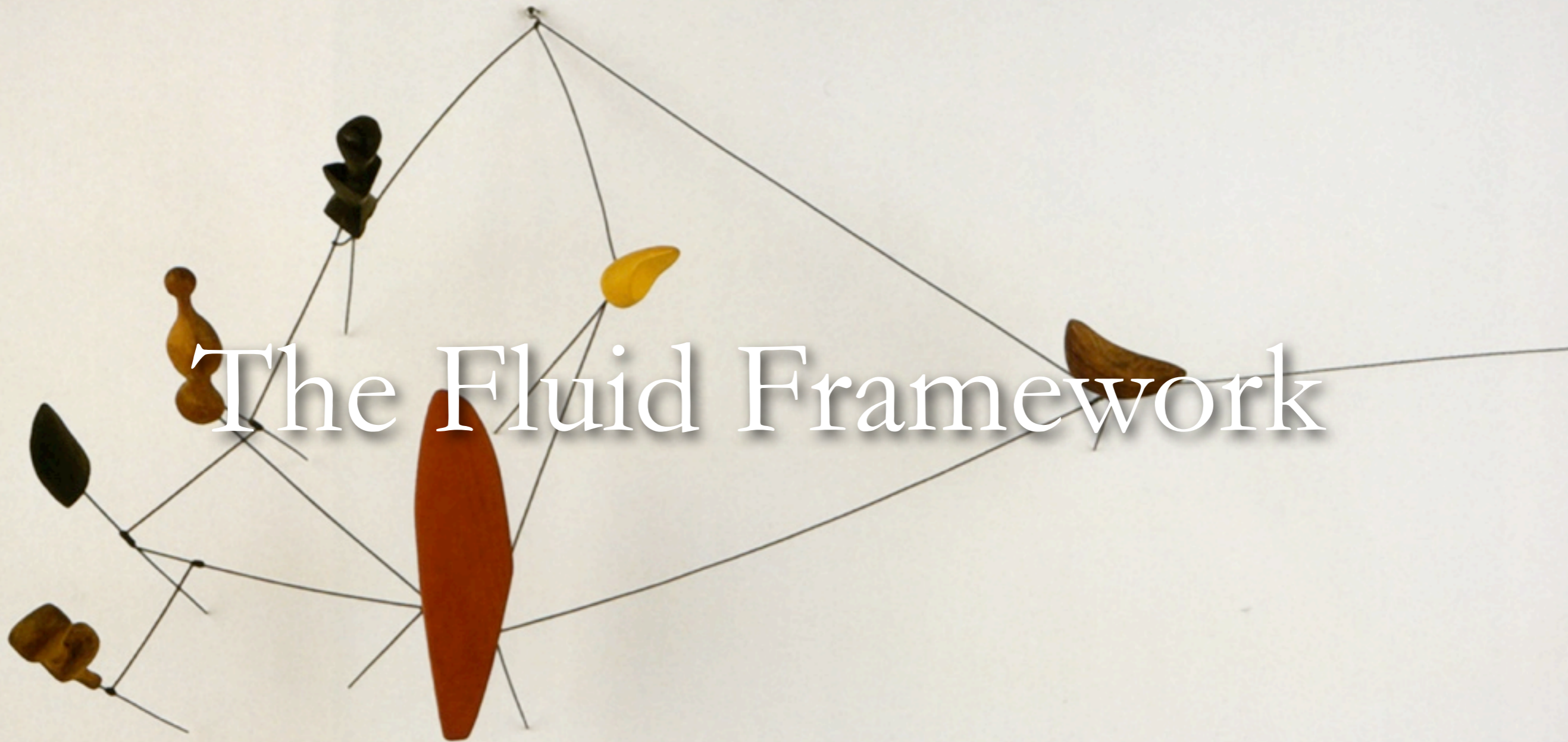


The Fluid Framework





Fluid technology goals

- Build an architecture to support user interfaces that can be shared and adapted.
- Develop tools that support the inclusive design process.
- Give users tools to personalize their environment.

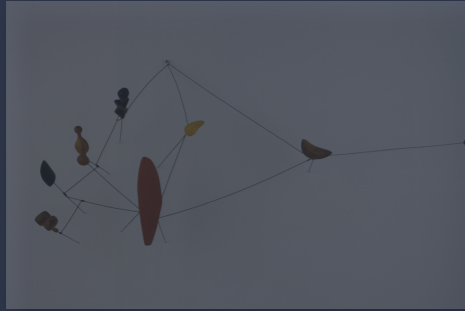


What is a framework?

A **software framework**, in computer programming, is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code providing specific functionality.

Frameworks are similar to software libraries in that they are reusable abstractions of code wrapped in a well-defined API. Unlike libraries, however, the overall program's flow of control is not dictated by the caller, but by the framework. This inversion of control is the distinguishing feature of software frameworks.

http://en.wikipedia.org/wiki/Software_framework

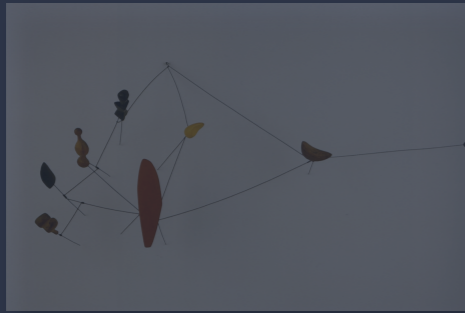


Huh?

