**Adapt Architecture**

**Introduction**

Adapt is a multiple device learning tool. Built in HTML and open web standards we aim to create a leading open source learning tool set that enables anyone to create engaging learning material. By using responsive design we’ve built a tool that can display courses across multiple devices, where the content responds to the users screen size.

Adapt was built with e-learning in mind and has core features like tracking completion status and assessments. By making this product open source we hope to create chances for people to develop their learning through online content.

**Principles**

**Structure**

Adapt has some key structural elements. We have a menu system (which is explained further down). Then we have a concept of Pages. Pages are where the main learning content is presented.

A **Page** can be broken down into smaller segments:

**Articles** - these are used to hold similar pieces on learning. They are also used for presentation purposes as Pages need to be split into relevant content.

**Blocks** - Blocks are placed within Articles and contain Components. You can have as many Blocks as you like in an Article, but we suggest no more than 3 or 4. Each Block in an Article should be related content.

**Components** - Components are the main interactions of Adapt. These are plugin based Components that can be added or taken away from a course. (See Components below). Some examples of Components would be a text, graphic or a multiple choice question. Components are either full spanned or single width. Full spanned components fill the containing Block whilst single width components fill only half and need another component along side it.

Components fall under two categories:

Question Component - Used to test knowledge (Multiple Choice Question).

Presentation Component - Used to display content (Narrative).

To summarise, a typical Adapt course could look like this:

// Insert image of basic course layout

**Responsive Layout**

Adapt is built around a responsive layout, where the breakpoints are based upon content. Although this is not fully responsive, it enables us to keep structure and components working across multiple devices. We’re able to keep our components from becoming too small to interactive with, whilst maintaining a responsive layout. Our core grid system is as follows (Based upon widths):

**960px and above -** Used mainly for desktop browsers. Single width Components sit side-by-side, whilst spanned Components are at their maximum width.

**959px to 760px -** Similar to above but all the components are now percentage widths based upon the wrapper.

**759px to 520px -** Between this breakpoint single width Components are snapped on top of each other. Whilst full spanned Components are either scaled down or transform into another Component (core Hot Graphic transforms into a Narrative).

**519px and below -** All Components are now percentage widths based upon the wrapper.

**Data & Security**

// How data is stored, created and loaded. Security measures.

**Completion of a course**

Every key structural element in Adapt (Menu Item, Page, Article, Block and Component) has a state of completion. As a default each element is set to incomplete. At it’s simplest form Adapt is completed from Component upwards - through to Blocks, then Articles, then Pages and finally any Menu Items.

Components can be set to complete in a variety of ways. For example, a text Component can be completed by coming into the viewport, a multiple choice question can be completed by answering the question and a narrative can be completed by visiting the last graphic.

Once a Component is set to complete the following process is executed:

1) The parent Block of the completed Component finds out if there are any other incompleted Components inside.

2) If the Block finds any incomplete Components it stops. If all Components are complete in that Block then the Article that contains this Block checks against its children (Blocks).

3) If the Article finds any incomplete Blocks it stops. If all Blocks are complete in that Article then the Page that contains this Article checks against its children (Articles).

4) If the Page finds any incomplete Articles it stops. If all Articles are complete in that Page then the Menu Item that contains this Page checks against its children (Pages).

5) This process continues until the Course is notified of every element being set to completed.

As well as setting a Component to complete, each question component should carry a score/weight and whether the Component has been passed or not.

**Browser support**

We recommend these browsers for displaying Adapt contenet: IE8+, Chrome (v30), Firefox v17, Safari v5 (desktop), iOS v5+ (Safari) and Andriod v2.3 (default browser).

**Core functionality**

**Libraries**

**Backbone -** Our Models, Views, Collections and Router are all based upon Backbone’s. When creating a new View or Model you’re actually creating a Backbone.View with a bit more functionality. We use Backbone as it’s super light, great for mobile devices and it enables us to have enough separation between our data and views.

Website: <http://www.backbonejs.org>

Version: 1.0.0

**Underscore -** Underscore is a hard dependency of Backbone. It’s a “utility-belt library for JavaScript” which comes with some handy functions. Underscore is used throughout Adapt, especially when dealing with objects.

Website: <http://www.underscorejs.org>

Version: 1.5.2

**jQuery -** jQuery is used for dom manipulation and some animations. It allows us to use a simple API that works across multiple browsers.

Website: <http://www.jquery.com>

Version: 1.10.2 or 2.0.3 based upon browser.

**Handlebars -** We use handlebars as our templating engine. It enables us to quickly build templates for all of our HTML markup which contains a level of logic.

Website: <http://handlebarsjs.com/>

Version: runtime-1.0.0

**Modernizr -** Modernizr is a feature detecting library that enables Adapt to display or work differently based upon the features of the browser. We also use Modernizr to load our conditional scripts with its built in yepnope loader. This enables us to serve up a faster and more lightweight version of Adapt if you’re on a mobile device or modern browser.

Website: <http://modernizr.com/>

Version: 2.6.2

**Inview -** The inview plugin allows us to find out when a component or dom element comes into view. This is used for setting complete on the text and graphic component. Inview could also be used for bespoke animations - so when an animation comes into view, the animation can start.

Website: <http://remysharp.com/2009/01/26/element-in-view-event-plugin/>

Version: N/A

**ImageReady -** This plugin allows us to trigger an event based upon whether an image has been cached/loaded. For example, ImageReady is used to check all the images on a page are loaded before showing the page.

Website: <http://remysharp.com/2009/01/26/element-in-view-event-plugin/>

Version: N/A

**JSON2 -** Some older browsers lack support for the JSON standard object. This polyfill enables support. This is conditionally loaded at load time.

