

JAMming with Gentics Mesh, React Static and Amazon S3

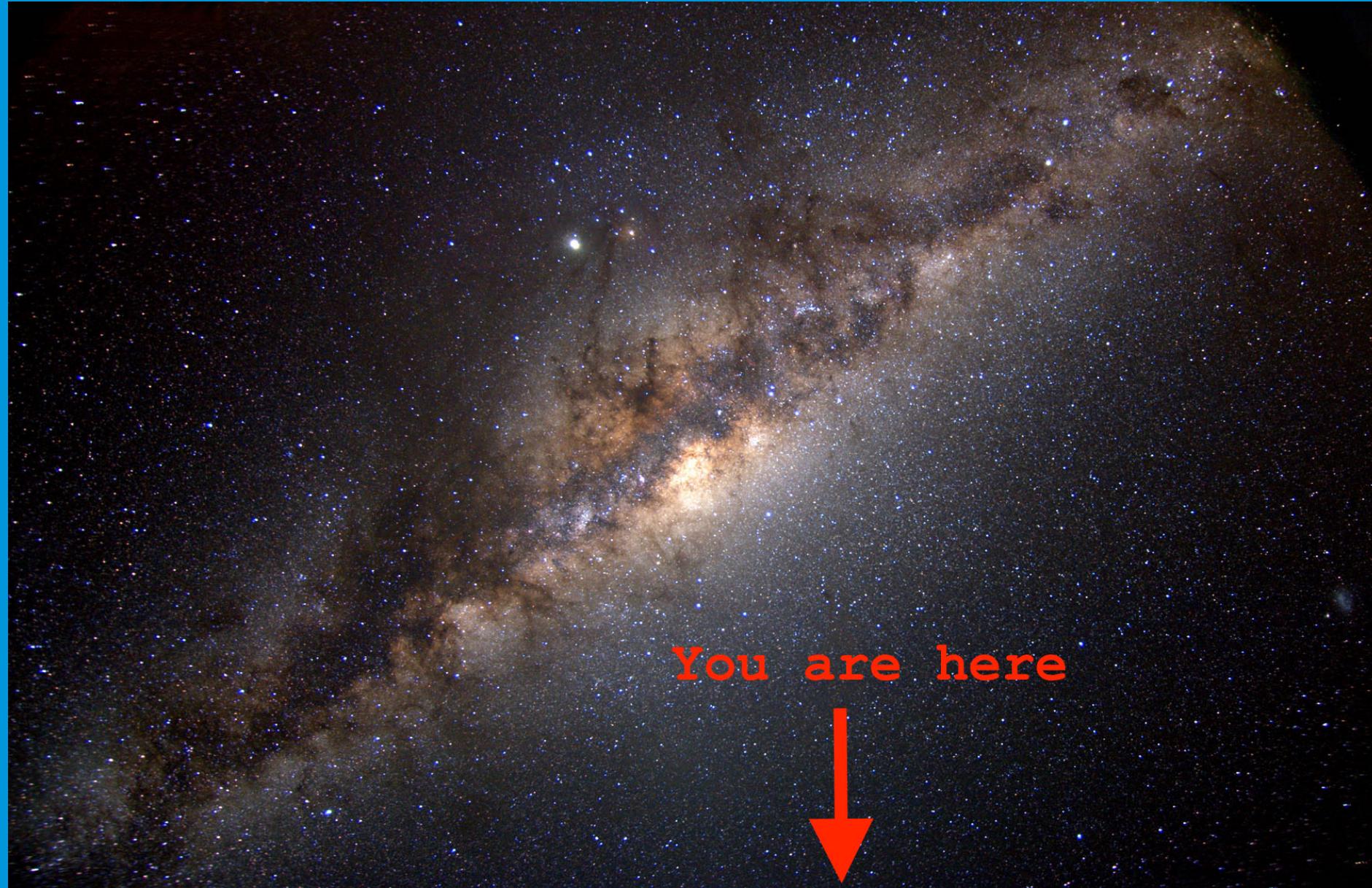
We Are Developers Workshop / Vienna, May 17th, 2018

Agenda

1. Intro talk
2. Gentics Mesh: tour of features
3. Break
4. React Static
5. Features implementation walk-through
6. Amazon S3
7. Wrap-up

Intro talk

What brought you here?



Credits

Episode VIII

THE FRONT-END STRIKES BACK

The credits and references for the images, quotes, ... are listed at the end of the presentation...

Web Content Projects



1. (Internal) content publication
2. (External) user generated content
3. 6-12 months projects
4. 10x of page types
5. 100x of components
6. 1,000x of content items
7. 100,000x of users

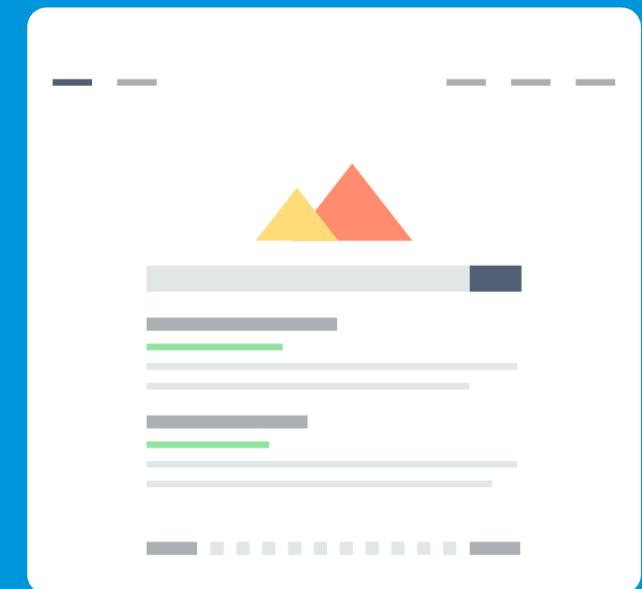
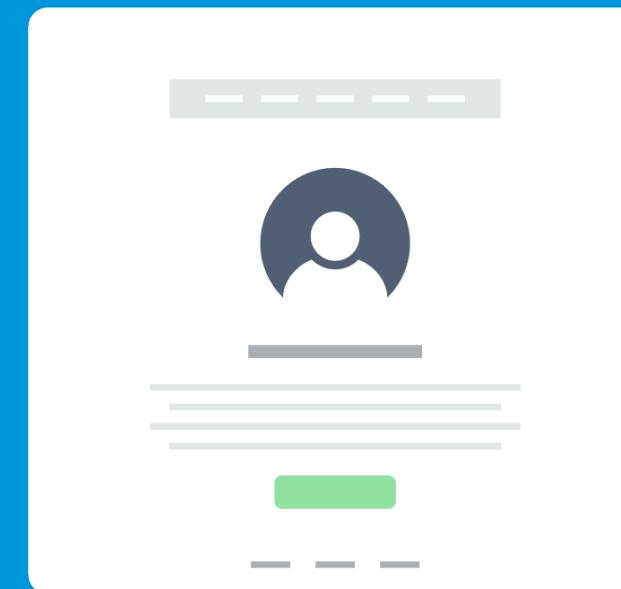
Implemented on a **coupled CMS**.