Stuff that helps us write great user interfaces faster by not having to solve the same problem over and over again.

Imagining the framework

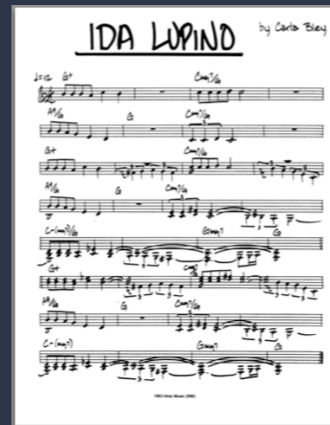




It's like music...



Component



Configuration



Integration



Framework



What is our framework?

- Code tools to help us:
 - build flexible designs
 - avoid errors
 - shape our code
 - write less code
 - build richly accessible code
- Built with open web technologies



The framework gives us...

- A **life cycle** for components
- A way to **configure** & wire up components
- Separation of **presentation from logic**
- A way to **change markup** and appearance

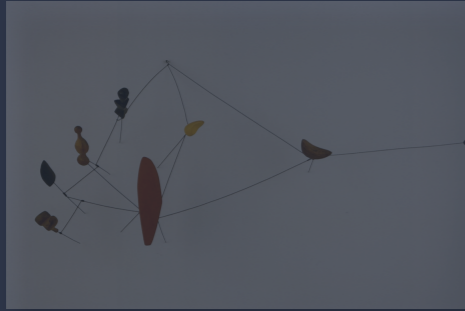


You can't bottle design

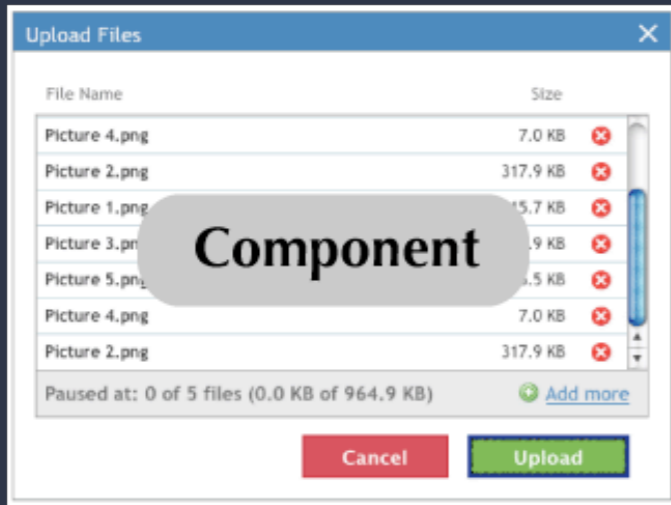
- Context is everything!
- Each new use case brings new design considerations
- We can't get away with shipping one specific design and assume we're done

Design for more design

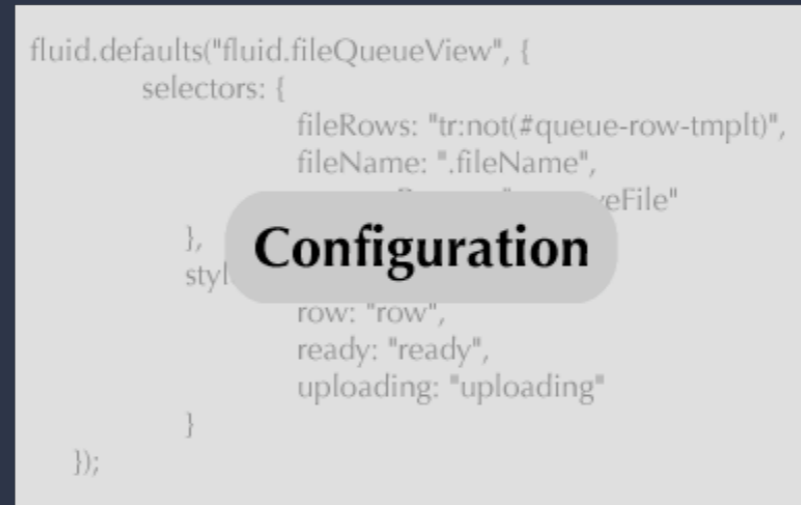
- Our designs should invite new designs
- What are the range of choices and needs for an interaction?
- How can we support people in making the right choices for their particular context?
- *The technology needs to help us...*



Components



+



=

Solution

Pattern



Context



Implementation

Component families

Inline Edit

Simple Text Inline Edit

Name	Grade
Lecture Sections	
Astronomy 7A P 411 LEC	Melis
Edit Details Assign GSIS Assign Students	Julie
Discussion Sections	
Astronomy 7A S 102 LAB	Sue

Rich Text Inline Edit

BioE 24 - Aspects of Engineering
Sibley Auditorium, **Wednesdays 4 - 5**

dolor sit amet, consectetur adipiscing elit. Suspendisse ut purus. Suspendisse
ut odio semper. Nulla. Donec. Nam. Donec. Donec. Donec. Donec. Donec. Donec. Donec.