Website: <https://github.com/douglascrockford/JSON-js>

Version: N/A

**ScrollTo -** This jQuery plugin is used to scroll the page to certain locations. It can take in dom elements as a selector, but can also animate the window scroll position. This is used to navigate back to a dom element on revisit and during animations.

Website: <http://flesler.blogspot.co.uk/2007/10/jqueryscrollto.html>

Version: 1.4.3.1

**swfObject -** This plugin unifies the different browser implementations of a Flash object across browsers. It enables a simple javascript API that can easily communicate between Flash and Javascript.

Website: <https://code.google.com/p/swfobject/>

Version: 2.2

**Build Process and Preprocessors**

**Grunt**

We use Grunt to do our build process. For further information on how to install Grunt and use it please visit our documentation here…

Grunt helps us automate our build process by doing such jobs as concatenating our files into one and compiling our Less files into CSS files.

**Less**

Less is a preprocessor for CSS that enables us to right more maintainable/extendable CSS code. Our Less preprocess is handled by Grunt and enables us to ‘watch’ files for changes and run a process in the background.

For more information on Less please see our developers guide...

**Adapt Core Code**

**Adapt**

Adapt has a small core which enables us to build a simple modular approach. By having Adapt only do two things, ‘register’ and ‘create’, we’re enabling other modules to take control of their own tasks.

Every plugin, whether that be a menu, theme, component or extension will need to register with Adapt through a simple API to access the internal event system (see Hub below). The API allows you to easily extend/inherit a previously registered view.

Once a plugin is registered, it can then be created. Adapt allows you to create a registered plugin, passing in all the arguments you need to start your plugin.

For more information please see the developers guide.

**Hub**

Adapt’s internal event system can be accessed from inside any registered plugin. This approach is based upon the Mediator/Observer pattern widely used throughout large scale javascript applications (similar to the Law of Demeter). This enables us to keep our code maintainable by avoiding strong coupling between our plugins. By strong coupling, we mean directly calling methods of other objects or affecting core code.

Instead we can tap into hooks/events that are fired off and build our plugins off these. No direct communication between core code and plugins should happen. By triggering an event and listening we can extend the Adapt core functionality without affecting any other plugin.

For more information please see ...

**Manager**

Adapt is a course manager who is in control of global and browser environment settings. Manager stores data such as the default language of a course, the current users accessibility needs and uses Hub to trigger events about updates to the browser settings like device orientation/screen size.

**Router**

Adapt is a single page app, and by using a router we are able to bookmark and navigate around a course without loading a new url. Although there is a longer upfront download time this approach enables us to have a fast and responsive navigation between content pages (This is vital for mobile devices where we don’t want to be downloading code after every time a user navigates through the course). We have two main places that the router routes to - a menu system and our content page.

**Utils**

What utilities are available to Adapt developers

...

**Models, Views and Collections**

Adapt’s core is built upon Backbone. Backbone enables us to decouple our data from our views and maintain a low download footprint. It comes with views, models and collections that already have handy methods. Adapt’s core models, views and collections add some extra functionality to these objects.

For more information on these please read the Developers Guide.

**Core bundle**

As a core package we intend to include some plugins as default. These plugins will be maintained by the core contributors, so there is a degree of testing and stability that a user can expect from these plugins.

The current core plugin this is below:

Components - Text, Graphic, Blank, Hot graphic, Narrative, Accordion, Multiple choice question, Graphical multiple choice question, Matching and Text input.

Extensions - Tutor and Block slider.

Themes - Adapt theme.

Menus - Adapt menu.

**Folder structure**

**Core**

All core code is located in here. This should never be touched. Instead we suggest you use the extension, component, theme, menu or bespoke folder for modifications.

**Extensions**

Extensions are extra functionality that adds to Adapt core. For example Tutor, the feedback plugin bundled with core is an extension. An extension is something that can easily add to core but isn’t relied upon.

For more information on how to develop an extension please see...

**Components**

Components are the main interactions that Adapt is based upon. Adapt can contain as many components as the course creator wishes. Examples of components are text, graphic or a multiple choice question.

For more information on how to develop a component please see...

**Themes**

Themes are interchangeable plugins that affect the appearance of Adapt. Each theme must contain the main layout templates such as the page and article. By putting the layout templates in the theme folder we’re able to creating a variety of themes based upon different HTML templates. Theming also contains some Javascript and Less/css files.

For more information on how to develop a theme please see...

**Menus**

Menus are similar to themes in that they can be interchangeable and contain their own HTML templates. Adapt has been built with a menu system that overlooks all the main content. It has been designed in such a way that the menu system can have as many layers as possible.

For instance, a user can enter a course and is given the choice of learning about “Planets” or “Stars”. The user makes a choice and is then presented with another menu layer of either a list of “Planets” or “Stars”. If the user progresses any further they will be presented with the learning content.

However any type of layering is allowed. A course could go straight to a piece of learning content or only have one menu layer.

For more information on how to develop a menu please see...

**Course**

The course folder contains all the assets and data for the course. Adapt’s data is stored in single JSON files based upon this structure - course.json, contentObjects.json, articles.json, blocks.json and components.json.

Inside the course folder are language folders. This allows separation of our data based upon language. Any graphical elements or files that are content based should go into their relevant language folder under assets. However there is a global assets folder for files that are used across all languages, for example, a background image for the page.

**Bespoke**

This folder is mainly used to create something that is bespoke to this course and is unlikely to be used again.

**Editor**

**How Adapt interacts with this**