The content stays the same

Web front-end design
changes every 2 years but
the content remains the same



Thinking (and implementing!) in components



SEO, SEO, SEO

The screenshot shows the Google PageSpeed Insights interface. At the top, it says "PageSpeed Tools > Insights". Below that is a blue navigation bar with "HOME", "GUIDES", "REFERENCE", and "SUPPORT" buttons. The main area is titled "PageSpeed Insights" and shows the URL "http://carmen-marcos.art/" in a field with an "ANALYZE" button. There are two tabs: "Mobile" (selected) and "Desktop". A summary card displays "Speed" as "Unavailable" and "Optimization" as "Good" (80 / 100). Below this, a message states that real-world performance data is unavailable but PageSpeed Insights can still analyze the page for optimizations. It also provides PSI estimates for page load times and resources. The "Page Stats" section includes a "Show how to fix" link for render-blocking content. The "Optimization Suggestions" section lists three items: "Eliminate render-blocking JavaScript and CSS in above-the-fold content" (with a "Show how to fix" link), "Leverage browser caching" (with a "Show how to fix" link), and "Enable compression" (with a "Show how to fix" link). On the right, there's a preview of a mobile phone displaying the website's content, which includes exhibition details and a painting of purple flowers.

1. Page speed: Time to first byte (TTFB)
2. Light-weight markup
3. Pre-rendered markup vs render markup per request
4. Nice & structured URLs
5. Breadcrumbs with metadata
6. ...



The Anatomy of a CMS

1. Content **modeling**
2. Existing content **migration**
3. Content **creation/editing**
4. Querying / search / API integrations
5. **Permissions:** who can do what (users, groups, ...)
6. **Content rendering (frameworks, libraries, ...)**
7. ...

What if...



1. CMS back-end & front-end were completely **separated**?
2. Content editors use an internal web interface?
3. Front-end could be **developed with any tech** or framework?

... the CMS would just focus on content



The CMS* is dead!

* Content Management System



The CMS* is dead!

