hgAi and a bunch of new code

Angie Hinrichs

March 11, 2015

Outline

- Why do we need an Annotation Integrator?
- hgAi demo
- A brief history of javascript in kent/src
- Enlightening info from Brian Craft
- New code in kent/src
- New dev dependencies and types of bugs

Why do we need hgAi?

- We send too many users to Galaxy for a commonly requested operation

  – “I am having trouble pulling out gene names for the corresponding probes. ... I've tried many of the settings, changing the group from custom to all tables, but still **I don't get any gene names, just an output of position**.
  ... Am I doing something wrong?”

Why do we need hgAi?

– "When I make a custom tract ... I am interested in getting the intersection of that with the TFBS track, **is there some easy way to label the output with the gene?** I get a bed file that does not directly indicate which gene it is associated with. Am I missing something?"

Why do we need hgAi?

- "There must be an easier way to **fill in the human gene symbol** on each line of a bed file than to manually look it up on the browser."
- "However, when I get the output it **doesn't match the gene name to the input list**, I only get the gene name."
- "This gives me the **regions that have genes, but not which ones** in the Table output."
- …

ulterior motives

- More modern interactive interface
- More elegant UI code

Demo time!

A brief history of JS in kent/src

- Before JS: all <FORM>s and submit buttons

- Minimal inline JS in HTML:

```
cgiMakeCheckBoxJS(buf, shared, "onchange=\"document.mainForm.submit();\"");
```

- More inline JS in HTML:

```
char *onChangeClade = "onchange=\"document.orgForm.clade.value =
document.mainForm.clade.options[document.mainForm.clade.selectedIndex].value;
document.orgForm.org.value = 0; document.orgForm.db.value = 0;
document.orgForm.submit();\"";
```

A brief history of JS in kent/src

- More and more inline JS in HTML:

```
// __detectback trick from http://siphon9.net/loune/2009/07/detecting-the-back-or-refresh-button-click/
printf("<script>\n"
       "document.write(\"<form style='display: none'><input name='__detectback' id='__detectback' "
       "value=''></form>\");\n"
       "function checkPageBackOrRefresh() {\n"
       "  if (document.getElementById('__detectback').value) {\n"
       "    return true;\n"
       "  } else {\n"
       "    document.getElementById('__detectback').value = 'been here';\n"
       "    return false;\n"
       "  }\n"
       "}\n"
       "window.onload = function() { "
       "  if (checkPageBackOrRefresh()) { window.location.replace('%s?%s'); } };\n"
       "</script>\n", getScriptName(), cartSidUrlString(cart));
```

A brief history of JS in kent/src

- More and more inline JS in HTML:

```
// __detectback trick from http://siphon9.net/loune/2009/07/detecting-the-back-or-refresh-button-click/
printf("<script>\n"
       "document.write(\"<form style='display: none'><input name='__detectback' id='__detectback' "
       "value=''></form>\");\n"
       "function checkPageBackOrRefresh() {\n"
       "  if (document.getElementById('__detectback').value) {\n"
       "    return true;\n"
       "  } else {\n"
       "    document.getElementById('__detectback').value = 'been here';\n"
       "    return false;\n"
       "  }\n"
       "}\n"
       "window.onload = function() { "
       "  if (checkPageBackOrRefresh()) { window.location.replace('%s?%s'); } };\n"
       "</script>\n", getScriptName(), cartSidUrlString(cart));
```

A brief history of JS in kent/src

- More and more inline JS in HTML:

```
// __detectback trick from http://siphon9.net/loune/2009/07/detecting-the-back-or-refresh-button-click/
printf("<script>\n"
       "document.write(\"<form style='display: none'><input name='__detectback' id='__detectback' "
       "value=''></form>\");\n"
       "function checkPageBackOrRefresh() {\n"
       "  if (document.getElementById('__detectback').value) {\n"
       "    return true;\n"
       "  } else {\n"
       "    document.getElementById('__detectback').value = 'been here';\n"
       "    return false;\n"
       "  }\n"
       "}\n"
       "window.onload = function() { "
       "  if (checkPageBackOrRefresh()) { window.location.replace('%s?%s'); } };\n"
       "</script>\n", getScriptName(), cartSidUrlString(cart));
```

A brief history of JS in kent/src

- Javascript in .js files!

```
setByCoordinates: function (chrom, start, end)
    {
        var newPosition = chrom + ":" + commify(start) + "-" + commify(end);
        genomePos.set(newPosition, commify(end - start + 1));
        return newPosition;
    },
```

