How I Learned to Stop Worrying and Love the KSS

Part 1: Client/Server Actions

Chris Calloway

TriZPUG

November 2007

Thanks to BubbleNet and Greenfinity

KSS?

- Kinetic Style Sheets
- Keep It Stupid Simple
- Javascript-free Ajax development

Ajax?

- Asynchronous Javascript and XML
- XMLHttpRequest (XHR)
- Web client request in background
- Server response updates client DOM

Ajax?


Get it in your Brain, FAST
Head Rush
Ajax
Learn how to make your web pages talk and listen all at the same time
Master out-of-this-world concepts, including the DOM, JSON, and XML
Watch Asynchronous and Synchronous Applications duke it out in an Espresso Talk
A caffeinated learning guide to the world of dynamic web pages
Load asynchronous programming directly into your brain
Make your clunky web apps feel like dynamic, responsive desktop applications
O'REILLY®
Brett McLaughlin

KSS

- Javascript meta-framework
- Ajax development without javascript
- Style sheets with CSS-compliant syntax
- Domain-specific language (DSL)

KSS Community

Home | Blog | Notes | Documentation (new section) | Bugs |

Search

search

« November 2007 »

Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

You are here: Home

Log in Join

KSS - Javascript-free Ajax

KSS is a javascript framework that aims to allow Ajax development without javascript. It uses stylesheets with CSS-compliant syntax to setup behaviours in the client and a set of well-defined commands that are marshalled back from the server to manipulate the DOM.

What is KSS?

KSS is a generic javascript AJAX framework.

The old name of KSS was kukit (cook-it). It is still in use for the javascript part of KSS.

Future release of **kss.base** will bring KSS working on pythonic platforms, such as:

- **pylons**
- **django**

It will also be possible to use just the javascript on non python platforms like PHP.. ***If you are a developer, please contact us if you have interest in porting KSS to the server environment you use!***

KSS Developers

Godefroid Chapelle
BubbleNet
Belgium

Balazs Ree
Greenfinity
Hungary

KSS Framework

- Formerly “kukit”
- Component architecture
  - kss.base
  - kss.core
  - plone.app.kss

kss.base

- Alpha
- Support for variety of web frameworks
  - Pylons
  - Django
  - non-Python (PHP)

kss.core

- Formerly “Azaz”
- Ajax for Zope
  - Zope 2.10
  - Five
  - Zope 3
  - Grok

plone.app.kss

- Formerly plone.azaz
- Included in Plone 3.0+
- Plone-specific plug-ins
- Consistent with Plone framework policies

KSS Framework

- Sets up DOM events on the client
- Selects from a set of well-defined rules on the server
- Trigger an action on the client or server

KSS Development

- Zope in debug mode
- Disable browser caching
- FireFox: Web Developer Extensions
- FireFox: FireBug
- FireBug: FireKiss.xpi
- portal_javascript in debug mode

KSS Development

- Disable in plone_kss
  - plone.kss
  - at.kss
- Develop HTML first

KSS Example: NavTree

- Create example1.kss
- In portal_skins/custom
- As a Zope file
- Content Type (MIME): text/kss
- In Zope 3, make it a browser:resource (through ZCML)

KSS Example: NavTree

- Register example1.kss in portal_kss
- Compression type = none
- Disable merging
- Disable caching

KSS Example: NavTree

```
<a href="http://localhost:8080/Plone/folder1"
   class="state-published navTreeCurrentItem"
   title="The first folder.">
      <img width="16" height="16"
           src="http://localhost:8080/Plone/folder_icon.gif"
           alt="Folder" />
      Folder 1
</a>
```

KSS Rule

```
a.navTreeCurrentItem:click {
  action-client: alert;
}
```

KSS Rule

```
a.navTreeCurrentItem:klick {
  action-client: alert;
}
```

KSS Properties

```
a.navTreeCurrentItem:click {
  evt-click-preventdefault: True;
  action-client: alert;
}
```