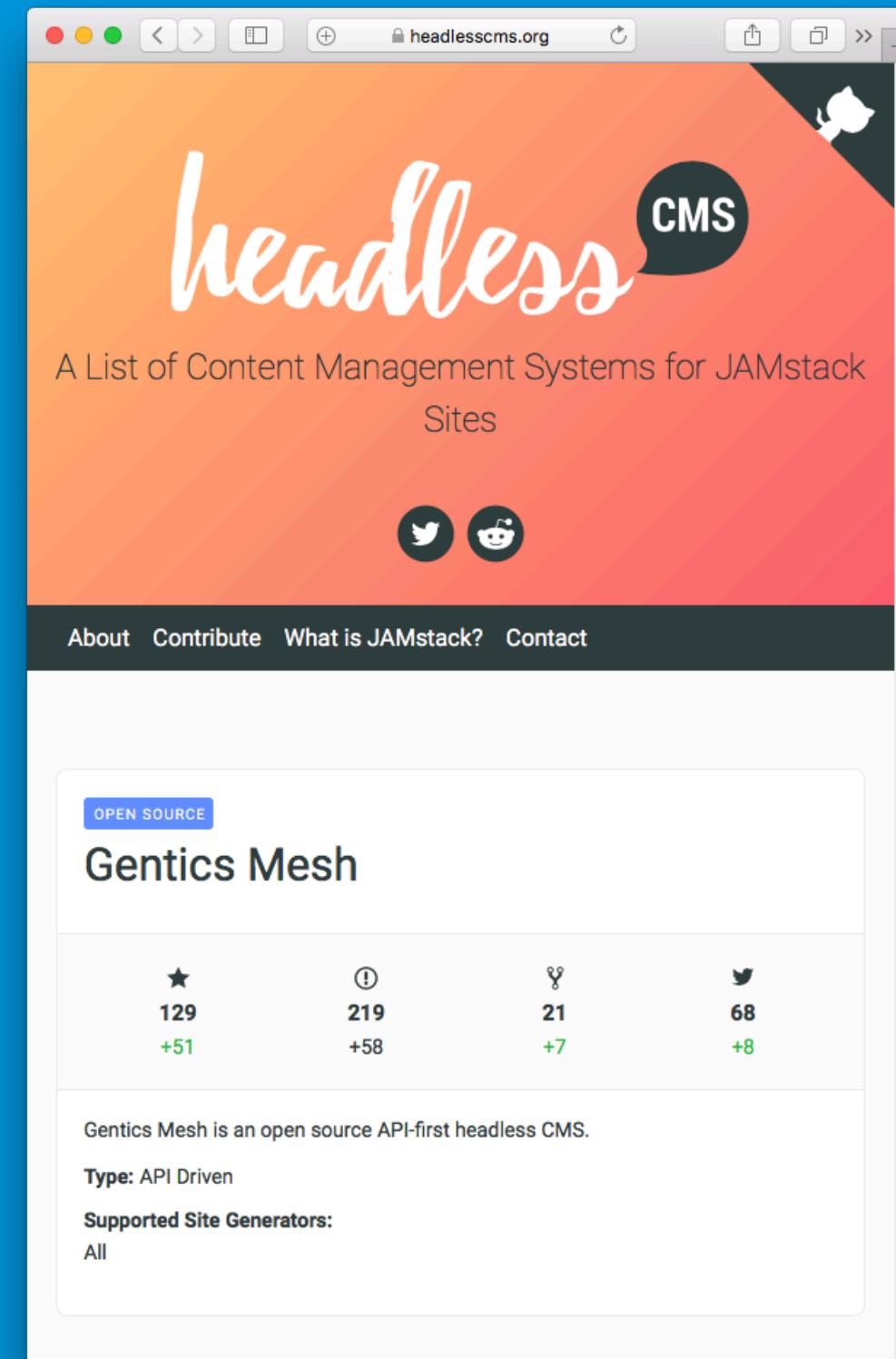
* Content Management System

Long Live the CMS*

* Content Micro-Service

a.k.a. Headless CMS

a.k.a. API-driven CMS



The content stays the same

Headless / API-first CMSS

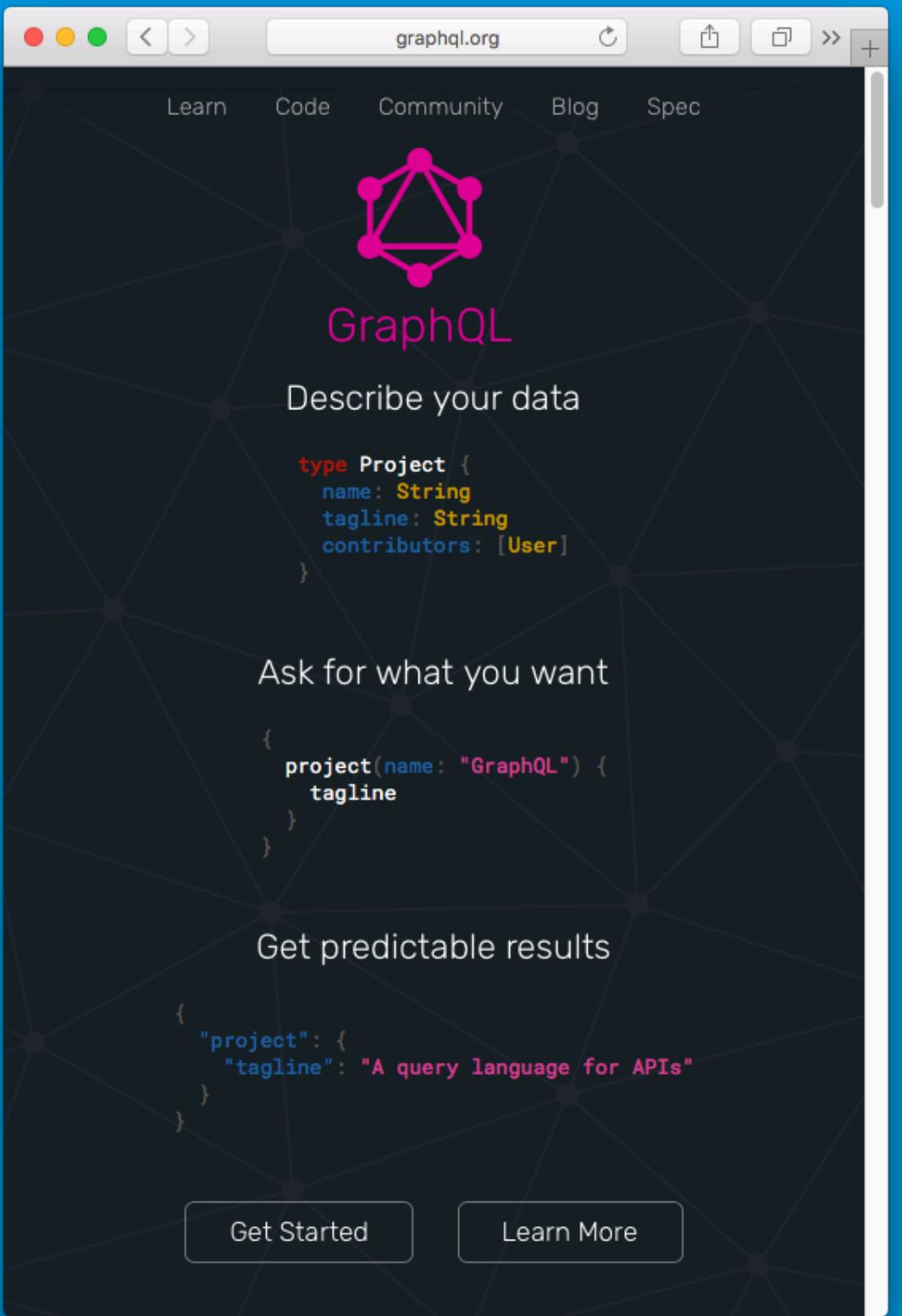
1. contentful (SaaS)
2. prismic (SaaS)
3. GraphCMS (SaaS)
4. Contenta CMS (Open-Source)
5. ...
6. Gentics Mesh (Open-Source)

... all with different **content APIs!**

Content APIs: O Content, where art thou?