Dropdown Inline Edit

Groups	Open
Choose one	07/02
Choose One	07/26
Section 1	
Section 2	
Section 3	

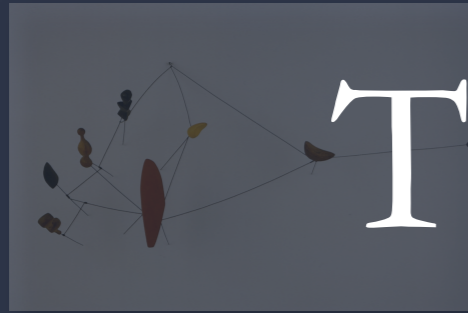


Value of the framework

- Positions us to write components faster
- Allows us to rework our designs for each new integration
- The framework is a design enabler
- Enables new developers to join our ranks and build their own solutions

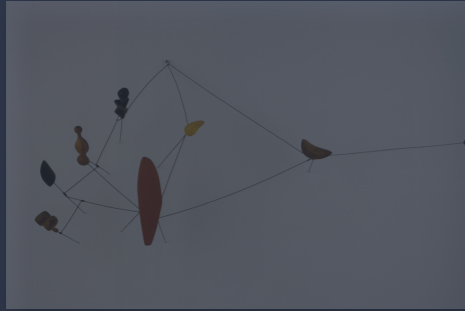
Building the framework

- We didn't build a framework because they're fun to write. They're not.
- We built a bunch of components, suffered, and then built a framework that addressed real challenges.



The wider context

- There are a lot of different JavaScript programming tools out there.
- Why did we build another one?



Measuring up

Foundational toolkits vs. application frameworks



Foundational toolkits

- Totally presentation focused
 - DOM manipulation
 - Event binding
 - Ajax
-
- eg. jQuery

Application frameworks

- **Model notifications** “something changed here”
- **Views** to help keep your presentational code clean
- **Data binding** to sync the display with your model

- eg. Sproutcore; Dojo + Dojox

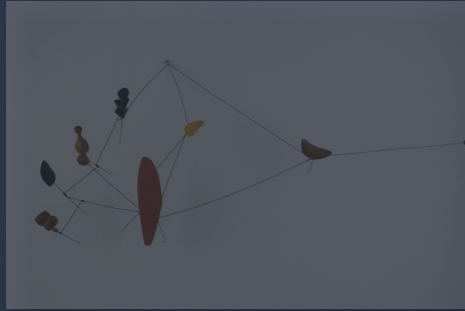
Where does Infusion fit?

- We recognize that we're not the only one in the browser: **we play nice with other toolkits.**
- We don't want to force adopters down a one-way technology street



Where does Fluid fit?

Infusion is an application framework designed to provide unprecedented flexibility while preserving interoperability.

