Cognizant
Passion for building stronger businesses

RADICAL CSS

An holistic Framework

Compiled by:
Tanmay A Navghare
Sr. Architect TAO-BFS
avra.banerjee@cognizant.com

Cognizant

TABLE OF CONTENTS

What Is RADICAL CSS? 3
ITCSS 3
Advantages 4
atomic css 4
Why atomic Design? 8
RADICAL CSS 9

WHAT IS RADICAL CSS?

Radical CSS is a combination of two highly scalable, extendable, and reusable framework derived from two methodologies, ITCSS and Atomic CSS.

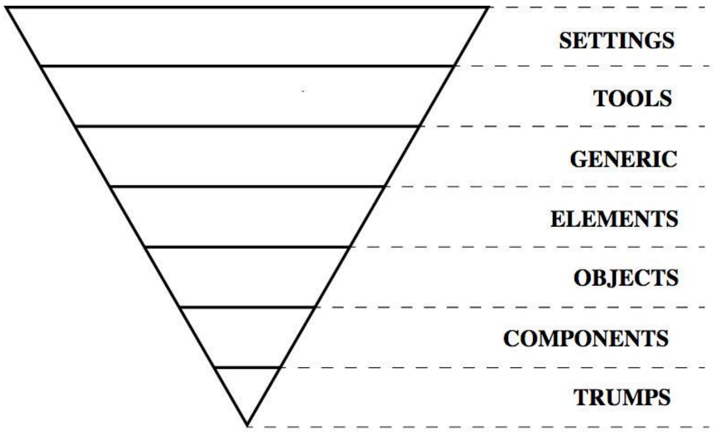
ITCSS

ITCSS is a sane, scalable and managed CSS architecture for larger UI projects.

ITCSS generally called “Inverted Triangle” is a methodology that involves visualizing the entire css as a layered, Inverted Triangle. This hierarchical structure helps in ordering your css in most effective and least wasteful way.

ITCSS can be used with preprocessors or without them and is compatible with CSS methodologies like BEM, SMACSS or OOCSS.

One of the key principles of ITCSS is that it separates your CSS codebase to several sections (called layers), which take form of the inverted triangle:



A directional flow from the flat top of the inverted triangle down to the tip at the bottom symbolizes an increase in specificity. This focus on less specificity makes it easier to reuse classes across a site multiple times.

Each subsection of the triangle may be considered a separate file or group of files, although you don’t need to write code in that way.

The layers are as follow:

- **Settings** – Preprocessor variables and methods (no actual CSS output)
- **Tools** – Mixins and functions (no actual CSS output)
- **Generic** – CSS resets which might include Eric Meyer’s reset, Normalize.css, or your own batch of code
- **Elements** – Single HTML element selectors without classes
- **Objects** – Classes for page structure typically following the OOCSS methodology
- **Components** – Aesthetic classes for styling any & all page elements (often combined with the structure of object classes)
- **Trumps** – The most specific styles for overriding anything else in the triangle

Each layer of the inverted triangle can be adjusted to suit your needs. So if you don't use a CSS preprocessor then you won't need the Settings or Tools layers. You should feel free to interpret each subsection as you see fit.

ADVANTAGES

Objects reusability for fast development

ITCSS helps in building a library of reusable components to allow fast frontend development. UI would then be composed of generic objects and project specific components.

