Riot.js cheatsheet

Component styling

This section only applies to Riot.js 2.x. Since 3.x, all styles are scoped *by default* and you can simply add a `style` tag to your component.

1. You can use a `<style>` tag within your tag. This style tag is **applied globally** by default.
2. You can **scope your style tag** to limit its effect to the component that you've defined it in. Note that scoping is **based on the tag name**. There are two options:
3. Use the `scoped` attribute, eg. `<style scoped> ... </style>`
4. Use the `:scope` pseudo-selector, eg. `<style> :scope { ... } </style>`
5. You can change where global styles are 'injected' by having `<style type="riot"></style>` somewhere in your `<head>`. This is useful for eg. controlling what styles are overridden.

Mounting

"Mounting" is the act of attaching a custom tag's template and behaviour to a specific element in the DOM. The most common case is to mount all instances of a specific top-level tag, but there are more options:

1. Mount all custom tags on the page: `riot.mount("*")`
2. Mount all instances of a specific tag name: `riot.mount("app")`
3. Mount a tag with a specific ID: `riot.mount("#specific_element")`
4. Mount using a more complex selector: `riot.mount("foo, bar")`

Note that "child tags" (that is, custom tags that are specified within other custom tags) are automatically mounted as-needed. You do not need to `riot.mount` them separately.

The simplest example:

```
<script>
// Load the `app` tag's definition here somehow...
```

```
document.addEventListener("DOMContentLoaded", (event) => {

    riot.mount("app");

});

</script>


<app></app>
```

Tag logic

- **Conditionally add to DOM:** `<your-tag if="{ something === true }"> ... </your-tag>`
- **Conditionally display:** `<your-tag show="{ something === true }"> ... </your-tag>` (but the tag always *exists* in the DOM)
- **Conditionally hide:** `<your-tag hide="{ something === true }"> ... </your-tag>` (but the tag always *exists* in the DOM)
- **For-each loop:** `<your-tag for="{ item in items }"> ... (you can access 'item' from within the tag) ... </your-tag>` (one instance of `your-tag` for each `item` in `items`)
- **For-each loop of an object:** `<your-tag for="{ key, value in someObject }"> ... (you can access 'key' and 'value' from within the tag) ... </your-tag>` (this is *slow!*)

All of the above also work on *regular* (ie. non-Riot) HTML tags.

If you need to add/hide/display/loop a *group* of tags, rather than a single one, you can wrap them in a `<virtual>` pseudo-tag. This works with all of the above constructs. For example:

```
<virtual for="{item in items}">

    <label>{item.label}</label>

    <textarea>{item.defaultValue}</textarea>

</virtual>
```

Revision #1
Created 11 December 2024 15:52:46 by joepie91
Updated 11 December 2024 18:44:19 by joepie91