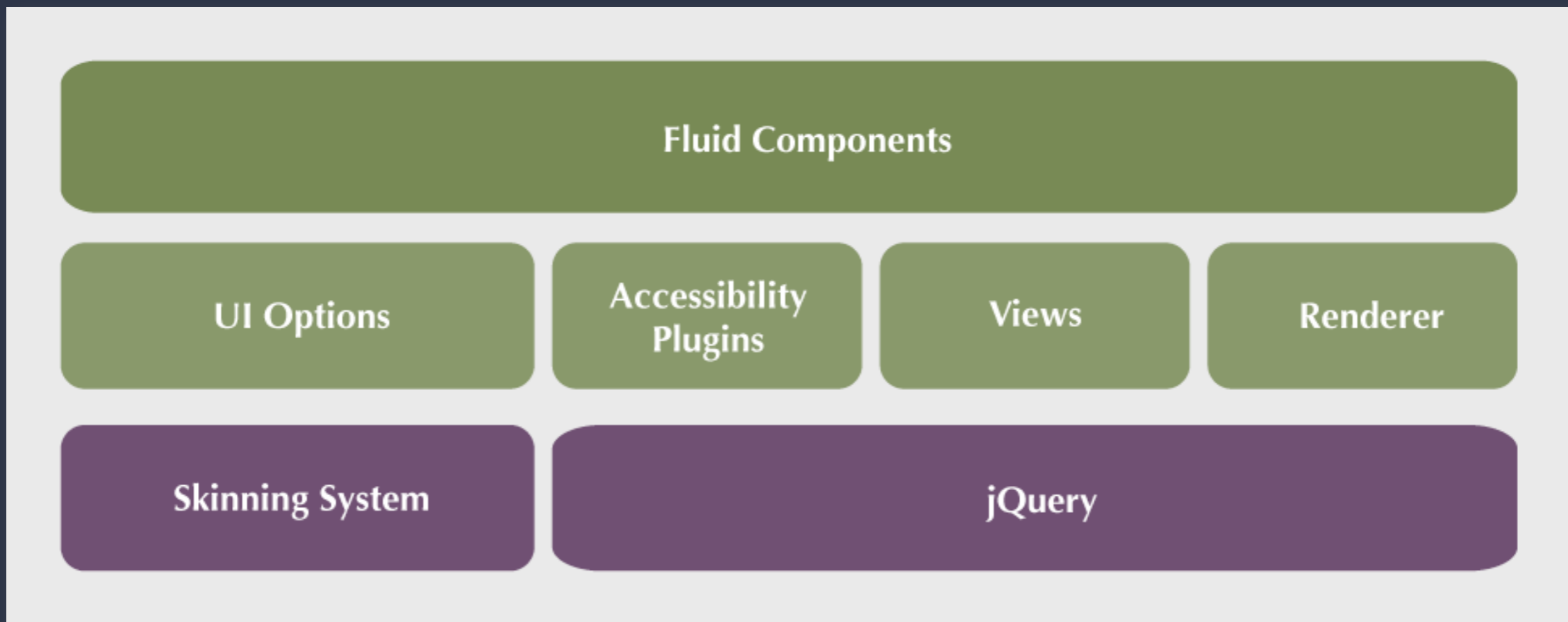


In summary

- Design for more design
- Offer a technology that isn't all-or-nothing
- Grow it based on real experience

getting geekier

Tasty framework sandwich



What's the framework?

- jQuery
- keyboard-a11y plugin
- that-ism
- Components & declarative options
- DOM Binder
- Views
- Events
- Subcomponents



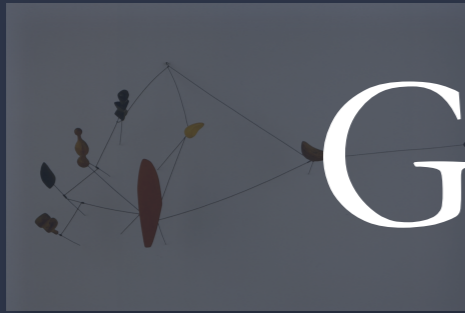
Component goals

- Thoroughly accessible
- Easy to share and reuse
- Can be personalized
- Plays nice with other technologies



Addressing real pain

- Behaviour and presentation logic tended to glob together as a single component
- Too easy to write clever logic that prevented any changes to the markup
- Handling configuration required lots of repetitive code



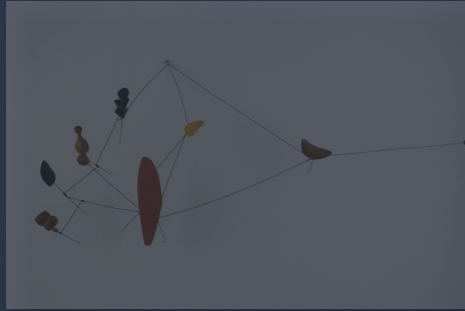
Goals and features

Change markup without breaking code	DOM Binder
Customize component	Declarative options
Inject custom behaviour into components	Events, subcomponents
Decouple presentation from model logic	Views
Easily testable	Events, views, subcomponents
Make accessibility easier	jquery.keyboard-a11y, ui.core
Stable and secure JavaScript objects	that-ism



Model View Controller

- Model is the application data and associated behaviour
- View presents the model and drives the interaction logic
- Controller is glue



Fluid MVC

- Controllers are the least interesting part of MVC
- Models are transparent
- Views can be easily swapped or altered



jquery.keyboard-a11y



Features

- `tabindex` normalization across browsers
- arrow key navigation
- activating elements
- migrating to `jquery ui-core.js`
- *makes keyboard accessibility super easy*



Keyboard Conventions

- **Tab** key focuses the control or widget
- **Arrow keys** select an item
- **Enter** or **Spacebar** activate an item

Tab is handled by the browser. For the rest, you need to write code.



TabIndex examples

Tabs.html

```
<!-- Tab container should be focusable -->
<ul id="animalTabs" tabindex="0">
  <!-- Individual Tabs shouldn't be focusable -->
  <!-- We'll focus them with JavaScript instead -->
  <li id="tab1" tabindex="-1">Cats</li>
  <li id="tab2" tabindex="-1">Dogs</li>
  <li id="tab3" tabindex="-1">Alligators</li>
</ul>
```



keyboard-a11y in code

Tabs.js

```
function keyNav(container, tabs, selectHandlers, activationHandlers) {  
  // Make the tablist accessible with the Tab key.  
  tabContainer.tabbable();  
  
  // Make each tab accessible with the left and right arrow keys.  
  tabs.selectable(tabContainer, selectionHandlers, {  
    direction: jQuery.a11y.orientation.HORIZONTAL  
  });  
  
  // Make each tab activatable with Spacebar and Enter.  
  tabs.activatable(activationHandlers);  
}
```



Further customization

jquery.keyboard-a11y.js

```
$.fn.selectable.defaults = {  
    direction: $.a11y.orientation.VERTICAL,  
    selectablesTabIndex: -1,  
    autoSelectFirstItem: true,  
    rememberSelectionState: true,  
    selectableSelector: ".selectable",  
    selectableElements: null,  
    onSelect: null,  
    onUnselect: null,  
    onLeaveContainer: null  
};
```

that-ism

Duchess
1964
Eugl.
Riviera



JavaScript pitfalls

- Lack of namespacing and privacy
- Confusing variability of `this`
- Security and stability issues: `prototype`
- No ability to link against multiple versions



Namespacing, privacy, and versioning

Fluid.js

```
var fluid_0_6 = fluid_0_6 || {};  
var fluid = fluid || fluid_0_6;  
  
(function ($, fluid) {  
  
    // Code goes here.  
  
})(jQuery, fluid_0_6);
```



that

- Define objects within a function
- Provides privacy and a bound context
- Types can't be maliciously altered
- Open for extension, not modification
- Douglas Crockford's pattern, not ours.



