



JAVASCRIPT IS COMING TO EAT YOU



Citizen Tim

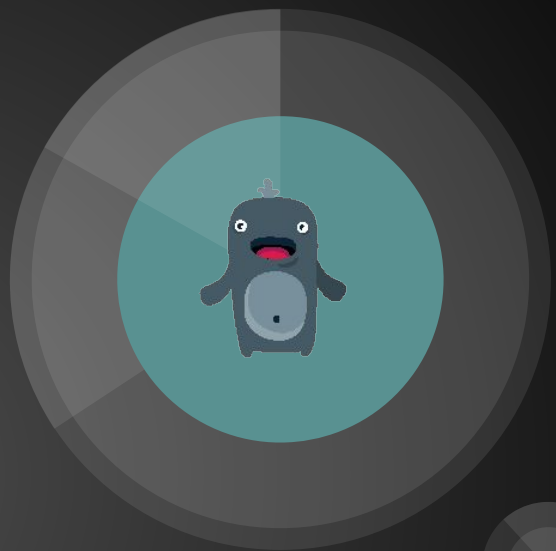
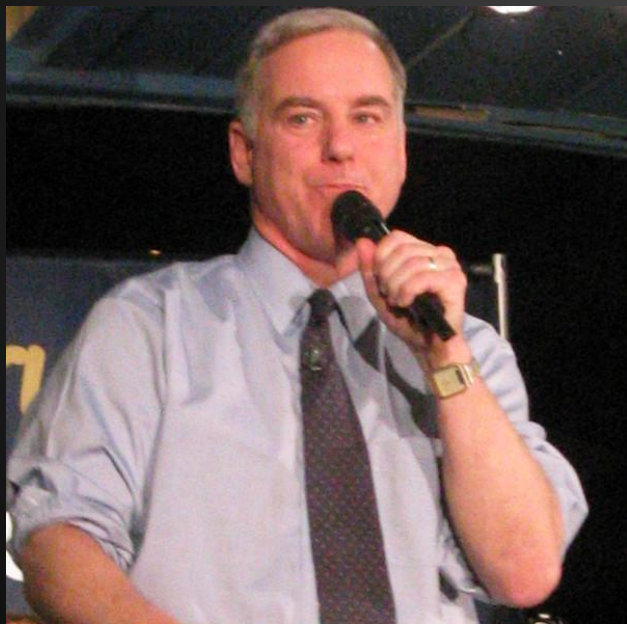
Electric Citizen | October 2019



January 2004



The Dean Screem



How we build the web

1994
to
2004

The Renaissance

2004
to
2014

The Industrial
Revolution

2014
to
Today

The Technological
Revolution



2004 to 2014

The Industrial Revolution

- Rise and domination of the CMS
- Drupal, Joomla, and WordPress
- SaaS/Commercial CMS
- LAMP stack
- The Cloud
- Ajax/Jquery (Gmail)



