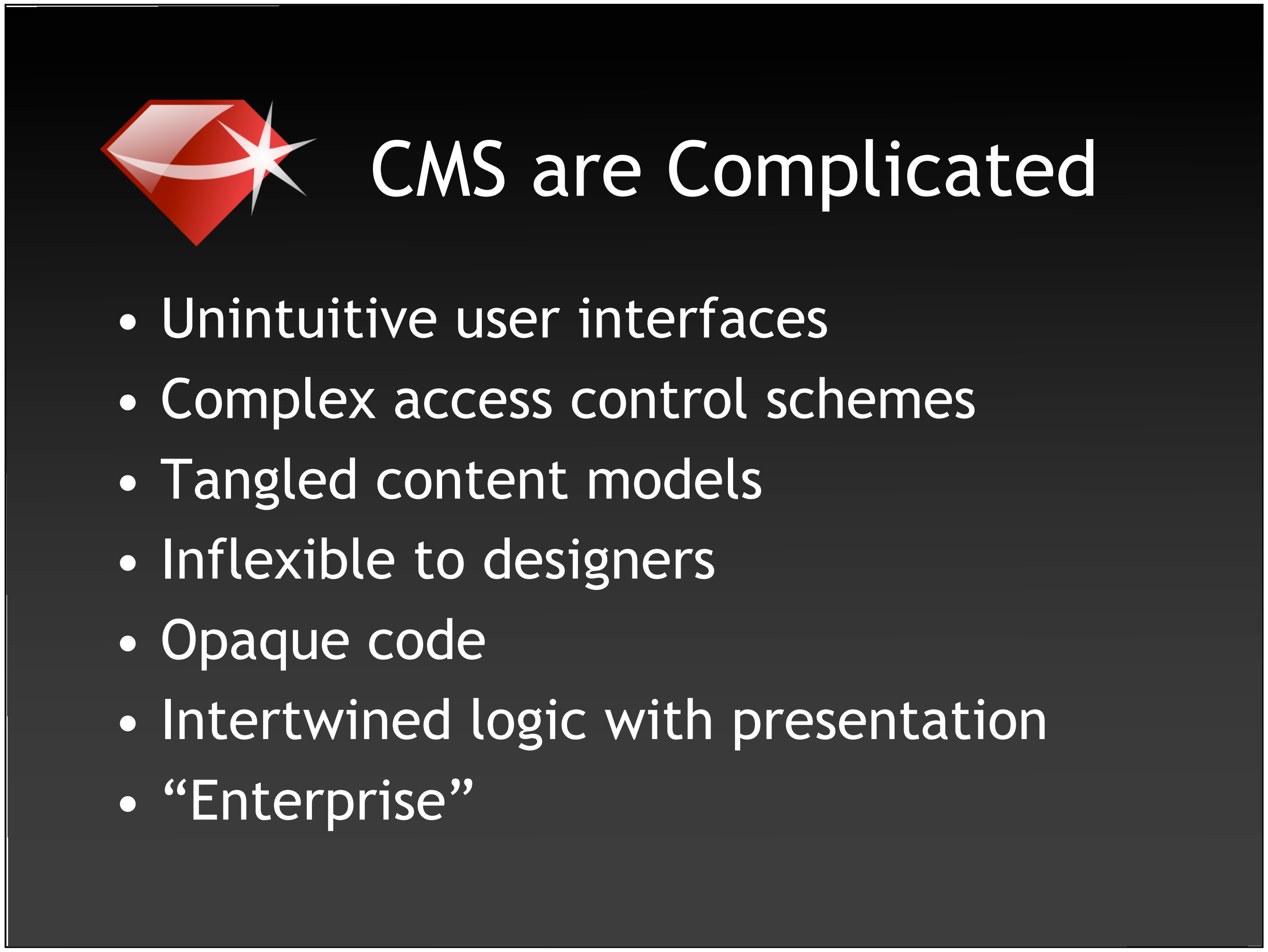
Pragmatic Content Management with Radiant

Sean Cribbs

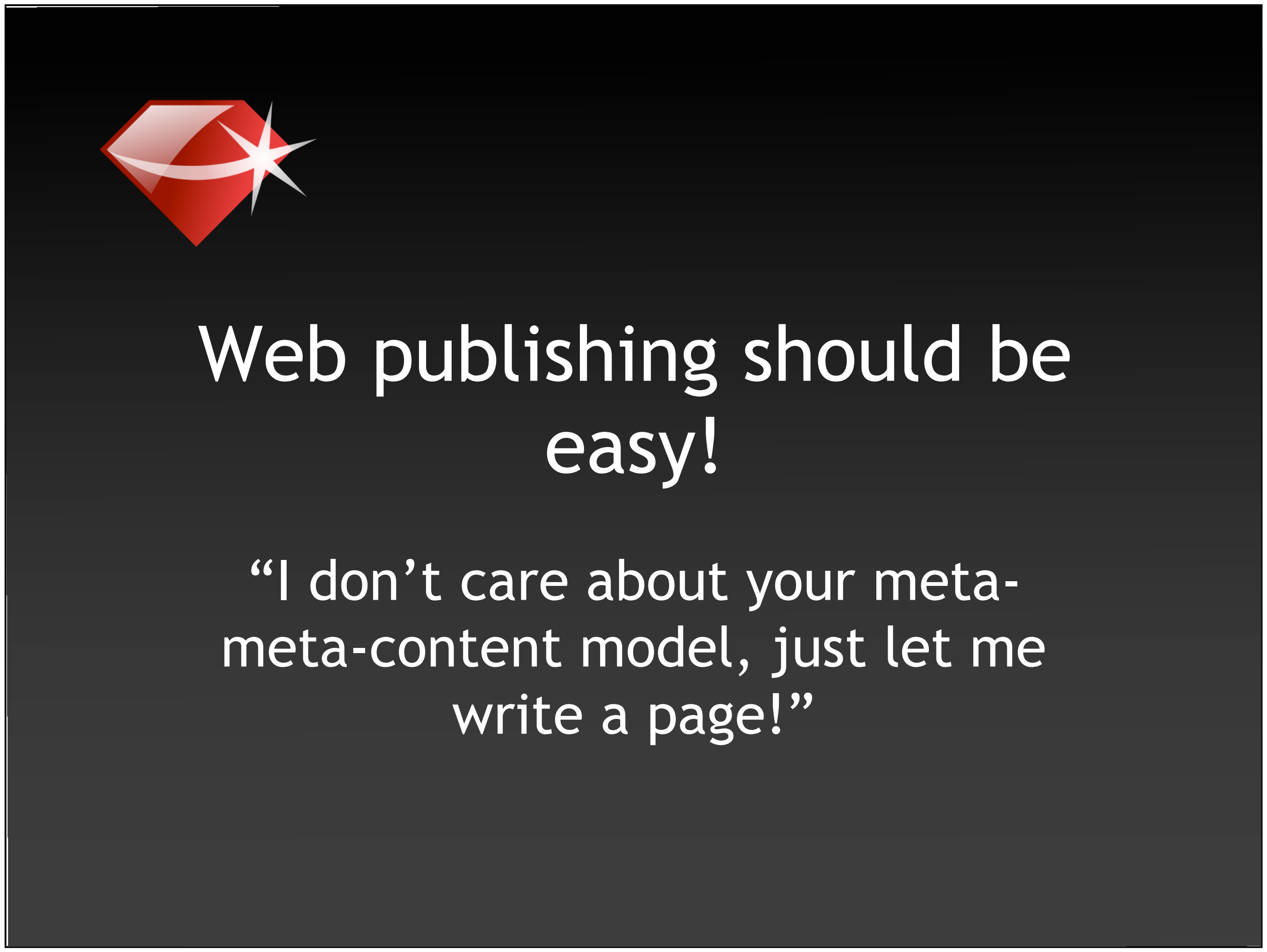
Lead Developer

Thanks to John Long, Author and Lead Designer, for some of these slides



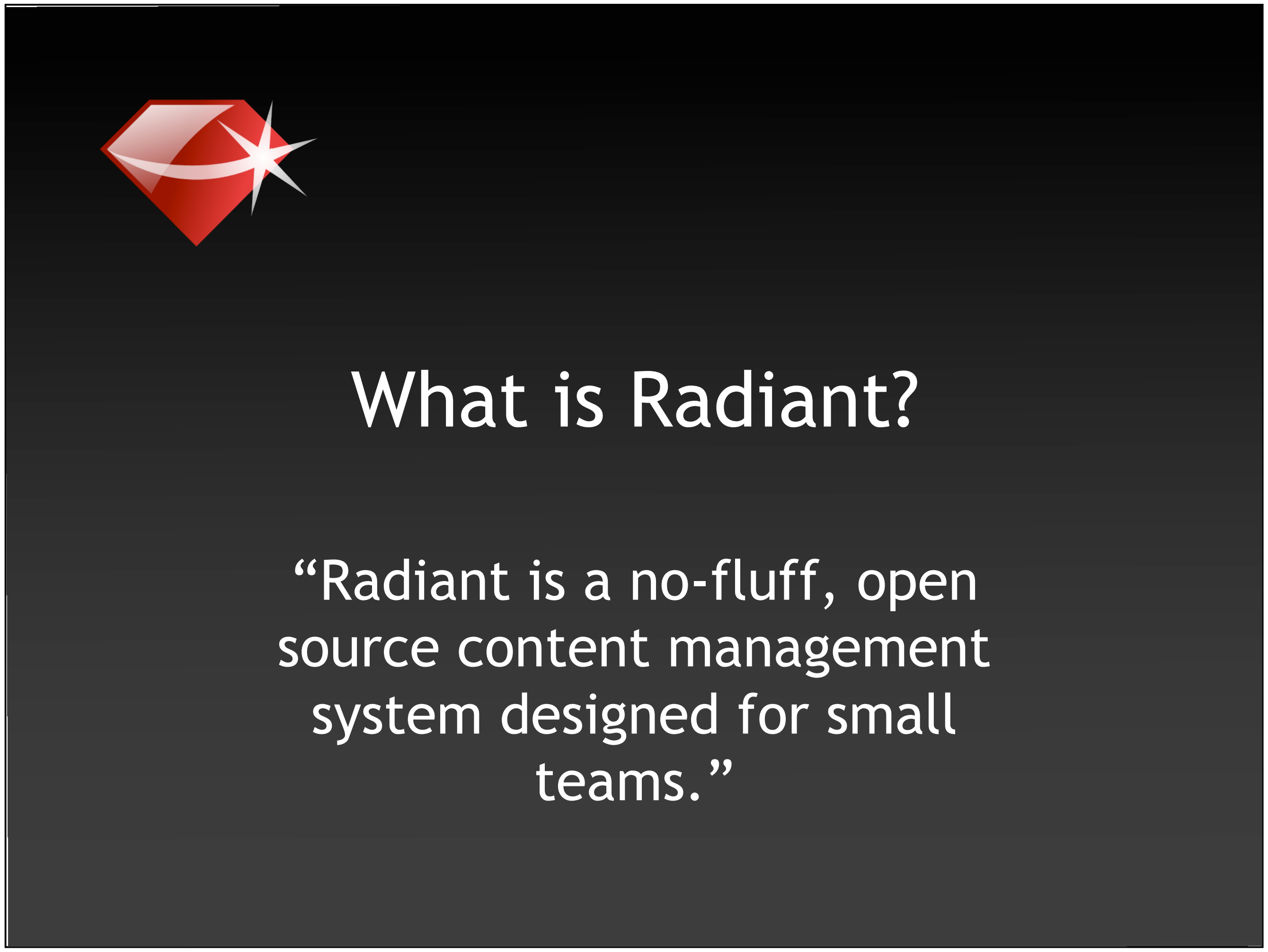
CMS are Complicated

- Unintuitive user interfaces
- Complex access control schemes
- Tangled content models
- Inflexible to designers
- Opaque code
- Intertwined logic with presentation
- “Enterprise”



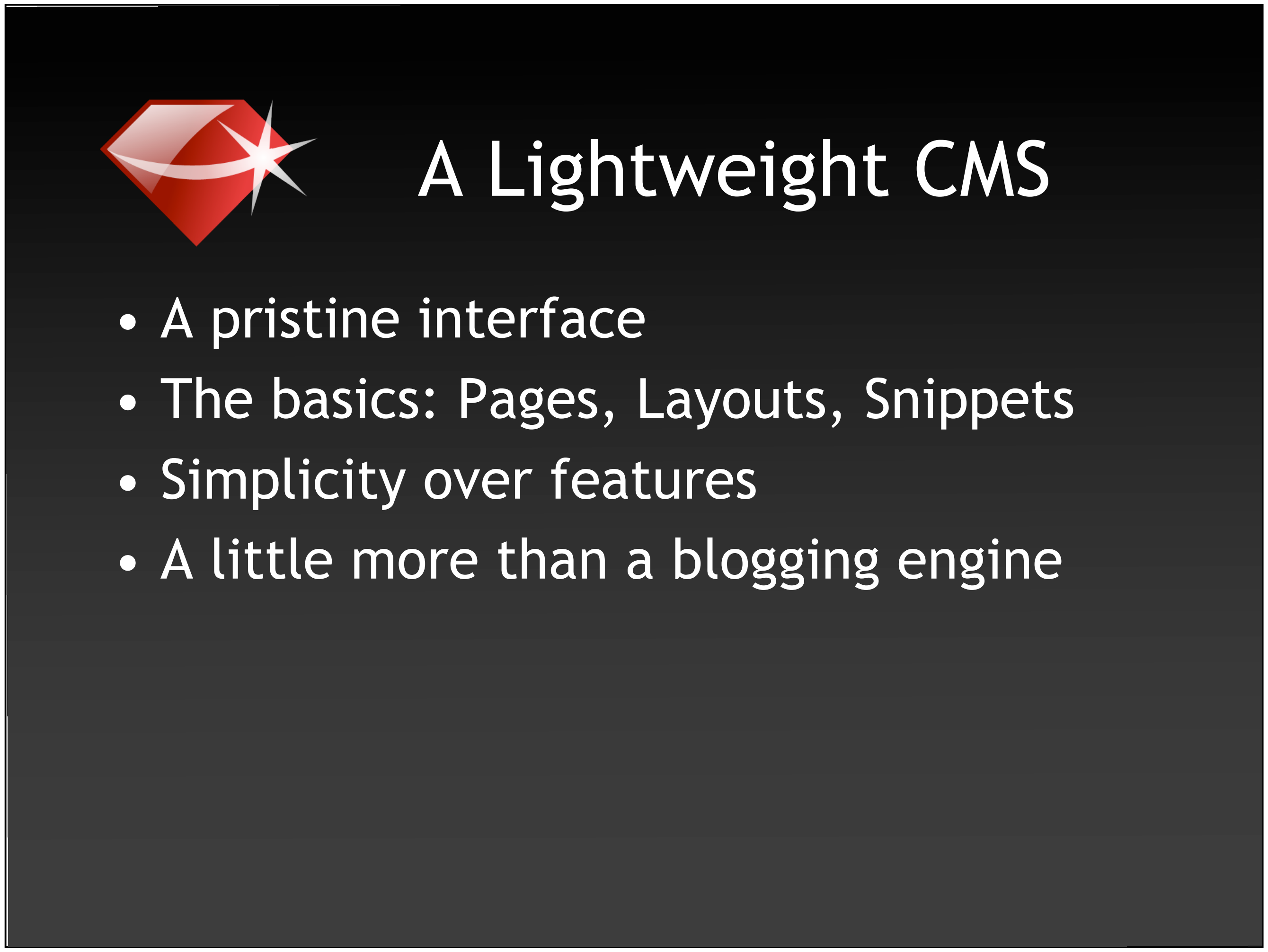
Web publishing should be easy!

“I don’t care about your meta-meta-content model, just let me write a page!”



What is Radiant?

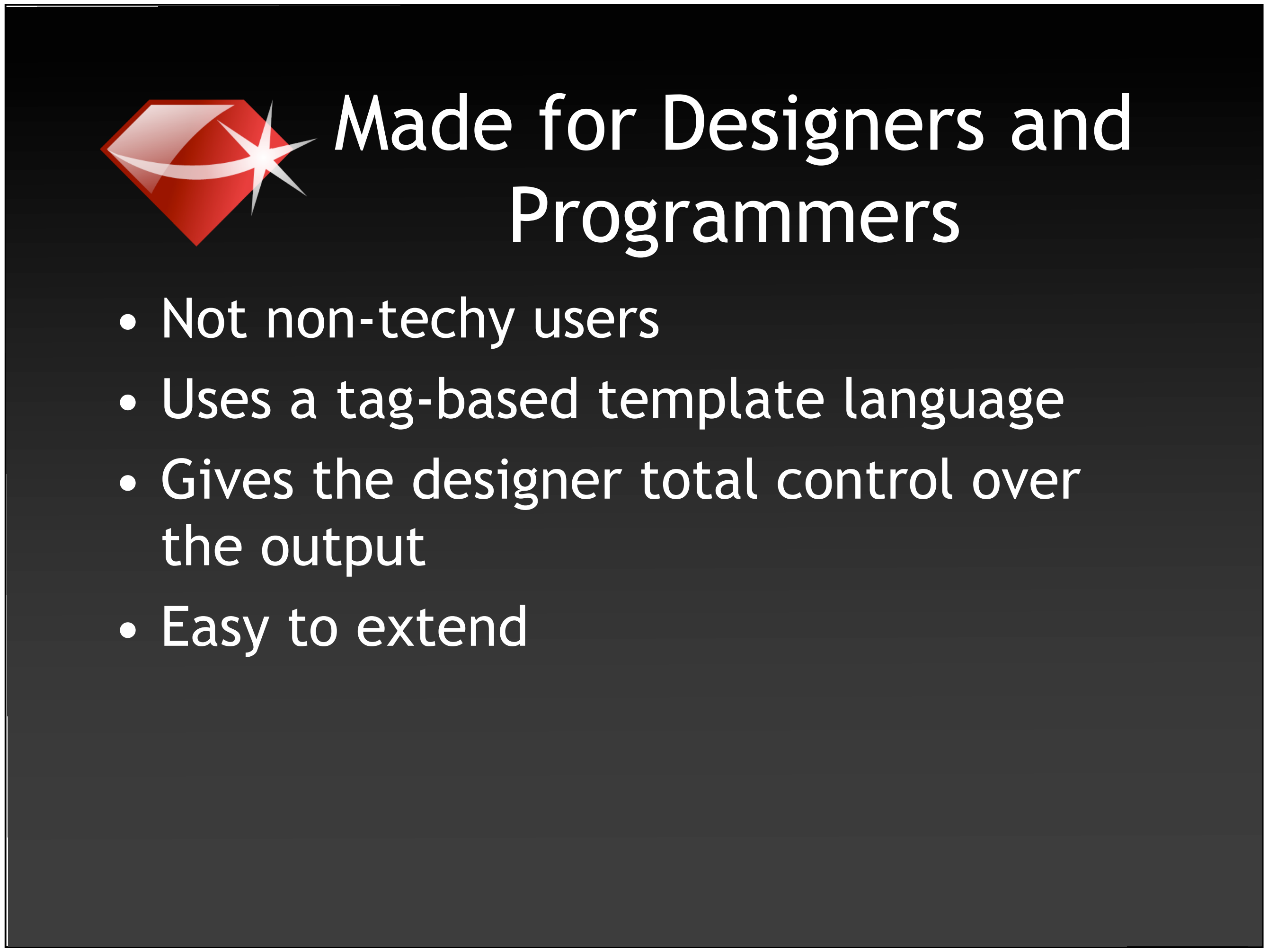
"Radiant is a no-fluff, open source content management system designed for small teams."