Putting it all together

UIOptions.js

```
fluid_0_6 = fluid_0_6 || {};  
(function ($, fluid) {  
    fluid.uiOptions = function (container, options) {  
        var that = fluid.initView("fluid.uiOptions", container, options);  
  
        that.save = function () {  
            that.events.onSave.fire(that.model);  
            fluid.applySkin(that.model);  
        };  
  
        that.refreshView = function () {  
            pushModelToView(that);  
        };  
  
        setupUIOptions(that);  
  
        return that;  
    };  
})(jQuery, fluid_0_6);
```




Components



What's a component?

- Central hub for:
 - Events
 - Configuration
 - Public API
- A composition of Views and model logic



Component contract

Uploader.js

```
/**
 * Instantiates a new Uploader component.
 *
 * @param {Object} container the DOM element containing the Uploader markup
 * @param {Object} options configuration options for the component.
 */
fluid.uploader = function (container, options) { ... }
```

WARMER

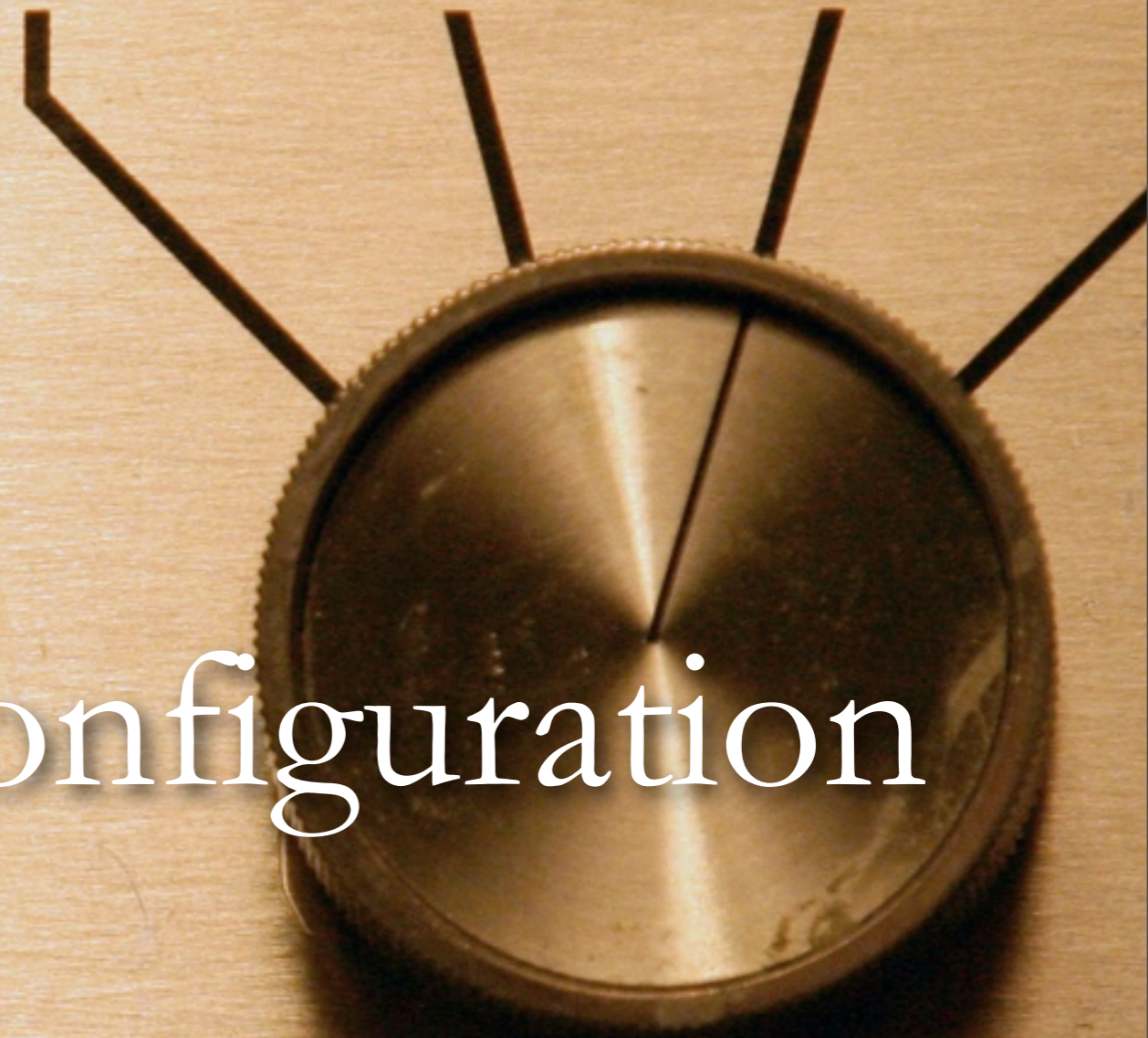
OFF

HI

MED



COOLER



Declarative Configuration



Tweaking components

- Transparent configuration
- Declarative: **ask, don't instruct**
- Mini IoC



What can be configured?