Event Parameter

```
evt-click-preventdefault: True;
```

- evt = parameter type (Event binding parameter)
- click = name of event
- preventdefault = name of parameter

KSS Properties

```
a.navTreeCurrentItem:click {
    evt-click-preventdefault: True;
    alert-message: "Love the KSS!";
    action-client: alert;
}
```

Action Parameter

```
alert-message: "Love the KSS!";
```

- alert = name of action
- message = name of parameter

KSS Example: Server Action

- Callable Method
  - Python Script
  - Zope3 views, multi-adapters, etc.
- Assemble commands
- Marshall back to client as XML
- Command calls DOM manipulation code on client
- Can have selector to control scope of nodes
- Can have parameters

KSS Example: Site Actions

- Create response1
- In portal_skins/custom
- As a Python Script

KSS Example: Site Actions

kss.core

```
# import Through-The-Web(TTW) API
from kss.core.ttwapi import startKSSCommands
from kss.core.ttwapi import getKSSCommandSet
from kss.core.ttwapi import renderKSSCommands
```

KSS Example: Site Actions

The response will contain DOM commands

```
# start a view for commands
startKSSCommands(context, context.REQUEST)
```

KSS Example: Site Actions

Add some commands to the response

```
# add a command
core = getKSSCommandSet('core')
core.replaceInnerHTML('#portal-siteactions',
                      '<h1>Love the KSS!</h1>')
```

KSS Example: Site Actions

Select some nodes for the command

```
# add a command
core = getKSSCommandSet('core')
core.replaceInnerHTML('#portal-siteactions',
                      '<h1>Love the KSS!</h1>')
```

KSS Example: Site Actions

Marshall commands back to client in XML

```
# render the commands
return renderKSSCommands()
```

KSS Example: Site Actions

Marshall commands back to client in XML

`http://localhost:8080/Plone/response1`

XML Payload Response

```
<!DOCTYPE html PUBLIC
  "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:kukit="http://www.kukit.org/commands/1.0">
  <body>
    <kukit:commands>
      <kukit:command selector="#portal-siteactions"
                     name="replaceInnerHTML"
                     selectorType="">
        <kukit:param name="html">
          <h1>Love the KSS!</h1>
        </kukit:param>
        <kukit:param name="withKssSetup">True</kukit:param>
      </kukit:command>
    </kukit:commands>
  </body>
</html>
```

Server Action Rule

```
a.navTreeCurrentItem:click {
  evt-click-preventdefault: True;
  action-server: response1;
}
```

Server Action

```
content = '<h1>Love the KSS!</h1><span>%s</span>' % DateTime()
core.replaceInnerHTML('#portal-siteactions', content)
```

Server Action Error

```
content = '<h1>Love the KSS!</h1><span>%s</span>' % DateTimeX()
core.replaceInnerHTML('#portal-siteactions', content)
```

Server Action Parameter

Don’t forget to add mymessage
to the Python Script parameter list!

```
content = '<h1>Love the KSS!</h1>' \
          '<span><b>%s</b> at %s</span>' \
          % (mymessage, DateTime())
```

Server Action Parameter

```
a.navTreeCurrentItem:click {
    evt-click-preventdefault: True;
    response1-mymessage: "Loving it";
    action-server: response1;
}
```

Multiple Rules per Event

```
ul#portal-globalnav li a:click {
  evt-click-preventdefault: True;
  action-server: response1;
  response1-mymessage: "Loving some KSS!";
}
```

Select nodes by HTML id

```
selector = core.getHtmlIdSelector('portal-personaltools')
core.replaceInnerHTML(selector, content)
```

Select nodes by CSS

```
selector = core.getCssSelector('dt.portletHeader a')
core.replaceInnerHTML(selector, content)
```

Virtual Nodes and Events

```
document:timeout {
  evt-timeout-delay: 8000;
  action-server: response1;
  response1-mymessage: "Loving the timeout!";
}
```

kssproject.org