A Lightweight CMS

- A pristine interface
- The basics: Pages, Layouts, Snippets
- Simplicity over features
- A little more than a blogging engine

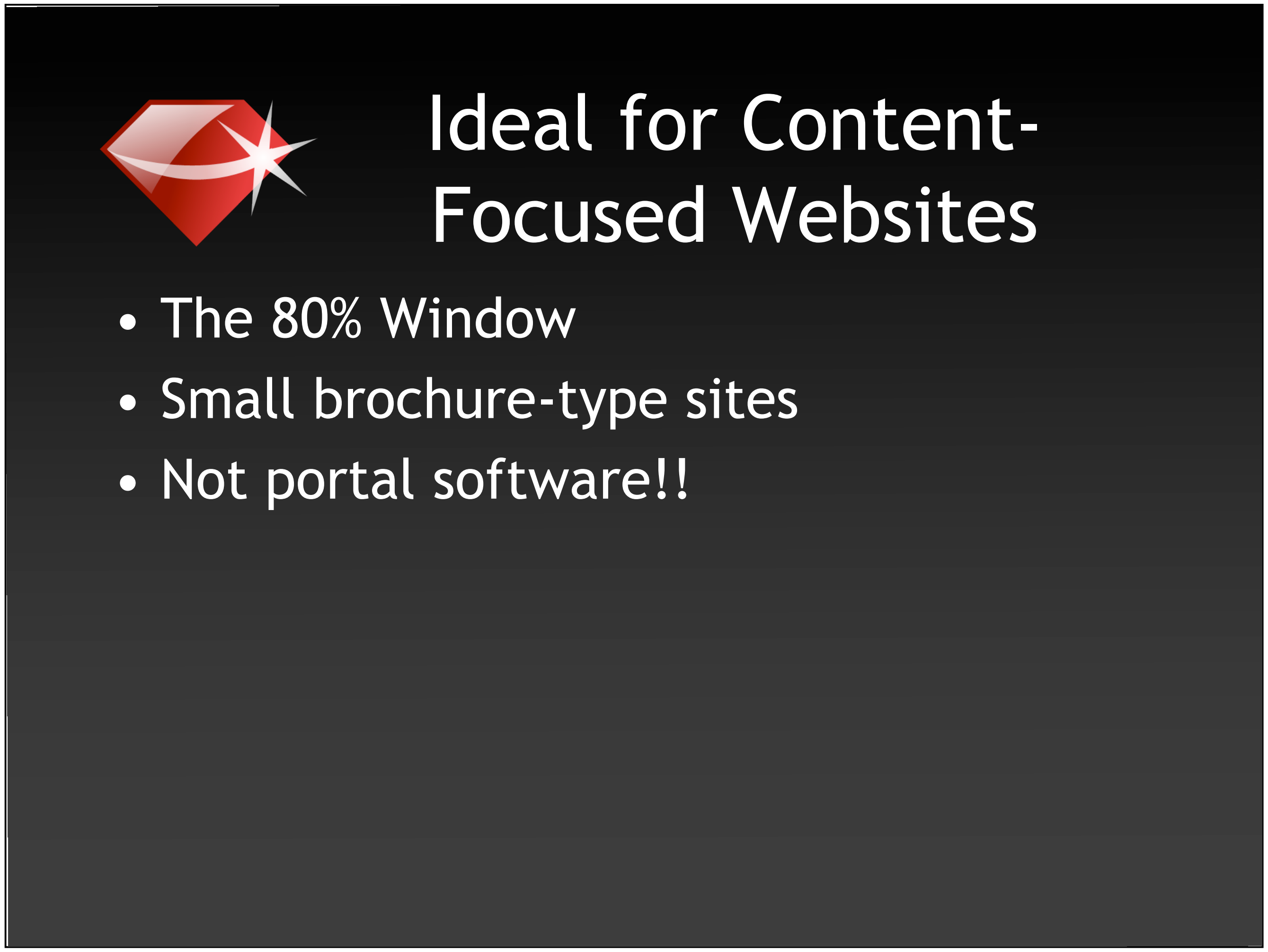
A lot less than an “Enterprise” CMS



Made for Designers and Programmers

- Not non-techy users
- Uses a tag-based template language
- Gives the designer total control over the output
- Easy to extend

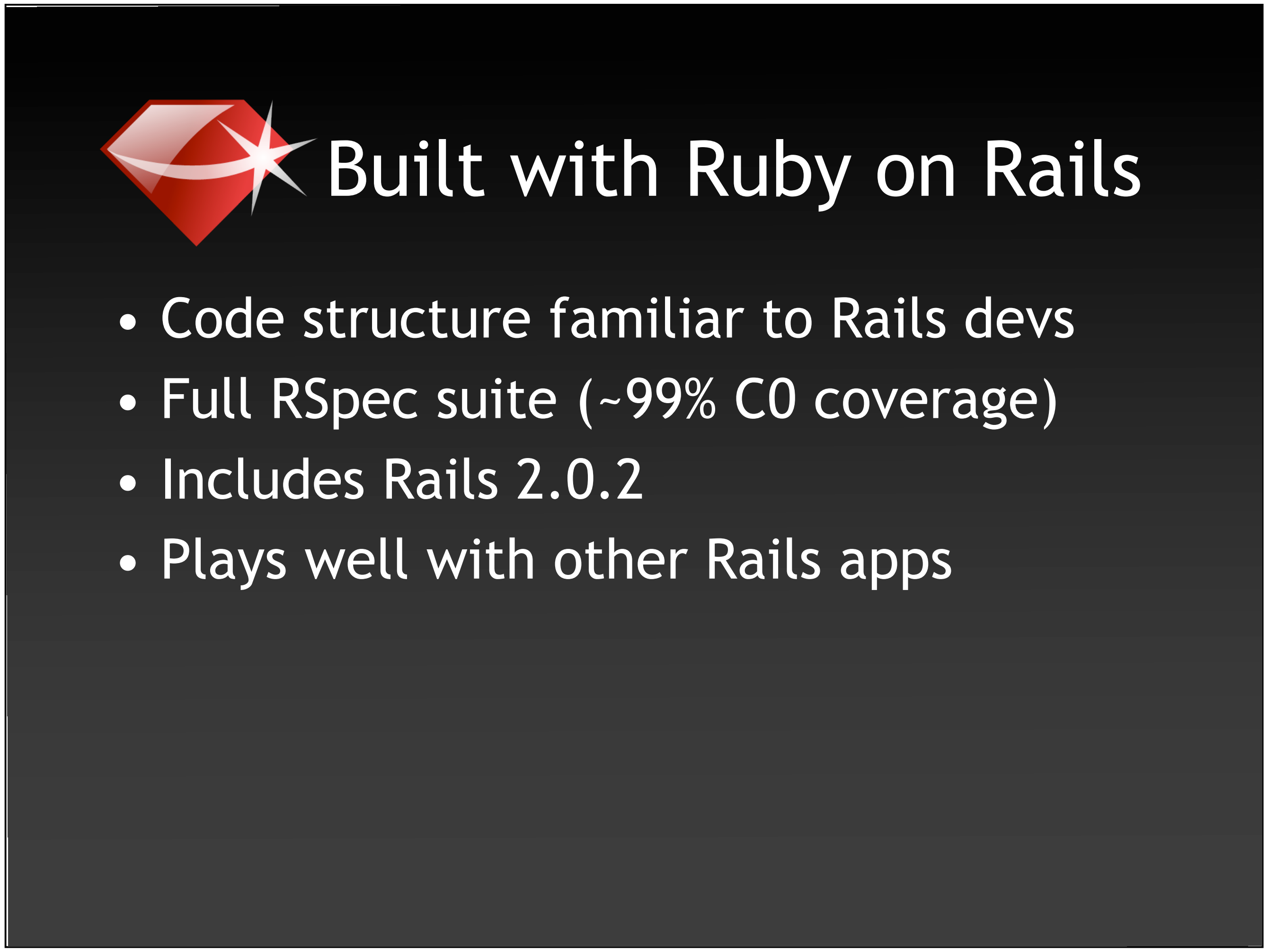
Non-techy users can be accommodated through customization



Ideal for Content-Focused Websites

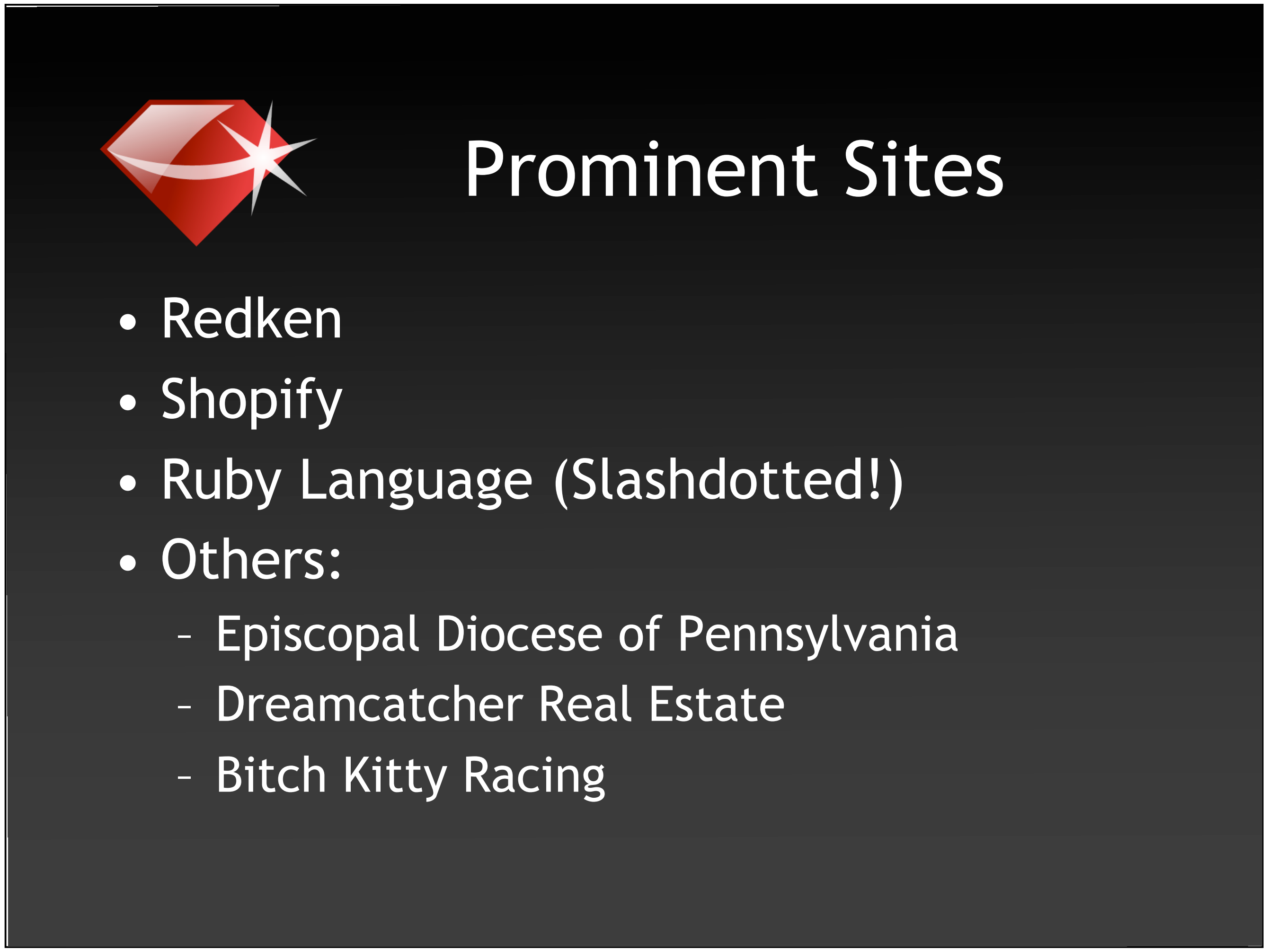
- The 80% Window
- Small brochure-type sites
- Not portal software!!

Perfect for brochure-type sites, great for others too



Built with Ruby on Rails

- Code structure familiar to Rails devs
- Full RSpec suite (~99% C0 coverage)
- Includes Rails 2.0.2
- Plays well with other Rails apps



Prominent Sites

- Redken
- Shopify
- Ruby Language (Slashdotted!)
- Others:
  - Episcopal Diocese of Pennsylvania
  - Dreamcatcher Real Estate
  - Bitch Kitty Racing

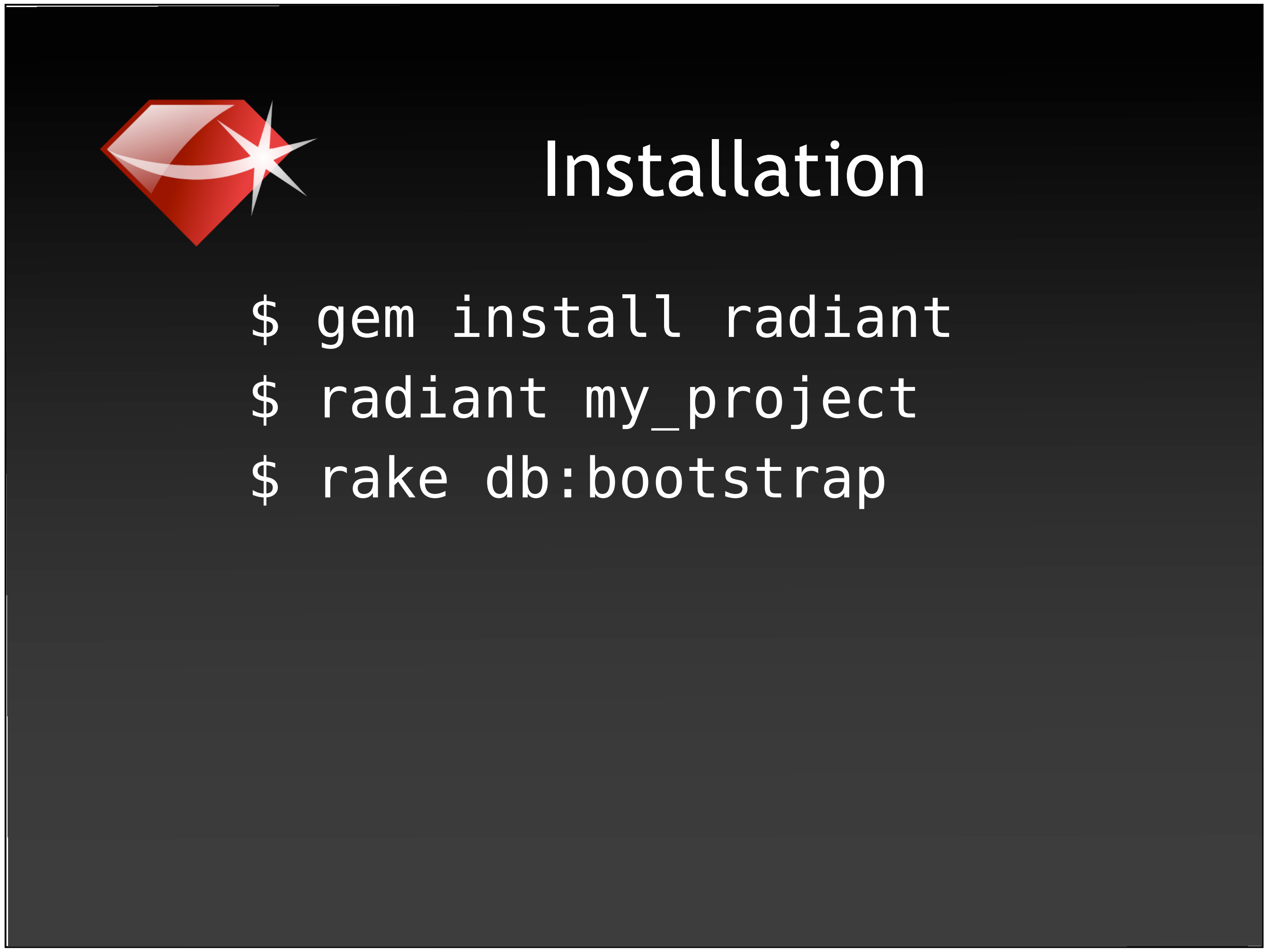
Redken.com and RedkenSalon.com (heavily customized)

Shopify.com front-facing brochure for e-commerce webapp

Ruby-lang.org original impetus for Radiant

Diopa.org & taosnewmexico.com - Loren and Paul

Bitchkittyracing.com - hilarious



Installation

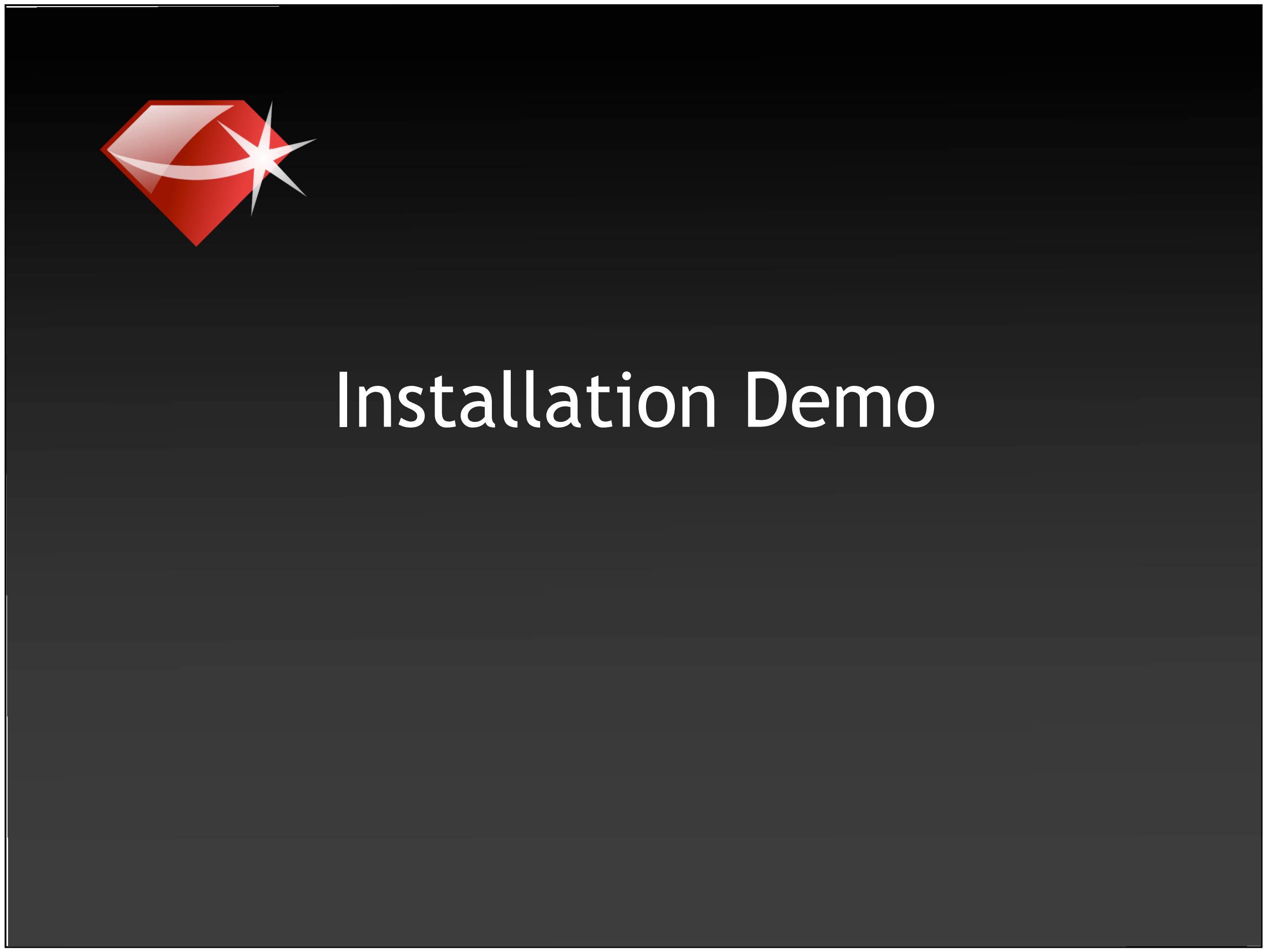
```
$ gem install radiant
$ radiant my_project
$ rake db:bootstrap
```

Need Ruby and RubyGems installed to do this:

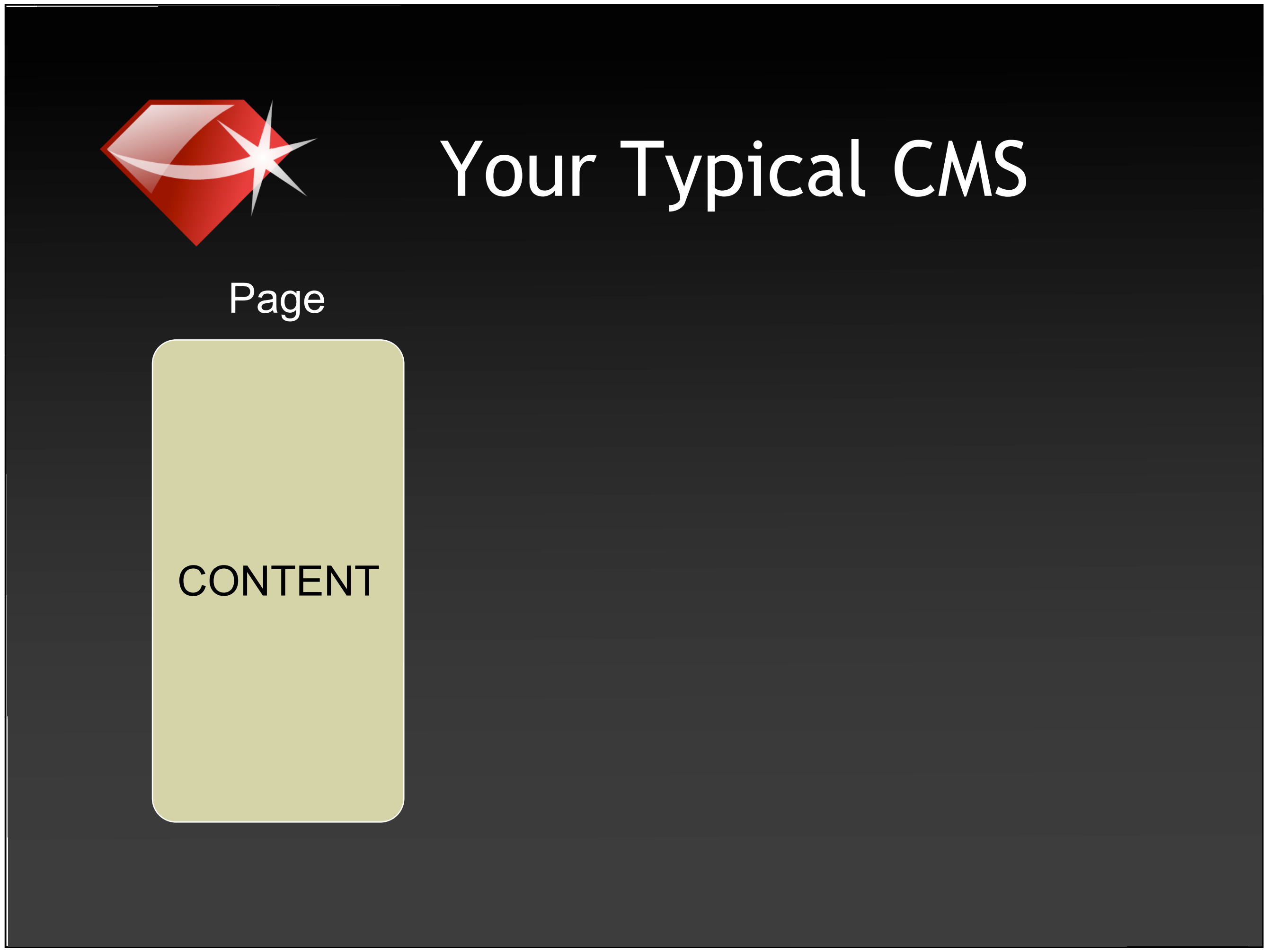
* Comes with OS/X Leopard

* Easy install for Linux (yum, apt-get, Synaptic)

