

Static Site Generation

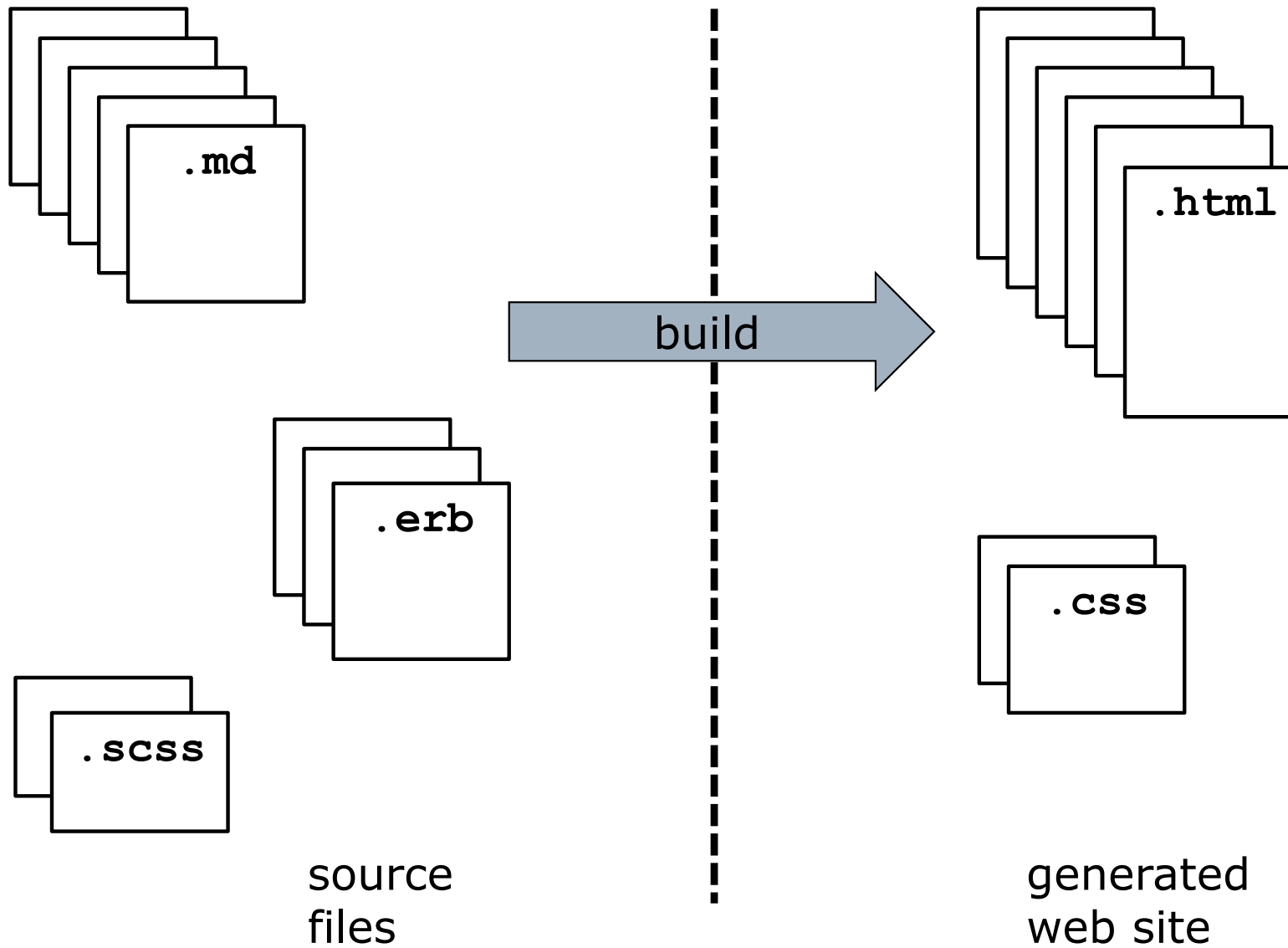
Computer Science and Engineering ■ College of Engineering ■ The Ohio State University

Lecture 21

What is Static Site Generation?