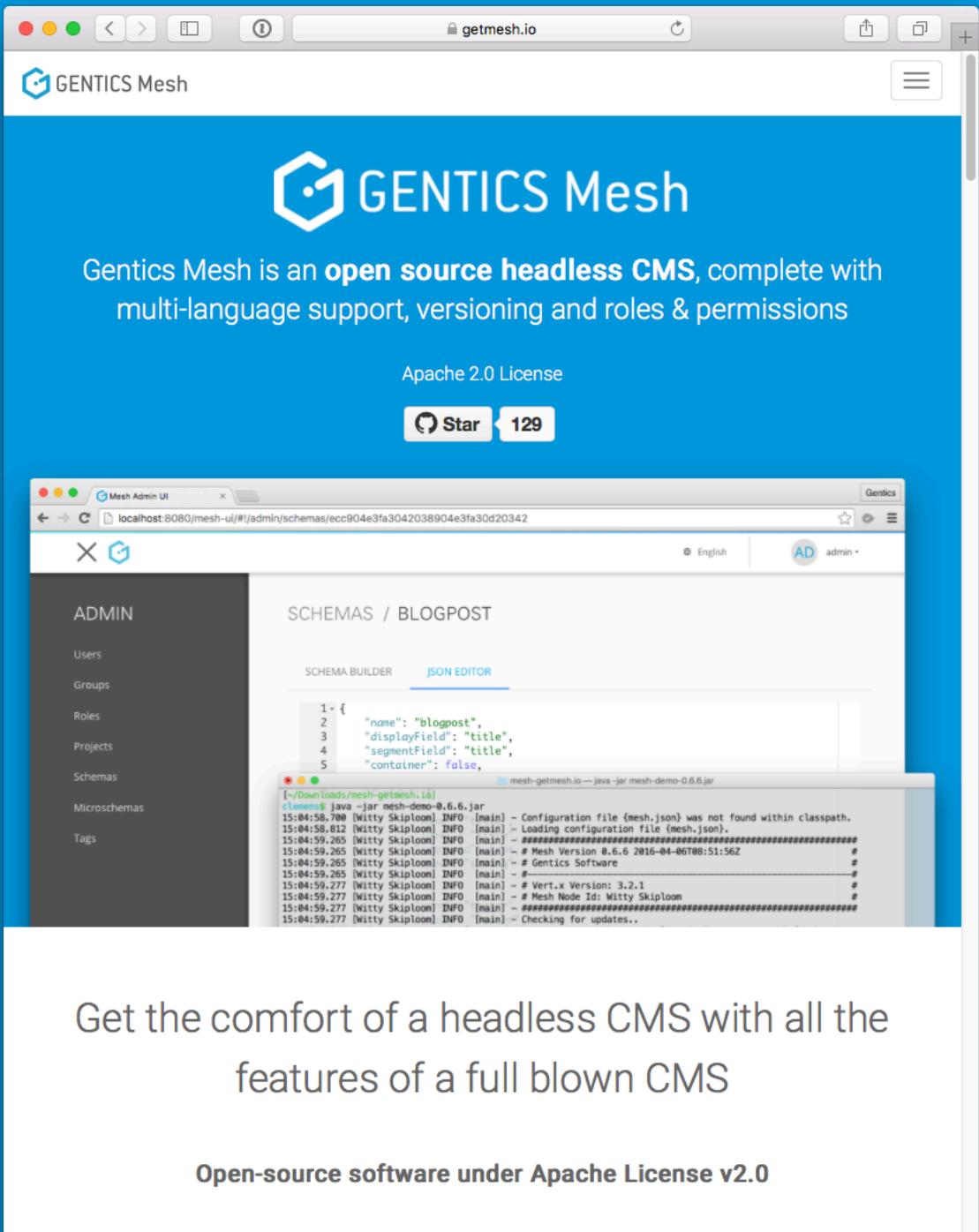
"I Am A [Dev] of Constant Sorrow"
— Soggy Bottom [Devs]



Content APIs: O Content, where art thou?

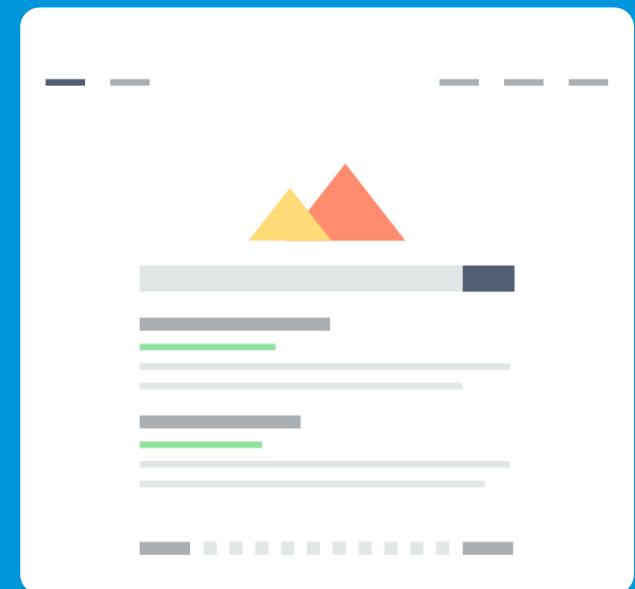
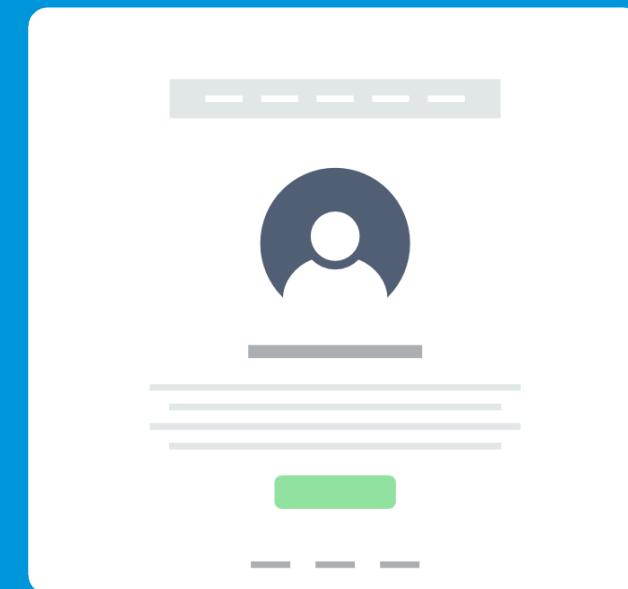
1. Proliferation of content repository APIs adds overhead for developers to learn about the **semantics** of the API
2. One query API language to rule them all?
3. **GraphQL support is a MUST HAVE.**

