Rubix Documentation: L20n

Mozilla L20n is a new localization framework developed by Mozilla for the Web. It allows localizers to put small bits of logic into localization resources to codify the grammar of the language. If you are unfamiliar with Mozilla's awesome framework we suggest you go through the learning material provided here: **L20n By Example (http://l20n.org/learn/)**

Of all the concepts there are three main concepts that are required to be understood by developers using L20n:

- **Entity:** Entities are containers for information. You use entities to identify, store, and recall information to be used in the software's UI.
- **Context:** Each context stores information about languages available to it, downloaded resource files and all entities in these resource files. Software developers can create contexts and query them for values of specific entities.
- **Context data:** Context data is how entities defined in L20n resources can interact with non-localizable variables provided by the developer. Context data is generally unknown at the time of writing the L20n code. By assigning values to it, the developer makes it known at runtime.

To use L20n you need to get localizers to write language specific files following the conventions set by the L20n framework. These files (should be named strings.l20n) should then be stored in the **public/locales/app** folder with names reflecting language codes (preferably ISO 639-1 format). We have also provided a handy flags icon set (courtesy GoSquared (http://gosquared.com)) for your use. You can use currenly only use 1 file per language and should follow this format: **public/locales/app/<two-letter-lang-code>/strings.l20n**. If in doubt, refer to the file structure in **public/locales/demo**.

The markup for invoking L20n entities is as follows:

```
<Entity entity='some-entity-here' data={{a: 'some context data', b: 'some other context d
ata'}} />
```

The Entity component defined above requires you to have 'entity' property defined alongwith an optional 'data' property. React takes care of keeping the rendered data in sync and the resultant output is actually a span element (so it can be used within Buttons or any other block element without compromising the interface).