DRUPAL DEVELOPER DAYS
LISBON 2018

Reusability is overrated, administrability overlooked

Enno Langelotz

Diamond Sponsor

ACQUIA®

Platinum Sponsors

wunder

platform.sh

Gold Sponsors

commerce guys

1x INTERNET

BLOOMIDEA

everis

NUVOLE

Thunder

Amazee Labs

About the speaker

Enno Langelotz

Drupal Frontend Developer since 2010

enno.langelotz@cocomore.com

CO|CO|MO|RE

German IT agency

Offices in Frankfurt, Cologne and Seville

Big on Drupal

www.cocomore.com

What this presentation is about

- Questions > compliance
- Context > fixed rules
- Establish simple rules for structuring your code
- **Concentrate on never losing control over your code base**

Assumptions for a real world project

- You can work with a preprocessor (Sass) and have a workflow setup (Gulp/Grunt)
- We are talking about a traditional Drupal website (not Headless Drupal, which will bring new complexities)
- Requirements will change, timelines are tight, design is not perfect. You need to work fast but stay in control.

Conditions may be harsh

- Design is not 100% ready when you start. Designer is external or otherwise not available for feedback.
- Client is unsure about what he wants. Changes opinions about desired outcome.
- PM has not worked with you before, does not know what you require in frontend
- Backend has completed their work and are unavailable for adaptions or questions

Special things about the frontend developer

- First to compare functionality with intended design
- First to notice usability problems
- First to create the intended visual behaviour

The frontend developer serves as a communication hub for potential show blockers. She/he has a direct impact on how smooth the project will go.

FIRST: Make time now to avoid problems later

	Old-fashioned project plan
Task 1	Create header styles
Task 2	Create footer styles
Task 3	Create styles for content type X
Task 4	Create styles for content type Y

	More proactive approach
Task 1	Check Design for problems/uncertainties
Task 2	Check Drupal HTML for design compatibility
Task 3	Setup basic Sass files and folders for further tasks
Task 4	Theme setup

Talk to the designer BEFORE you start coding

- Is the intended responsive behaviour sufficiently clear?
- What can be solved with CSS? Do you have a clear idea where you need preprocess functions, Twig changes or Javascript?
- Does the design cause problems on small/big screens? Are the edge cases clearly defined?

Make sure you have the time to solve any open points with your designer before you start

Talk to backend BEFORE you start coding

- Is the HTML markup sufficient for your needs?
- Can everything be addressed properly with useful selectors?
- Do we have unexpected behaviour or pages that are not defined by design?

Make sure you have the time to solve any open points with your backend developer before you start

