Known issues - Zenterio “RocketApp”

*** Does not render RCSS styled tags at all**, no known cause or workaround
Images and text does work however (Text has some issues though described below)
This makes it currently impossible to decorate focus of elements using RCSS, The sample application is therefor hard to navigate on the STB.

*** Images need to be PNG, it won’t load JPG etc.** Workaround: Convert your image assets to PNG.

*** Images does not have the proper colours**, No known workaround, looks like RGB(A) are incorrectly swizzled.

*** Text does not clip according to the elements clip box**, No known workaround; Might be related to point 1

*** RocketApp seg faults upon launch**- Workaround: Use full path to the assets, this appears to be only an issue when assets are loaded/refered to by Lua and not actually RML. See my start.lua how I went about this; what’s not obvious in my start lua is that I double mounted the USB stick to a known location, in my case /tmp/lua

You can do that quite simply by writing
```
mkdir -p /tmp/lua
mount /dev/sda1 /tmp/lua
```

*** Timers are simply not useable**, RocketApp exposes an API sys.new_timer(<delay>, <callback>) to create timers however at the writing time, this will indeed create a timer and execute the callback once after the specified time. The problem is that the timer is repeating and never stops repeating (although it doesn’t call the callback ever again) if you try to nest timer creating to create a repeating timer you quickly overflow the event queue and the application hangs.

*** Event handling differs between libRocket and the STB**, this is not a major functional problem as it’s quite simple to work around; however it makes it more cumbersome to develop a “libRocket” application when parts of libRocket aren’t supported on the STB, and further more it will make scaling difficult; writing a complex system modular without additional Lua support there will simply be whoever overwrites onKey most recently that gets the keyInput; the only way to workaround this is to have some Lua framework that handles key routing (that’s probably something you would want to do in the end anyway due to the fact libRocket is very oriented around mouse navigation and that’s obviously not an option here.)
LibRocket key handling works like the traditional DOM model and are provides context variables with the event so you know the originating element etc (which as already mentioned is not the complete solution on a device lacking mouse/touch input, however it allows for writing input handling emulation fairly easily as you can dispatch events on a specified element and libRocket will just “work”)
It also solves the issue with multiple input handlers being active at once.

*** Documentation is sparse**, Essentially there are no documentation at all at this

time for the APIs exposed by Rocket App. As RocketApp is under ongoing development there simply don't exist any documentation making it difficult to develop anything that's not just purely libRocket based.

*** Default output commands such as "print()" etc are not output to the console**, Workaround: Override the global print with print = sys.zenterio_debug; This should be adequate for most situations.

*** RCSS changes to display, does not always work.** In the example code when you press the "trailer" button the entire presentation screen is hidden using "display: none", on the STB this does not occur and you left with the previous buffer as a background.

Known issues - libRocket/Lua general

*** Lua support is somewhat spartan**, it provides just the bare minimum and some things are one way unlike the native C++ API.
For example you can't query hidden elements using the Lua bindings; this is rarely an issue but limits the capability of Lua- if supported it could allow for writing pure Lua generators for custom elements.
Some APIs are one way, for example the Lua bindings allows for registering event handlers on object but there is no way to remove these- I'm not certain of the implications of this but it could theoretically mean lua objects will be impossible to garbage collect once bound to a even handler scope (but as stated it has not been verified if this is the case), the lack of functionality itself is not a show stopper as it can easily be workaround by a lookup table based event handler and simply "nil" the handler when you no longer want to handle the the event.

*** No support for timers**, libRocket has no builtin support for timers making simple things such as updating a clock or animating something impossible.

*** Graphical expression**, maybe not technically an issue, but libRocket feels quite dated in it's capabilities; as long as you don't want to generate tons of image assets to simulate things such as gradients, rounded corners etc you'll have to be prepared that the layout will be square and somewhat spartan. On the positive side it does support RGBA so theoretically gradients could be simulated with dynamical generation of tags of various alpha.

*** Some RCSS tags don't dynamically update**, of the RCSS attributes I've experimented with and unfortunately some of the most useful ones (the decorator attributes) does not appear to support being dynamically updated, changing the source for a *-decorator-image-src just doesn't work; it seems these decorators are only evaluated when the view is originally rendered.
It's a pity as it could make layouts more powerful such as supporting theming etc to some extent the <img> tag does mitigate this but as far as I know the <img> tag don't support tiling etc.

*** Text won't render at all unless a valid font face is specified**, there is no "fallback" / "default" font face for text rendering, this is more of a "good to know" and it's hard to mis as the log will spam errors when this happens.

*** <img> tag don't accept data uri scheme**, this is more of a "good to have" feature, as it allows for sending image data in band with other metadata; and a even bigger benefit would be that it would be possible to create significantly more complex layouts as rounded corners etc could be generated procedurally from Lua.

*** Unicode support is limited**, libRocket by default renders just a subset of the Unicode space; (Internally it converts it to another character encoding) additional (at the cost of higher memory usage and slower start up) can be defined via the RCSS property font-charset: <range>. However the Lua version used 5.2.3 does not have proper unicode support (something fixed in 5.3) making it very cumbersome to deal with unicode in general.
RML does not support unicode escape sequences, but should theoretically display them fine if the actual RML file was saved with UTF-8 encoding. It should be noted though that attempts to load, use a custom Unicode TTF with symbols far off the regular coding range has been unsuccessful and images been used for symbols instead. It's not known why attempts failed.