[i]}}" />
            {{#state rejected}} <img src = "error.gif"   />
         {{/diagram}}
         {{friend.name}}
      {{/each}}
{{/diagram}}
```



```
friends = cjs.async(fb_request("/me/friends"));
pics    = friends.map(function(friend) {
              return cjs.async(fb_request( "/" + friend.id
                                               + "/picture"));
          });

//...

{{#diagram friends.state}}
   {{#state pending }} Loading friends...
   {{#state rejected}} Error
   {{#state resolved}}
      {{#each friends friend i}}
         {{#diagram pics[i].state}}
            {{#state pending }} <img src = "loading.gif" />
            {{#state resolved}} <img src = "{{pics[i]}}" />
            {{#state rejected}} <img src = "error.gif"   />
         {{/diagram}}
         {{friend.name}}
      {{/each}}
{{/diagram}}
```






iPad
4:32 PM

ConstraintJS

- Combines constraints & FSMs

ConstraintJS

- Combines constraints & FSMs
    - Enable more controllable constraints

ConstraintJS

- Combines constraints & FSMs
    - Enable more controllable constraints
- Integrates with Web languages

ConstraintJS

- Combines constraints & FSMs
  - Enable more controllable constraints
- Integrates with Web languages
- Efficient implementation

ConstraintJS

- Combines constraints & FSMs
  - Enable more controllable constraints
- Integrates with Web languages
- Efficient implementation
- Demonstration through examples

ConstraintJS

http://cjs.from.so/

Stephen Oney (CMU)
Brad Myers (CMU)
Joel Brandt (Adobe)

Thanks to Adobe, Microsoft SEIF, NSF, and the Ford Foundation for funding