- Use a *program* to produce HTML pages
 - Analogous to compiling programs
 - Translation: source code → machine code
- Development cycle:
 - Write source
 - Compile (aka *build*)
 - Test/inspect result
- Examples of translators
 - Jekyll (used for GitHub Pages, aka github.io)
 - Middleman
 - Lots more, see: staticsitegenerators.net

Picture



Middleman: A Ruby Gem

- Project is a directory (eg myproj)
\$ **middleman** **init** myproj
 - Configuration files, README, Gemfile, etc
- Create source files in **myproj/source**
 - Subdirectories for CSS, images, etc
- Compile all the source files
\$ **bundle exec middleman build**
- Result is placed in **myproj/build**
- Deploy: copy/rsync/ftp contents to server
\$ **rsync -avz --del myproj/build ~/WWW**
- Preview locally (no build needed)
\$ **bundle exec middleman server**

Deployment: GitHub

- GitHub Pages: serves repo as web site
 - URL `https://org.github.io/repo/`
 - Settings > Pages > Build > Source
 - Branch (gh-pages), subdirectory
 - Alternative: GitHub Action
- GitHub Action
 - Responds to an event (eg push on main)
 - Runs the build process
 - Deploys the result
- Use relative links (notice path in URL)
 - `# config.rb`
 - `activate :relative_assets`
 - `set :relative_links, true`
- Helpful: add URL to repo's About

Why Bother?

1. Code reuse and single-point-of-control over change
2. Authoring of content in a language that is more human-friendly
3. Parameterized generation of markup and content

Let's look at each of these benefits in turn...

Motivation #1: Visual Identity

The image displays a collage of overlapping screenshots from the CSE 3901 website, illustrating visual identity through consistent branding and layout. The screenshots show various pages including the course overview, meeting schedule, projects/tasks, submission instructions, and a development environment setup guide. Each screenshot maintains a consistent header with the Ohio State University logo and navigation links, and a consistent footer with contact information. The text is clear and legible, and the overall design is professional and user-friendly.

THE OHIO STATE UNIVERSITY Map Buckeye Link Webmail Search Ohio State

Dept. of Computer Science & Engineering
CSE 3901: Web Applications (Autumn 2024)

Home Meetings Labs Dev Environment Resources Syllabus

Overview

Number	CSE 3901
Title	Project: Web Application Design, Development, and Deployment
Instructor	Prof. Paul Sivilotti
Credits	U 4
Grading	Group projects and individual exams. To pass the course, a minimum grade of C- (midterms and final) is required. See the syllabus for more details.

THE OHIO STATE UNIVERSITY Map Buckeye Link Webmail Search Ohio State

Dept. of Computer Science & Engineering
CSE 3901: Web Applications (Autumn 2024)

Home Meetings Labs Dev Environment Resources Syllabus

Class Meeting Schedule

Note: Information that appears in this font, is subject to change before its official posting.

Meeting	Day	Date	Topic
1	W	Aug 21	Architectural
2	F	Aug 23	Git Version Control
3	M	Aug 26	Git Distribution
4	W	Aug 28	Git Extensions

THE OHIO STATE UNIVERSITY Map Buckeye Link Webmail Search Ohio State

Dept. of Computer Science & Engineering
CSE 3901: Web Applications (Autumn 2024)

Home Meetings Labs Dev Environment Resources Syllabus

Projects, Tasks, and Submission

Note: Information that appears in this font, is subject to change before its official posting.

Submission

Projects and tasks must be turned in by the following deadlines:

- a group submission of the code on GitHub
- an individual peer evaluation submitted by the deadline

Each group member must submit their own code.