* One-click installer for Windows



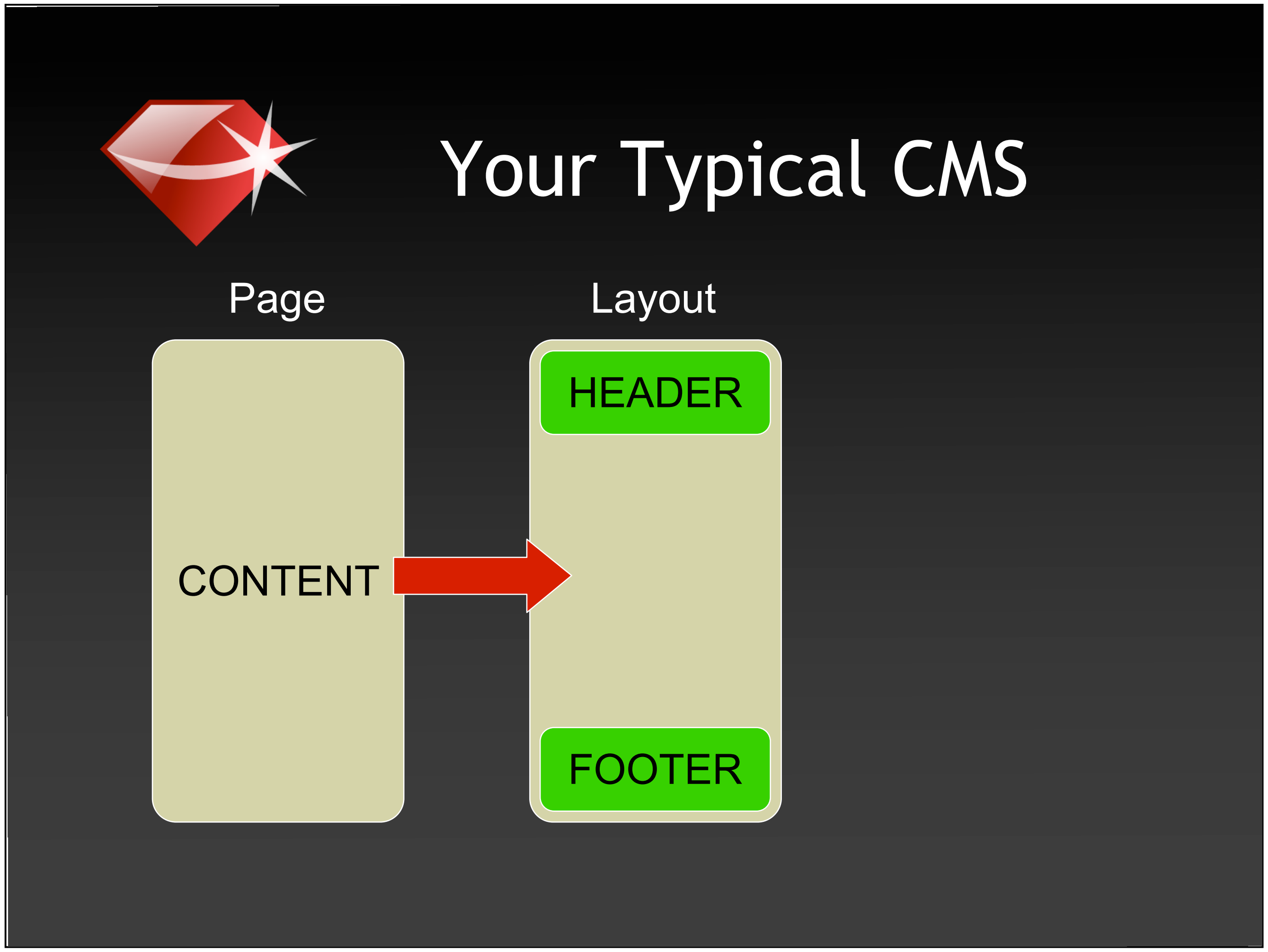
Installation Demo



Your Typical CMS

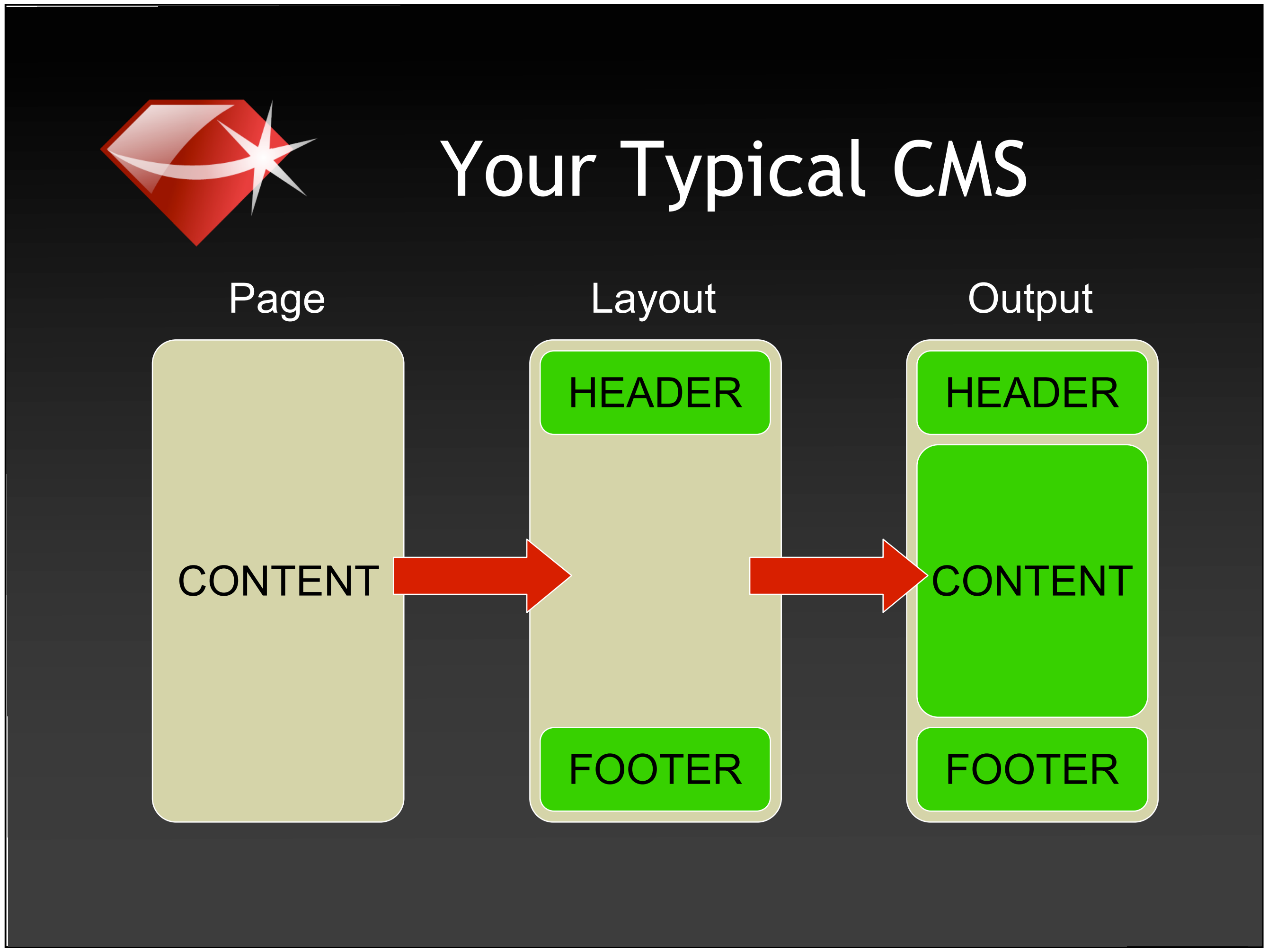
How is Radiant different? Let’s see an example of a Typical CMS structure

Editing content for a page is generally pretty flat



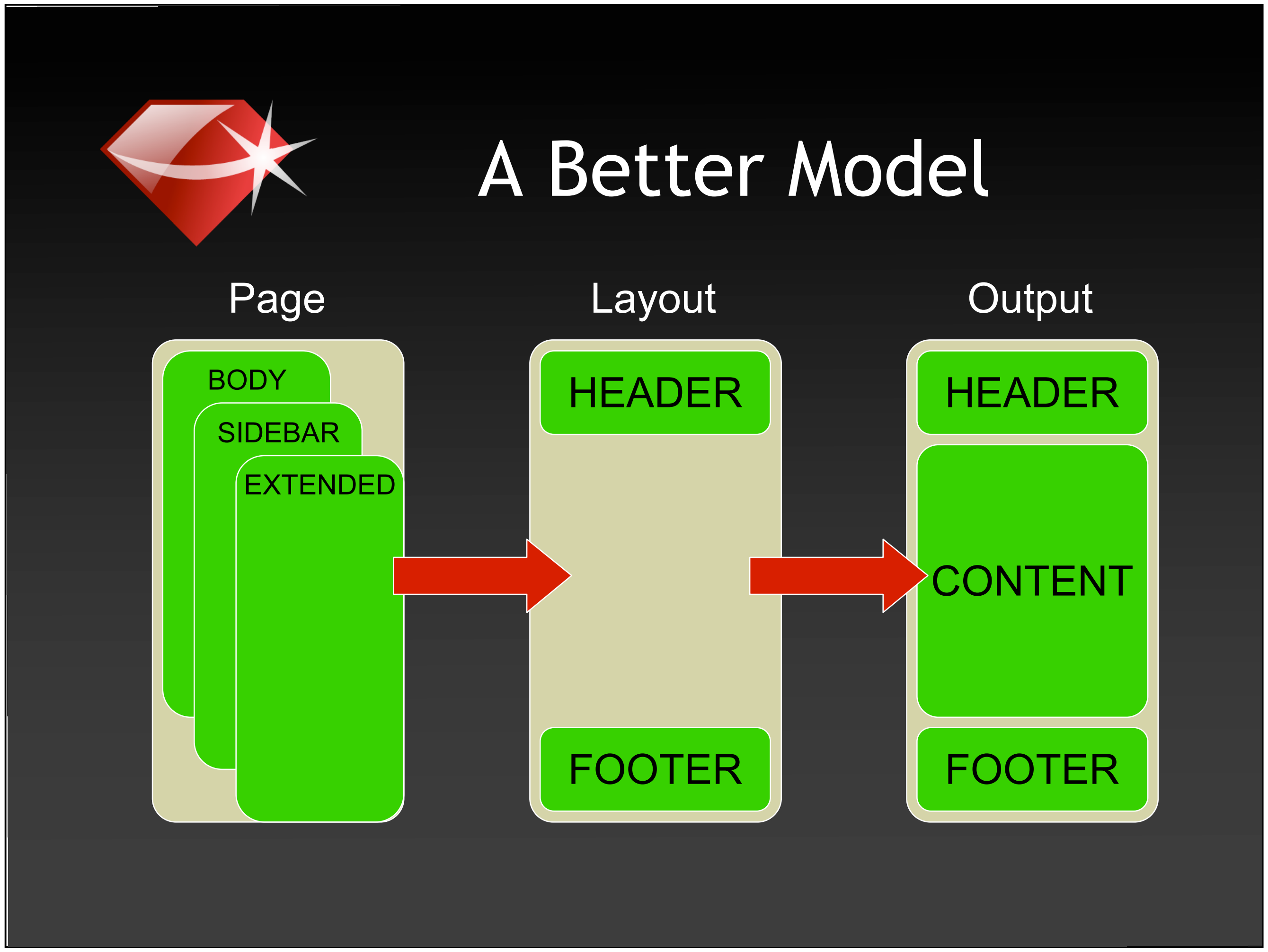
Your Typical CMS

Once editing is done and the page is to be generated, a layout is selected that has a site-wide header and footer


Your Typical CMS

Then the content of the page is inserted into the middle of the layout and the page is generated.

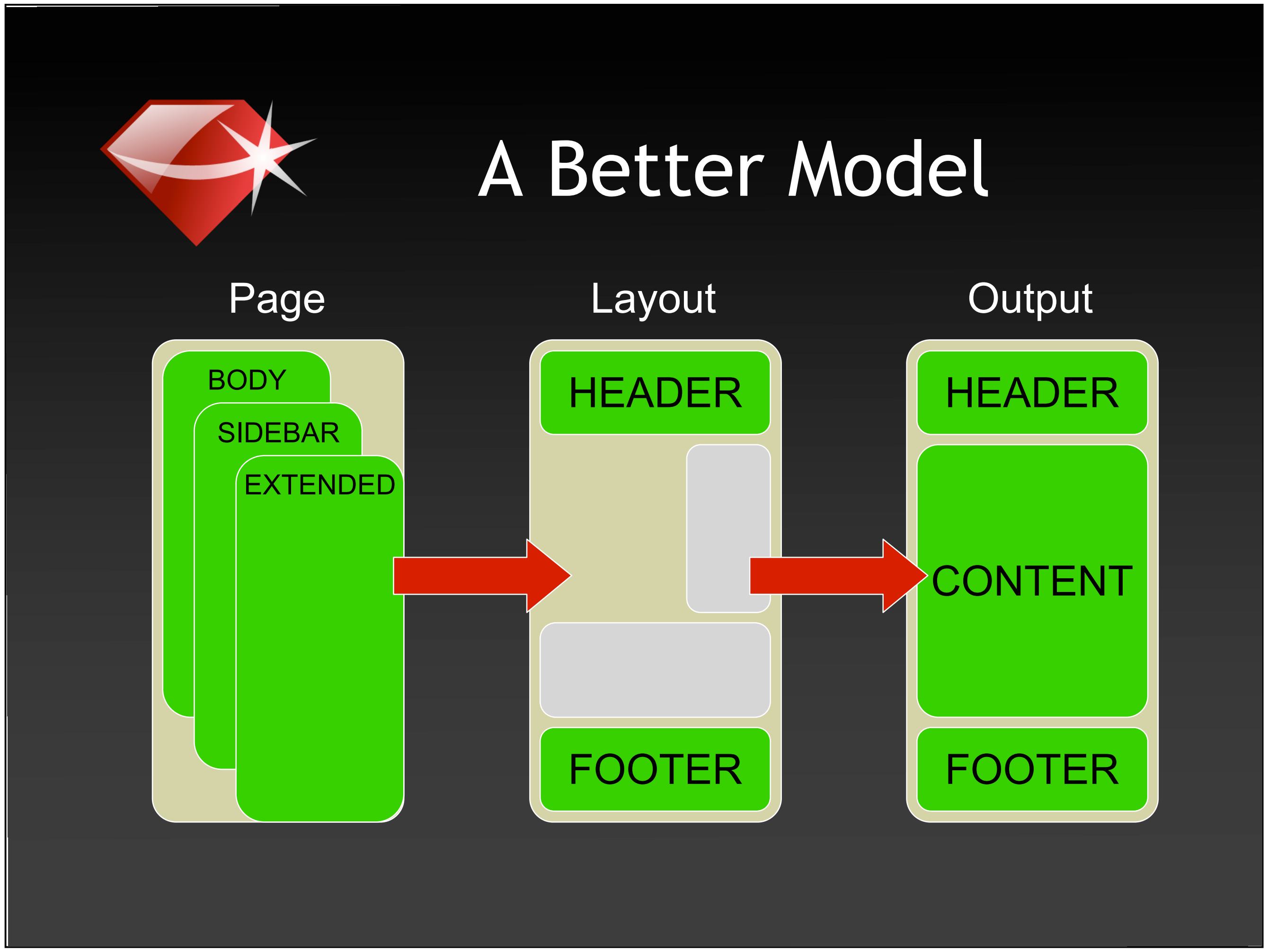
However, typical designs rarely involve such a simple structure. Instead, each page of content consists of several different pieces or blocks that flow within the content area



A Better Model

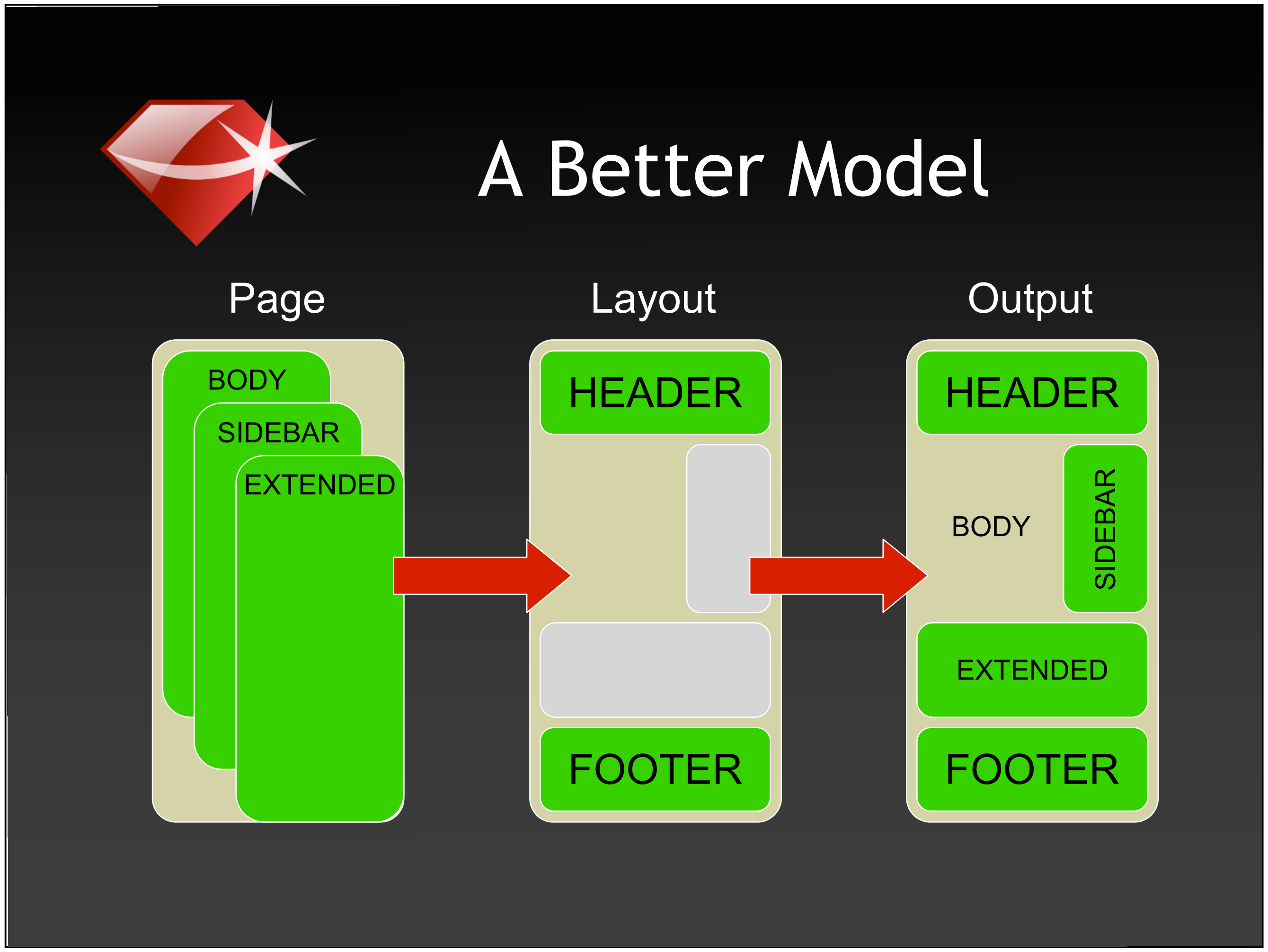
So let’s model the page to accommodate more complex designs.

First we break the page into individual pieces



A Better Model

Then we'll create specific spaces for those pieces in the layout



A Better Model

Page

BODY

SIDEBAR

EXTENDED

Layout

HEADER

FOOTER

Output

HEADER

BODY

SIDEBAR

EXTENDED

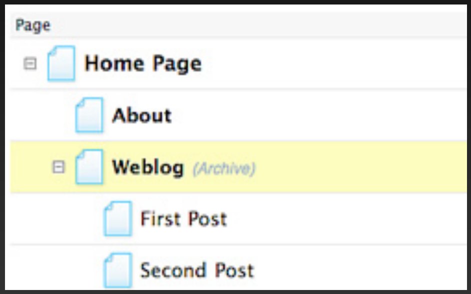
FOOTER

And finally, render the individual pieces into their respective places.