Headless CMS: Gentics Mesh

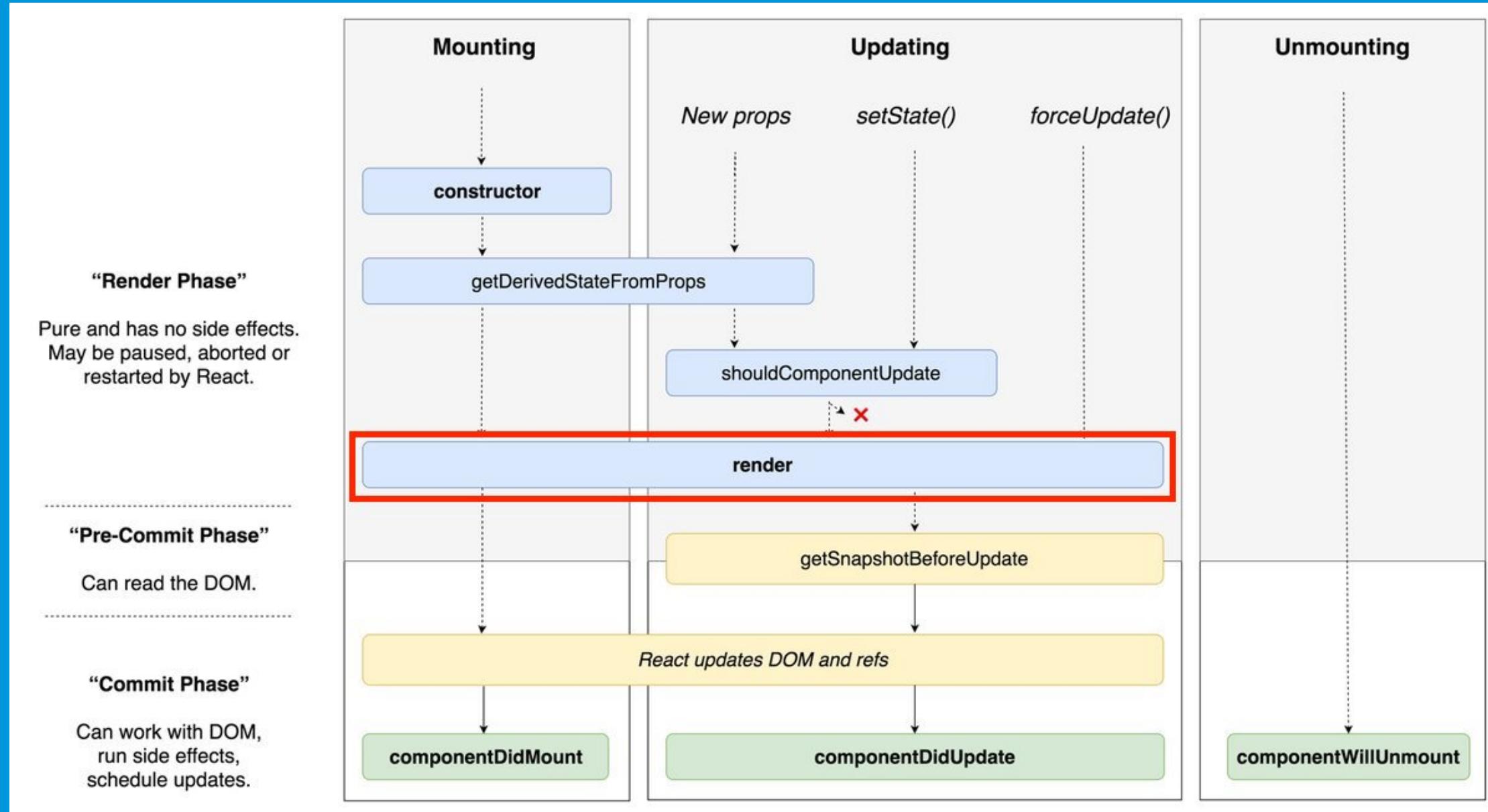


1. (Hierarchical) **content tree**: scalability!
2. APIs: REST, **GraphQL**, ElasticSearch
3. **Image manipulation**: via API & via management UI
4. **Multi-lingual support**
5. On-(cloud)-premise: **own your content!**
6. Users, groups, roles & permissions
7. Content **migrations**

React: it's components all the way down!



React Static: my gateway drug to React



SEO, SEO, SEO



SEARCH OPTIMIZATION

JAMstack

1. **JavaScript:** the language of the web
2. **APIs:** to access content
3. **Markup:** pre-rendered HTML markup

Amazon S3 for content delivery

1. Managed, scalable and cheap

/ The browser is the new server! /

Main takeways (1/2)



As a back-end developer:

1. Do you build your own search engine, business process engine, ...?
No you don't.
2. Do you build your own content management infrastructure?
You should not.

