Technical Debt

#techdebt

Steve Hartzog, Travel Channel

Public Service Announcement

These are my thoughts...

Based on my experience at the Travel Channel.

No guarantees expressed or implied.
Blame nobody but me. :)

But first... a little history

TECHTOBERFEST

Software Engineering History

1980: SmallTalk-80 calls out the MVC pattern...
2005: PHP5 adds support for MVC...
2009: CQ 5.3 is released along with JSP 2.2.
2010: Adobe buys CQ from Day Software.
2013: Adobe renames CQ and releases 5.6 with JSP 2.3.
2014: AEM 6.0 released. Still no support for MVC, but Java 8 finally supported.
2015: AEM 6.1 released. Still no support for MVC.

What is technical debt?

Over time, developers make minor changes.

These fixes often break other things…

Gradually these build up....

“If it’s too complex, it will be difficult to maintain.”

“Complexity is the enemy.”

“Technical debt is anything that makes the dev’s life ***harder***.”

Examples of Complexity

- Extra folders...
- Extra divs that merely add complexity…
- CSS classes without distinct targets…
- Forking a popular library like Bootstrap...
- A faulty Continuous Deployment tool...
- Builds take several minutes to test a change…
- Templates in multiple places...
- Data transformations in multiple places…
- And generally, challenges of doing Front End Dev on AEM…

Folder Debt: Tree of Doom

Our folders:

/src	What? Of course this is source.
/main	Why? Is there a second? Or alt?
/content	Hold on. This is not content. It’s code.
/jcr_root	There is no sane reason for this…
/etc	Is this app really etcetera?
/clientlibs	The whole app is just a client library?
/tccom	Don’t we already know the site name?
/assets	Is source code an asset?
/js	Finally, the application
/images	Wait. Images are at the same level?

Folder Debt: Tree of Doom

Let’s delete a few folders:

/~~src~~	What? Of course this is source.
~~/main~~	Why? Is there a second? Or alt?
~~/content~~	Hold on. This is not content. It’s code.
~~/jcr_root~~	There is no sane reason for this…
~~/etc~~	Is this app really etcetera?
~~/clientlibs~~	The whole app is just a client library?
~~/tccom~~	Don’t we already know the site name?
~~/assets~~	Is source code an asset?
/js	Finally, the application
/images	

Folder Debt: Tree of Win

Then add (2) and move (1):

/webapp	
/jsps	Currently this is at /app/tccom for TC
/images	
/js	Hidden from production, not served to browser
/less	
/dist	Production sees only bundled & minified source

“What if it was this simple?”

DOM Debt: Tree of Fail

Our Content DOM:

```
<section id=”site” class=”flush-top”>
    <div class=”area” data-sni-area=”content”>
        <div class=”site-container”>
            <div class=”site-container-inner”>
                <div class=”layout-contentwell-container” />
                    <div class=”layout-contentwell-row”>
                        <div role=”home” class=”home_leftcontent”>
                    <div class=”layout-rightrail-container”>
```

DOM Debt: Tree of Fail

Let’s delete a few div’s:

<section ~~id=”site” class=”flush-top”~~>
~~<div class=”area” data-sni-area=”content”>~~
~~<div class=”site-container”>~~
~~<div class=”site-container-inner”>~~
<div ~~class=”layout-~~contentwell~~-container”>~~
~~<div class=”layout-contentwell-row”>~~
~~<div role=”home” class=”home_leftcontent”>~~
<div class=”~~layout-~~rightrail~~-container~~”>

DOM Debt: Tree of Win

Our Content DOM, simplified:

```
<section>
	<div class="contentwell">
	<div class="rightrail">
```

“What if it was this simple?”