This is the Radiant content model. Now let's talk about the specific aspects of that model

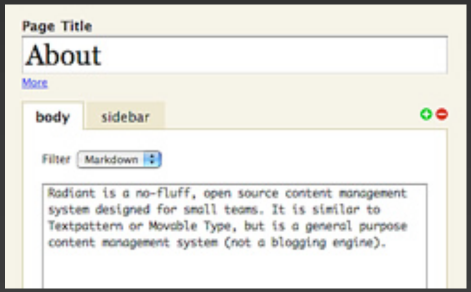
Pages

- Content
- Hierarchical
- “Folder” and “index”
- Composed of “parts”
- Five-minute automatic cache



Page

- Home Page
  - About
  - Weblog (Archive)
    - First Post
    - Second Post



Page Title

About

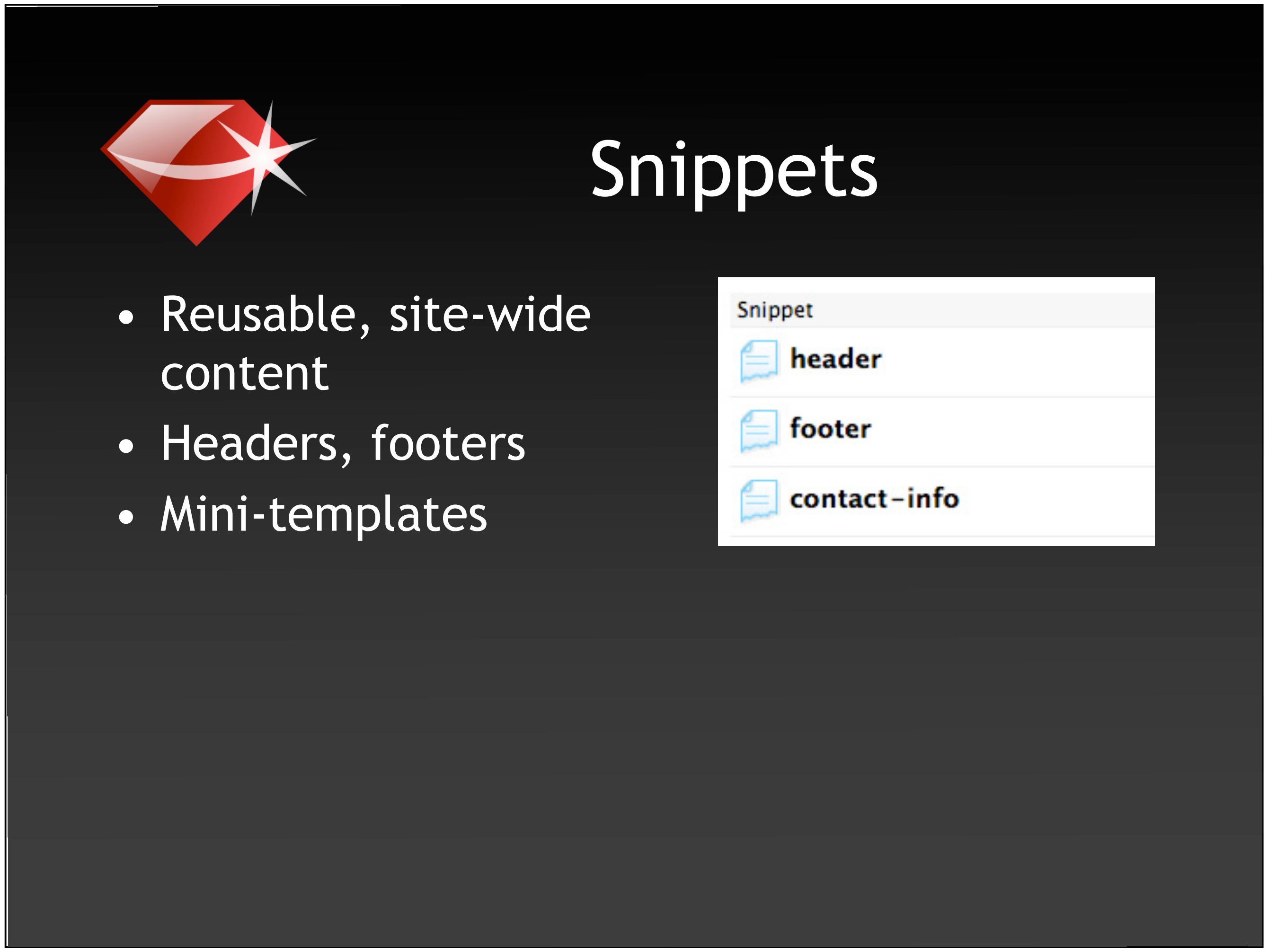
More

body | sidebar

Filter: Markdown

Radiant is a no-fluff, open source content management system designed for small teams. It is similar to Textpattern or Movable Type, but is a general purpose content management system (not a blogging engine).

“folder/index” - if you were publishing a static site in Apache



Snippets

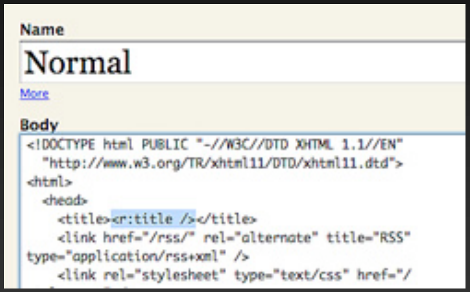
- Reusable, site-wide content
- Headers, footers
- Mini-templates

Snippet
header
footer
contact-info

Mini-templates: used in iteration schemes

Layouts

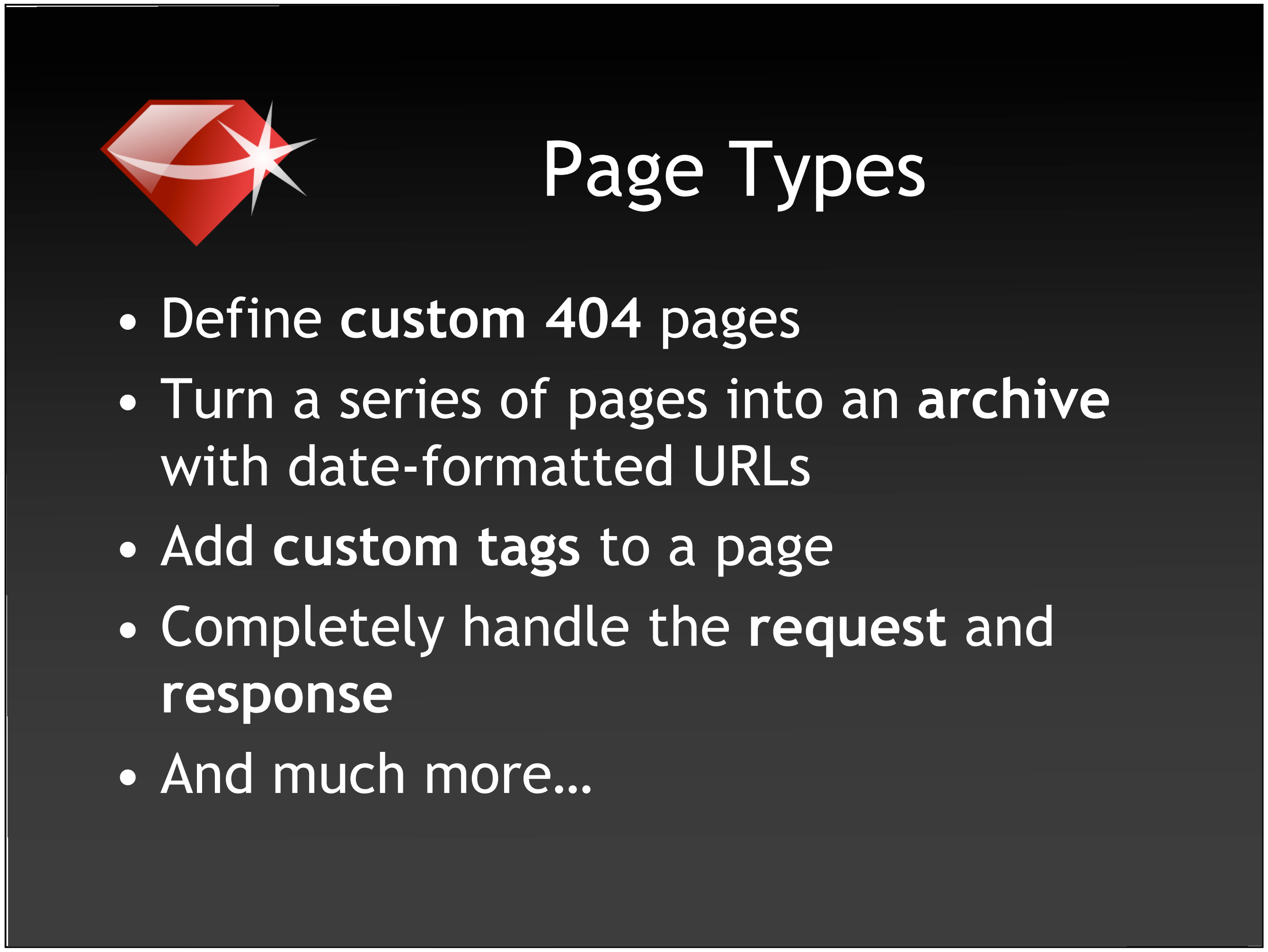
- Site wireframe or "template"
- Renders parts and snippets
- Determines content type - HTML, XML, RSS, Javascript, CSS, Text, etc



Normal

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <title><r:title /></title>
    <link href="/rss/" rel="alternate" title="RSS"
type="application/rss+xml" />
    <link rel="stylesheet" type="text/css" href="/
```

Important to note: Radiant can serve any type of content that can be represented in plain text, including things you wouldn't normally put in CMS

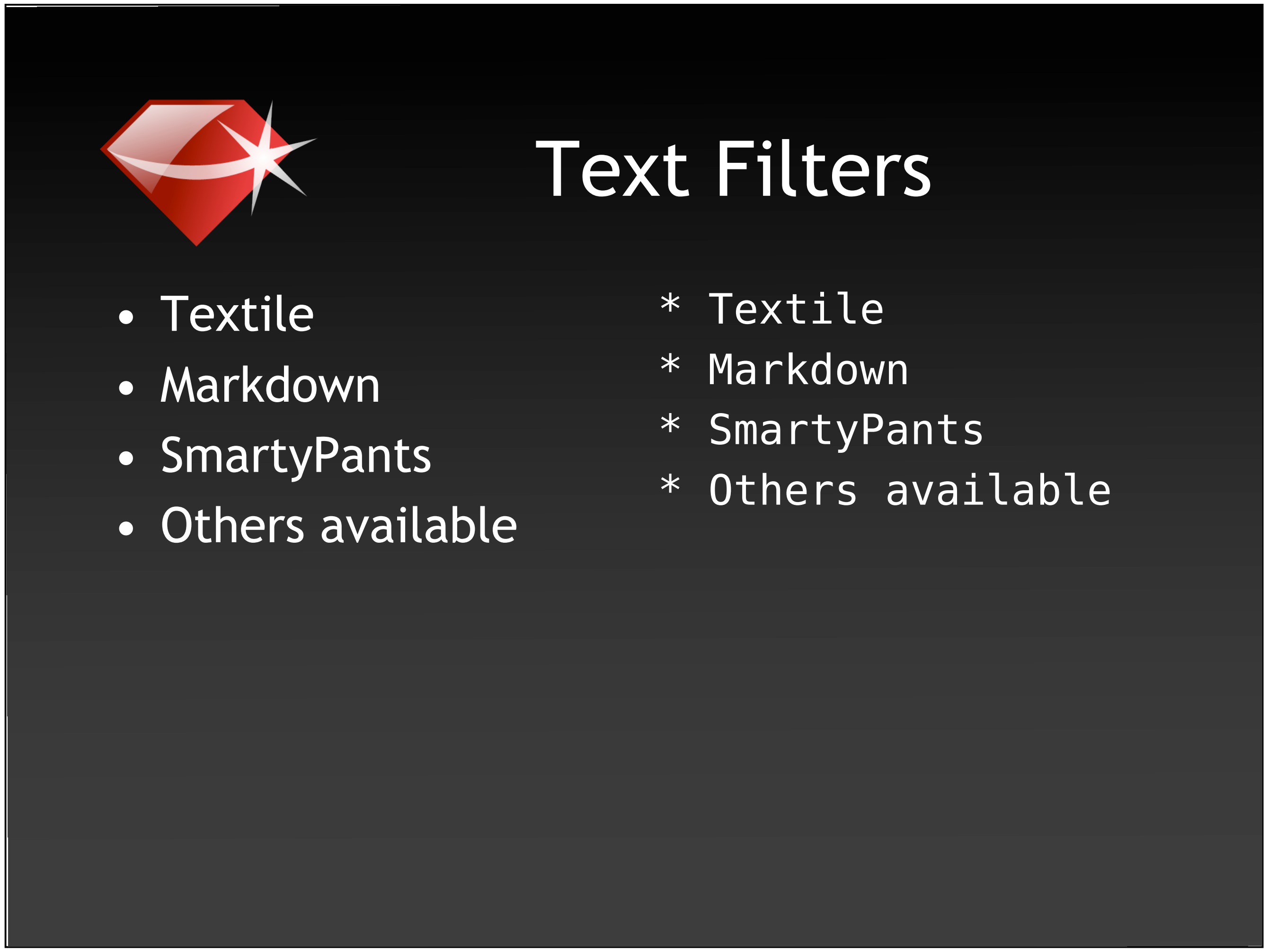


Page Types

- Define **custom 404** pages
- Turn a series of pages into an **archive** with date-formatted URLs
- Add **custom tags** to a page
- Completely handle the **request** and **response**
- And much more...

404 pages, Archive are included in the core

Custom tags, etc can be done in extensions



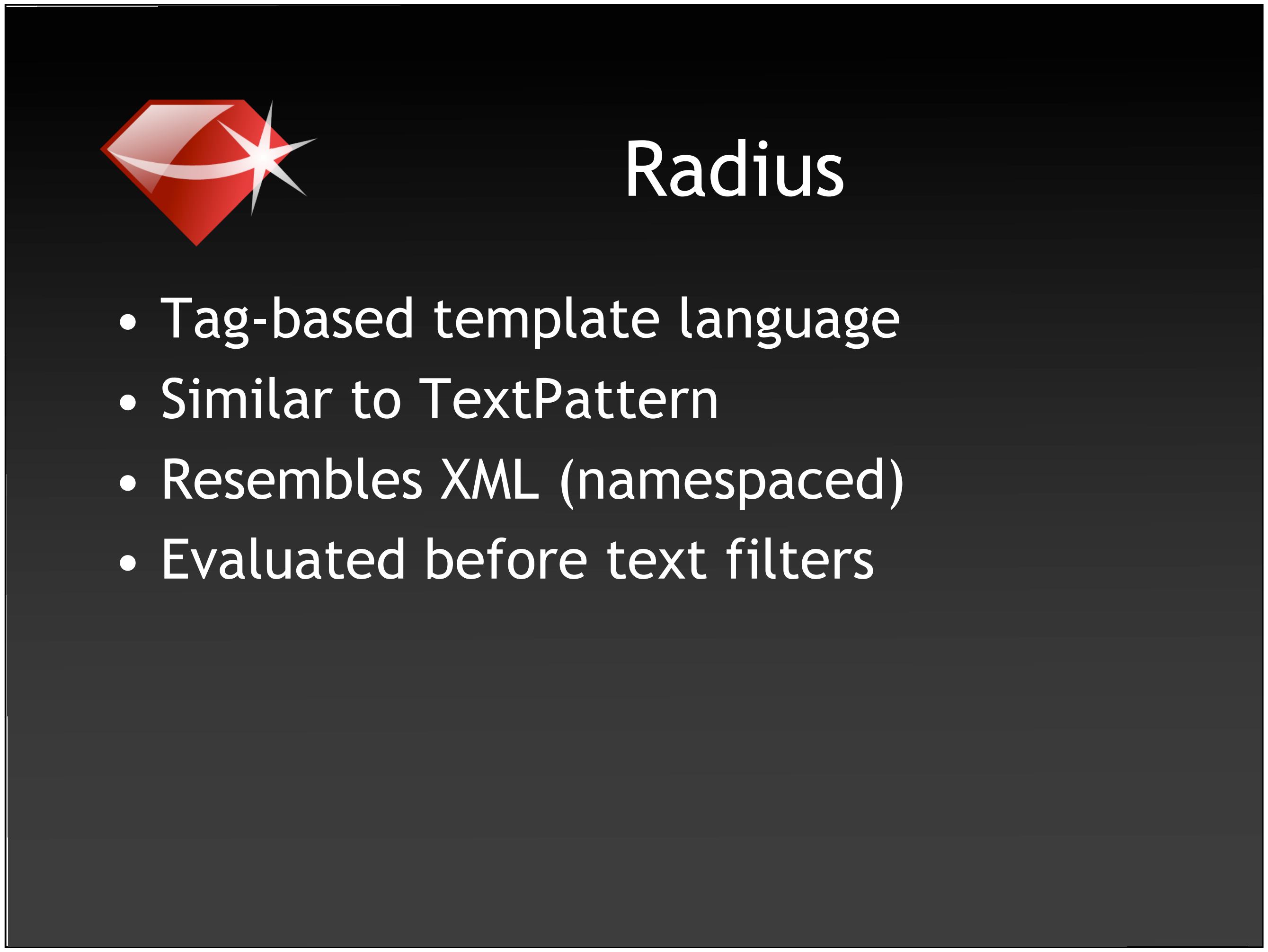
Text Filters

- Textile
- Markdown
- SmartyPants
- Others available