To submit the code, create a tag called "submission" and push it to the origin.

```
$ git tag -a submission -m "complete"
$ git push origin submission
```

THE OHIO STATE UNIVERSITY Map Buckeye Link Webmail Search Ohio State

Dept. of Computer Science & Engineering
CSE 3901: Web Applications (Autumn 2024)

Home Meetings Labs Dev Environment Resources Syllabus

Setting up a Development Environment

Last updated: Aug 5, 2024

Overview

There are two ways to set up a development environment for CSE 3901:

- Create a [virtual machine](#) running locally, or
- Install the tools natively on your machine's OS:
 - [Windows](#)
 - [MacOS](#)
 - Linux (follow the instructions for option 1 above, a [virtual machine](#), but starting with Step 3, "Customize the Ubuntu Installation")

The first way is fully supported, but requires a more powerful (memory, processor) machine. The second way produces a more responsive environment and overall better development experience, but is not directly supported by the instructional staff.

THE OHIO STATE UNIVERSITY
Dept. of Computer Science & Engineering
395 Dreese Labs
2015 Neil Avenue

If you have a disability and experience difficulty accessing this page, please contact the Digital Accessibility Center for assistance at accessibility@osu.edu or 614.292.4242.

Motivation #1: Visual Identity

- Common headers & footers
 - Example: OSU web sites share nav bar
 - Example: course web site
- Duplication of code is evil
 - Corollary: cut-and-paste is evil
 - Destroys single-point-of-control over change
- Solution:
 - Put common HTML in one file (a *partial*)
 - Every document includes that file

ERb: Embedded Ruby

- General templating mechanism
 - “Template” = a string (usually contents of some file)
 - Contains (escaped) bits of ruby
 - `<% code %>` execute ruby code (“scriptlet”)
 - `<%= expr %>` replace with result of ruby expr
 - `<%# text %>` ignore (a comment)
- Example: a text file

```
This is some text.  
<% 5.times do %>  
Current Time is <%= Time.now %>!  
<% end %>
```
- Process using erb tool to generate result

```
$ erb example.txt.erb > example.txt
```
- Naming convention: *filename.outputlang.erb*
 - Example `index.html.erb`
- Many alternatives, eg HAML

Generation of Site

- Source files in myproj/source

```
$ ls source
```

```
index.html.erb  syll.html.erb
```

```
meet.html.erb
```

- Compile

```
$ bundle exec middleman build
```

- Result after building

```
$ ls build
```

```
index.html  meet.html  syll.html
```

Partials

- A document *fragment* included in other documents
- Include in template with `partial` function

```
<body>  
  <%= partial "navigation" %>  
  ...  
  <%= partial "footer" %>  
</body>
```

- Partial's *filename* begins with '_'

- ie `_navigation.erb`

```
<div class="navbar">  
  <ul id="site-nav"> <li> ... </li> </ul>  
</div>
```

- Note: '_' omitted in argument to function

Generation of Site with Partials

- Source files in myproj/source

```
$ ls source
```

```
_footer.erb          meet.html.erb
```

```
_navigation.erb     syll.html.erb
```

```
index.html.erb
```