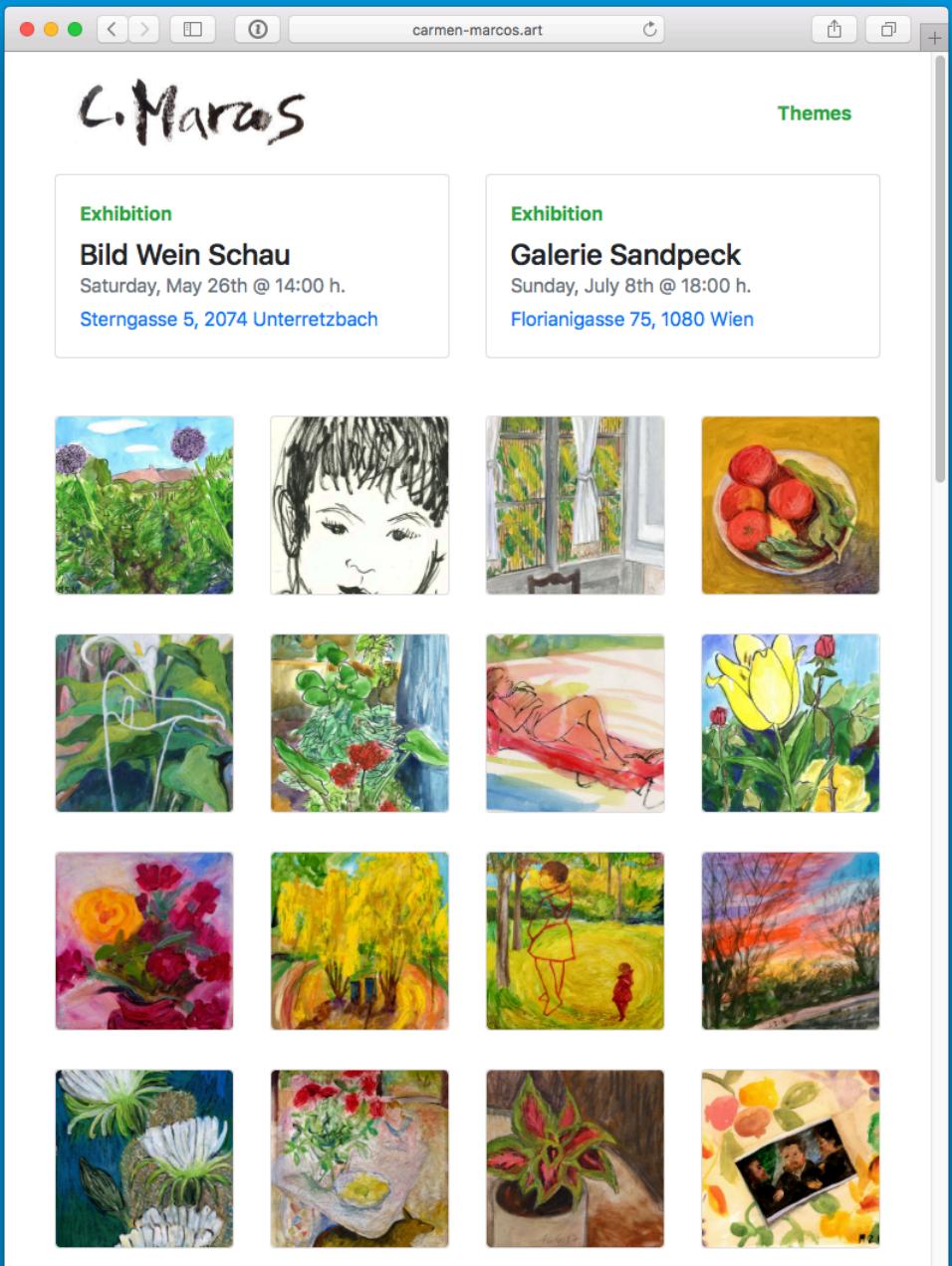
Main takeways (2/2)



As a front-end developer

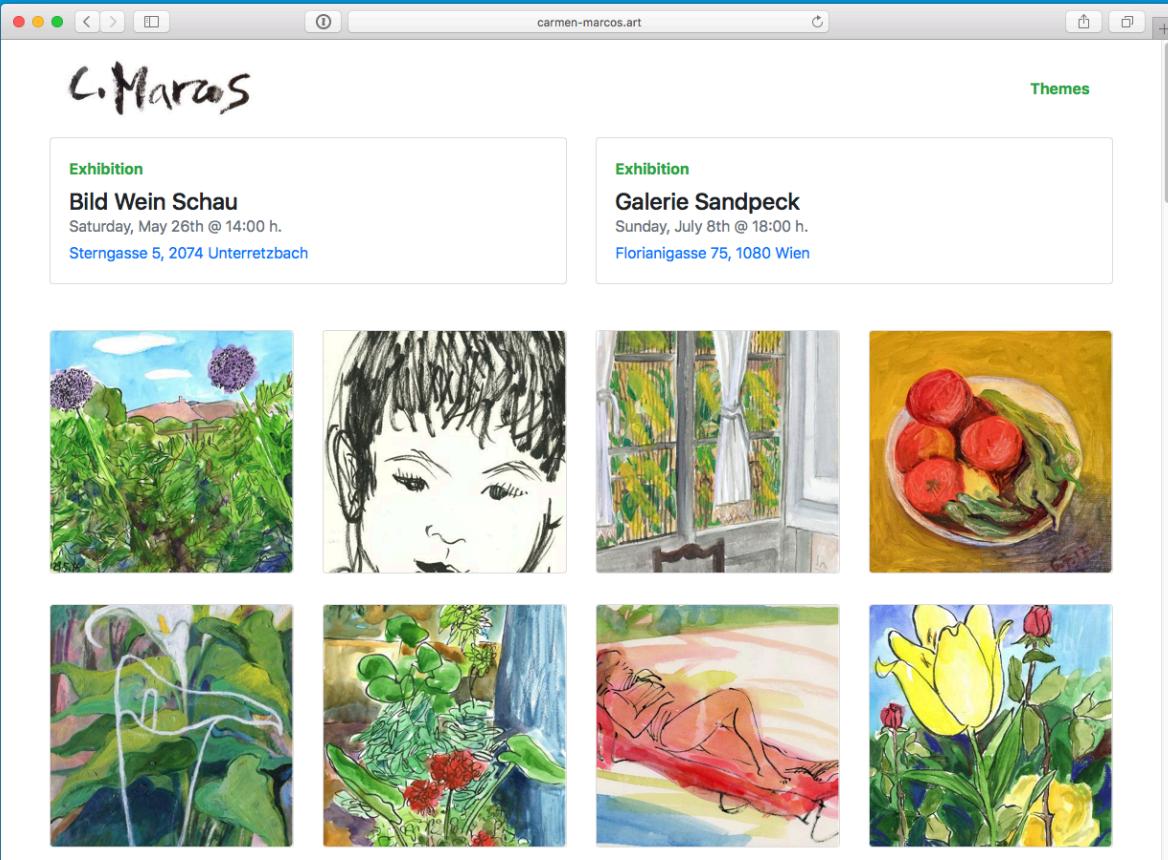
1. Abstract away the management aspect of content
2. Focus on the front-end implementation with freedom of choice

The Project



1. ...
2. prismic.io + (Scala) Play! Framework
3. ...
4. Contentful + (JavaScript) Angular 1.x
5. ...
6. GraphCMS + (JavaScript) GatsbyJS
7. **Gentics Mesh + (JavaScript) React Static**