Flexibility

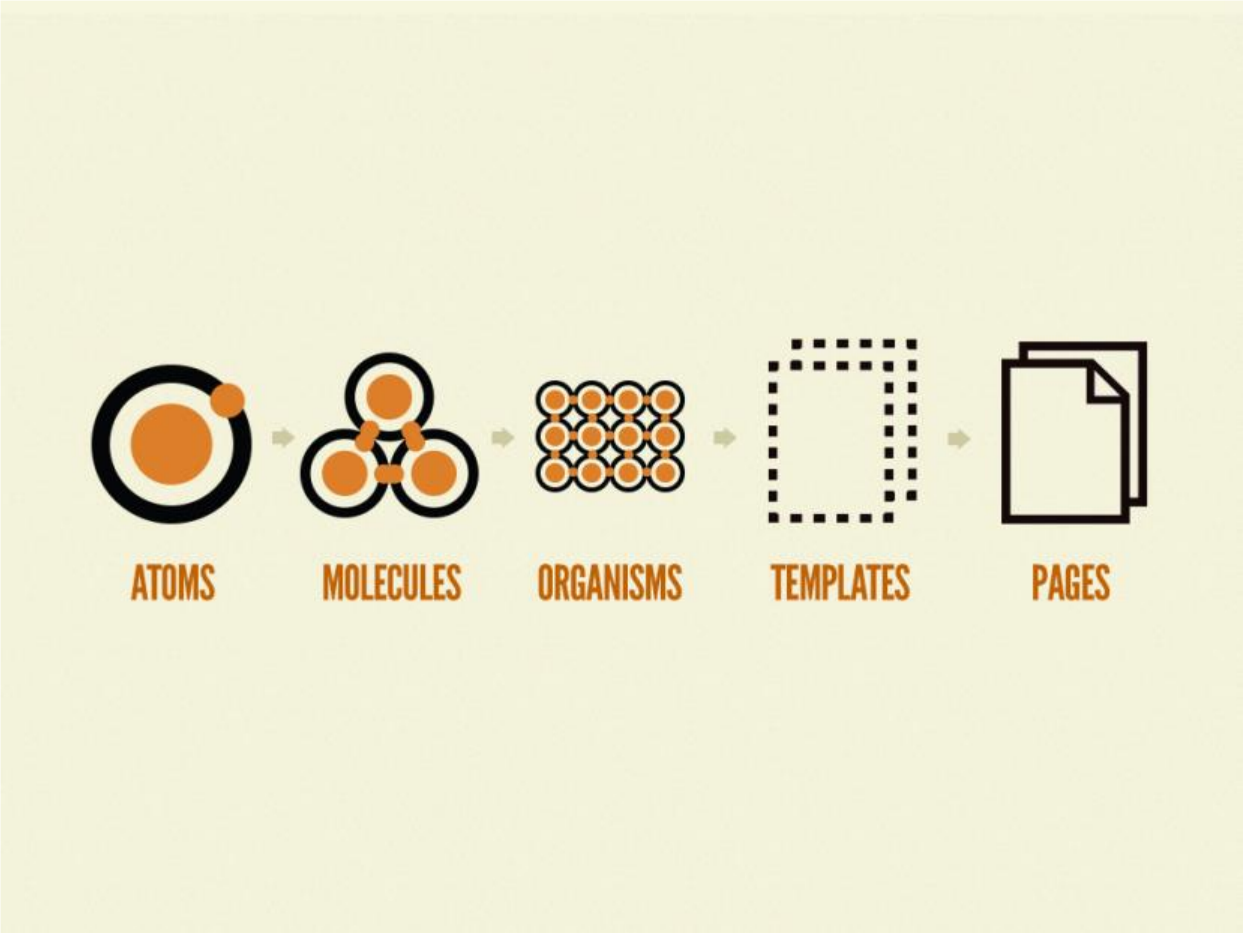
ITCSS is quite flexible in terms of your workflow and tools. It can be combined with any of the leading standard methodologies like BEM, OOCSS, SMACSS etc.

File size and styles duplication

It helps in keeping your file size at the minimal and avoids in style duplication.

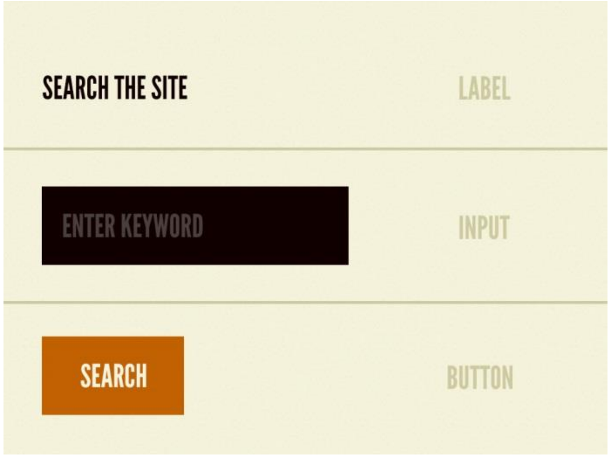
ATOMIC CSS

Atomic design in web design is a methodology which draws a parallel to the real world. The thought is that all matter is comprised of atoms, Atoms bond together to form molecules , which inturn combine to form more complex organisms to ultimately create all matter in the universe.



Simillarly Interfaces are composed of smaller components. This means that the interface can be broken down into fundamental building blocks and work up from there. This is the basic concept behind Atomic design.

