Concept - STK improvements

Goal

This page is to collect and discuss ideas for possible STK improvements. This is a rough, unculled list of wishes, not yet a roadmap.

Motivation

Why does the STK need improvements?

- Good impression - Developers evaluating magnolia should get a good impression that our offering is up to date.
- Best practices - Front end best practices are evolving for good reason - we should ensure that our offering incorporates them.
- In-place dynamic ajax content updating should be supported
- Not Monolithic - Some people are scared away by the standard templates that we provide, we should make it clear that there are different parts of STK that can be mixed and matched.

Mission

STK must support the reality of how frontend engineers build modern websites.
STK should enable and encourage front-end best practices.

Topics / Wishlist

- **Separate STK foundation from templates**
  - Just needs to be clear. Documentation should distinguish them - help people to not be scared away
    - You can use the foundation without using the templates.
    - You can use the components without using the templates.
  - magnolia-empty-webapp should probably already have apps for Templates and Resources, and should support inplace templating.
  - Perhaps the STK app group should be renamed to Templating .

- **Create alternative templatesets**
  - Classic templateset (what we use now)
  - Grid layout
    - Bootstrap templateset
    - Foundation?
    - Grid thing that SWM guy presented (advantages to this, if we have one of the above?
  - JQuery Mobile templateset
  - Simple demoproject for each template set.
  - Documentation showing that its easy to create your own templateset.
  - Encourage companies to create their own templateset and add to forge.
    - ie - we could build bootstrap - and someeone else might be motivated and do foundation or jquery UI or extJS

- **Front end best practices**
  - Be light-weight
  - Start minimal - let people add things. Rather then starting monolithic and cluttered and people have to rip things out.
  - Only include needed css & javascript
  - Only initialize needed javascript
  - Clientside form validation (falling back to server side)

- **Improve Javascript handling**
  - Revise onload script initialization.
    - so that only used things are initialized
    - customers can easily / intelligently plug into that system.

- **AJAX content loading**
  - Standard way to load dynamic content?
  - Bookmarkable URLs
  - SEO of dynamically loaded content
  - Support single page websites
    - Tablets / smart phones /

        - apps (ie phonegap / firefox OS app / windows app)
        - usatoday.com
    - Content transitions
        - Fade / Slide / Zoom

- **Frontend Templating**
    - ie Mustache/Handlebars/Hogan
    - This supports client-side javascript frameworks (backbone / angular)
    - Compiled templates ? Hulk?
    - Rest API to retrieve data for templates.

- **Update components**
    - Make sure our components are up to date and working well.

- **Support LESS / SASS** out of the box.
    - Experienced front end dev do not want to use freemarker to process css.

- **Documentation**
    - A clear demo of the components that are available
        - Pizza topping list.
        - Maybe something like the Vaadin sampler.

- **New standard theme**
    - pop looks outdated.
    - Hire professional designeers? - or find really nice themes on web to borrow from.
    - Have 3 themes that right away show off how much they can change things.

- **New templates for running instance**
    - / that no-one can change.
    - What the SWM guy presented - Philipp seemed interested - is this not already possible?

- **Make working with Resources easy and as a developer would expect.**
    - There is a lot of confusion about how to use resources. People want to just work on files on the filesystem as they are used to - but the handling of the Resources module - and storing resources in a workspace + using the bypass switch is confusing to them.
    - (Example: http://forum.magnolia-cms.com/forum/thread.html?threadId=e6fb47f8-33f8-4ccd-abaa-c088eba35274)

Advantages of STK that we want to honor / not break

- Fast startup - Cool components ready for you to use.
- Availablity of components - control which components can go in which areas.
- Responsiveness - Send less resources to mobile devices
- Channels
- Image variations
- +Personalization
- Easy upgrade to new versions of magnolia
- Current STK is very good for HTML compliance and accessibility. (But this should be verified, tests added, and an accessibility showcase page should be added.)
- ...

Concrete Customer / Partner Input

- Sebastian had a customer recently asking if we supported Bootstrap framework.
- Mark from Virgin Holidays
    - Frontend developer wanting to use front end frameworks that he already has experience with.
    - "Its offputting to see bloat or bad front-end design"
    - STK Feels too heavy
    - Scared about having to do lots of configuration in the tree.
    - Dont want to be locked in.
    - Is there a way to encapsulate an HTML "module"? It includes its own CSS and Javascript? (Sounds like Angular thing)

Inspiration / References

- https://prismic.io/faq#how-is-this-different-from-a-good-old-cms

- http://www.magnolia-cms.com/conference/program/presentations/rich-javascript-apps.html
- STK 2.0 Workshop
- http://wiki.magnolia-cms.com/download/attachments/42270723/stk-remarks-vpro.pdf

Minimal Templating Kit

Use Cases

To ensure that what we are designing works - we should have several (3?) use cases in mind. These could be used for thought experiments, or if time permits we could build out actual examples.
Better yet - if we could use these use cases internally at Magnolia then they would be even more realistic and we would understand them better.

Potential MTK use cases

- Blog/News site with categories (Magnolia developer blog?) - Blog posts are content from a Blog/News app, not pages.
- Web application - Angular? Classic MVC TODO list?
- Shop / Catalog
- Landing page / Microsite
- Developer quiz (Magnolia Certification Test?)
- Intranet: ContentApps for people, interest groups, interests, abilities, places, and times.And modules, departments, cells, and concepts.
- Twitter Bootstrap site
- Presentation (ie slide deck)
- Admin UI components
- Template Apps
- Magazine
- Iphone/android/mobile native app.
- Game?

Potential content-set use cases

Maybe we should rather start with a sets of content as use cases. The templating kit must support easily creating a meaningful content experience with any set of content.
As the CMS cell - we should probably get some inspiring high quality content to work with. (Including high quality design)
This will help us when we want to create convincing demo projects.

- Intranet - people, projects, processes, interests, roles
- Product information - brochureware
- Branded content experience (redbull)
- Interest / Content (One of a kind cars & photos/videos/stories)
- News / Blog
- TODOList - User application data.
- Organizational + Forms (City / municipality / School)
- Shop

I guess this goes in the directions of verticals or "Solutions" based on magnolia.