CSS does not require any structure, but your project does

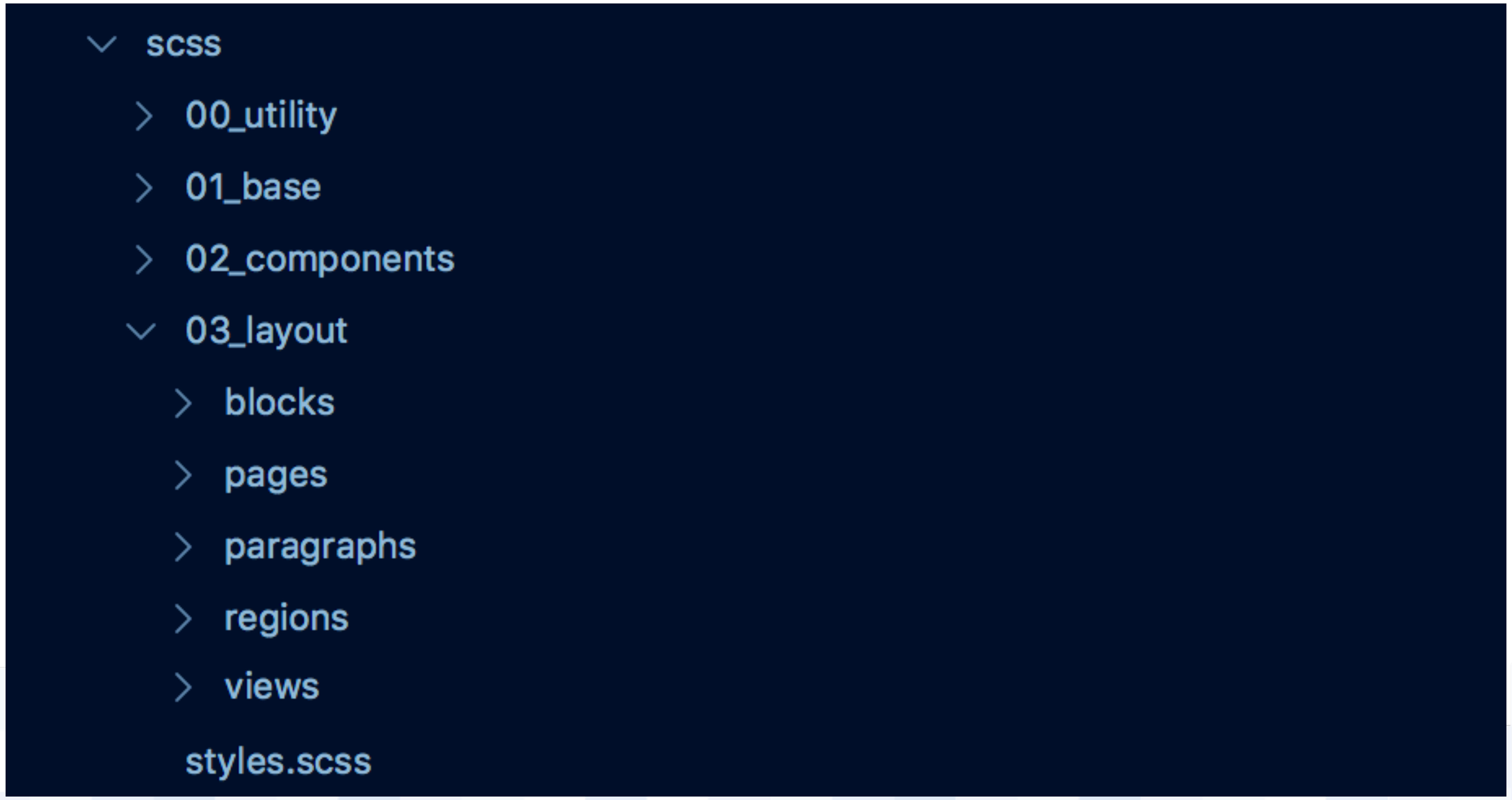
- Have a clear order of tasks and dependencies
- Have a clear idea of how you structure your Sass files and folders
- Have a clear idea about what CSS selectors you are going to use where and how

Goal: Any styles changes at the end of the project are equally as easy to implement as at the beginning