The Project



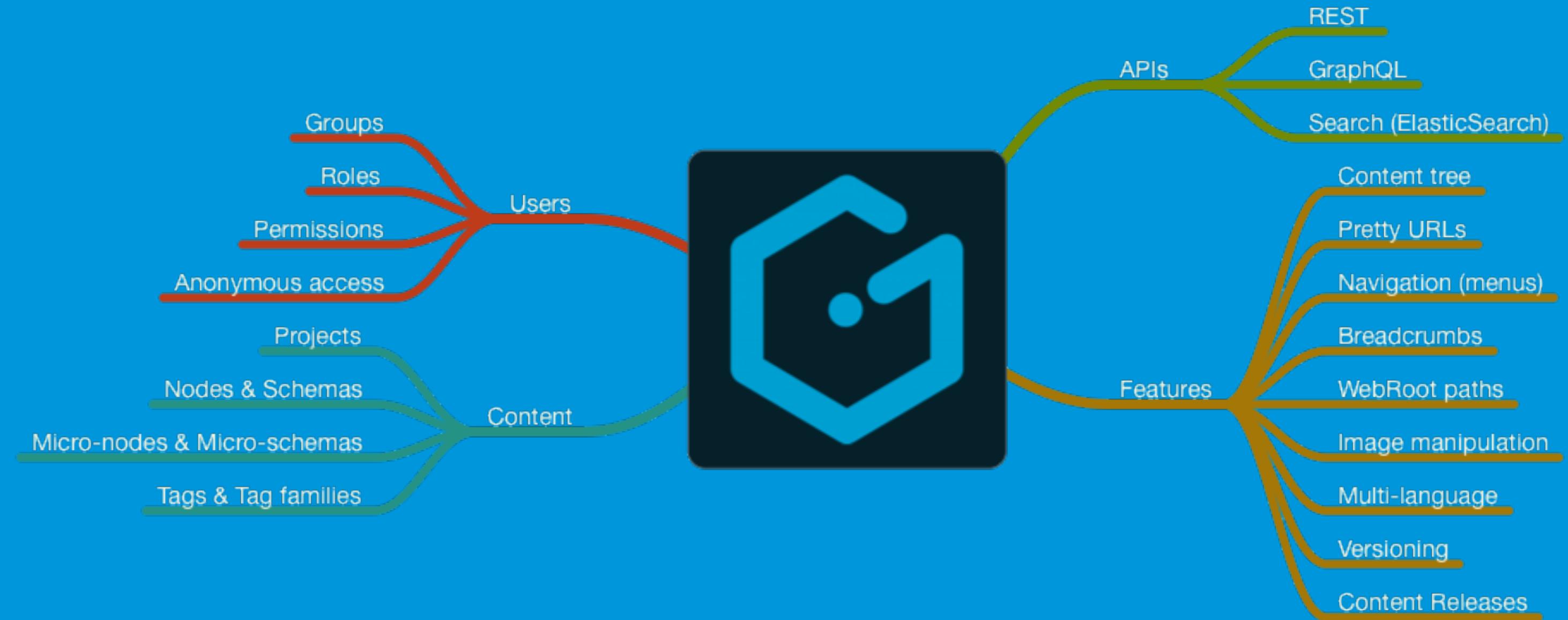
<https://github.com/rafacm/carmen-marcos-art-portfolio>

Any questions?!