- Modes and optional features
- Selectors
- Styles
- Subcomponents
- Events
- Language bundles

reorderer.js

```
fluid.defaults("fluid.reorderer", {
  instructionMessageId: "message-bundle:",
  styles: {
    defaultStyle: "orderable-default",
    selected: "orderable-selected",
    dragging: "orderable-dragging",
    mouseDrag: "orderable-dragging",
    hover: "orderable-hover",
    dropMarker: "orderable-drop-marker",
    avatar: "orderable-avatar"
  },
  selectors: {
    dropWarning: ".drop-warning",
    movables: ".movables",
    grabHandle: "",
    stylisticOffset: ""
  },
  avatarCreator: defaultAvatarCreator,
  keysets: fluid.reorderer.defaultKeysets,
  layoutHandler: "fluid.listLayoutHandler",
  events: {
    onShowKeyboardDropWarning: null,
    onSelect: null,
    onBeginMove: "preventable",
    onMove: null,
    afterMove: null,
    onHover: null
  },
  mergePolicy: {
    keysets: "replace",
    "selectors.selectables": "selectors.movables",
    "selectors.dropTargets": "selectors.movables"
  }
});
```

nam sed utaris tui.
si confirmame
uas tuas et impu
nuer tenentur ::
guinibus dñs salutis
tabit lingua mecc
niam ::
aperies. & os meum
abit laudem tuam ::
isses sacrificium dedis
olocaustis non delectaberis.
sp̄s contribulatus. cor
amiliatum dñs non dispiciet ::
in bona uoluntate tua
entur muri hierusalem ::
is sacrificium iustitiae.
tholocausta. tunc im
altare uisum unum

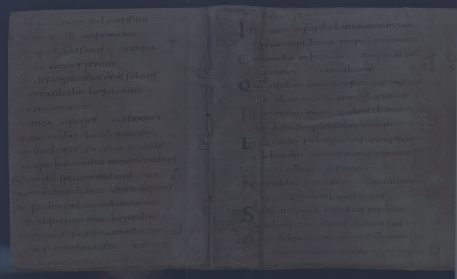
DOM Binder

mei pradicium ::
In parabola aurem meam
aperiam in psalterio propositionem meam ::
Curamebo indicem. & iniquitas cal
pate mei. & uim labium me ::
Qui confidunt in uirtute sua. cum multo
dicuntur in seipsum gloriantur ::
Ferter non redemur. redimet homo.
ardabit de placationem suam ::
Etiam in impuonem anime suae.
laborabit in aeternum. et uict
adhuc. in finem ::
Non uidebit in uirtute. cum uiderit
sapientes merentes ::
Sil insipiens et stultus peribant.
relinquent aliena diuitias suas ::
Pulchritudo in deum illorum in
cipem. tabernaculum in impo



Decoupling code from markup

- The most common component pitfall is hard-baking assumptions about markup.
- Use named selectors to separate the component implementation from the markup.
- Let users specify alternative selectors.



We'll take anything

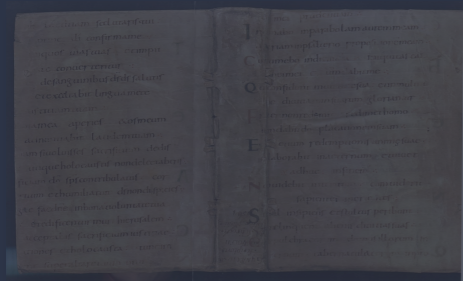
- The DOM Binder supports:
 - jQuery selectors
 - Elements
 - Arrays of elements
 - jQuery objects
 - Functions



Declaring interesting things

Uploader.js

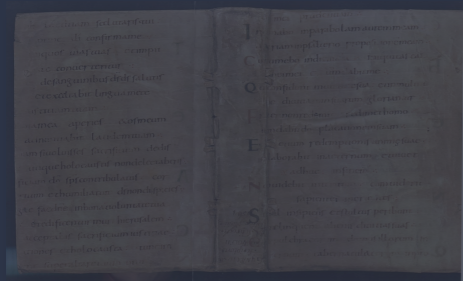
```
selectors: {  
  fileQueue: ".fluid-uploader-queue",  
  browseButton: ".fluid-uploader-browse",  
  uploadButton: ".fluid-uploader-upload",  
  resumeButton: ".fluid-uploader-resume",  
  pauseButton: ".fluid-uploader-pause",  
  totalFileProgressBar: ".fluid-scroller-table-foot",  
  stateDisplay: "div:first"  
}
```



locate()