If you don’t know where to start, do this

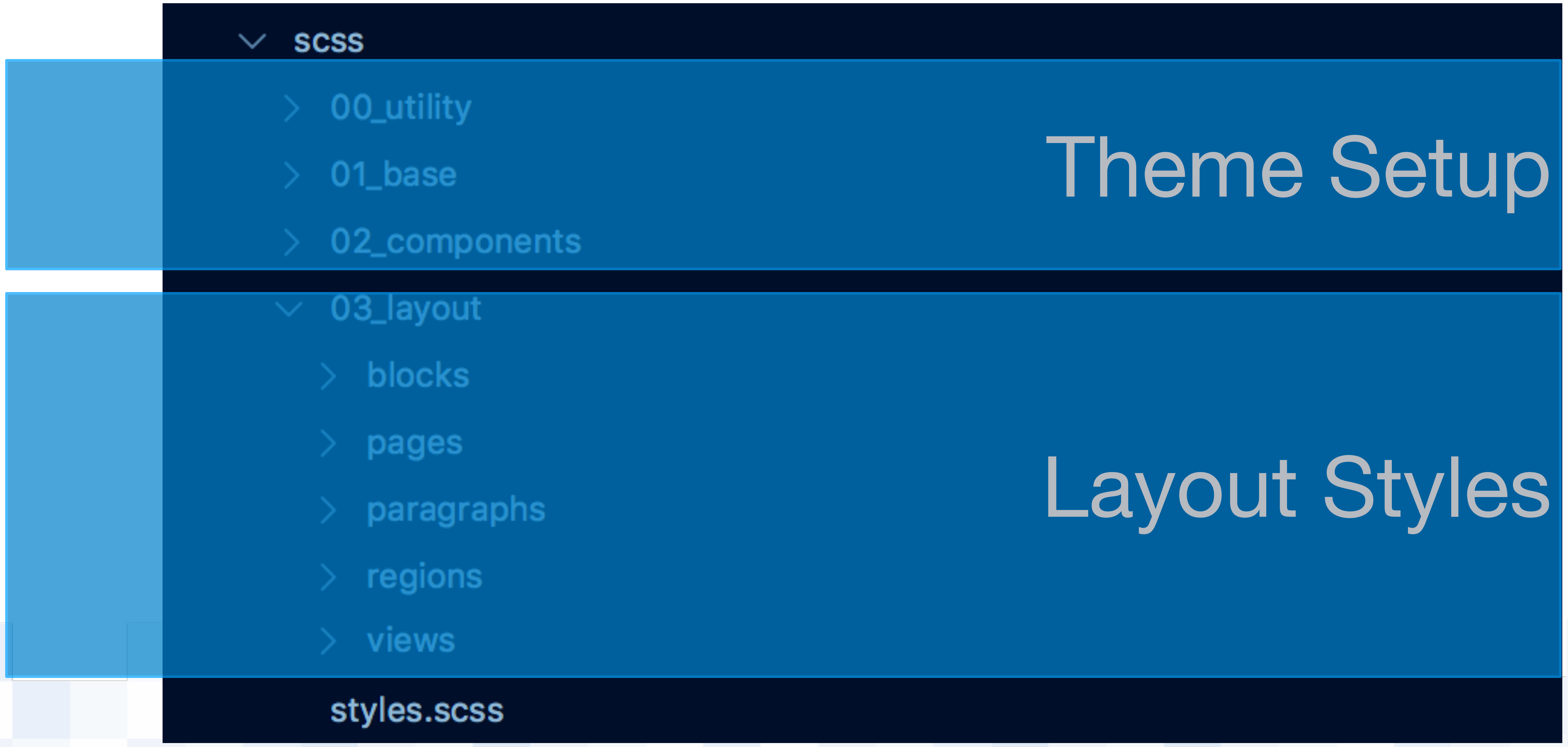
- Use a SMACSS approach
- Use the Drupal HTML markup for structuring your code
- Divide your work into three phases:
  - design and functionality check
  - theme setup
  - specific layout styles

SMACSS for DRUPAL



```
scss
  00_utility
  01_base
  02_components
  03_layout
    blocks
    pages
    paragraphs
    regions
    views
  styles.scss
```

Two phases for the actual development



```
scss
  00_utility
  01_base
  02_components
  03_layout
    blocks
    pages
    paragraphs
    regions
    views
  styles.scss
```