```
* Textile
* Markdown
* SmartyPants
* Others available
```

Text filters ease the writing of HTML

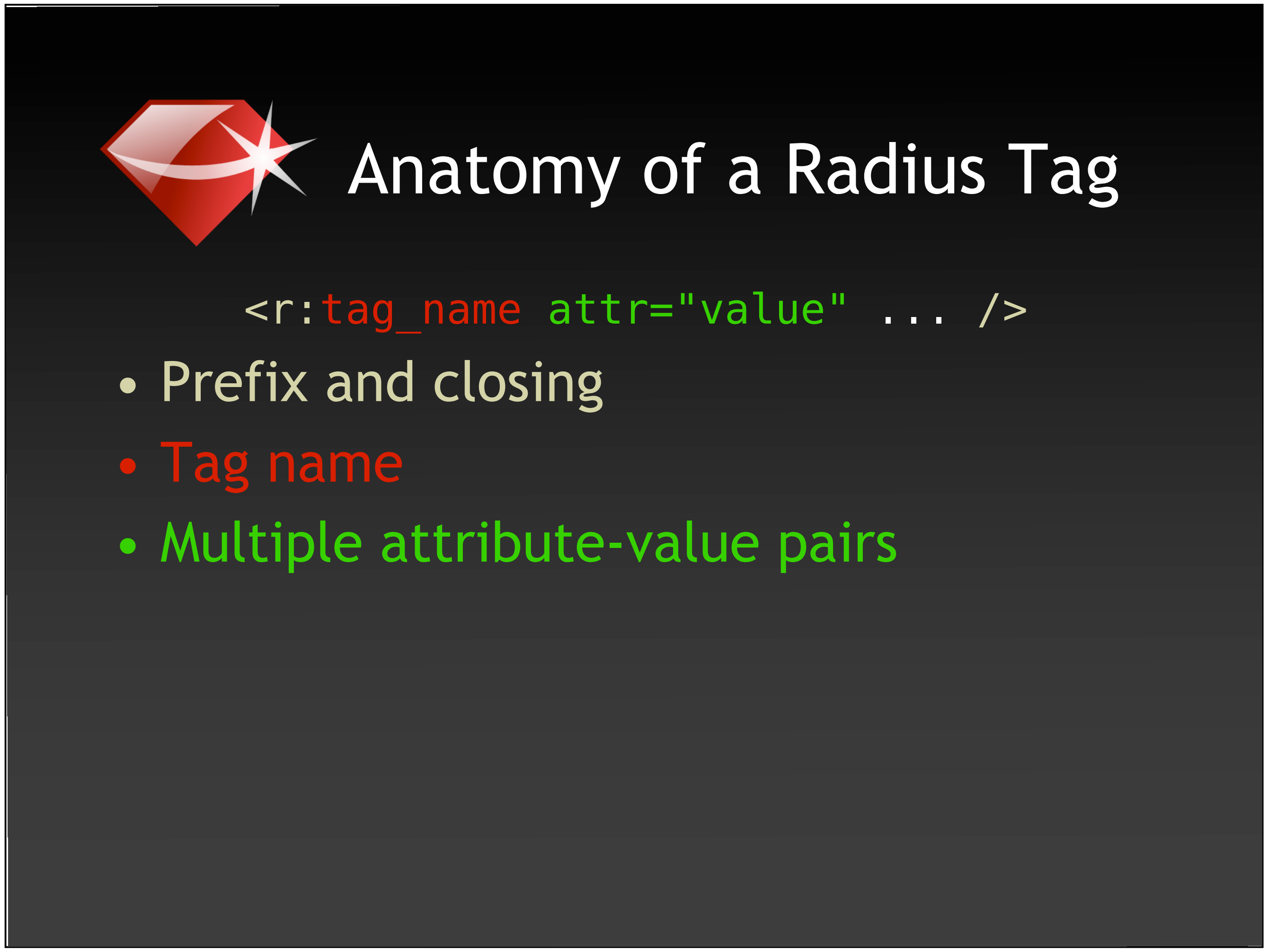
Right column shows how input would look in Textile, left formatted



Radius

- Tag-based template language
- Similar to TextPattern
- Resembles XML (namespaced)
- Evaluated before text filters

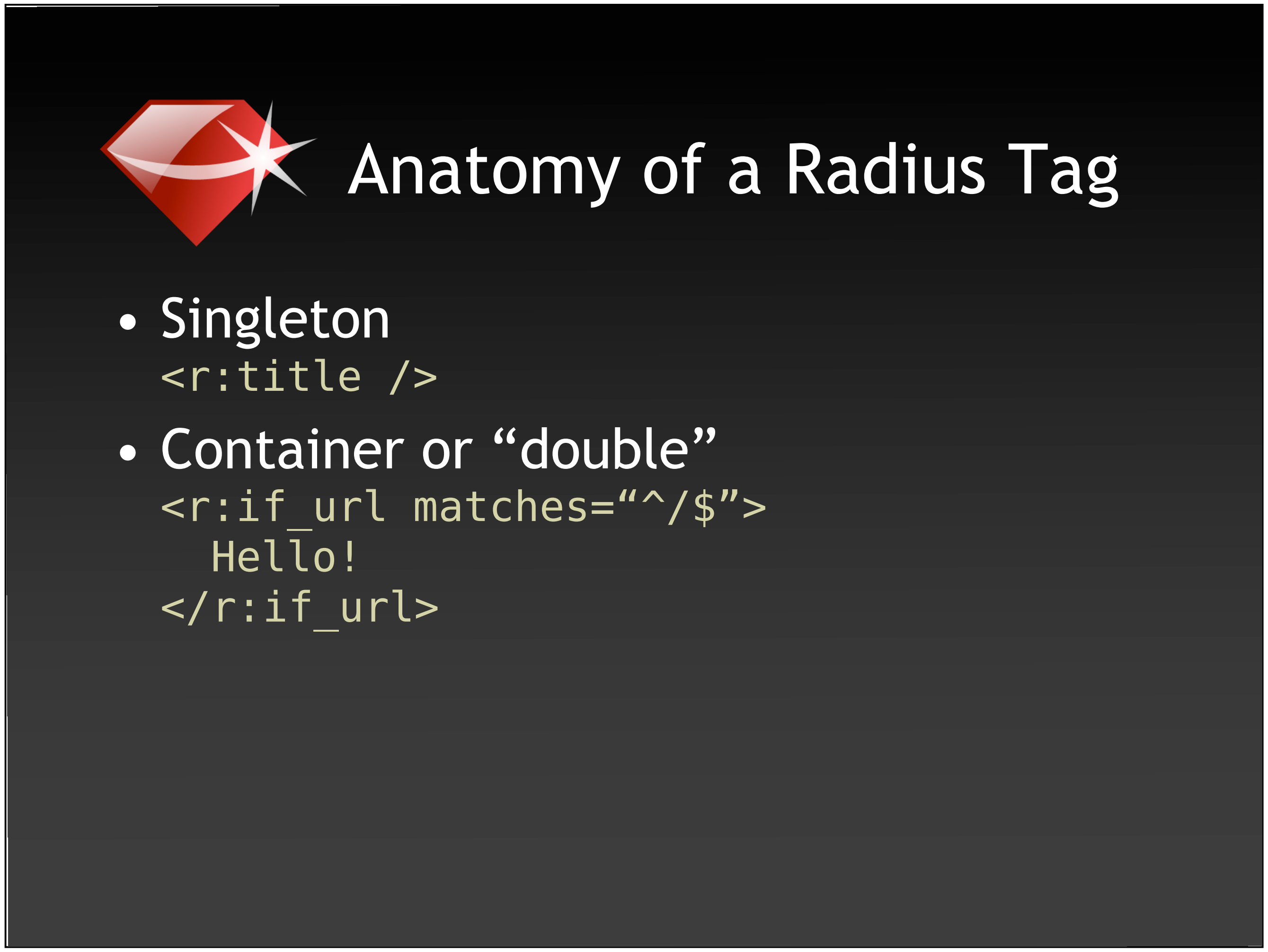
Radius.rubyforge.org - can be used in your own projects!



Anatomy of a Radius Tag

```
<r:tag_name attr="value" ... />
```

- Prefix and closing
- Tag name
- Multiple attribute-value pairs



Anatomy of a Radius Tag

- Singleton

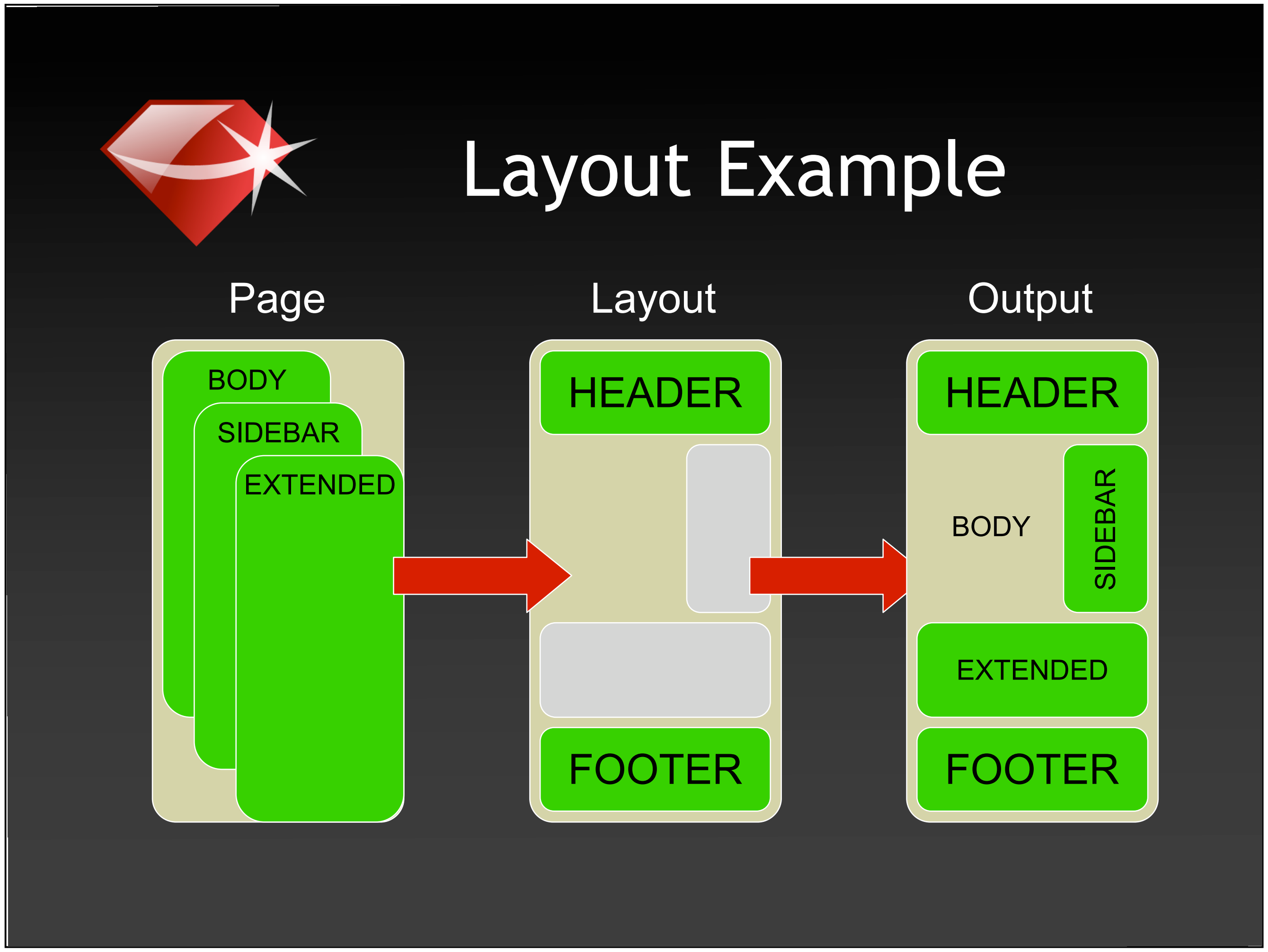
```
<r:title />
```

- Container or “double”

```
<r:if_url matches=“^/$”>
  Hello!
</r:if_url>
```

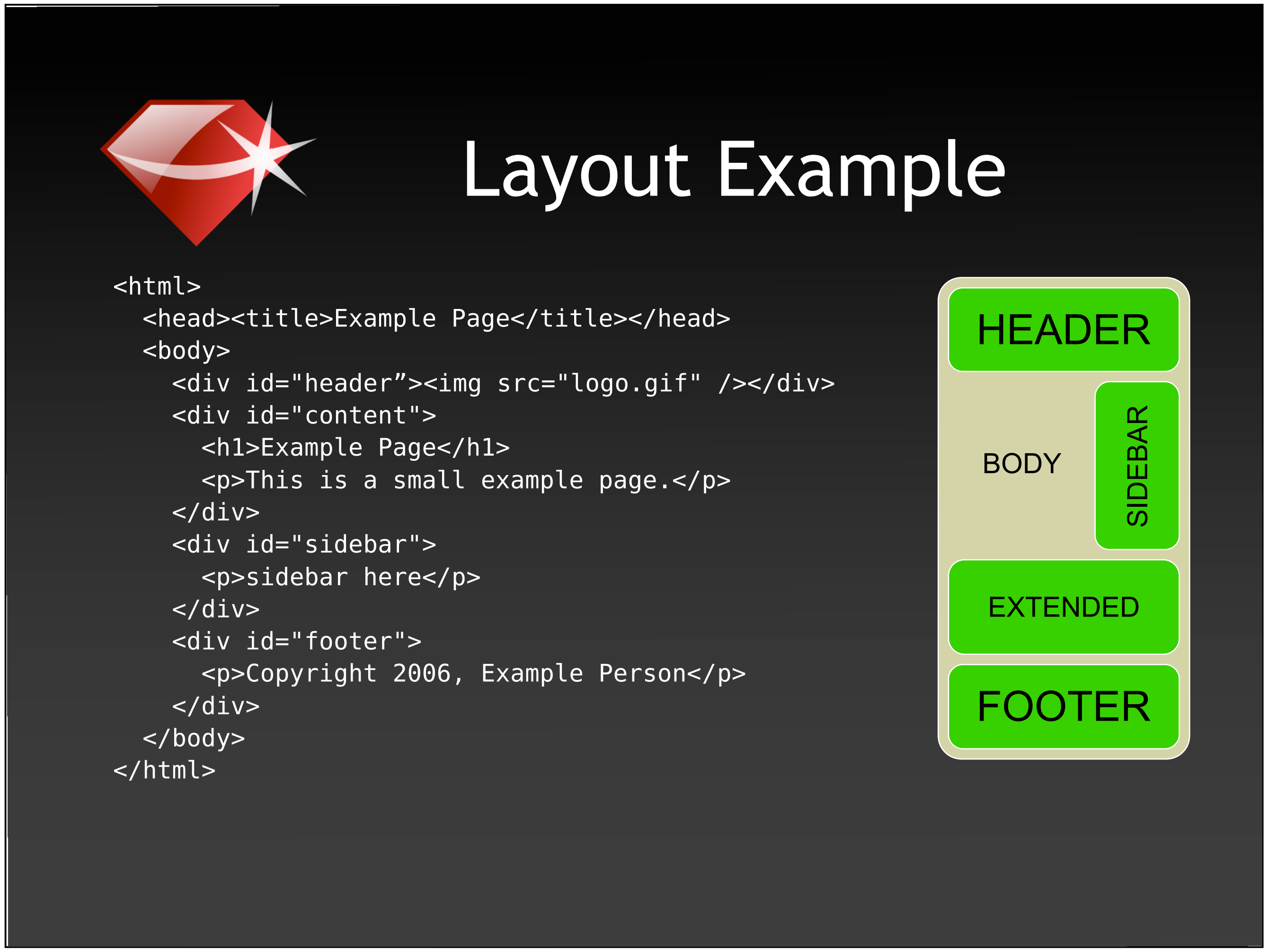
<r:title> displays the title of the current page

<r:if_url> displays its contained content if the current page’s URL matches the given regexp



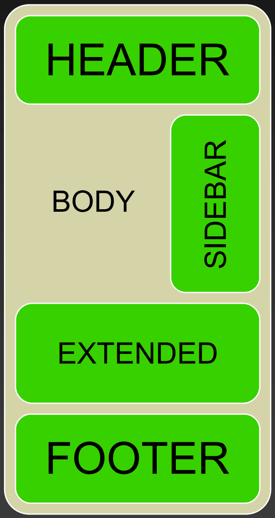
Layout Example

Let’s turn our previous example into a real layout using Radius



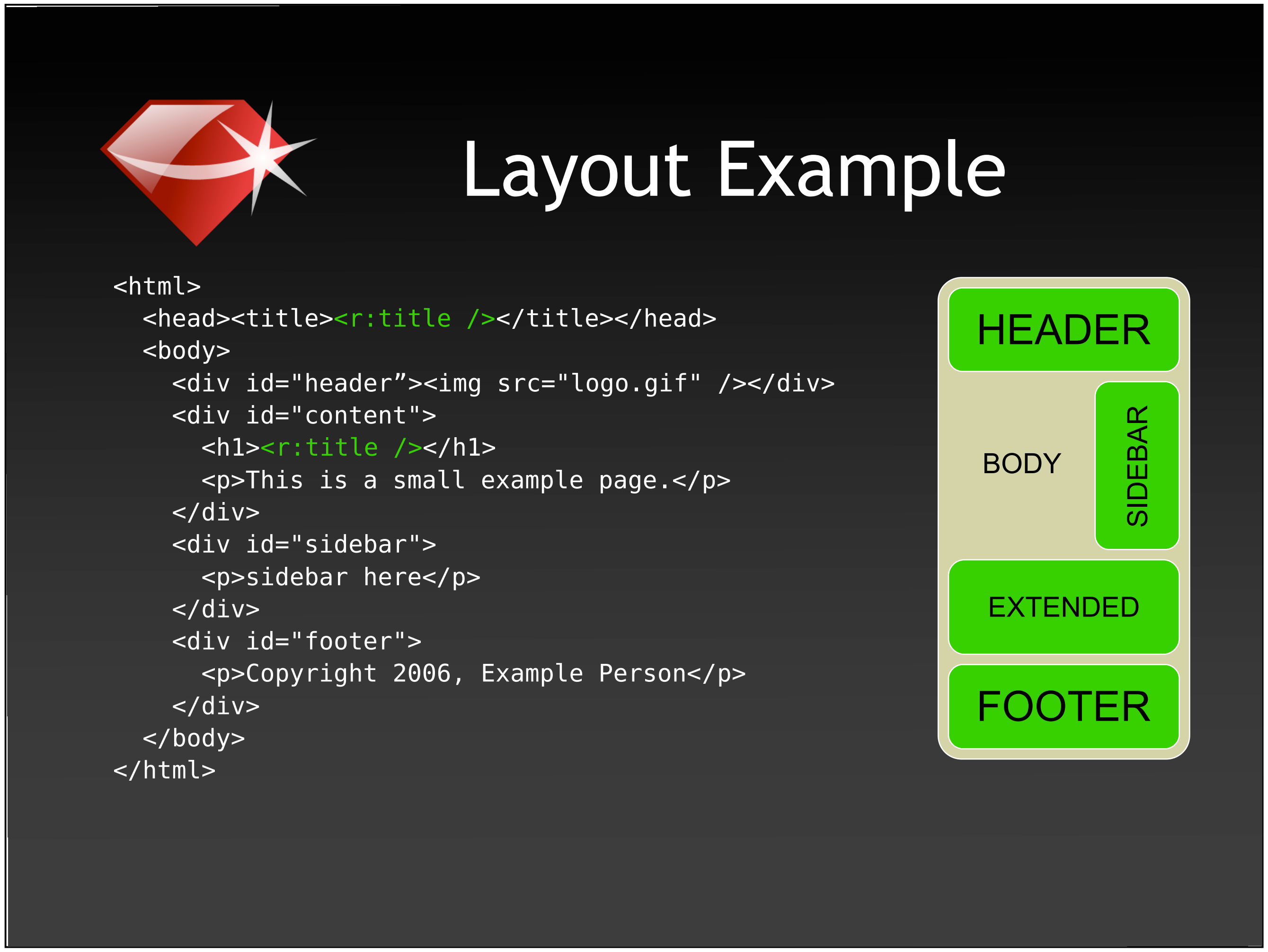
Layout Example

```
<html>
  <head><title>Example Page</title></head>
  <body>
    <div id="header”><img src="logo.gif" /></div>
    <div id="content">
      <h1>Example Page</h1>
      <p>This is a small example page.</p>
    </div>
    <div id="sidebar">
      <p>sidebar here</p>
    </div>
    <div id="footer">
      <p>Copyright 2006, Example Person</p>
    </div>
  </body>
</html>
```



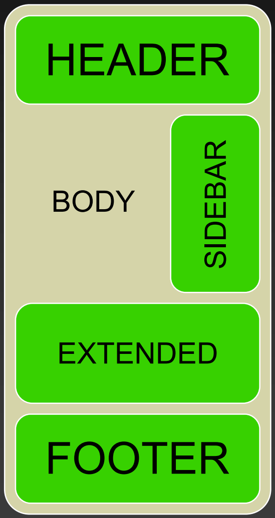
Here’s what it might look like in raw HTML

Let’s start from the top and turn the code into a Radiant layout

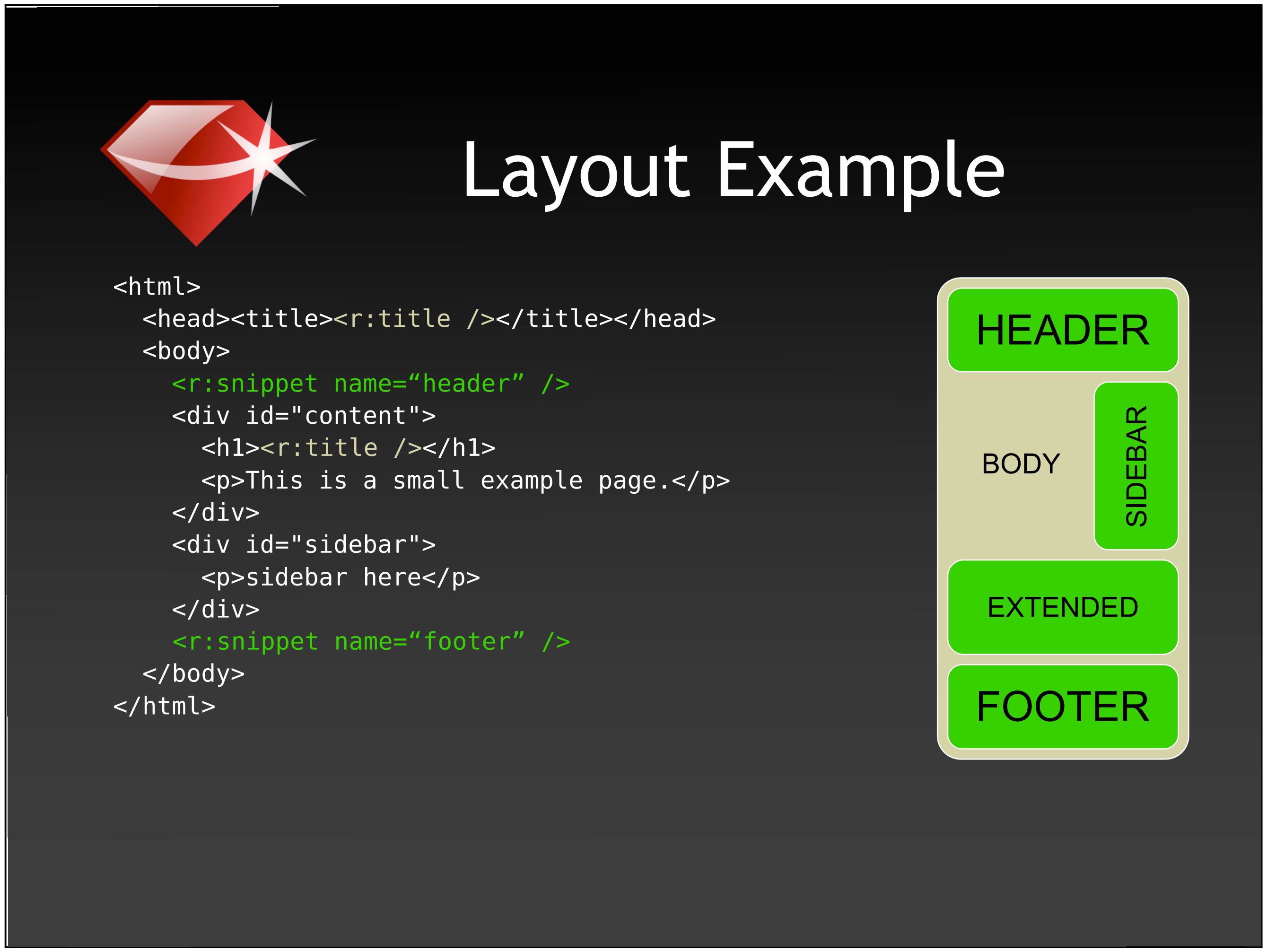


Layout Example

