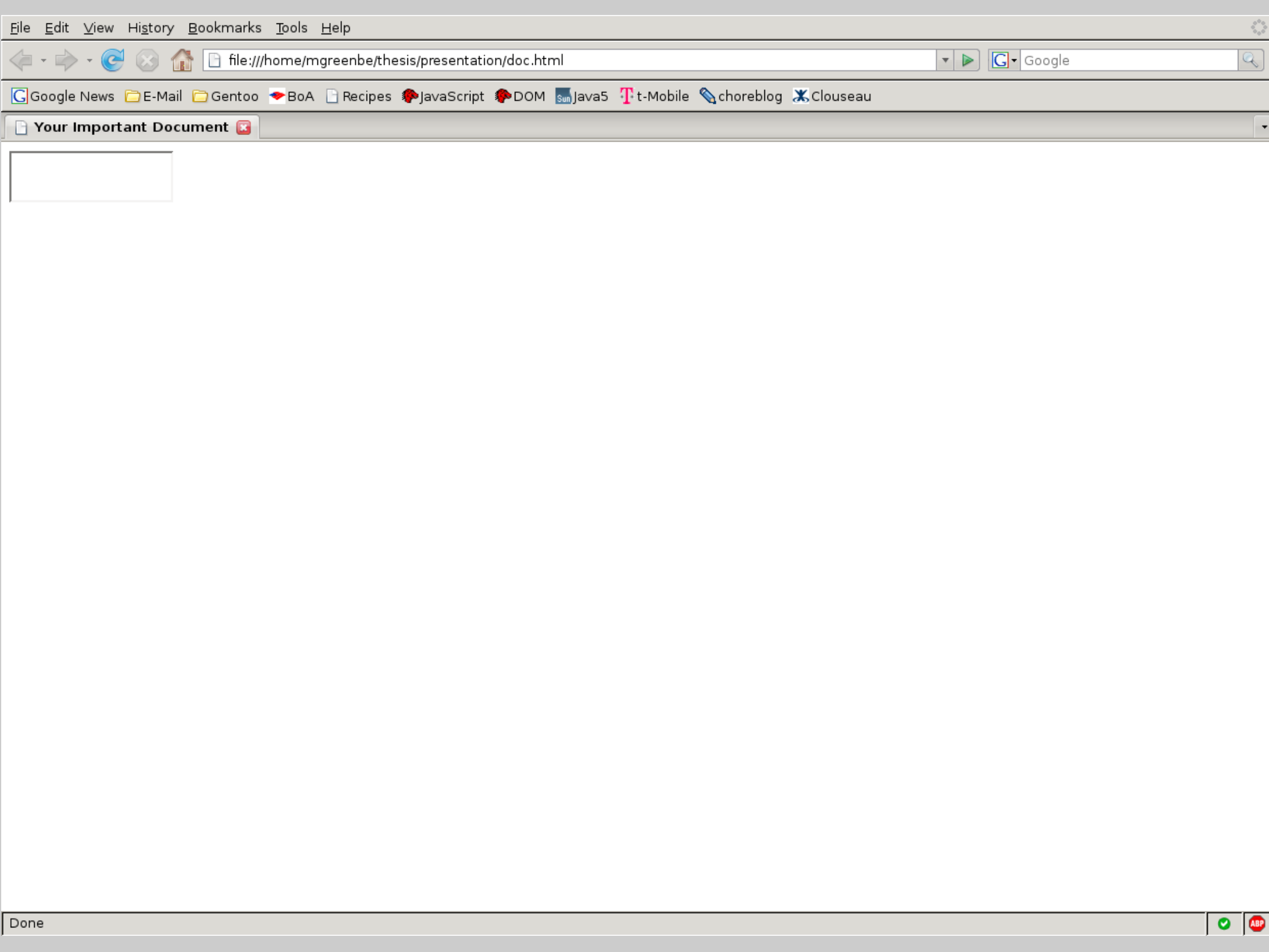
# Haplography and hendiadys in the Aramaic of Daniel