- Compile

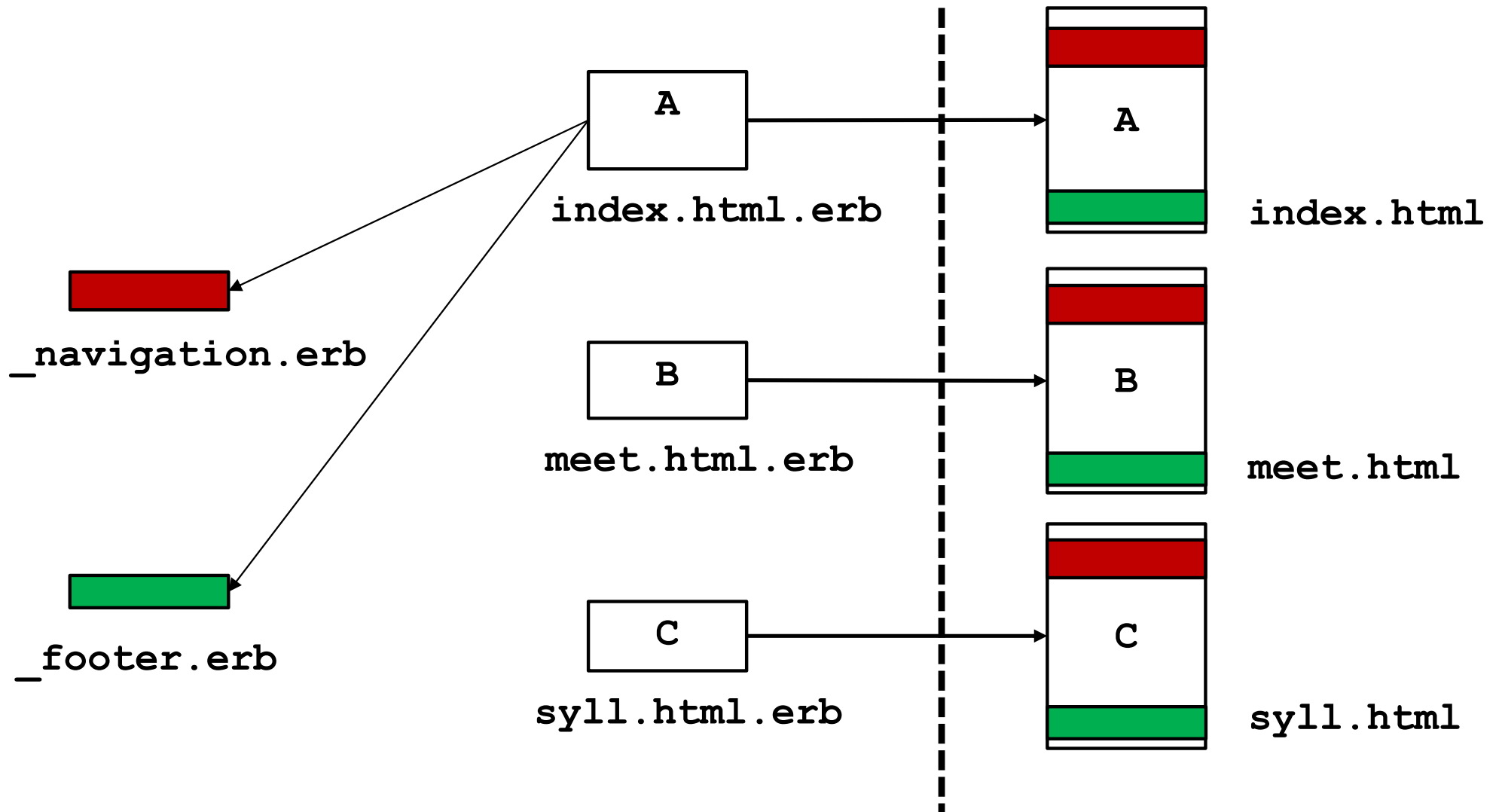
```
$ bundle exec middleman build
```

- Result after building

```
$ ls build
```

```
index.html meet.html syll.html
```

Site Generation With Partial



Tricks with Partials

□ Content of partial can be customized with arguments in call

□ In call: pass a hash called `:locals`

```
<%= partial "banner",  
    locals: { name: "Syllabus",  
             amount: 34 } %>
```

□ In partial: access hash with *variables*

```
<h3> <%= name %> </h3>
```

```
<p> Costs <%= "#{amount}.00" %></p>
```