```
<html>
  <head><title><r:title /></title></head>
  <body>
    <div id="header”><img src="logo.gif" /></div>
    <div id="content">
      <h1><r:title /></h1>
      <p>This is a small example page.</p>
    </div>
    <div id="sidebar">
      <p>sidebar here</p>
    </div>
    <div id="footer">
      <p>Copyright 2006, Example Person</p>
    </div>
  </body>
</html>
```

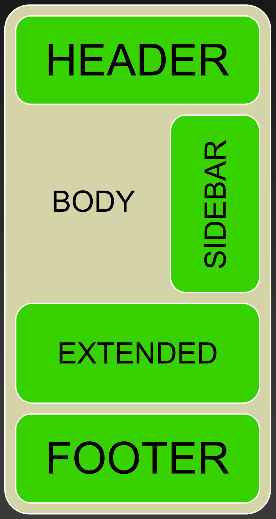


First, we’ll replace the title of the page, both places it occurs with <r:title />

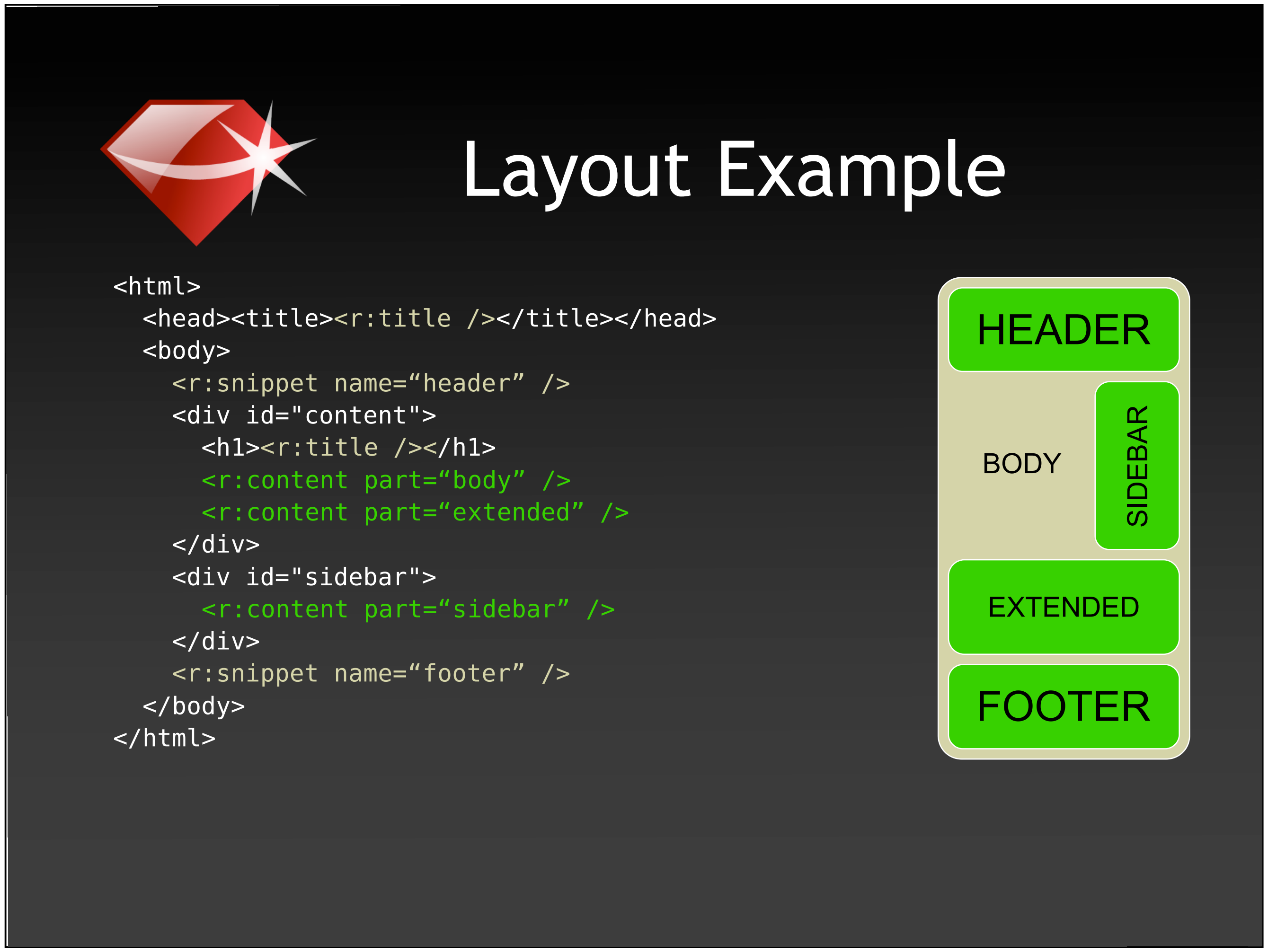


Layout Example

```
<html>
  <head><title><r:title /></title></head>
  <body>
    <r:snippet name="header" />
    <div id="content">
      <h1><r:title /></h1>
      <p>This is a small example page.</p>
    </div>
    <div id="sidebar">
      <p>sidebar here</p>
    </div>
    <r:snippet name="footer" />
  </body>
</html>
```

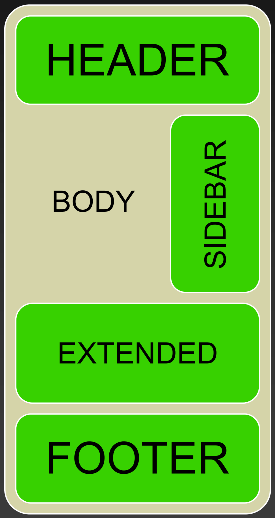


Next, we’re going to extract the header and footer into snippets that we can edit separately, so we’ll replace them with <r:snippet> tags



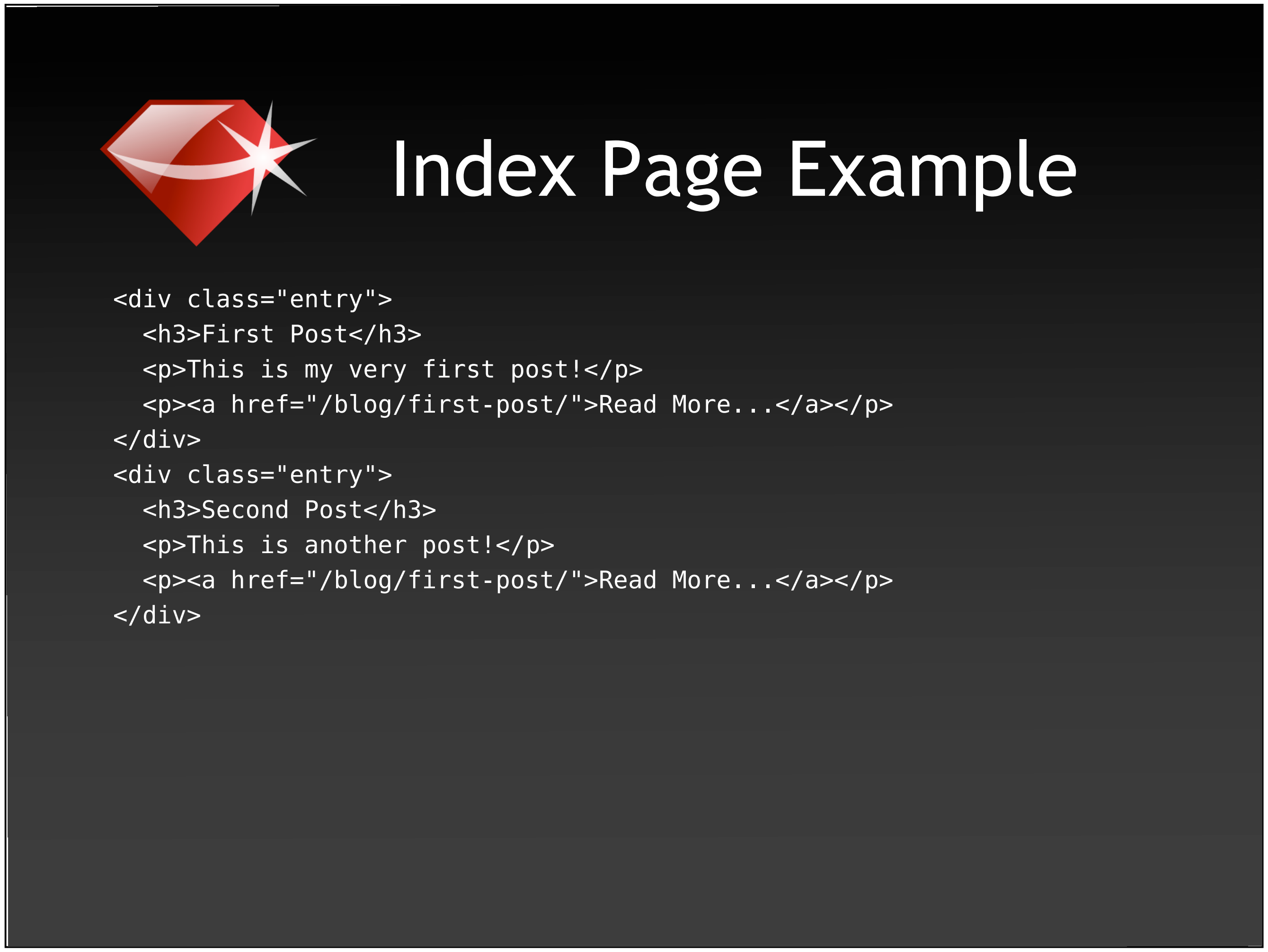
Layout Example

```
<html>
  <head><title><r:title /></title></head>
  <body>
    <r:snippet name=“header” />
    <div id="content">
      <h1><r:title /></h1>
      <r:content part=“body” />
      <r:content part=“extended” />
    </div>
    <div id="sidebar">
      <r:content part=“sidebar” />
    </div>
    <r:snippet name=“footer” />
  </body>
</html>
```



Now let’s emit the body, extended, and sidebar parts into their respective blocks using the <r:content /> tag

If tags are good, why not use them everywhere?



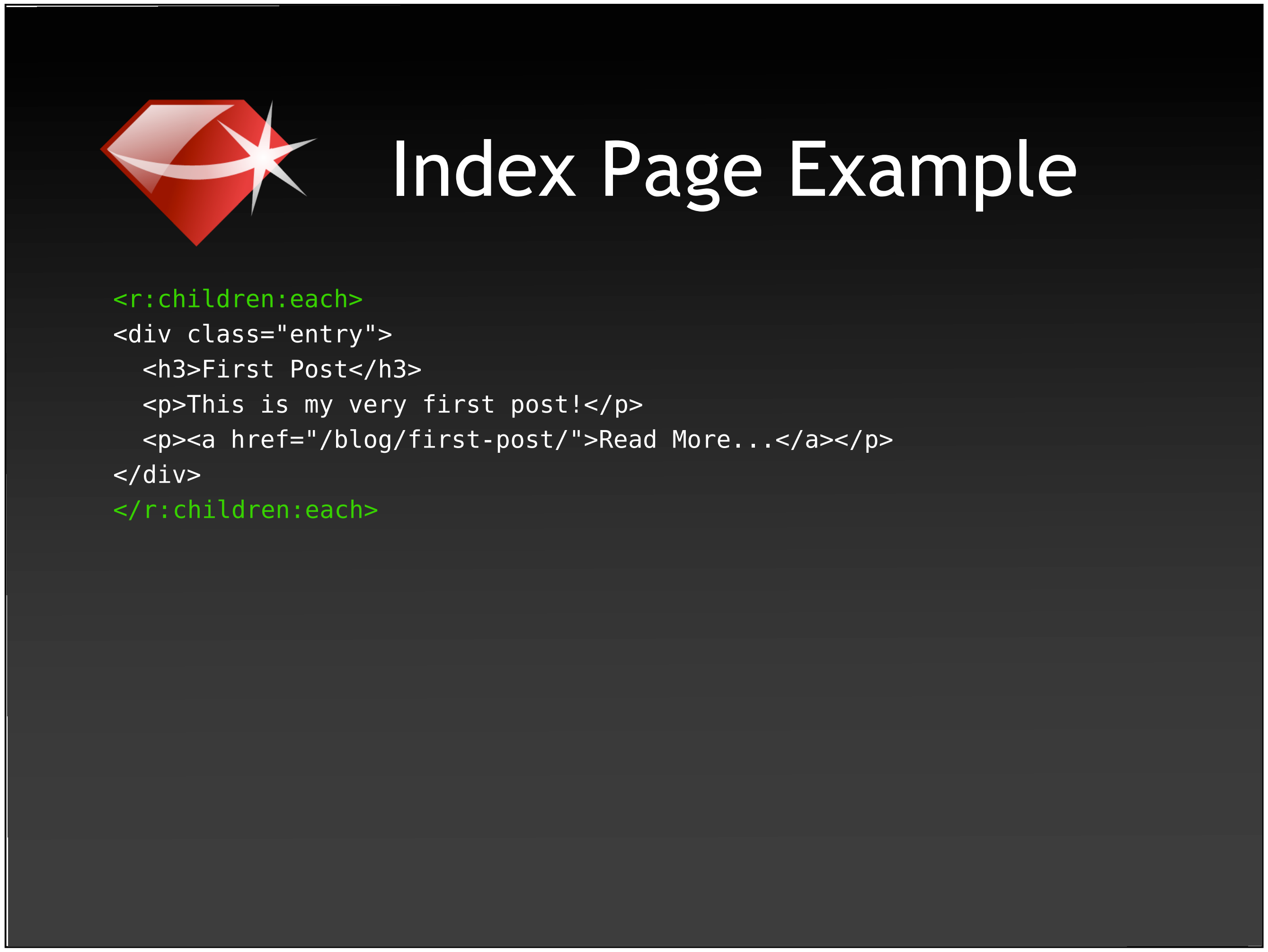
Index Page Example

```
<div class="entry">
  <h3>First Post</h3>
  <p>This is my very first post!</p>
  <p><a href="/blog/first-post/">Read More...</a></p>
</div>
<div class="entry">
  <h3>Second Post</h3>
  <p>This is another post!</p>
  <p><a href="/blog/first-post/">Read More...</a></p>
</div>
```

Here’s a typical block of HTML code one might find in, say, a blog

You have multiple entries with title, a short summary and a link to read more of the entry.

First, notice the duplication of the code here -- let’s automate that with Radius

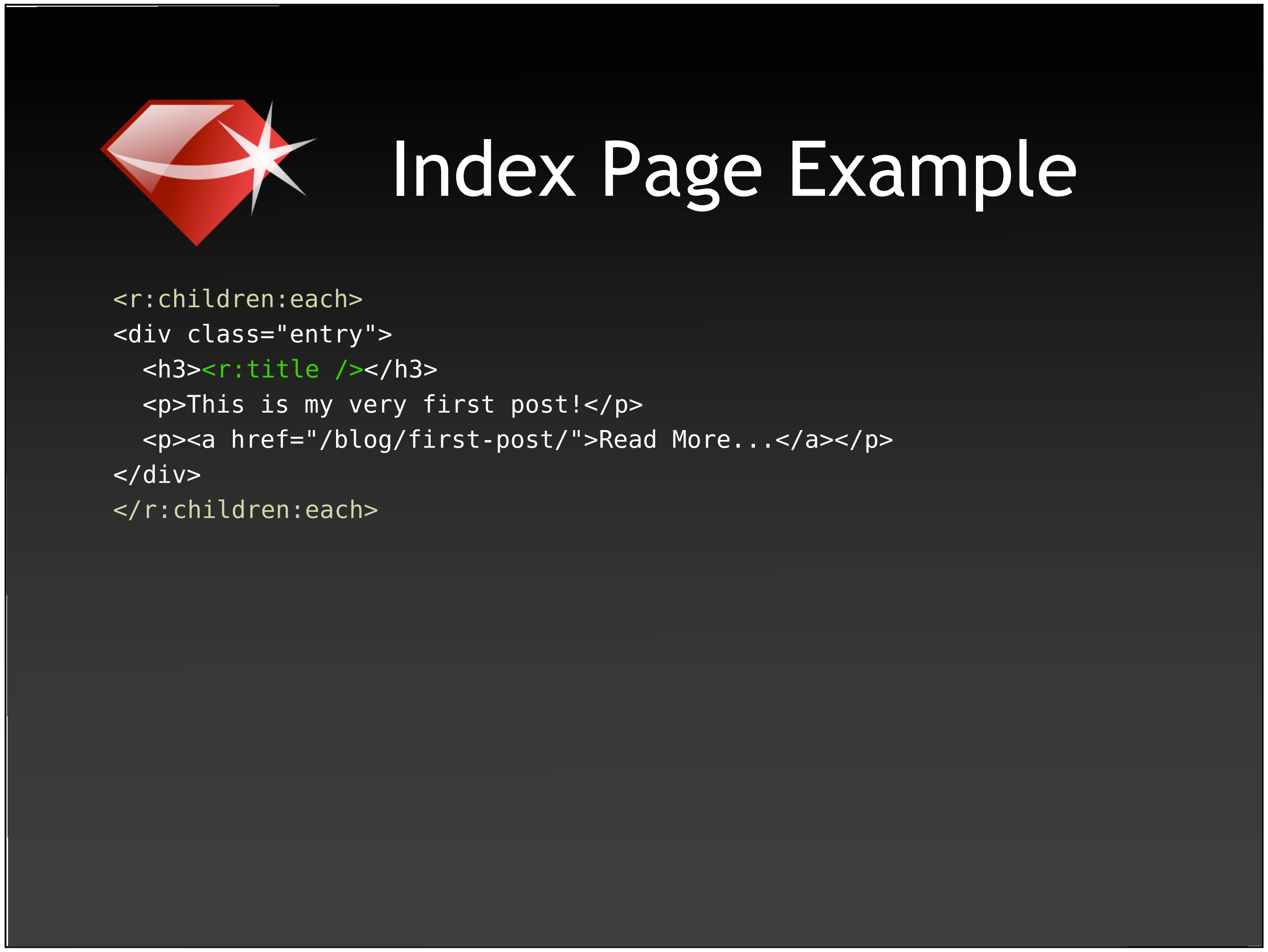


Index Page Example

```
<r:children:each>
<div class="entry">
  <h3>First Post</h3>
  <p>This is my very first post!</p>
  <p><a href="/blog/first-post/">Read More...</a></p>
</div>
</r:children:each>
```

First, since they have the same format, let’s just put those posts as child pages of the index page and iterate over them with <r:children:each>

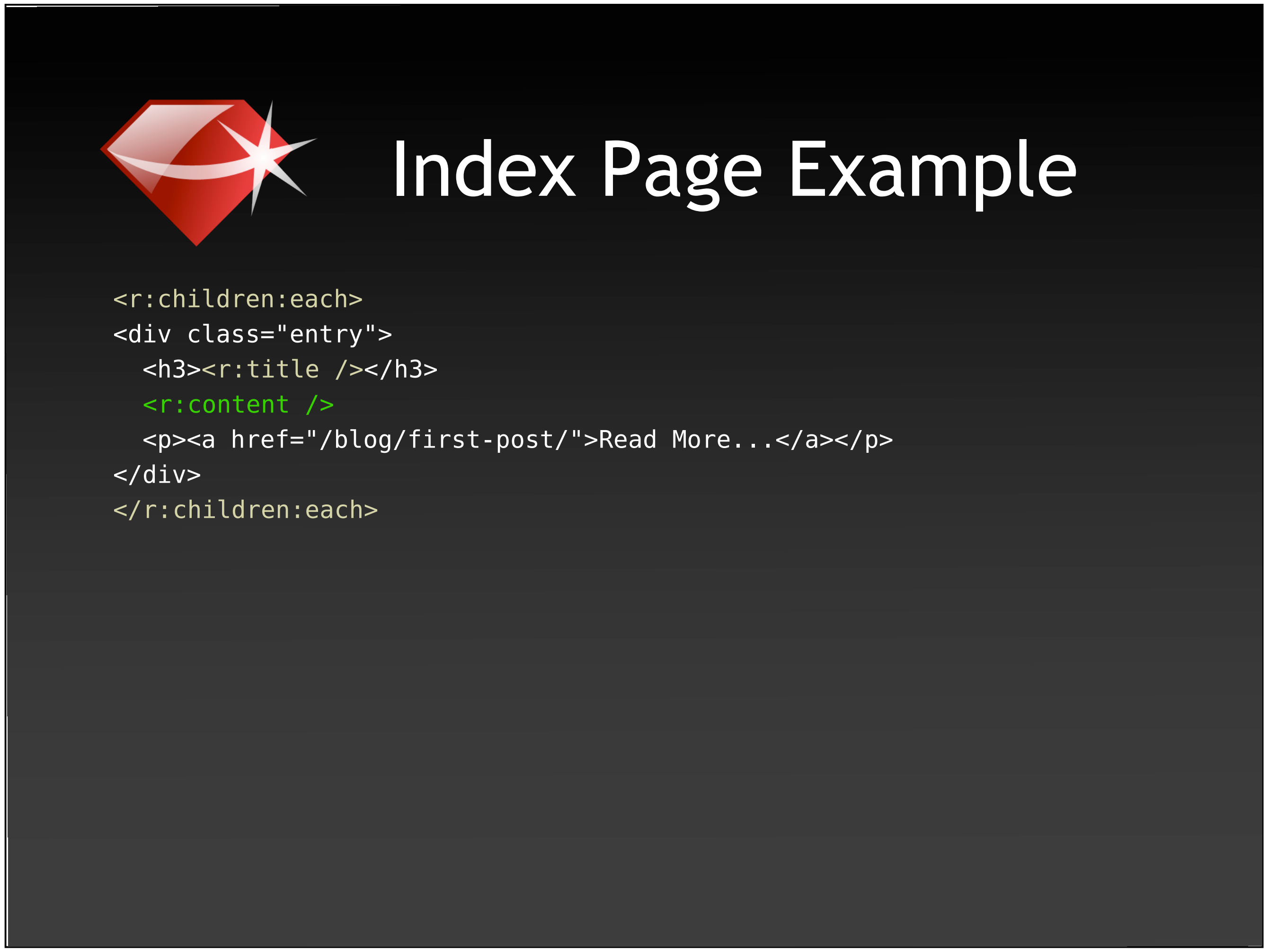
Inside this block, all tags refer to the current child in the iteration



Index Page Example