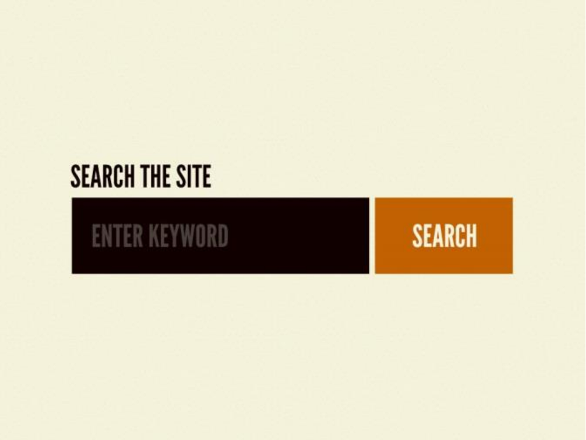
Atoms



Atoms are the basic building blocks of matter. Applied to web interfaces, atoms are our HTML tags, such as a form label, an input or a button.

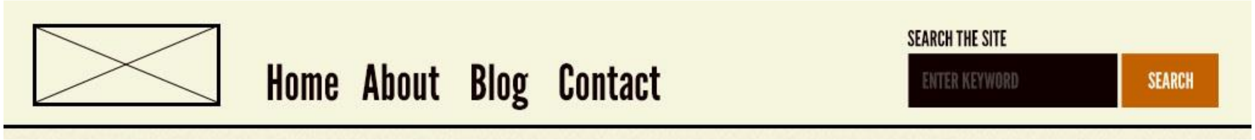
Atoms can also include more abstract elements like color palettes, fonts and even more invisible aspects of an interface like animations.

Molucules



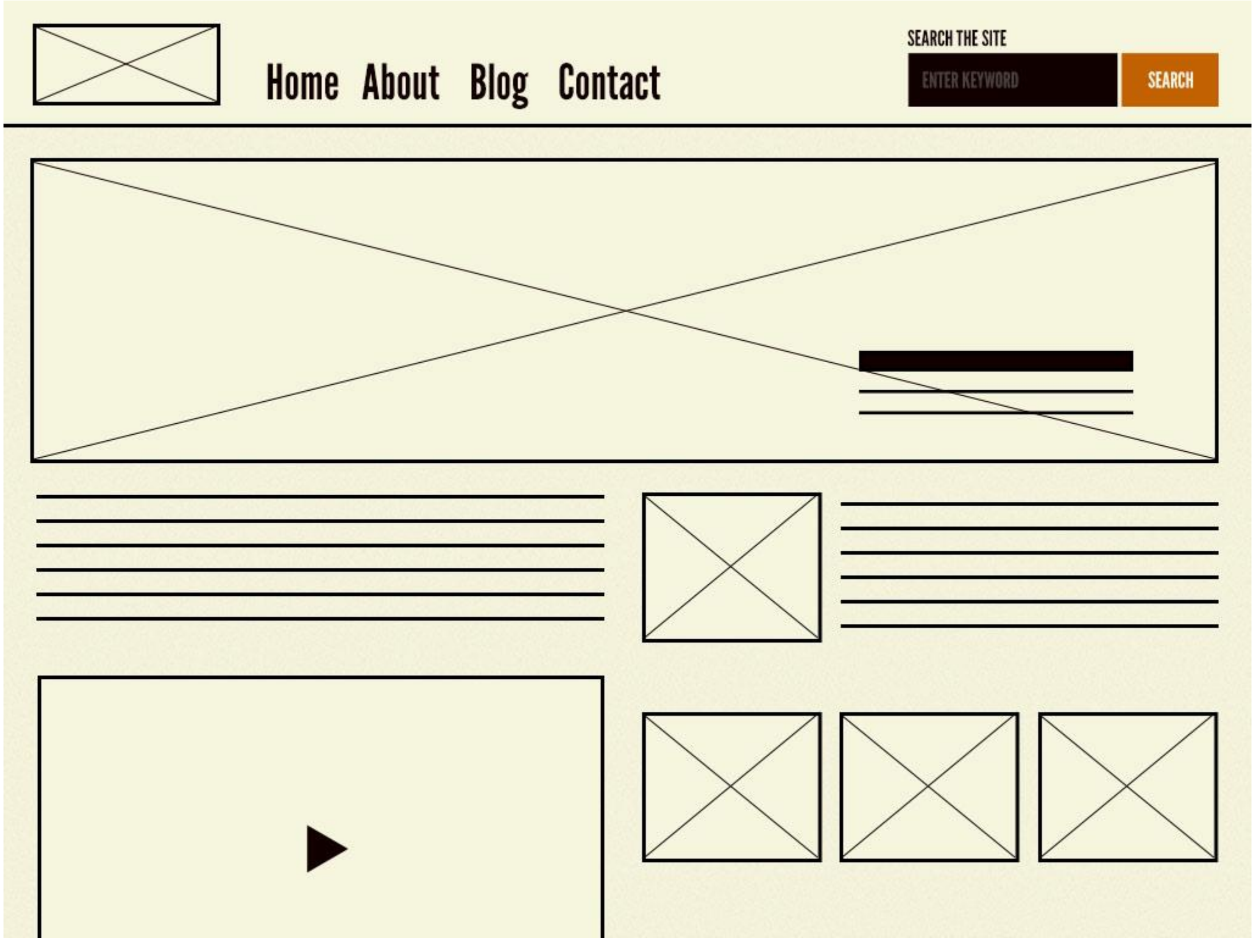
Molecules are groups of atoms bonded together and are the smallest fundamental units of a compound. These molecules take on their own properties and serve as the backbone of our design systems.