Problem

- How to guarantee every page includes partial(s)
 - Partials don't ensure one page *structure* across the site
- Every page should look like:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Class Meetings</title>
    <link rel="stylesheet" type="text/css"
      href="osu_style.css">
  </head>
  <body>
    <%= partial "navigation" %>
    ... <!-- different for each page -->
    <%= partial "footer" %>
  </body>
</html>
```

Solution: Layouts

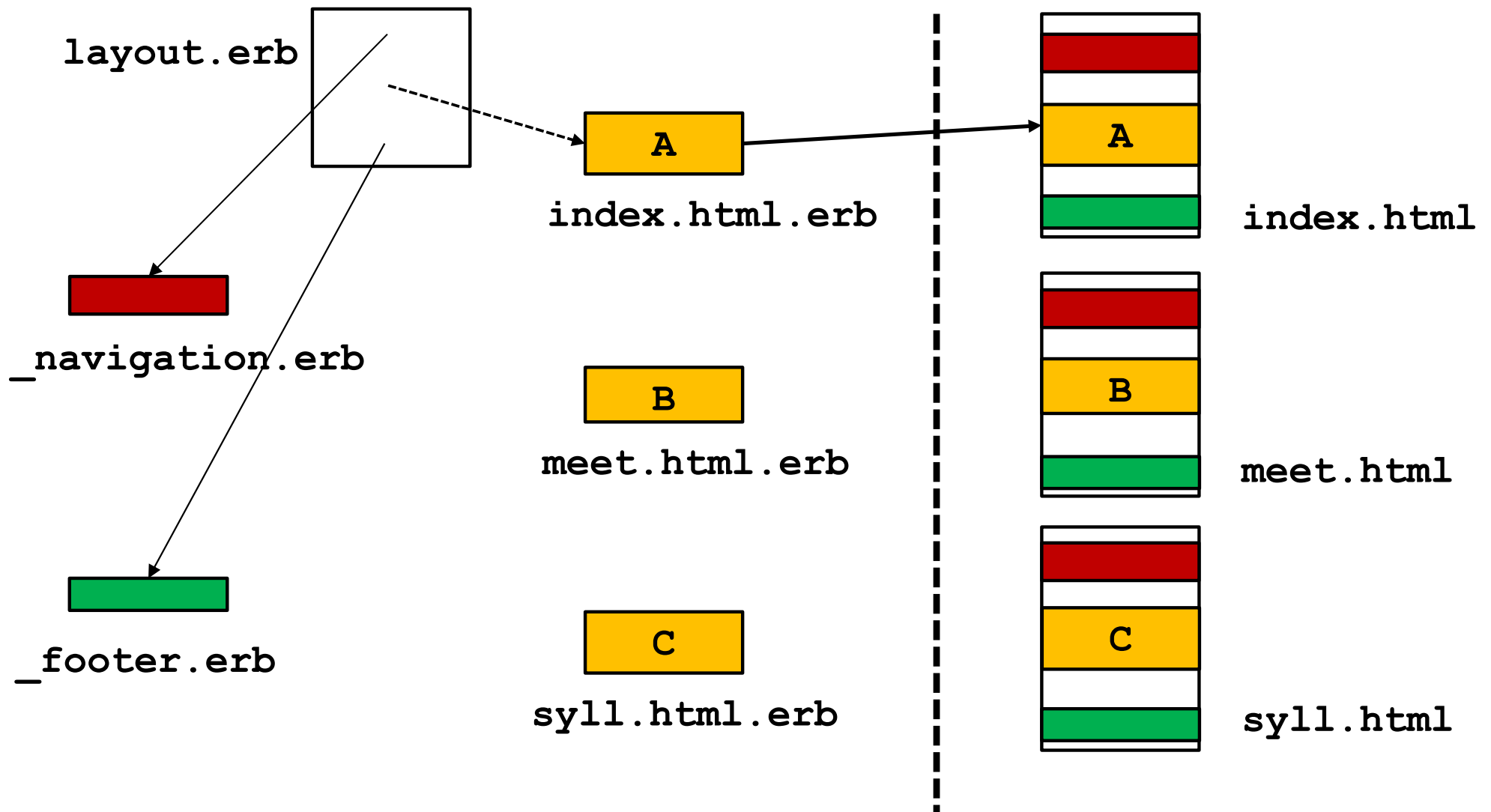
- HTML formed from: **Layout** + **Template**
 - Layout is the common structure of HTML pages
 - Layout uses `yield` to include (page-specific) template

- File: **layout.erb**

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title> ... etc
  </head>
  <body>
    <%= partial "navigation" %>
    <%= yield %>
    <%= partial "footer" %>
  </body>
</html>
```

- Layout is where you put site-wide styling
 - e.g., navigation bar, div's with CSS classes, footers

Site Generation With Layouts



Generation of Site with Layouts

- Default layout in `source/layouts/layout.erb`

```
$ ls -F source  
index.html.erb meet.html.erb  
layouts/          syll.html.erb  
$ ls source/layouts  
_footer.erb _navigation.erb layout.erb
```
- Result after building

```
$ ls build  
index.html meet.html syll.html
```