A brief history of JS in kent/src

- JQuery
  - Finding DOM elements is easy!
  - Changing DOM elements is easy!
  - AJAX is easy!
  - Helps w/browser incompatibilities!
  - JQueryUI plugins!
    - Autocomplete
    - Drag & drop reordering
    - Tabs
    - Context menu (right-click)

A brief history of JS in kent/src

- Using DOM elements as storage for app state

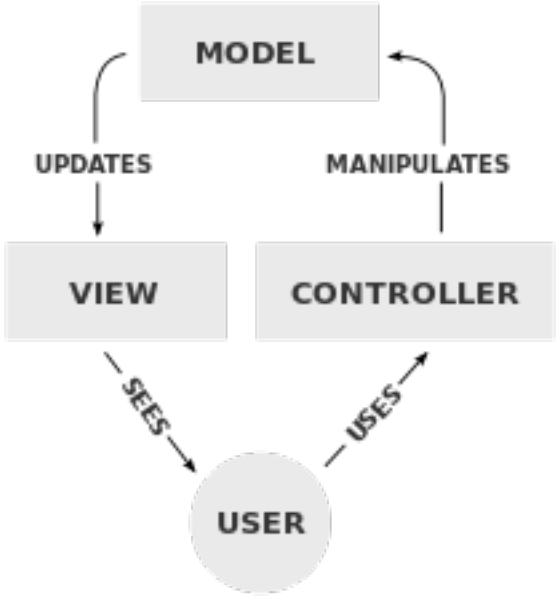
```
// Only look at visible views
subCBs = $(subCBs).not('.disabled').not(":disabled");

if (subCBs.length > 0) {
    var CBsChecked = subCBs.filter(":checked");
    if (!isABC) {
        if (CBsChecked.length === subCBs.length) {
            matCbComplete(matCB,true);
            $(matCB).attr('checked',true);
        } else if (CBsChecked.length === 0) {
            matCbComplete(matCB,true);
            $(matCB).attr('checked',false);
        } else {
            matCbComplete(matCB,false);
            $(matCB).attr('checked',true);
        }
    }
```

…

There are other ways…

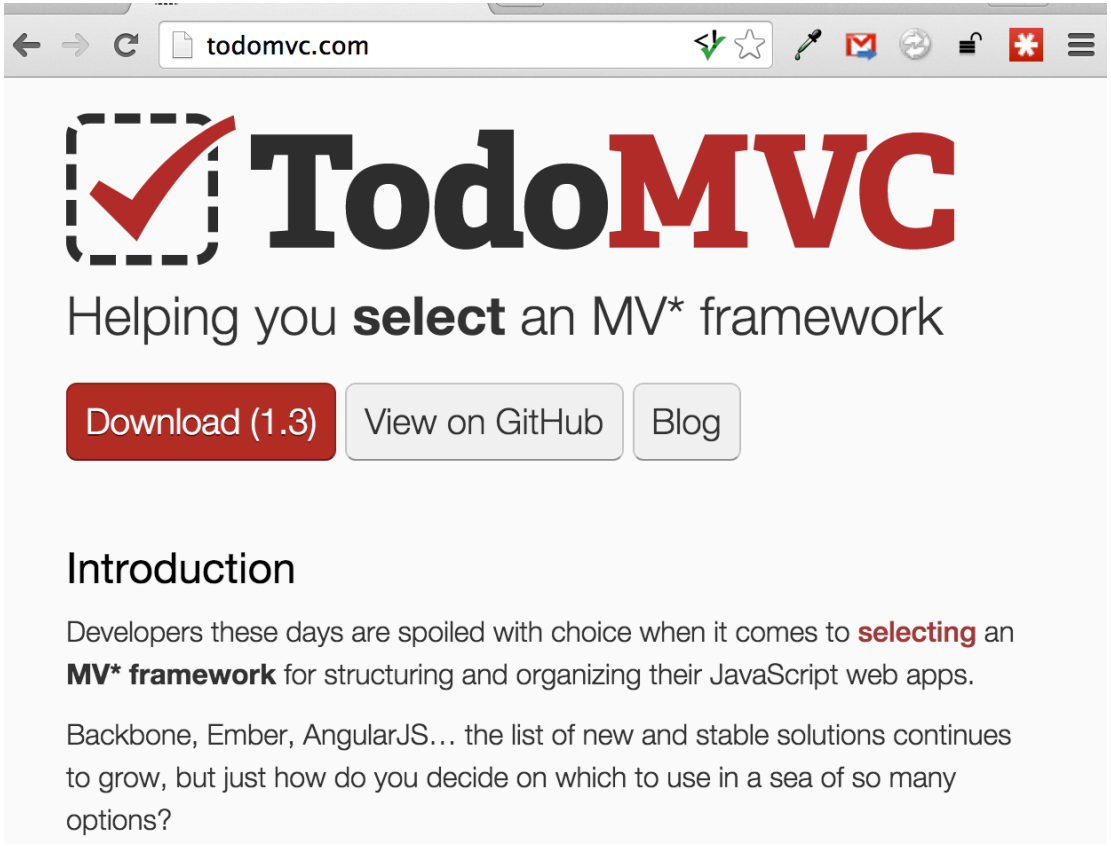
- MVC: Model-View-Controller (1970s)
  - Model: application state & logic
  - View: display rendering
  - Controller: dispatch input from user