Organisms



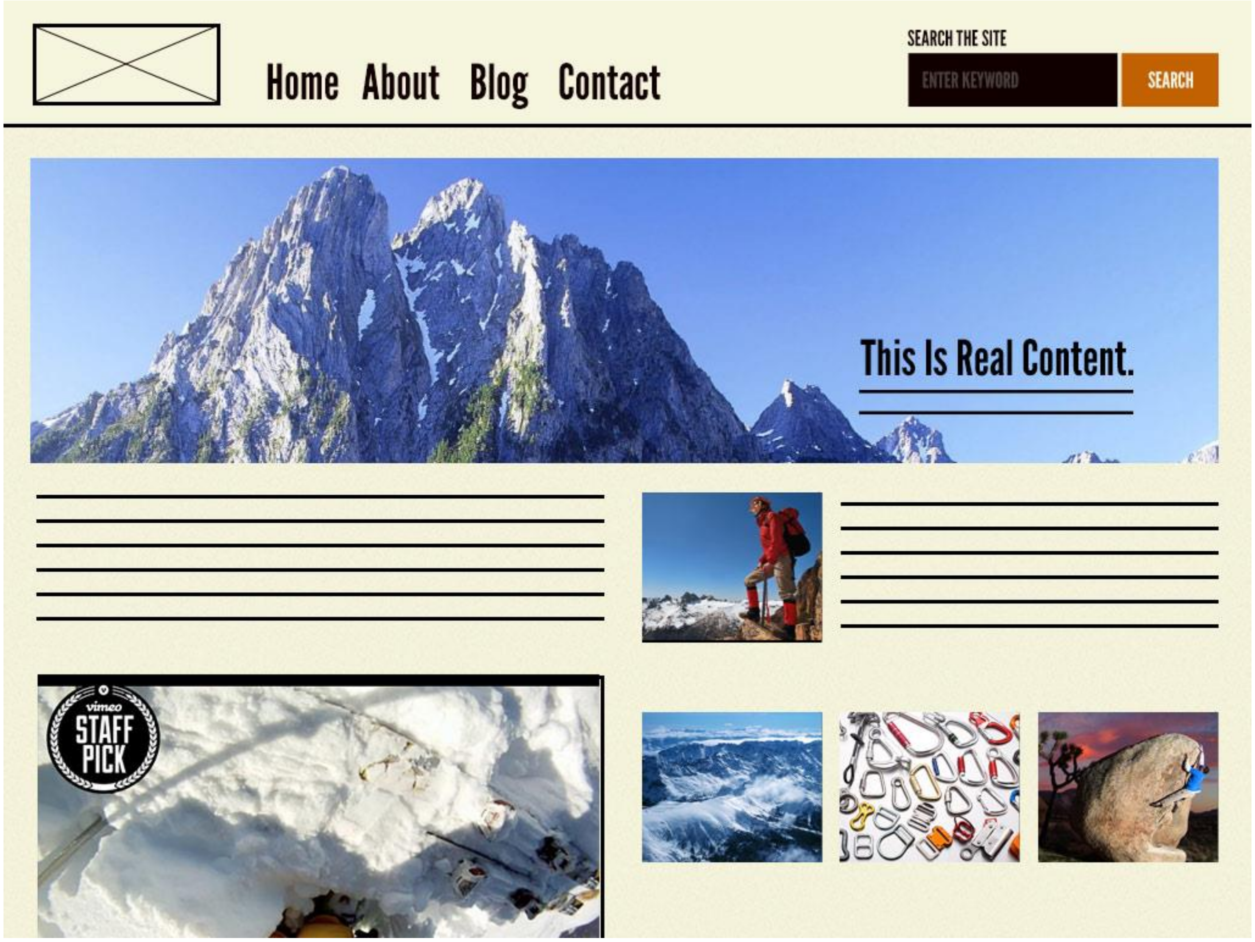
Molecules give us some building blocks to work with, and we can now combine them together to form organisms. Organisms are groups of molecules joined together to form a relatively complex, distinct section of an interface.