FileQueueView.js

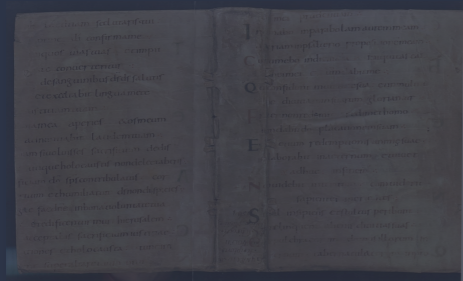
```
that.events.onFileSuccess.addListener(function (file) {  
  var row = rowForFile(that, file);  
  that.locate("removeButton", row).unbind("click");  
  that.locate("removeButton", row).tabindex(-1);  
  changeRowState(row, that.options.styles.uploaded);  
});
```



fastLocate()

Reorderer.js

```
function firstSelectable(that) {  
    var selectables = that.dom.fastLocate("selectables");  
    if (selectables.length <= 0) {  
        return null;  
    }  
    return selectables[0];  
}
```



refresh()

Reorderer.js

```
thatReorderer.refresh = function () {  
    thatReorderer.dom.refresh("movables");  
    thatReorderer.dom.refresh("selectables");  
    thatReorderer.dom.refresh("grabHandle",  
        thatReorderer.dom.fastLocate("movables"));  
    thatReorderer.dom.refresh("stylisticOffset",  
        thatReorderer.dom.fastLocate("movables"));  
    thatReorderer.dom.refresh("dropTargets");  
    initItems();  
    thatReorderer.selectableContext.selectables = thatReorderer.dom.fastLocate("selectables");  
    thatReorderer.selectableContext.selectablesUpdated(thatReorderer.activeItem);  
};
```

A close-up photograph of a white-framed window with a view of a green landscape. The window has a silver handle on the right side. The word "Views" is written in white, serif font in the center of the window. The background is a blurred green landscape with a hint of a sunset or sunrise in the upper left corner.

Views



Managing the presentation

- Views are DOM-oriented objects
- They encapsulate the presentational behaviour of a component
- They show a view on model-sourced data
- They often represent only a portion of the overall component's screen real estate



View Contract

- Views:
 - Are automatically DOM-bound
 - Have a container
 - May share with their parent component
 - May have options
 - May use events
 - Should implement `refreshView()`



Becoming a View

FileQueueView.js

```
fluid.fileQueueView = function (container, events, parentContainer,
                                uploadManager, options) {
    var that = fluid.initView("fluid.fileQueueView", container, options);
```



refreshView()

FileQueueView.js

```
that.refreshView = function () {  
    that.scroller.refreshView();  
    that.container.getSelectableContext().refresh();  
};
```

A top-down view of a wooden board with a grid of holes. Numerous metal rods of varying diameters and finishes (some polished, some matte) are inserted into the holes. The rods are arranged in a somewhat regular pattern, though some are missing or partially inserted. The word "Events" is written in a white, serif font in the center of the board.

Events



About the events system

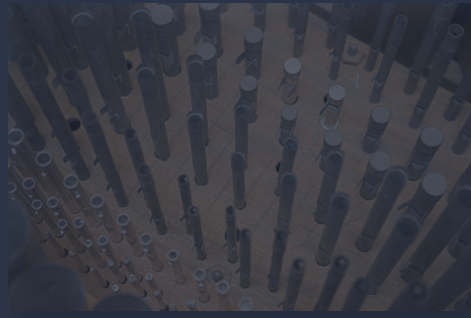
- Pure model-based events
- Designed for sending messages between JavaScript objects
- Not encumbered by the DOM or presentational concerns
- Analogous to jQuery events, but intentionally a bit different



Declaring Events

Reorderer.js

```
events: {  
    onShowKeyboardDropWarning: null,  
    onSelect: null,  
    onBeginMove: "preventable",  
    onMove: null,  
    afterMove: null,  
    onHover: null  
}
```



Types of events

`null` “hey everyone, something is happening”

`preventable` “should I do this?”

`unicast` “our little secret”



Listening for events

section-info-inner.html

```
listeners: {  
  afterFinishEdit: function (newValue, oldValue) {  
    // Save the data to the server.  
  },  
  modelChanged: function (newValue, oldValue, that) {  
    // Update state.  
  }  
}
```




Using events in code

FileQueueView.js

Firing events:

```
var finishUploading = function (that) {  
    that.events.afterUploadComplete.fire(that.queue.currentBatch.files);  
    that.queue.clearCurrentBatch();  
};
```

Listening for events programmatically:

```
that.events.afterFileQueued.addListener(function (file) {  
    that.queue.addFile(file);  
});
```

Subcomponents

- Provides very loose coupling between parts
- Look up dependencies by name, and the framework will instantiate them for you
- Share portions of overall configuration
- Users can implement their own version or configure alternatives

Instantiating subcomponents

Uploader.js

```
var setupUploader = function (that) {  
  // Instantiate the upload manager and file queue view,  
  // passing them smaller chunks of the overall options for the uploader.  
  that.uploadManager = fluid.initSubcomponent(that,  
                                              "uploadManager",  
                                              [that.events, fluid.COMPONENT_OPTIONS]);  
  
  that.fileQueueView = fluid.initSubcomponent(that,  
                                              "fileQueueView",  
                                              [that.locate("fileQueue"),  
                                              that.events,  
                                              that.container,  
                                              that.uploadManager,  
                                              fluid.COMPONENT_OPTIONS]);  
}
```