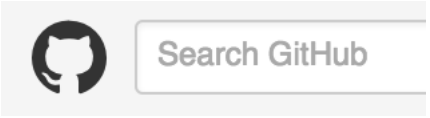
http://en.wikipedia.org/wiki/Model-view-controller

There are other ways…

- Modern web development: MV* frameworks



There are other ways…



https://www.google.com/#q=javascript+the+good+parts
https://developer.mozilla.org/en-US/

https://github.com/
https://nodejs.org/
https://www.npmjs.com/

Enlightenment from BC

- Functional programming concepts
- Declarative UI rendering (ReactJS)
- Sequential immutable states (ImmutableJS)

Functional programming

- Imperative programming with mutable state: fertile ground for bugs
- Pure functions behave predictably
- JS's first-class functions support FP

http://en.wikipedia.org/wiki/Functional_programming
http://en.wikipedia.org/wiki/Comparison_of_programming_paradigms

FP for dynamic web app?

- View:

  web page = F($state_n$)

- Model/Controller:

  $state_{n+1}$ = G($state_n$, event)

View: Imperative C…

```
if ((row = sqlNextRow(sr)) != NULL)
    {
    unsigned scafCount = sqlUnsigned(row[0]);
    unsigned totalSize = sqlUnsigned(row[1]);
    cgiTableRowEnd();
    safef(msg1, sizeof(msg1), "contig/scaffold<BR>count:");
    safef(msg2, sizeof(msg2), "total size:");
    cgiSimpleTableRowStart();
    cgiSimpleTableFieldStart();
    puts(msg1);
    cgiTableFieldEnd();
    cgiSimpleTableFieldStart();
    puts(msg2);
    cgiTableFieldEnd();
    cgiTableRowEnd();
    cgiSimpleTableRowStart();
    cgiSimpleTableFieldStart();
    printLongWithCommas(stdout, scafCount);
    cgiTableFieldEnd();
    cgiSimpleTableFieldStart();
    printLongWithCommas(stdout, totalSize);
    cgiTableFieldEnd();
    }
   cgiTableRowEnd();
```

… vs. declarative HTML

```
if ((row = sqlNextRow(sr)) != NULL)
    {
    unsigned scafCount = sqlUnsigned(row[0]);
    unsigned totalSize = sqlUnsigned(row[1]);
    cgiTableRowEnd();
    safef(msg1, sizeof(msg1), "contig/scaffold<BR>count:");
    safef(msg2, sizeof(msg2), "total size:");
    cgiSimpleTableRowStart();
    cgiSimpleTableFieldStart();
    puts(msg1);
    cgiTableFieldEnd();
    cgiSimpleTableFieldStart();
    puts(msg2);
    cgiTableFieldEnd();
    cgiTableRowEnd();
    cgiSimpleTableRowStart();
    cgiSimpleTableFieldStart();
    printLongWithCommas(stdout, scafCount);
    cgiTableFieldEnd();
    cgiSimpleTableFieldStart();
    printLongWithCommas(stdout, totalSize);
    cgiTableFieldEnd();
    }
   cgiTableRowEnd();
```

```
<tr>
  <td>contig/scaffold<br />
      count:</td>
  <td>total size:</td>
</tr>
<tr>
  <td>13,563</td>
  <td>2,453,304,946</td>
</tr>
```