Templates



Templates are very concrete and provide context to all these relatively abstract molecules and organisms. Templates are also where clients start seeing the final design in place. In my experience working with this methodology, templates begin their life as HTML wireframes, but over time increase fidelity to ultimately become the final deliverable.

Pages



Pages are specific instances of templates. Here, placeholder content is replaced with real representative content to give an accurate depiction of what a user will ultimately see.

Pages are the highest level of fidelity and because they're the most tangible, it's typically where most people in the process spend most of their time and what most reviews revolve around.

WHY ATOMIC DESIGN?

Atomic design provides a clear methodology for crafting design systems. Clients and team members are able to better appreciate the concept of design systems by actually seeing the steps laid out in front of them.

Atomic design gives us the ability to traverse from abstract to concrete. Because of this, we can create systems that promote consistency and scalability while simultaneously showing things in their final context.

RADICAL CSS

As discussed both methodologies are with its advantages like Flexibility, Reusability and real world concept, by combining both we can have best of both the worlds and create a framework which can be used for large scale application.

Being flexible has its advantage, as we can use BEM to have a more verbose and declarative naming convention.

The proposed framework would include 9 layers as follow.

1. Settings
2. Tools
3. Generic
4. Elements (Similar to Atoms)
5. Modules (Similar to Molecules)
6. Components (Similar to Organisms)
7. Trumps
8. Templates
9. Pages

Settings - This holds any global settings for your project. Examples of global settings might be things like the base font size, colour palettes, config (for example, $environment: dev;) and so on.

Tools – This contains the globally available tooling like mixins and functions. Any mixin that does not need accessing globally would belong in the partial to which it relates.

Generic - The Generic layer is the first one that actually produces any CSS. It houses very high-level, far reaching styles. This layer is seldom modified, and is usually the same across any projects you work on. It contains things like Normalize.css, global box-sizing rules, CSS resets and so on. The Generic layer affects a lot of the DOM, hence it being nice and wide in the Triangle model, and occurring very early on.

Elements – This contains the bare unclassed HTML elements like the H2, LABEL, etc. It being a less generic but still covers a wide range, hence its position. Once the styles are defined the deviations need to be implemented using classes.

Note - All the above layers put together form the abstract framework for creating more detailed and precise pages.

Modules - This is the first layer in which we find class-based selectors. These are concerned with styling non-cosmetic design patterns, or 'objects'. This layer affects less of the DOM than the last layer, has a higher specificity, and is slightly more explicit in that we are now targeting sections of the DOM with classes. This draws similarity to molecules which are formed from Atoms.

Components – Like Organisms which are formed from molecules, Components are based out of modules. Groups of modules put together to form a relatively complex section of the user interface. Going down from here the layers are more specific to the application being developed.