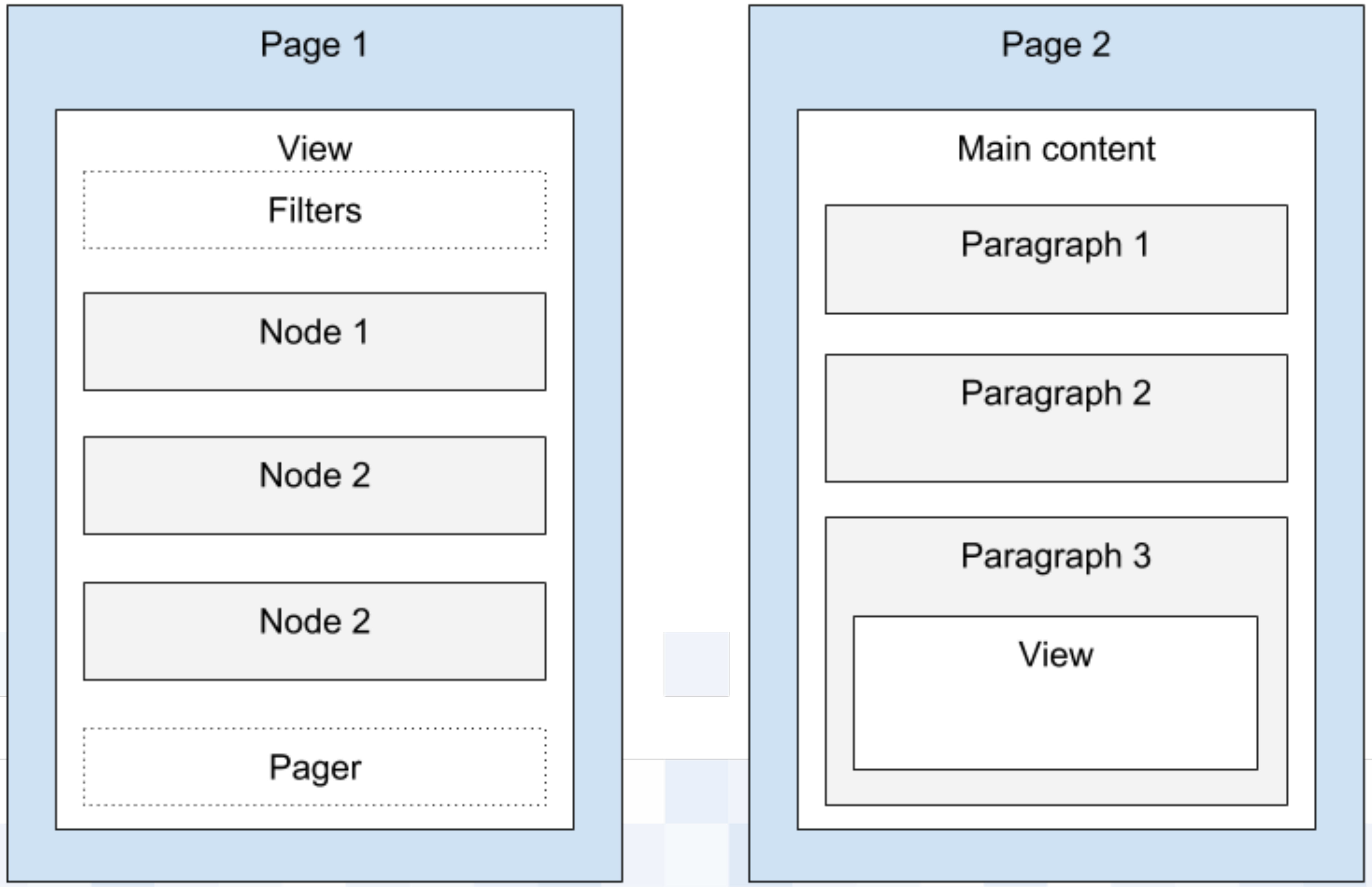
Theme setup

- Grid
- Typography
- Variables
- Basic (!) layout
- Useful Mixins, functions, etc.
- Basic styles for components

An extensive theme setup serves various purposes:

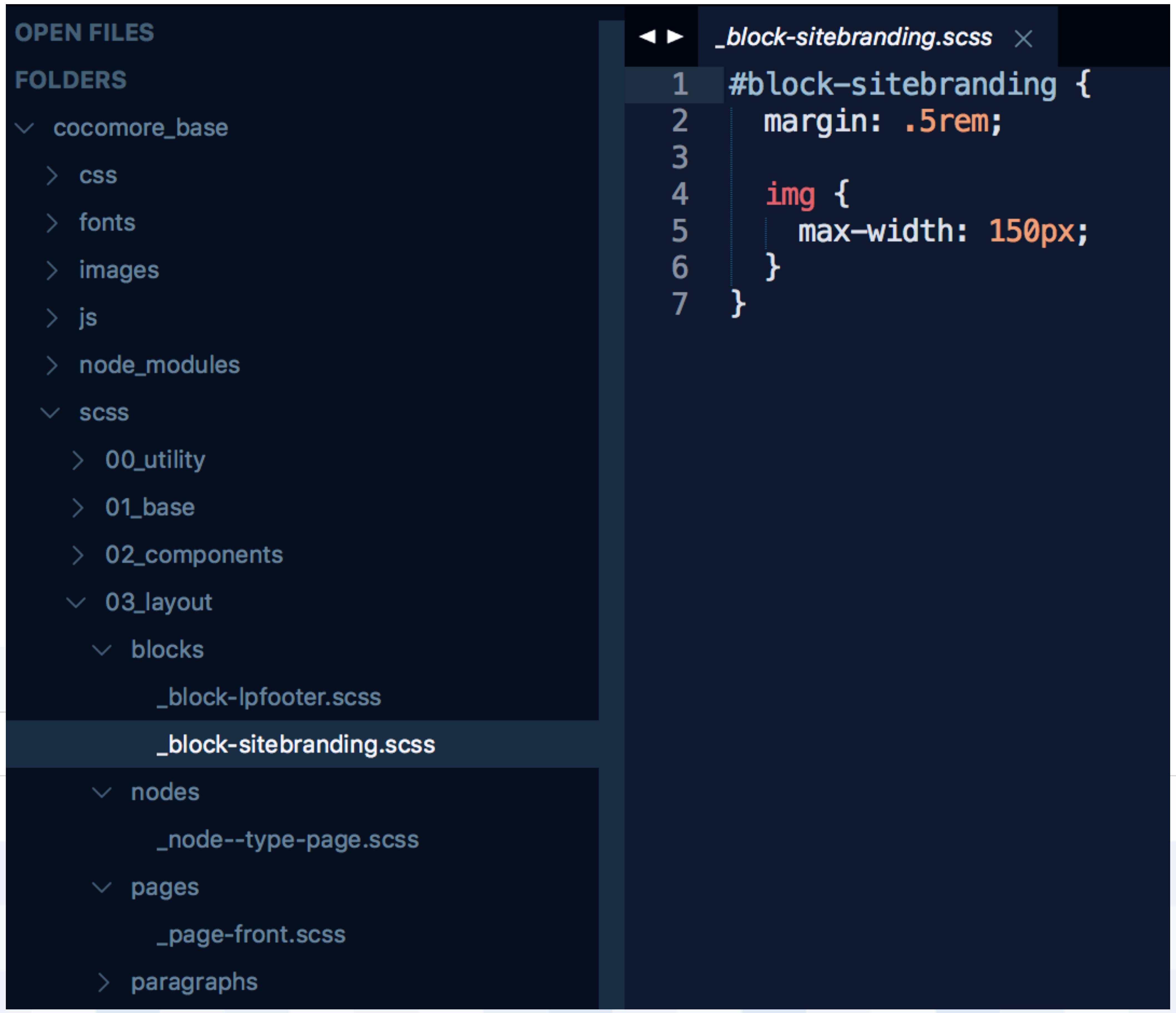
- You will find flaws and inconsistencies that can be corrected in time.
- You will better understand the intended design.

Layout styles - context is everything



- Work from the inside out.
- Add modifications depending on context

Sass structure follows simple rules



```
#block-sitebranding {
  margin: .5rem;

  img {
    max-width: 150px;
  }
}
```

- archive name represents the chosen selector
- folder represents the type
- no exceptions!

An approach that let's you take control

- Whole Sass archives can be deleted according to content or requirement changes
- Relevant Sass code is easily found by anyone by inspecting the Browser and comparing class names with Sass files
- No hidden dependencies

An approach that let’s you work in imperfect conditions

- In theory these simple rules allow you to start working without having the full design and/or understanding of the functionality
- Code can easily be refactored and/or extended
- You don’t need to have reusability of code snippets figured out at the beginning of the project. You implement it on the way.

A few restrictions apply

- Once you have finished work on the theme setup, you are not allowed to touch those folders anymore, unless you have special circumstances and work in coordination with the whole frontend team.
- Do no include selectors outside your file scope defined by its name. This kills the administrability.
- If you include external CSS/Sass libraries, they should let you opt-in on their code. If they bring their own default styles that you didn't ask for, they do more harm than good.

Questions?

Thank you for your attention