Page-Specific Data in Layout

- Some layout content is page-specific
 - Example: `<title>` in document's head
- Solution: Ruby variable `current_page`
 - Example: `current_page.path`
- Template contains "frontmatter" that sets the value of `current_page.data`
 - In template (`meet.html.erb`)

```
---  
title: "Class Meetings"  
---
```
 - In layout (`layout.erb`)

```
<title> <%= current_page.data.title %>  
</title>
```

Example: Navbar Highlights

The image displays a sequence of overlapping screenshots of a web page for 'CSE 3901: Web Applications (Autumn 2024)'. Each screenshot highlights a different navigation link in the top menu with a red circle:

- Home**: The first screenshot shows the 'Home' link highlighted.
- Meetings**: The second screenshot shows the 'Meetings' link highlighted.
- Labs**: The third screenshot shows the 'Labs' link highlighted.
- Dev Environment**: The fourth screenshot shows the 'Dev Environment' link highlighted.
- Resources**: The fifth screenshot shows the 'Resources' link highlighted.

The page content includes a table with the following information:

Number	CSE 3901
Title	Project: Web Application Design, Development, and Deployment
Instructor	Prof. Paul Sivilotti
Credits	U 4
Grading	Group projects and individual exams. To pass the course, a minimum grade of C- (midterms and final) is required. See the syllabus for more details.

The page also features a 'Class Meeting Schedule' table:

Meeting	Day	Date	Topic
1	W	Aug 21	Architectural
2	F	Aug 23	Git Version Control
3	M	Aug 26	Git Distribution
4	W	Aug 28	Git Extensions

Additional sections include 'Projects, Tasks, and Submission' and 'Setting up a Development Environment'.

Why Bother?

1. Code reuse and single-point-of-control over change
2. Authoring of content in a language that is more human-friendly
3. Parameterized generation of markup and content

Let's look at each of these benefits in turn...

Motivation #2: Improved Syntax

- HTML tags make content hard to read
 - `<p>`, `<h2>`, ``, `` etc
 - vs plain text, which is easier to read
- Common plain text conventions:
 - Blank lines between paragraphs
 - Underline titles with `-`'s or `=`'s
 - Emphasize `*words*`, `_words_`, `**words**`
 - Links as `[text](url)`
 - Unordered lists with bullets using `*` or `-`
 - Numbered lists with `1.`, `2.`, `3.`

Why Middleman?

The last few years have seen an explosion in the amount and variety of tools developers can use to build web applications. Ruby on Rails selects a handful of these tools:

- [Sass](#) for DRY stylesheets
- [CoffeeScript](#) for safer and less verbose javascript
- Multiple asset management solutions, including [Sprockets](#)
- [ERb](#) & [Haml](#) for dynamic pages and simplified HTML syntax

Middleman gives the stand-alone developer access to all these tools and many, many more. Why would you use a

Why Middleman?

The last few years have seen an explosion in the amount and variety of tools developers can use to build web applications. Ruby on Rails selects a handful of these tools:

- [Sass](http://sass-lang.com/) for DRY stylesheets
- [CoffeeScript](http://coffeescript.org/) for safer and less verbose javascript
- Multiple asset management solutions, including [Sprockets](https://github.com/sstephenson/sprockets)
- [ERb](http://ruby-doc.org/stdlib-2.0.0/libdoc/erb/rdoc/ERB.html) & [Haml](http://haml.info/) for dynamic pages and simplified HTML syntax

Middleman gives the stand-alone developer...

Why Middleman?

The last few years have seen an explosion in the amount and variety of tools developers can use to build web applications. Ruby on Rails selects a handful of these tools:

- * **[Sass]** (<http://sass-lang.com/>) for DRY stylesheets
 - * **[CoffeeScript]** (<http://coffeescript.org/>) for safer and less verbose javascript
 - * Multiple asset management solutions, including **[Sprockets]** (<https://github.com/sstephenson/sprockets>)
 - * **[ERb]** (<http://ruby-doc.org/stdlib-2.0.0/libdoc/erb/rdoc/ERB.html>) & **[Haml]** (<http://haml.info/>) for dynamic pages and simplified HTML syntax
- **Middleman**** gives the stand-alone developer...

