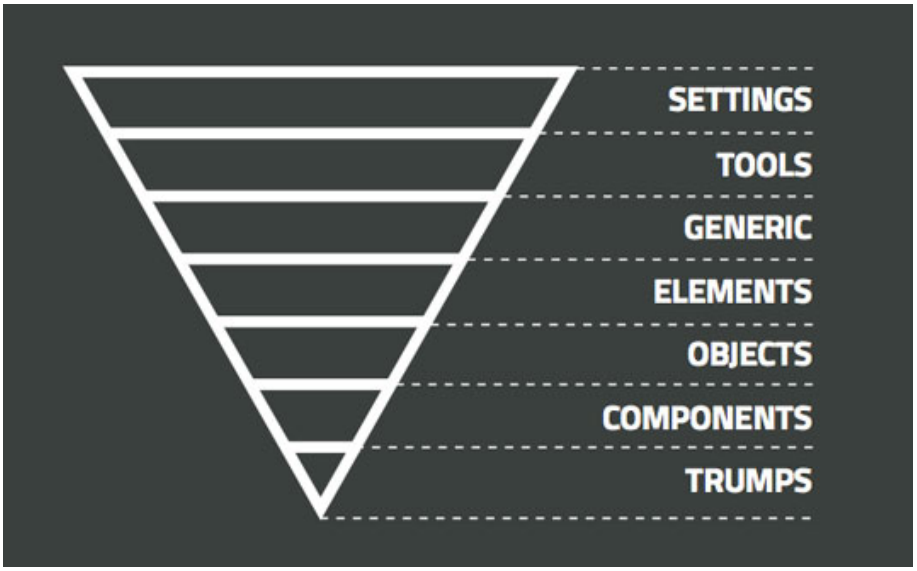
CSS Architecture (ITCSS – Inverted Triangle CSS)

- Layered architecture where specificity slowly increases layer by layer
    - Settings – globally-available variables, and config switches
    - Tools – globally-available tools, mixins, and helper functions
    - Generic – ground-zero styles, low specificity, far-reaching, resets, normalize.css, etc.
    - Elements – unclassed HTML element(s), last layer we see is type selectors, H1-H6, basic links, lists, etc.
    - Objects – OOCSS (Object Oriented CSS), cosmetic-free design patterns, begin using classes exclusively, and agnostically named
    - Components – designed pieces of UI, still only using classes, and more explicitly named
    - Theme – design skin or overall look, brand colors, etc.
    - Test – used to isolate temporary styles for testing
    - Trumps – helpers, overrides, utilities, only affect one piece of the DOM at a time, and usually carry !important
- Remove Settings and Tools layers if not using a preprocessor
- Add Theme and Test layers if needed
- Add brand colors to Settings layer if not using Theme



YouTube - https://www.youtube.com/watch?v=1OKZOV-iLj4

Creative Bloq Article - http://www.creativebloq.com/web-design/manage-large-scale-web-projects-new-css-architecture-itcss-41514731

GitHub - https://github.com/itcss

CSS Structure (combination of inuitcss, OOCSS, BEM)

- Name and separate CSS files based on architecture layers
    - Include table of contents CSS for overall view of what's included in project
    - Multiple files may be necessary for each layer
- Styles will be component-based and portable by removing location dependence (what not where)
- Class names are independent of context
- Use UI component for meaningful name in HTML (how and where)
- This approach uses more classes but will take the complexity out of the CSS and move it to the HTML markup
- Media queries are included with the rules they affect

CSS Formatting & Syntax

- Consistent so it looks and feels familiar
    - Table of contents
        - Provides name of section, files associated with it, and brief summary of what it does (based on CSS architecture)
    - Rules
        - Four (4) space indents, no tabs
        - 80 character wide columns
        - Multi-line CSS
        - Meaningful use of whitespace
    - Titling
        - Begin each new major section of a CSS project with a title
        - Prefixed with hash (#) to allow targeted searches
    - Commenting
        - Anything that isn't immediately obvious from the code alone
    - Naming conventions
        - BEM-like
            - Block – root of component (.person {})
            - Element – delimited by 2 underscores (.person_ _head {})
            - Modifier – delimited by 2 hyphens (.person- -tall {})
        - Selectors
            - Use good intent and make unambiguous (.primary-nav {})
            - Shorter and child selector are better performance (.foo > .bar)
            - No ID's
            - Component-based (.btn {} not .promo a {})
            - Portable (.btn {} not input.btn {})
            - Quasi-qualified (/*ul*/.nav {})
        - UI components
            - Provide a meaningful name alongside an ambiguous class
            - Data-ui-component attribute (class="ui-list" data-ui-component="products")

CSS Guidelines - http://cssguidelin.es/

GitHub - https://github.com/stubbornella/oocss/wiki

Smashing Magazine - http://www.smashingmagazine.com/2011/12/an-introduction-to-object-oriented-css-oocss/

Matt Stauffer - https://mattstauffer.co/blog/organizing-css-oocss-smacss-and-bem

CSS Implementation

- Take stock of all CSS we have (overview of entire project)
- Put classes on everything creating one component layer that is huge
    - Use intuitcss, OOCSS, and BEM methodologies
    - Find bad selectors and put a class on them
    - Remove unnecessary ID's
- Find common traits between components and then refactor
- Move refactored styles to CSS file for the layer they belong in