DOM Debt: Tree of Fail

Our Components:

```
<div class=”parbase baseComponent mediaStrip jukebox section”>
    <div class=”tc-border-control”>
        <div class=”tc-base-component”>
            <section class=”mediastrip-wrapper”>
                <header class=”mediastrip-header” />
                <div data-grid-columns=”1” class=”mediastrip-inner”>
                    <div class=”mediastrip-item” />
```

DOM Debt: Tree of Fail

Let’s delete a few lines:

```
<div class=”~~parbase baseComponent~~ mediaStrip jukebox ~~section~~”>
    ~~<div class=”tc-border-control”>~~
        ~~<div class=”tc-base-component”>~~
            ~~<section class=”mediastrip-wrapper”>~~
                <header ~~class=”mediastrip-header”~~ />
                ~~<div~~ data-grid-columns=”1” ~~class=”mediastrip-inner”~~>
                    <div class=”~~mediastrip-~~item” />
```

DOM Debt: Tree of Win

Our Component DOM, simplified:

```
<div class=”mediaStrip jukebox” data-grid-columns=”1”>
        <header />                          // target with .mediastrip header
        <div class=”item” />                // target with .mediastrip .item
```

“What if it was this simple?”

Real World Debt: How did it happen?

TECHTOBERFEST

Modal Dialogs

1. We fork bootstrap.
2. Design stipulates a different base modal dialog style.
3. Later, we implement a photo-gallery...
4. Then we start gallery-voting…

Modal Dialog Solution

1. Make the base .modal-close a mixin().
2. Reference the mixin locally, directly on the gallery class.
3. Target the gallery’s modal instead of the global modal.
4. Delete the various javascript references to scrollTop.

Sharebars

1. First sharebar is created, with two styles.
2. 3rd style is created, but hidden on mobile.
3. Client template (duplicate of server) created.
4. Constant tweaks and one off fixes jammed into Less.
5. Then desktop style shown on mobile…

Sharebar Solution

```
@media @screen-med {
 div#gigya-share-jst_1reaction0:hover:before {
   color: @facebook-icon-color;
 }
}

// styles that both types of sharing share
.social-share, .social-share-this-page {
 width:50%;
 position: relative;
 &:extend(.component-wrapper all);
 @media @screen-xsmall {
   width: auto;
   position: fixed;
   top: 0;
   left: 0;
   vertical-align: top;
   display: inline-block;
   z-index: 999;
   background-color: #303030;
```

```
// Global Icons
.gig-button-container-facebook { }
.gig-button-container-twitter { }
.gig-button-container-pinterest { }
.gig-button-container-googleplus { }
.gig-button-container-email { }

// ShareBar Treatments
.grayScaleTreatment() { }
.desktopTreatment() { }
.mobileTreatment() { }
```

JS Build Evolution

1. RequireJS: Let’s async everything!
2. Let’s bundle the commonly used modules, then only async a few.
3. Let’s implement fingerprinting.
4. Fine, let’s not async anything....

Real World Debt: Current Struggles

TECHTOBERFEST

General Build Struggles

Outdated tool chain.

- What if we used TypeScript? Gulp? Browserify?

Cloudbees is a fail.

- What if we used Bamboo?

Incredibly complicated AEM configuration

- What if we had an Docker container already configured?

Image & Video Content is a PITA

- What if we pointed all images to a central point, and just updated content?

AEM Struggle

But most importantly…

Remember that AEM 5.6 has no way to separate the view from the controller.

AEM Struggle

- Sling is not a controller. It’s a router.
- JCR is an extremely slow and inefficient model (which is why we use Solr).
- JSP does not support MVC, and mixes business logic, cq:includes/taglibs and DOM.

Simple Solutions:

1. Sightly with the AEM 6.x JavaScript-Use API
2. Use neba.io to bring MVC to AEM (via Spring MVC)

Can we prevent this?

We can lessen the impact

1. Avoid complexity… make it simple.
2. Constantly reevaluate and refactor.
3. Make sure that new code is isolated and clean.
4. Simplify the build environment.
5. Separate the controller from the view.

But remember… keep it *simple* !

Questions?

Session Feedback Code

GUDJQT

#techdebt