```
<r:children:each>
<div class="entry">
  <h3><r:title /></h3>
  <p>This is my very first post!</p>
  <p><a href="/blog/first-post/">Read More...</a></p>
</div>
</r:children:each>
```

That means we can replace the title

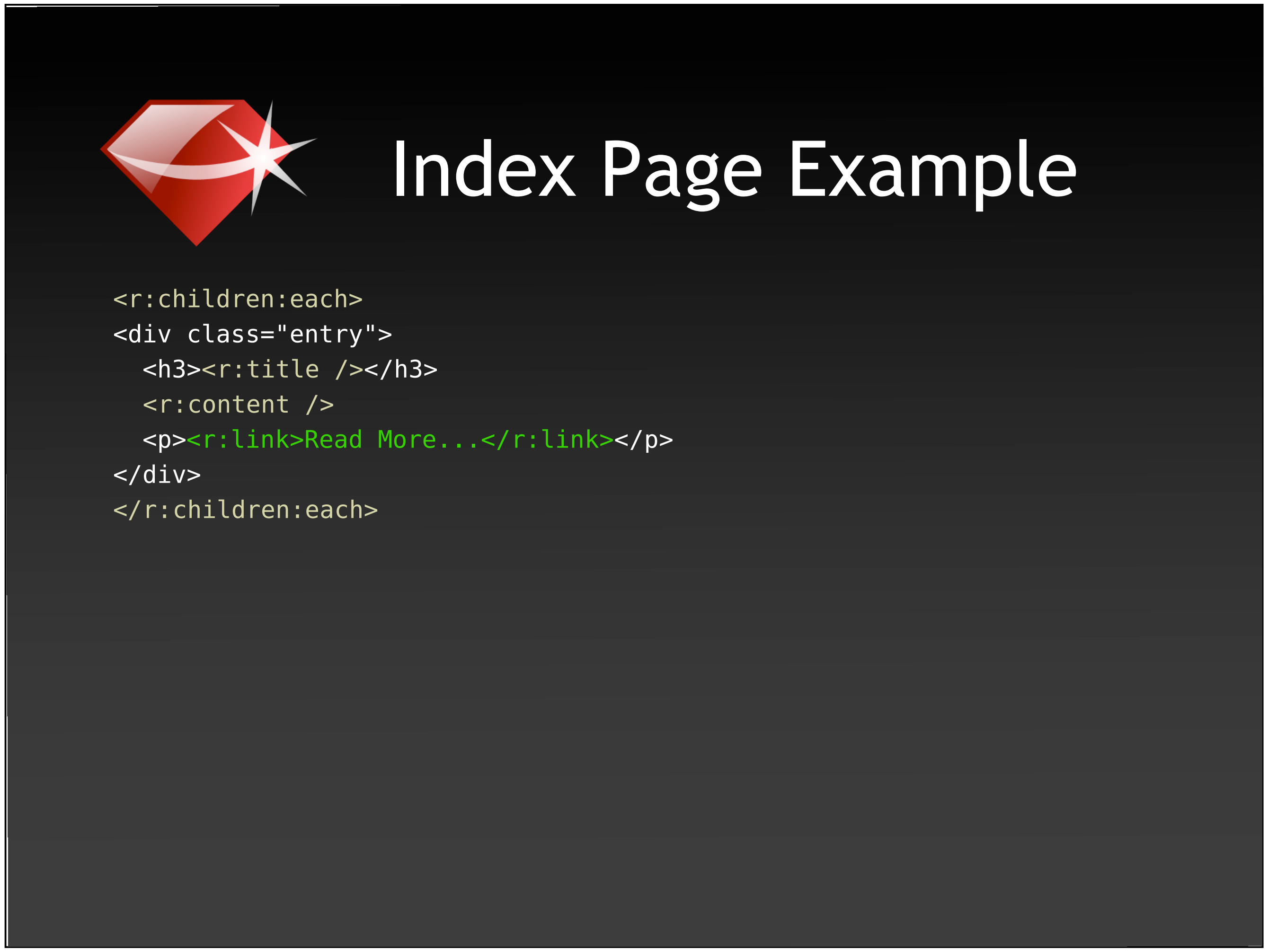


Index Page Example

```
<r:children:each>
<div class="entry">
  <h3><r:title /></h3>
  <r:content />
  <p><a href="/blog/first-post/">Read More...</a></p>
</div>
</r:children:each>
```

And the summary

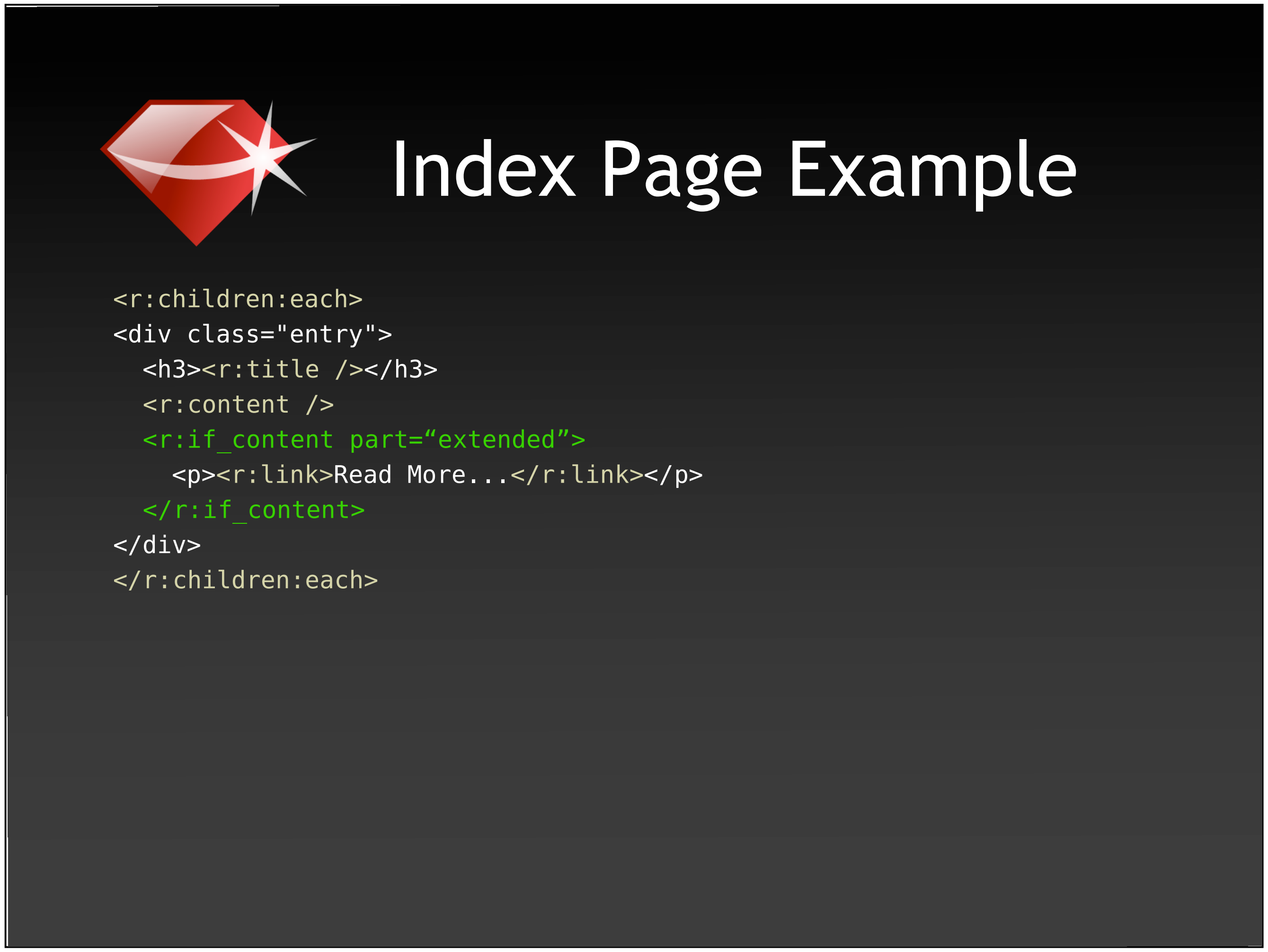
Note that without the “part” attribute, <r:content> renders the “body” part by convention



Index Page Example

```
<r:children:each>
<div class="entry">
  <h3><r:title /></h3>
  <r:content />
  <p><r:link>Read More...</r:link></p>
</div>
</r:children:each>
```

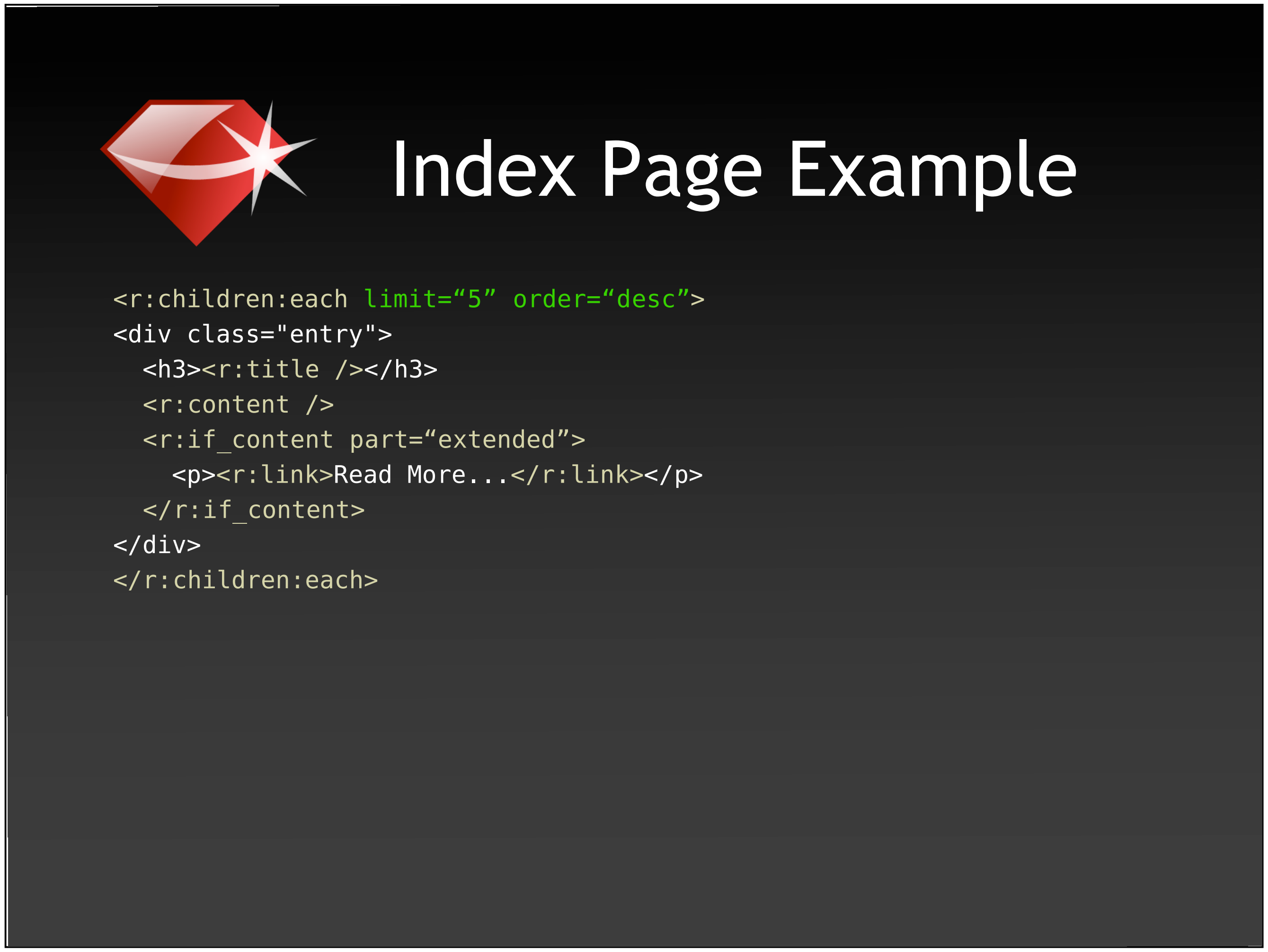
We can also generate a link to the full post with “Read more” in it



Index Page Example

```
<r:children:each>
<div class="entry">
  <h3><r:title /></h3>
  <r:content />
  <r:if_content part=“extended”>
    <p><r:link>Read More...</r:link></p>
  </r:if_content>
</div>
</r:children:each>
```

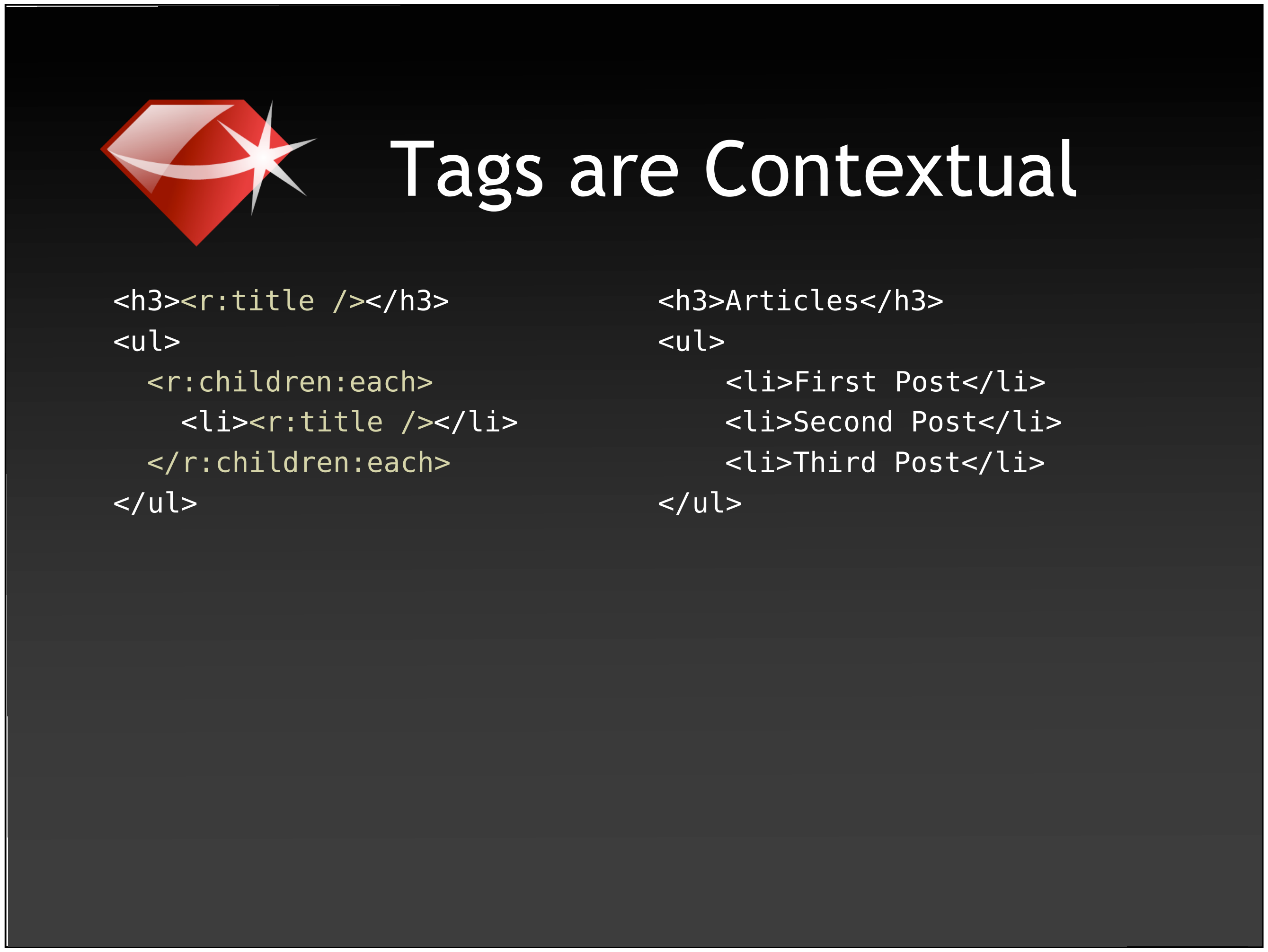
Not all blog posts might have more than the summary, so let’s only see the “Read more” link if there’s an “extended” part.



Index Page Example

```
<r:children:each limit=“5” order=“desc”>
<div class="entry">
  <h3><r:title /></h3>
  <r:content />
  <r:if_content part=“extended”>
    <p><r:link>Read More...</r:link></p>
  </r:if_content>
</div>
</r:children:each>
```

Finally, since this is a blog, let’s limit the front page to 5 posts and order them descending by publish date



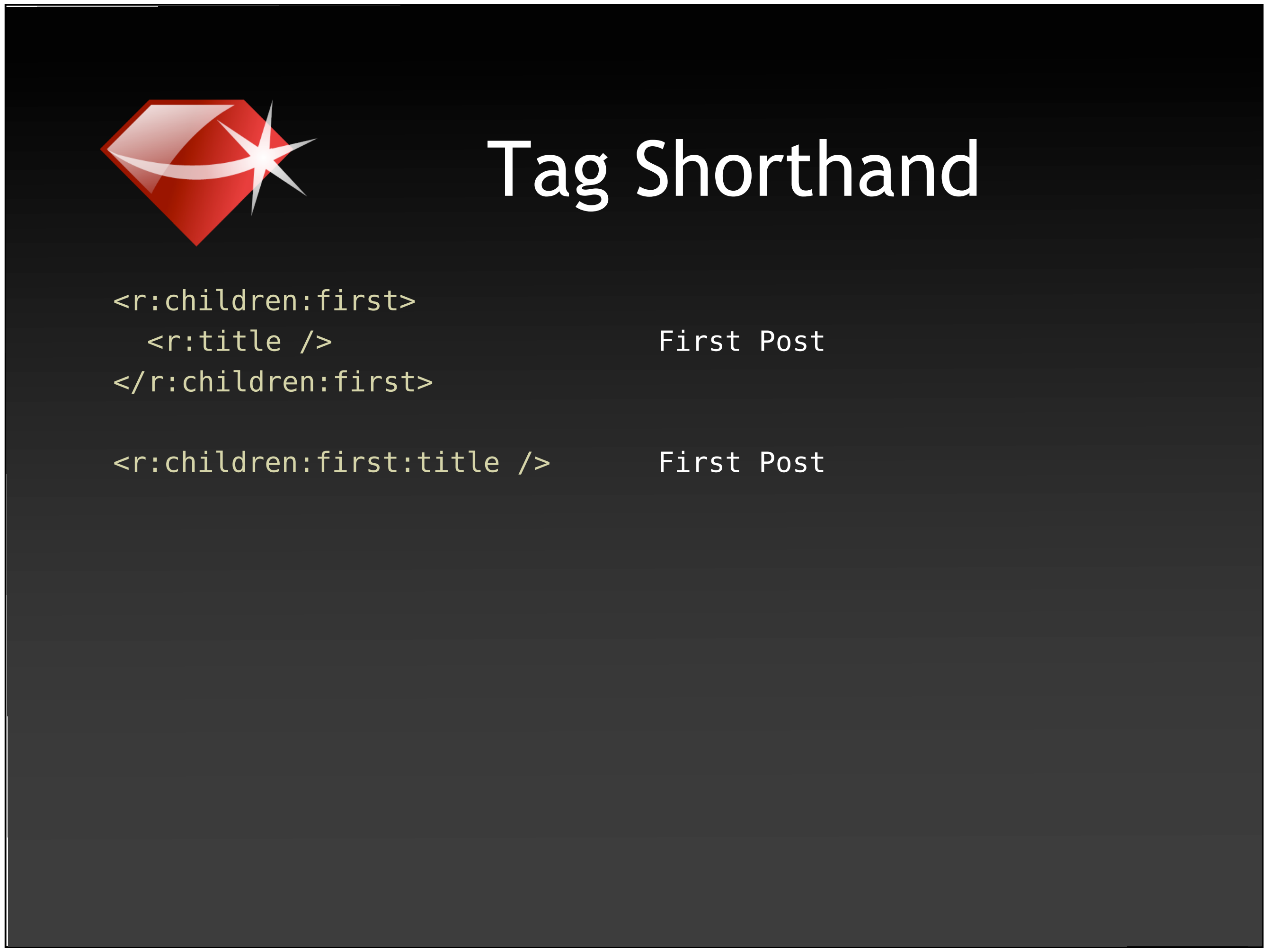
Tags are Contextual