Markdown

- Formalizes these ASCII conventions
 - Filename extension: `.md`
 - Adds some less familiar ones (eg ```)
- Translator generates HTML from markdown
 - Examples: GitHub readme's, user-posted comments on web boards (StackOverflow)
 - Other target languages possible too
- See Middleman's README.md
 - [Regular](#) view
 - [Raw](#) view
- Warning: many Markdown dialects/engines
 - daringfireball.net (original, 2004, stale)
 - Common Mark, GitHub-flavored markdown (GFM), Markdown Extra
 - kramdown, rdiscount, redcarpet, ...

CSS: Magic Numbers

- Literals are common in CSS

```
h1 { background-color: #ff14a6; }  
h2 { color: #ff14a6; }
```

- Result: Lack of single-point-of-control
- Solution: SASS allows variables

```
$primary: #ff14a6;  
h1 { background-color: $primary; }  
h2 { color: $primary; }
```

- Translator generates CSS from SASS
- Note: CSS has something similar (custom properties)

CSS: Repeated Ancestors

- CSS requires separate rules for different elements with same ancestor

```
.navbar ul { ... }
```

```
.navbar li { ... }
```

- Changing classname requires changing all these rules

- Solution: SASS allows nested selectors

```
.navbar {  
  ul { ... }  
  li { ... }  
}
```

Why Bother?

1. Code reuse and single-point-of-control over change
2. Authoring of content in a language that is more human-friendly
3. **Parameterized generation of markup and content**

Let's look at each of these benefits in turn...

Motiv'n #3: Content Generation

- Problem: Parameterized/repeated content
 - Example: Course offering term
- Solution: Read content from data
 - Files in subdirectory data/ define variables
 - # *data/dates.yml*
 - term*: "Autumn 2024"
 - Variables then available in templates
 - <%= *data.dates.term* %>
- Problem: Repeated structure
 - Example: Each row in table
- Solution: Generate structure with code
 - Iterate over array, creating table rows
 - See course web site
 - <% *meetings.each* do |meet| %>
 - <tr> <td> <%= *meet.date* %> </td>...

Generating Random Content

- Want placeholder content for prototype
 - Useful for making style/layout decisions
 - Don't care about actual content
- Solution: use a *method* that returns an HTML string

```
<%= lorem.sentence %>
```

- Many lorem methods available

```
lorem.paragraphs 2
```

```
lorem.date
```

```
lorem.last_name
```

```
lorem.image('300x400')
```

```
#=> http://placeholder.it/300x400
```

Helper Functions

- Used to generate common HTML snippets

- Example: hyperlinks

```
<a href="/about.html">About us</a>
```

- With `link_to` helper in template:

```
<%= link_to('About us', '/about.html') %>
```

```
#=> <a href="/about.html">About us</a>
```

- Many optional arguments

```
<%= link_to('My Blog', '/blog.html',
```

```
      class: 'happy') %>
```

```
#=> <a href="/blog.html" class="happy">
```

```
  My Blog</a>
```


(Many) More Helper Functions

□ Format helpers

```
pluralize 2, 'person' #=> "2 people"
```

□ Tag helpers

```
tag :img, src: '/kittens/png'  
content_tag :p, class: 'warning' do ... end
```

□ Form helpers

```
form_tag '/login', method: 'post'  
button_tag 'cancel', class: 'clear'
```

□ Asset helpers

```
stylesheet_link_tag 'all'  
javascript_include_tag 'jquery'  
favicon_tag 'images/favicon.png'  
image_tag 'padrino.png',  
  width: '35', class: 'logo'
```

Summary

- ERb
 - Template for generating HTML
 - Scriplets and expressions
- Reuse of views with partials
 - Included with partial (eg `<%= partial...`)
 - Filename is prepended with underscore
 - Parameter passing from parent template
- Layouts and templates
- Markdown, SASS
- Content generation and helpers