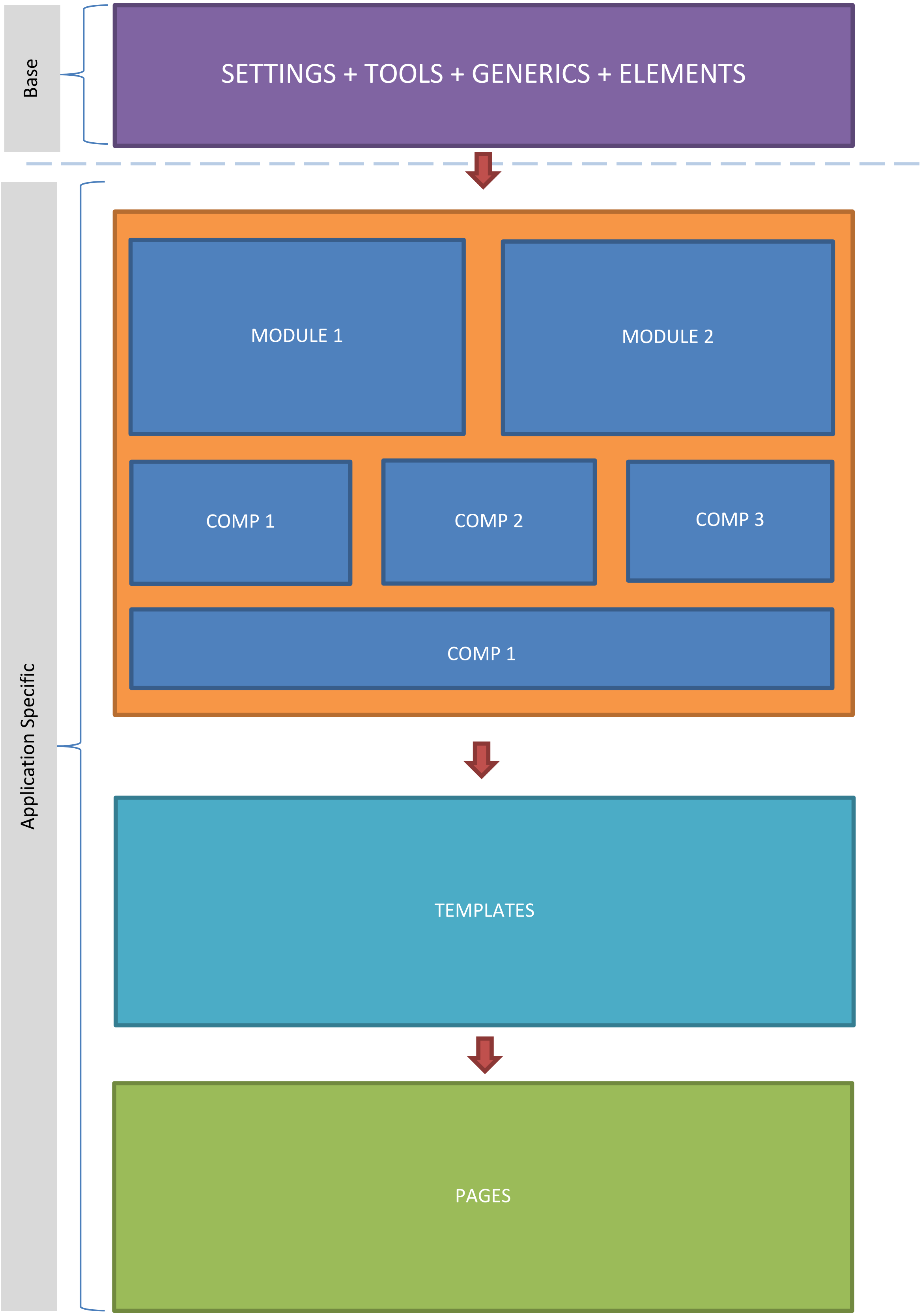
Trumps - This layer beats – or 'trumps' – all other layers, and has the power to override anything at all that has gone before it. It is inelegant and heavy-handed, and contains utility and helper classes, hacks and overrides.

A lot of the declarations in this layer will carry “!important” (e.g. .text-center { text-align: centre !important; }). This is the highest specificity layer – it includes the most explicit types of rule, with the most narrow focus. This layer forms the point of the Triangle.

Templates – These are composed of groups of components put together to form pages. It is where we start seeing the layout or the design being shaped.

Pages – These are formed from various templates put together. The placeholders are replaced with actual images as per the design. Pages are the highest level of fidelity.

The complete CSS framework is shown in the following diagram.


Base
SETTINGS + TOOLS + GENERICS + ELEMENTS
Application Specific
MODULE 1
MODULE 2
COMP 1
COMP 2
COMP 3
COMP 1
TEMPLATES
PAGES

This framework being flexible can be combined with other methodology like BEM and BEM-IT, CSS preprocessors like SASS.

This makes it easier to implement such frameworks as a holistic framework in a large organizations.

When put all together the main scss file should look something like this

```scss
@import "settings.global";
@import "settings.colors";

@import "tools.functions";
@import "tools.mixins";

@import "generic.box-sizing";
@import "generic.normalize";

@import "elements.headings";
@import "elements.links";

@import "modules.wrappers";
@import "modules.grid";

@import "components.site-nav";
@import "components.buttons";
@import "components.carousel";
@import "trumps.clearfix";
@import "trumps.utilities";
@import "trumps.ie8";
```

This ensure that each layers contain css of:

- Similar specificity
- Similar explicitness
- Similar reach

This approach would give us a much more manageable and a holistic CSS architecture.