Gentics Mesh: Tour de Features

Gentics Mesh: Tour de Features



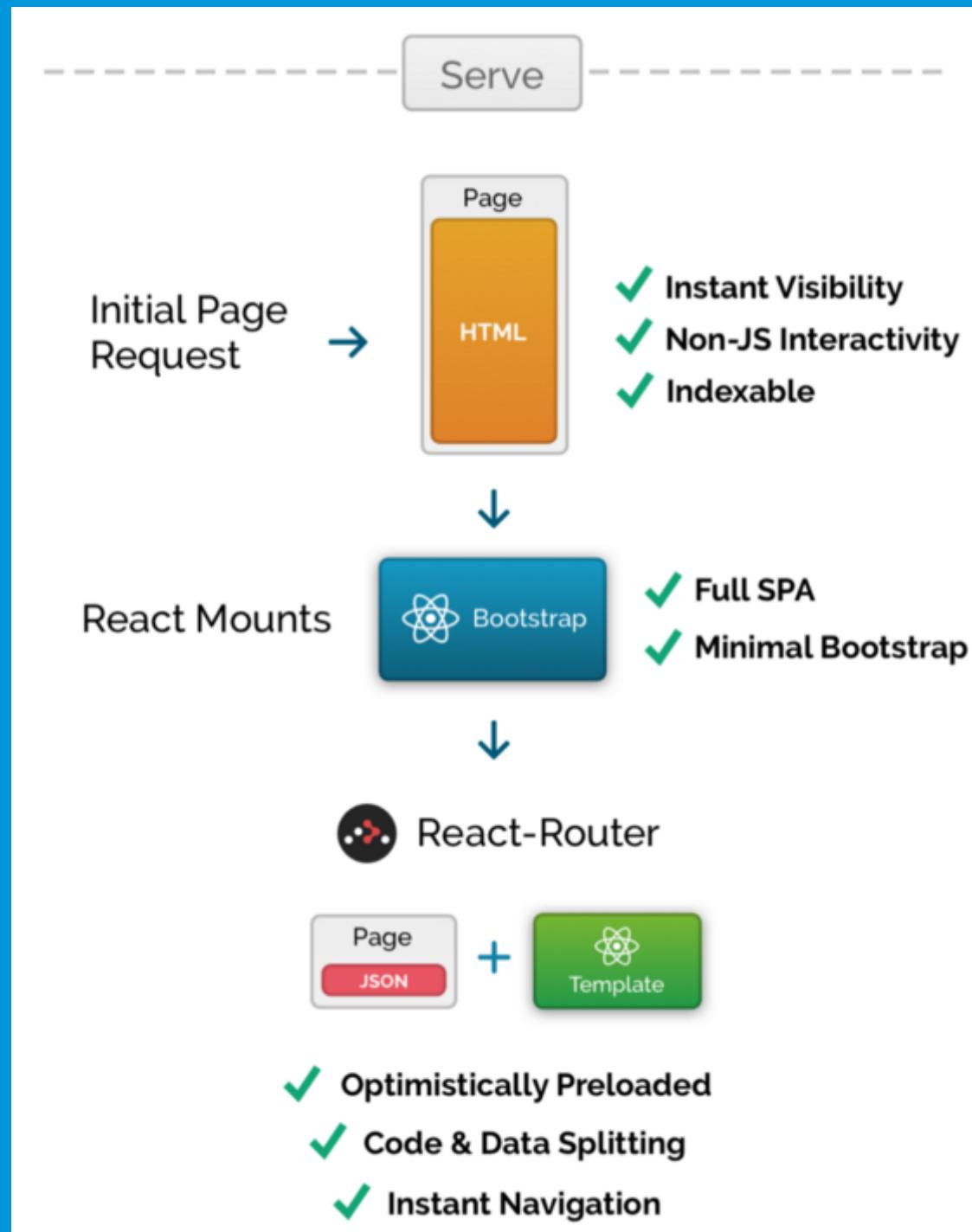
React Static

React Static: Develop



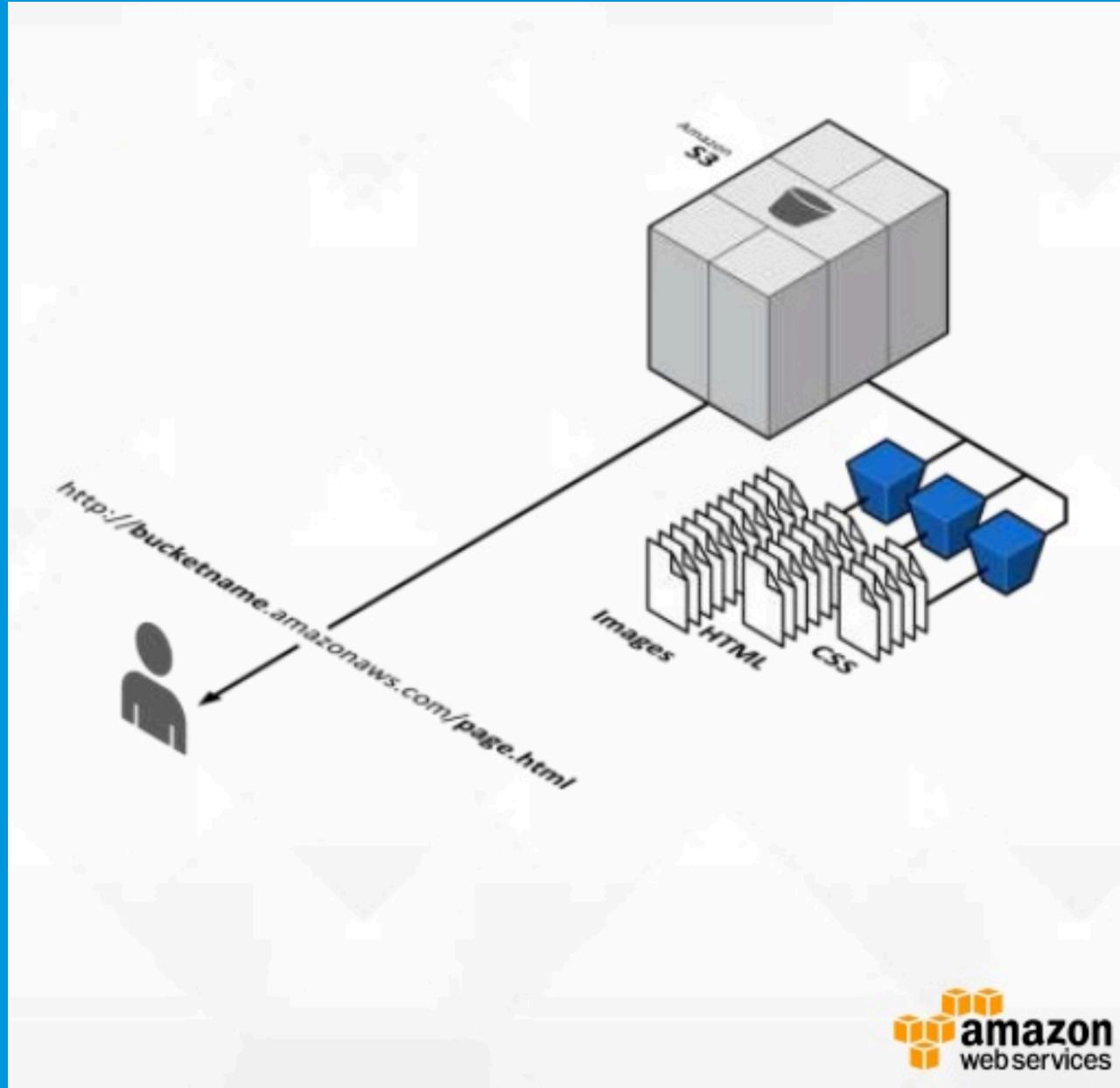
1. Low conceptual complexity:
 1. `getSiteData() -> HOC`
 2. `getRouteData() -> HOC`
2. Source data agnostic
3. Hot-reload
4. Webpack customization possible

React Static: Serve



1. Not only **static**
2. Full-blown React SPA available too
3. Pre-fetches `<Link to={}>` components

Amazon S3 for content delivery



1. S3 is the static web server
2. S3 is fully managed and scales
3. (Optionally) add CDN capabilities with CloudFront
4. ACHTUNG! Images are still hosted on Gentics Mesh
5. JavaScript in the browser is still available (JAM)
6. JavaScript in the browser has access to Gentics Mesh APIs (JAM)

Thanks for your time!

Credits & References

- Slide 4: [Astronomy Picture of the Day](#)
- Slide 5: [Star Wars Opening Crawl generator](#)
- Slide 9: [PageSpeed Insights](#)
- Slide 10: [The Anatomy Lesson of Dr. Nicolaes Tulp](#)
- Slide 12: [Execution of Louis XVI](#)
- Slide 13: [Execution of Louis XVI](#)
- Slide 15: [Man of constant sorrow](#)

Credits & References

- Slide 16: [GraphQL](#)
- Slide 20: [Headless CMS](#)
- Slide 17: [Gentics Mesh](#)
- Slide 19: [React lifecycle methods](#)
- Slide 21: [JAMStack](#)
- Slide 23: [Website & Source](#)
- Slide 24: [Website & Source](#)

Credits & References

- Slide 25: [Spies Like Us](#)
- Slide 27: [Gentics Mesh](#)
- Slide 29: [React Static](#)
- Slide 30: [React Static](#)
- Slide 31: [AWS May Seminar Series](#)