Configuring a subcomponent

Uploader2.html

```
var myUploader = fluid.uploader(".fluid-uploader", {  
    uploadManager: "fluid.demoUploadManager"  
});
```

Overriding subcomponent options

```
var myUploader = fluid.uploader("#simple_uploader", {  
  fileQueueView: {  
    type: "fluid.fileQueueView",  
    options: {  
      selectors: {  
        fileRows: ".row",  
        fileName: ".fileName",  
        fileSize: ".fileSize",  
        removeButton: ".removeFile"  
      }  
    }  
  }  
});
```

Where are we going?

UI Options

User Interface Options

Personalize the display and interaction of site content in [3 easy steps](#).

1 [Select a template](#) > 2 **Set Preferences** > 3 [Apply Preferences](#)

Customize the template by setting preferences to suit your needs. Changes will be reflected in the Preview Window.

Color

- Color Palette 1 - [link_color](#)
- Color Palette 2 - [link_color](#)
- Color Palette 3 - [link_color](#)
- Color Palette 4 - [link_color](#)

Layout

Layout	Graphics	Table of Contents
<input checked="" type="radio"/> Default <input type="radio"/> Simple	<input checked="" type="radio"/> Default <input type="radio"/> Simple	<input checked="" type="radio"/> No <input type="radio"/> Yes

Text

Font	Size	Spacing
<input checked="" type="radio"/> Default <input type="radio"/> Arial <input type="radio"/> Verdana <input type="radio"/> Courier <input type="radio"/> Times	<input type="radio"/> -2 <input type="radio"/> -1 <input checked="" type="radio"/> Default <input type="radio"/> +1 <input type="radio"/> +2 <input type="radio"/> +3 <input type="radio"/> +4 <input type="radio"/> +5 <input type="radio"/> +6	<input checked="" type="radio"/> Default <input type="radio"/> Wide <input type="radio"/> Wider <input type="radio"/> Widest


Links

Highlight links (on hover) No Yes

Preview Window

Standard

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur interdum, mi sed ultricies vehicula, nibh tortor hendrerit tellus, ut feugiat pede nunc in eros. Nullam convallis arcu quis ante.



Sed nisi. Fusce at nulla ut nulla euismod posuere. Curabitur id nulla id neque faucibus.

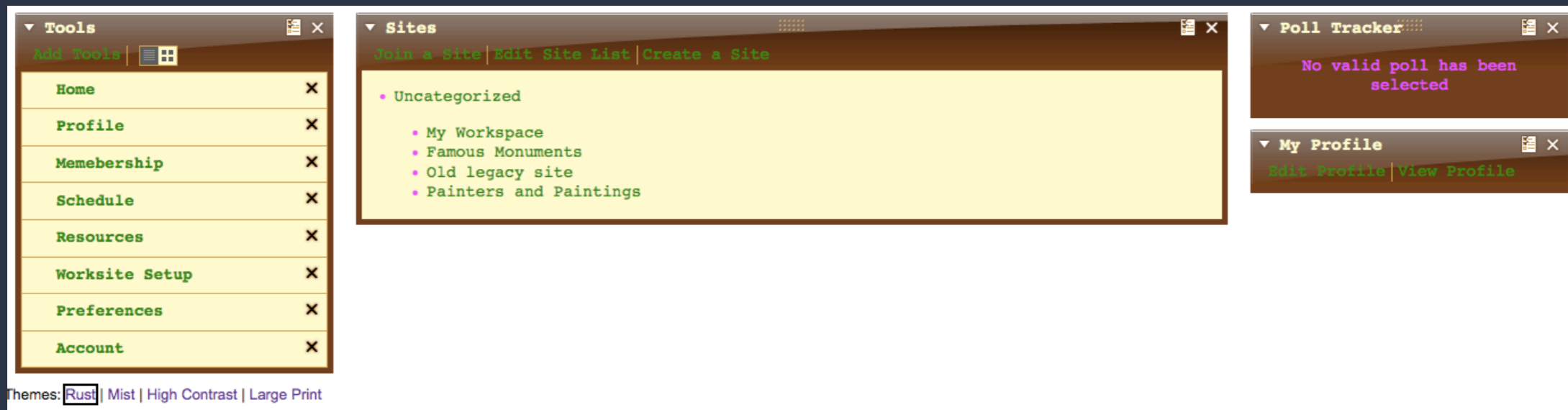
- [Quisque consectetur](#)
- [Enim ac magna](#)
- [Sed quis ipsum eu lorem cursus scelerisque](#)
- [Nam a mi id eros pharetra suscipit](#)

Fusce at nulla ut nulla

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur interdum, mi sed ultricies vehicula, nibh tortor hendrerit tellus, ut feugiat pede nunc in eros. Nullam convallis arcu quis ante.

[← Select a different template](#)

Skinning system



Renderer