Grammuelle

A Grammacology of Design

Grammuelle is a Declarative CSS (SASS) framework for building cellular designs (all puns intented).

Example

Try

```
@include Do('moduleN') {
  @include If($type: "form") {
    @include Then($type: "modal");
  };
}
```

To get

```
.__moduleN__ form [capabillity="cancel"] {
  background: #777; }
.__moduleN__ form [role] {
  outline: 1px solid gold; }
.__moduleN__ form [rel] {
  background: limegreen;
  color: #fff; }
.__moduleN__ form [cancel] {
  color: red; }
.__moduleN__ modal [role] {
  outline: 1px solid gold; }
.__moduleN__ modal [rel] {
  background: limegreen;
  color: #fff; }
.__moduleN__ modal [cancel] {
  color: red; }
```

Maybe (not working yet)

```
$ grammuelle do "aNewModuleName" "background-image: url('http://placehold.it/350x150'); colo
```

To update the `__stylebook__` with

```
@include Do('aNewModuleName') {
  background-image: url('http://placehold.it/350x150');
  color: gold;
}
```

Get weird

```
// @fileOverview ./src/__templates__.scss
$tachyonsOnceExampleRedux: (
  __typeplate__: _(
    (extend, f6),
    (extend, grow),
    (extend, no-underline),
    (extend, br-pill),
    (extend, ph3),
    (extend, pv2),
    (extend, mb2),
    (extend, dib),
    (extend, white),
    (extend, bg-black),
    (extend, hint--bottom)
  ),
);

// @fileOverview ./__interface__.scss

@include Do('module__tachyonsOnceExample') {
  a { @include Once(
    $model: "anchor",
    $collection: $tachyonsOnceExampleRedux
  ) }
}
```

Install

```
$ bower|npm install grammuelle
```

Now update

```
// your theme.scss
@import "path/to/gramuelle/__stylebook__";
```

Grammar books

```
__basics__.scss     (physical and geometric denominations)
__core__.scss       (just grammatical stuff)
__interface__.scss  (idiomatic stuff, web stuff, thematic media queries)
__stylebook__.scss  (a new book for *grammuratificographs*)
```

Names are up to you. What problems are we solving?

Problems

vjuex's `big css` problems:

P1. Don't mess with other CSS on the page

Guard against globals:

```
@include Do ... If ... Then ...
```

P2. Dependencies

Pair up thematic media queries with JS expectations:

```
@include Do ... Responses ...
```

P3. Dead Code Elimination

See pynaximander. Otherwise

```
@include Do ... With ...
```

encourages template sharing which can be used in build automation pruning.

P4. Minification

Use BEM classes and reduce to hashes:

```
@include Do ... Class ...
```

P5. Sharing Constants between CSS and JS

```
@include Do ... Once ... [getComputedStyled or requestAnimationFrame with basics]
```

P6. Non-deterministic Resolution

```
@include Do ... Class ...
```

P7. Isolation

```
@include Do ... Rules ...
```

Overview

Module-level interplay goes here, or meta-CSS type junk like:

1. `modules`
2. `grids`
3. `displays`
4. `hooks` (`before`, `after` order serializations)
5. `models`
6. `basics` (wip)

Generally the design ontology should be implemented within an **internal**, so that we provide CSS middleware from admixtures of the following:

1. `Ifs`
2. `Onces`
3. `Spreads`
4. `Classes`
5. `Withs`
6. `Responses`
7. `Wraps`
8. `Rules`

Overview

`theme.scss` and `__interface__.scss` are good places to start. The latter is like a `master style` booklet for your project. After you are done preparing your grids, modules, rules, hooks, responses, etc., `import` your interface:

Minimalism

```
@import "__stylebook__";
```

Maximalism

```
@import "__interface__";
```

Extremalism

```
@import "__theme__";
```

Is style for humans possible?

Probably not, but until neversville we've got non-atomic declarative CSS to make what's possible explicit.

Documentation

See `./docs/` or Web docs.

Glossary

See `./glossary.md`

Idioms

See `./idioms.md`

Contribute

See `./CONTRIBUTE`

License

See `./LICENSE`