… vs. declarative HTML

```
if ((row = sqlNextRow(sr)) != NULL)
    {
    unsigned scafCount = sqlUnsigned(row[0]);
    unsigned totalSize = sqlUnsigned(row[1]);
    cgiTableRowEnd();
    safef(msg1, sizeof(msg1), "contig/scaffold<BR>count:");
    safef(msg2, sizeof(msg2), "total size:");
    cgiSimpleTableRowStart();
    cgiSimpleTableFieldStart();
    puts(msg1);
    cgiTableFieldEnd();
    cgiSimpleTableFieldStart();
    puts(msg2);
    cgiTableFieldEnd();
    cgiTableRowEnd();
    cgiSimpleTableRowStart();
    cgiSimpleTableFieldStart();
    printLongWithCommas(stdout, scafCount);
    cgiTableFieldEnd();
    cgiSimpleTableFieldStart();
    printLongWithCommas(stdout, totalSize);
    cgiTableFieldEnd();
    }
   cgiTableRowEnd();
```

contig/scaffold count:	total size:
13,563	2,453,304,946

```
<tr>
  <td>contig/scaffold<br />
      count:</td>
  <td>total size:</td>
</tr>
<tr>
  <td>13,563</td>
  <td>2,453,304,946</td>
</tr>
```

ReactJS: declarative UI done efficiently

- View: web page = F($state_n$)
- Rebuild entire page with JQuery DOM actions? Too slow!
- ReactJS:
  - Rebuild a data structure of virtual DOM elements
  - React detects diffs, makes minimal DOM changes

http://facebook.github.io/react/
https://www.youtube.com/watch?v=DgVS-zXgMTk

ReactJS: declarative UI done efficiently

- JSX: Javascript in which you can embed this:

```
<tr>
  <td>contig/scaffold<br />
      count:</td>
  <td>total size:</td>
</tr>
<tr>
  <td>{summary.scafCount}</td>
  <td>{summary.totalSize}</td>
</tr>
```

ReactJS: declarative UI done efficiently

- JSX: Javascript in which you can embed this:

```
<tr>
  <td>contig/scaffold<br />
      count:</td>
  <td>total size:</td>
</tr>
<tr>
  <td>{summary.scafCount}</td>
  <td>{summary.totalSize}</td>
</tr>
```

- And make your own components:

```
<TableRow valueList={[ scafCountLabel,    totalSizeLabel ]} />
<TableRow valueList={[ summary.scafCount, summary.totalSize ]} />
```

http://jsfiddle.net/sorfx2tp/4/

ReactJS: declarative UI done efficiently

- Make your own component

```
var TableRow = React.createClass({
    // Transform a list of values into <tr> with <td> for each value

    render: function() {
        var cells = this.props.valueList.map(function(value) {
            return <td>{value}</td>;
        });
        return (
          <tr> {cells} </tr>
        );
    }
});
```

http://jsfiddle.net/sorfx2tp/4/

C writes JSON instead of HTML

```
if ((row = sqlNextRow(sr)) != NULL)
    {
    unsigned scafCount = sqlUnsigned(row[0]);
    unsigned totalSize = sqlUnsigned(row[1]);
    jsonWriteObjectStart(jw, NULL);
    jsonWriteListStart(jw, "summaryRow");
    jsonWriteString(jw, “scafCount”, printLongWithCommas(stdout, scafCount));
    jsonWriteString(jw, “totalSize”, printLongWithCommas(stdout, totalSize));
    jsonWriteListEnd(jw);
    jsonWriteObjectEnd();
    }
```

(still pretty verbose compared to JS)

```
{ summary: { scafCount: commify(scafCount),
             totalSize: commify(totalSize)
           }
}
```

What kind of Model for React View?

- Model/Controller:

  $state_{n+1} = G(state_n, event)$

- Make a completely new copy of state every time? Inefficient!

- ImmutableJS: efficient implementation of immutable data structures

ReactJS + immutable state

- Functional programming style
- Efficiency
- Easy undo/redo!

https://www.youtube.com/watch?v=DMtwq3QtddY

New javascript code

- New subdirectories of kent/src/hg/js/
  - external/: open source libs
  - react/
    - lib/
    - plugins/
    - hgAi/
    - bundle/
  - model/
    - lib/
    - hgAi/

New lib for CGI as JSON backend

- hg/lib/cartJson.c

```
struct cartJson *cj = cartJsonNew(cart);
cartJsonRegisterHandler(cj, "getFields", getFields);
cartJsonExecute(cj);
```

- JS sends a JSON command to CGI like this:

```
{ getFields: { tables: “knownGene,snp142” } }
```

- CGI returns JSON response:

```
{ tableFields: { knownGene: [ ‘name’, ‘chrom’, ‘strand’, … ], snp142: …
```

New dev dependencies

- node
- jsx
- jsxhint
- uglifyjs

New types of bugs

- Responses arriving late / out of sync
- Developer forgetting to commit compiled JSX
  - git commit hook?

Thanks!

- Ann, b0b, Jim
- Brian Craft
- Kate Rosenbloom
- Jonathan Casper, Matt Speir
- Genome Browser Group