```
<h3><r:title /></h3>
<ul>
  <r:children:each>
    <li><r:title /></li>
  </r:children:each>
</ul>
```

```
<h3>Articles</h3>
<ul>
    <li>First Post</li>
    <li>Second Post</li>
    <li>Third Post</li>
</ul>
```

Here’s a few more tips to getting a hang on tags

Tags respond to context, so if a containing tag changes something in the context, the contained tags will pick that up.

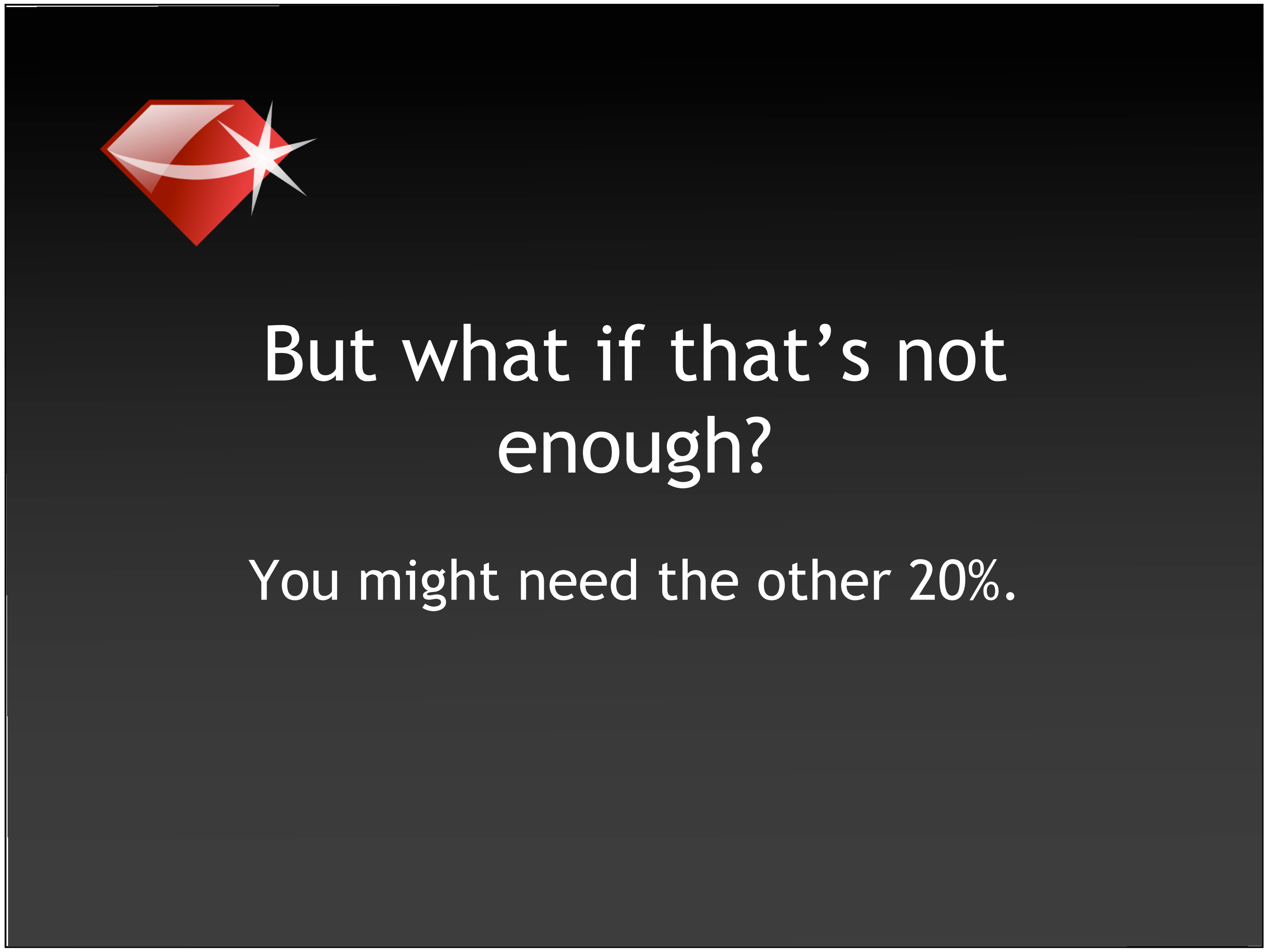


Tag Shorthand

```
<r:children:first>
  <r:title />                       First Post
</r:children:first>

<r:children:first:title />      First Post
```

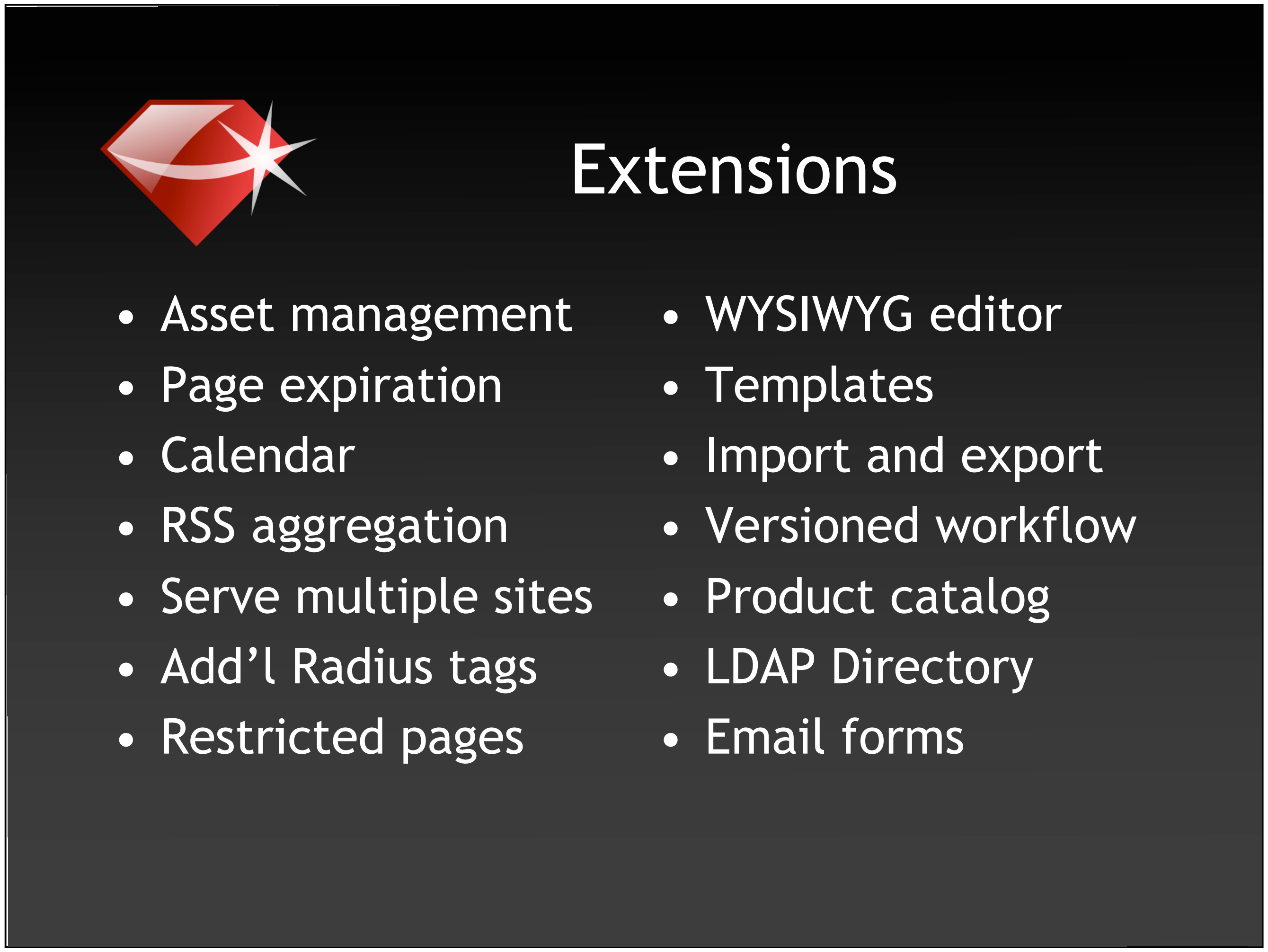
Radius Tags can be abbreviated when they don’t have any attributes or intervening text by chaining together colons inside the tag.



But what if that's not enough?

You might need the other 20%.

We realize not every content-management problem is going to be solved just by what Radiant provides. We also want to keep the core code of the application clean and free of cruft. So we created a first-class system to customize Radiant to suit your needs, which we call "extensions".

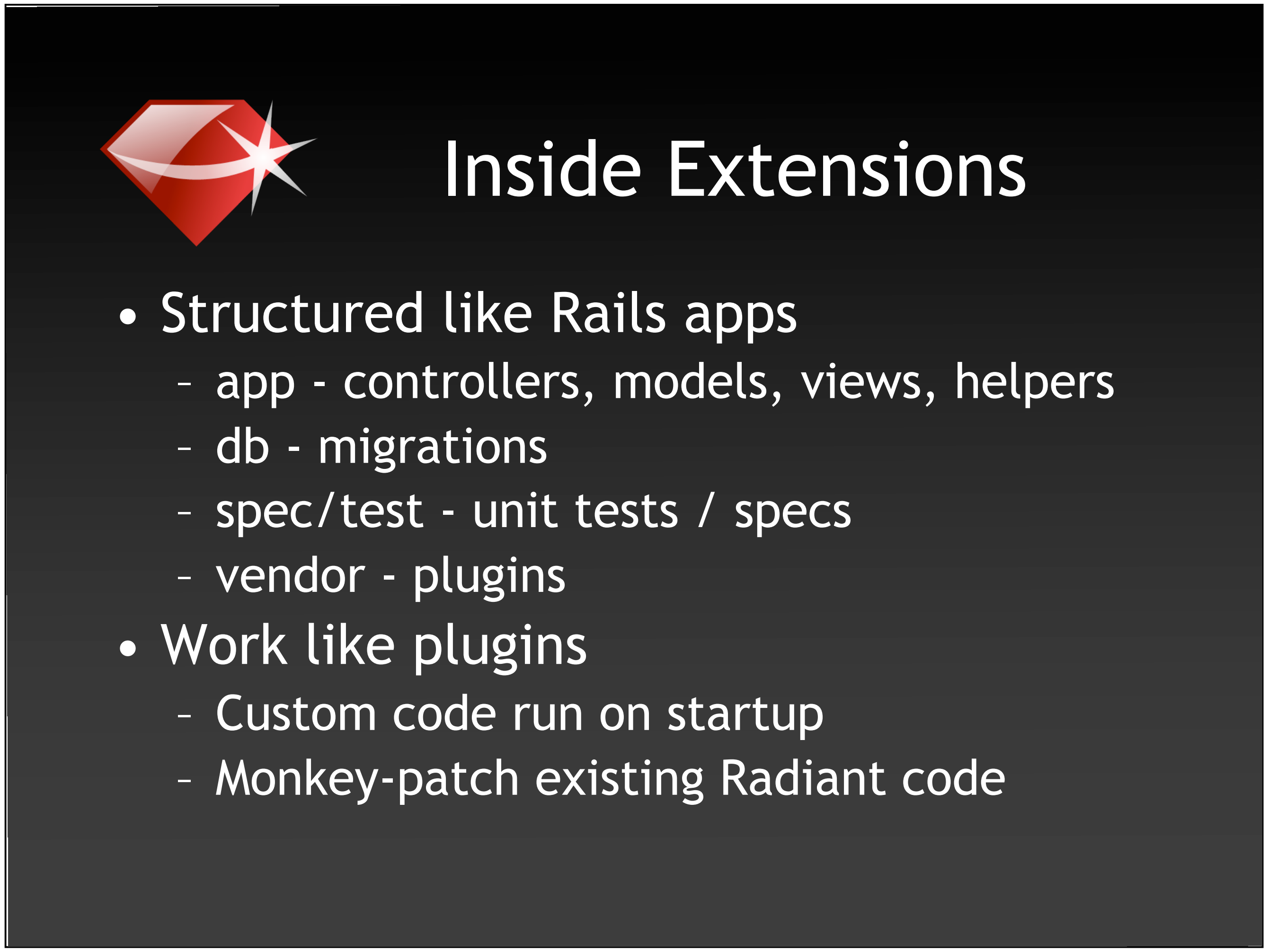


Extensions

- Asset management
- Page expiration
- Calendar
- RSS aggregation
- Serve multiple sites
- Add’l Radius tags
- Restricted pages
- WYSIWYG editor
- Templates
- Import and export
- Versioned workflow
- Product catalog
- LDAP Directory
- Email forms

Extensions can do pretty much anything you can program in Ruby.

These are some examples of extensions that the community has produced. (list a few)

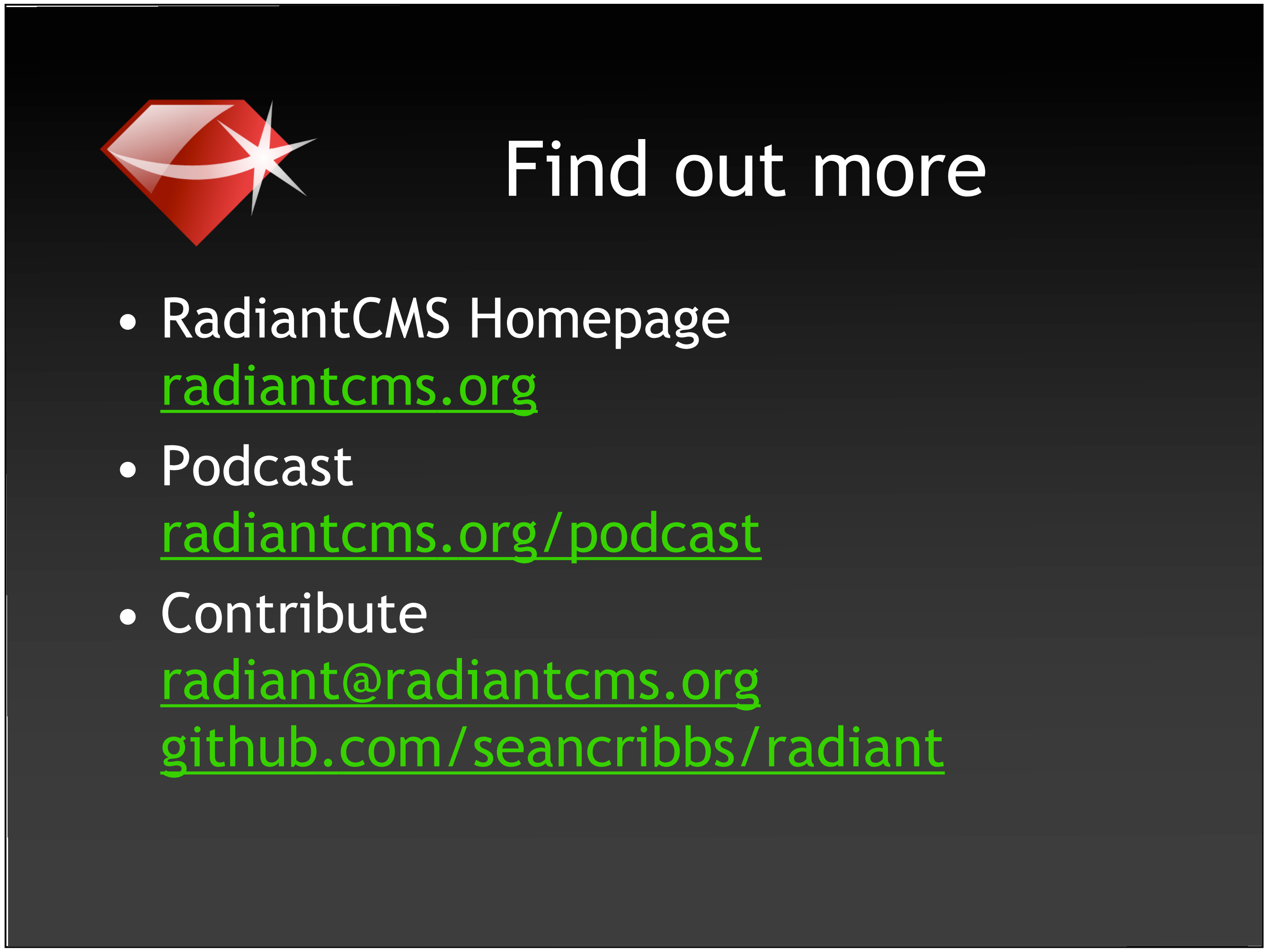


Inside Extensions

- Structured like Rails apps
  - app - controllers, models, views, helpers
  - db - migrations
  - spec/test - unit tests / specs
  - vendor - plugins
- Work like plugins
  - Custom code run on startup
  - Monkey-patch existing Radiant code

We also made it fairly easy to create them, at least for devs familiar with Rails.

Site and Extensions Demo

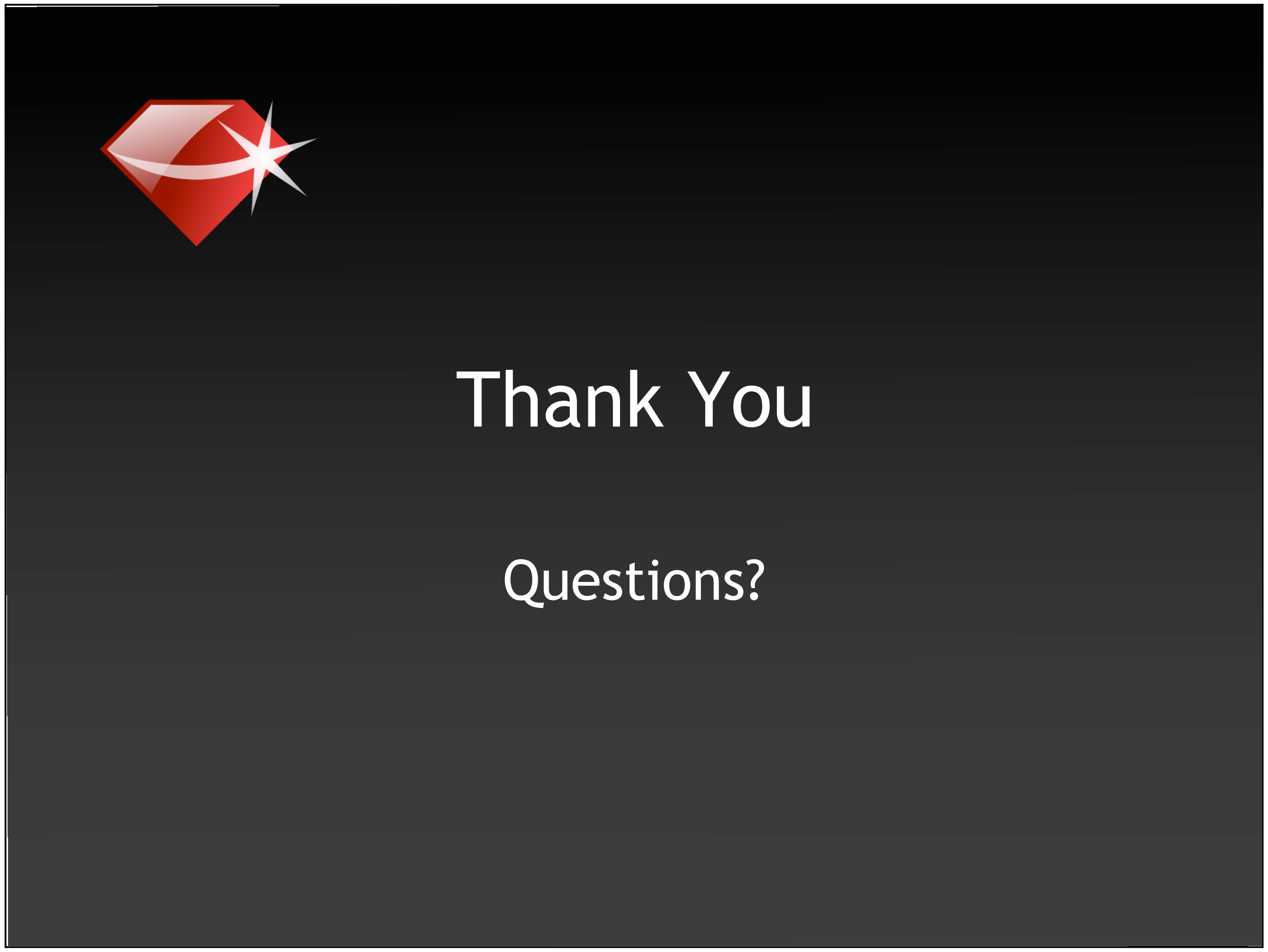


Find out more

- RadiantCMS Homepage
  radiantcms.org
- Podcast
  radiantcms.org/podcast
- Contribute
  radiant@radiantcms.org
  github.com/seancribbs/radiant

Coming Soon

SaaS Radiant Hosting
by Yours Truly



Thank You

Questions?