Michael Greenberg

# Declarative, composable views

Michael Greenberg


File Edit View History Bookmarks Tools Help
file:///home/mgreenbe/thesis/presentation/doc.html
Google
Google News E-Mail Gentoo BoA Recipes JavaScript DOM Java5 t-Mobile choreblog Clouseau
Your Important Document
Done

# HTML

```
<html>
<head>
  <title>Your Important Document</title>
  <script>
    // INSERT MAGIC, DISK 1 OF 6
  </script>
</head>
<body>
  <textarea id="doc" />
</body>
</html>
```

# JavaScript

```
function make_textarea(id, v, dom_update_callback) {
    var node = make_dom_node('textarea', { 'id': id }, [v]);
    add_event_handler(node, 'change', dom_update_callback);
    add_event_handler(node, 'keypress', dom_update_callback);
    return node;
}

function extract_textarea(id) {
    var node = get_dom_object(id);
    return node.value;
}
```

**...plus about 10-20 lines of DOM manipulation.**

**...plus about 20-30 lines of AJAX calls.**

# Flapjax

```
var d = valFromServer('path.to.doc');
insertDomB(TEXTAREAB({'id':'doc'}, [v]),
           'doc');
valToServer(d, 'path.to.doc');
```

# The Data Model

**Our data model, d, never gets updated!**

**The fix:**

```
var d = valFromServer('path.to.doc');
insertDomB(TEXTAREAB({'id':'doc'}, [v]),
           'doc');
valToServer(extractValue_b('doc'),
            'path.to.doc');
```

# Lenses

- Pair of functions

$$l.get :: JS \rightarrow DOM$$

$$l.putback :: DOM * JS \rightarrow JS$$

- Laws

*GETPUT*

$$l.putback(l.get(j), j) = j$$

*PUTGET*

$$l.get(l.putback(d, j)) = d$$

# Examples

focus *n*

(focus *n*).get o = o[n]

(focus *n*).putback v o = o with n = v

plunge *n*

(plunge *n*).get v = { n: v }

(plunge *n*).putback o v = o[n]

l ; k

(l ; k).get j = k.get(l.get(j))

(l ; k).putback d j = l.putback(k.putback(d, l.get(j)), j)

# DOM Lenses

span_tag *attribs kids*

(span_tag *attribs kids*).get v =

    <span *attribs*>*kids* v

(span_tag *attribs kids*).putback h v = h.lastChild

textarea_tag *attribs*

(textarea_tag *attribs*).get v =

    <textarea *attribs*>v</textarea>

(textarea_tag *attribs*).putback h v = h.value

# DOM Lenses, continued

*l* = textarea_tag({}) ; span_tag({ 'id': 'doc' }, [])

*l*.get v =

```
    <span id="doc">
        <textarea>v</textarea>
    </span>
```

*l*.putback h v = *t*.putback(*s*.putback(h, *t*.get), v)

= *t*.putback(h.firstChild, v)

= h.firstChild.value

# Tree sequencing

/ = span_tag({ 'id': 'doc' }, textarea_tag({}))

Works like ;-sequencing, but looks like HTML!

# Flapjax with Lenses

```
var d = valFromServer('path.to.doc');
textarea_tag({ 'id': 'doc' })
  .bind_to_b(d, 'doc');
valToServer(d, 'path.to.doc');
```

**Or, with a wrapper:**

```
textarea_tag({ 'id': 'doc' })
  .bindToServer('path.to.doc', 'doc');
```

# Contributions

A declarative, composable language for constructing and deconstructing HTML

Programmers no longer need to work with the DOM directly

Now possible to define generally composable widgets

# Not Mentioned Here

Combinators for ordered data, order and list_map – a contribution to the theory of lenses

A solution to the “Table of Contents” problem

# Future Work

- Widgets
  - Fancy looking
  - Fancy acting
- Synchronization policies
- Other list combinators
- Compilation