2014 to Today

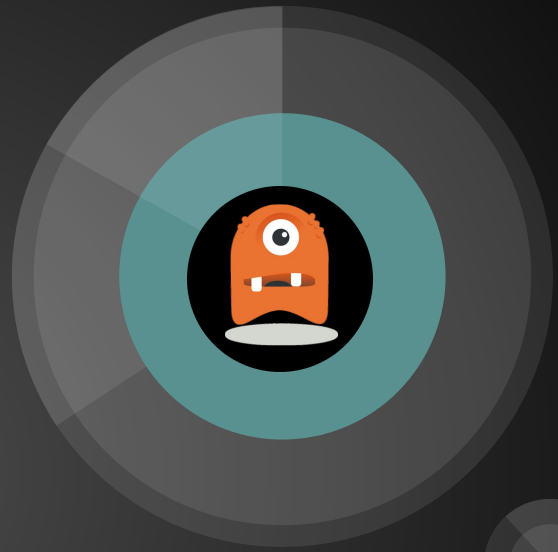
The Technological Revolution

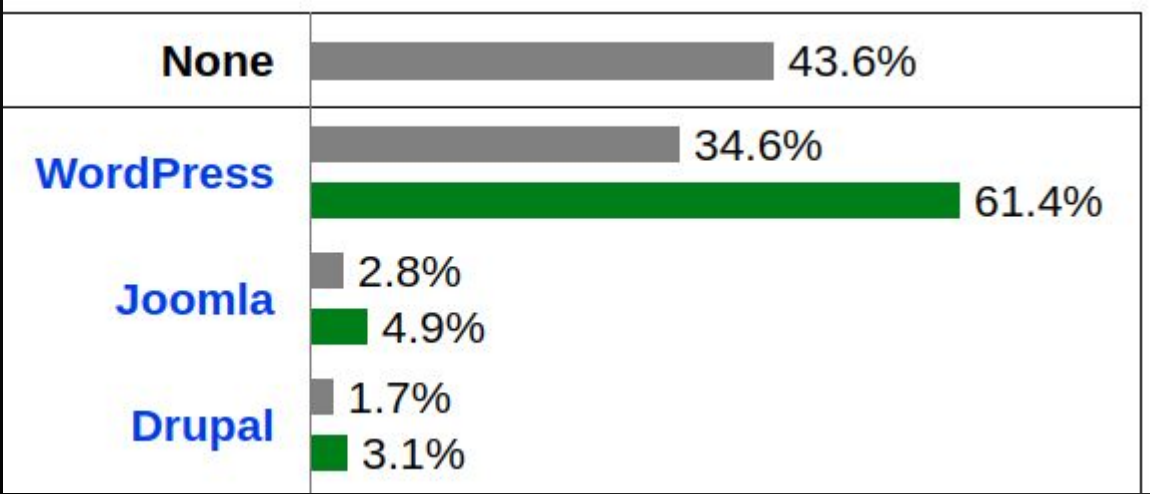
- React/Angular/Vue
- Decoupled CMS
- Node.js and JS libraries
- JAMstack and APIs
- Managed services
- Serverless



**Most of us are still
living in 2014.**

And it is a great place to be.





Winning all the things.

- We enjoy a billion dollar global marketplace
- Innovation continues at great speed
- Drupal or WordPress are still an obvious choice for most websites



Come, Watson, come!
The game is afoot.

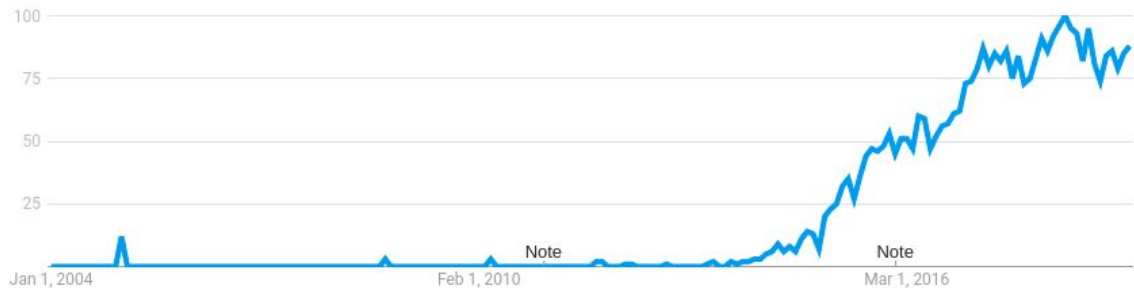


REACT & ANGULAR

2004 to present

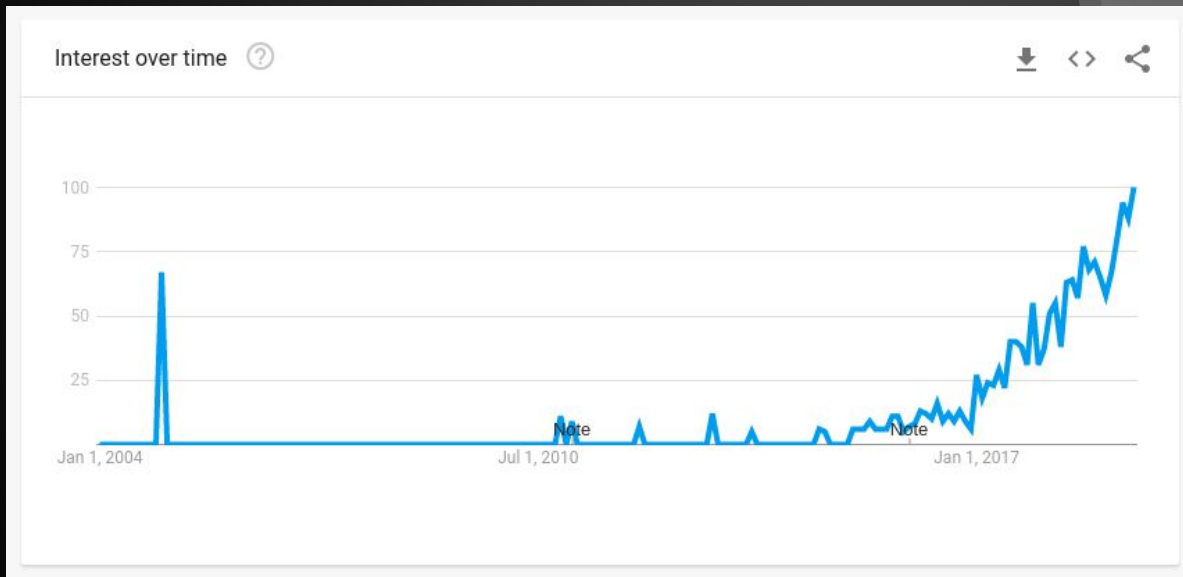


Interest over time ?



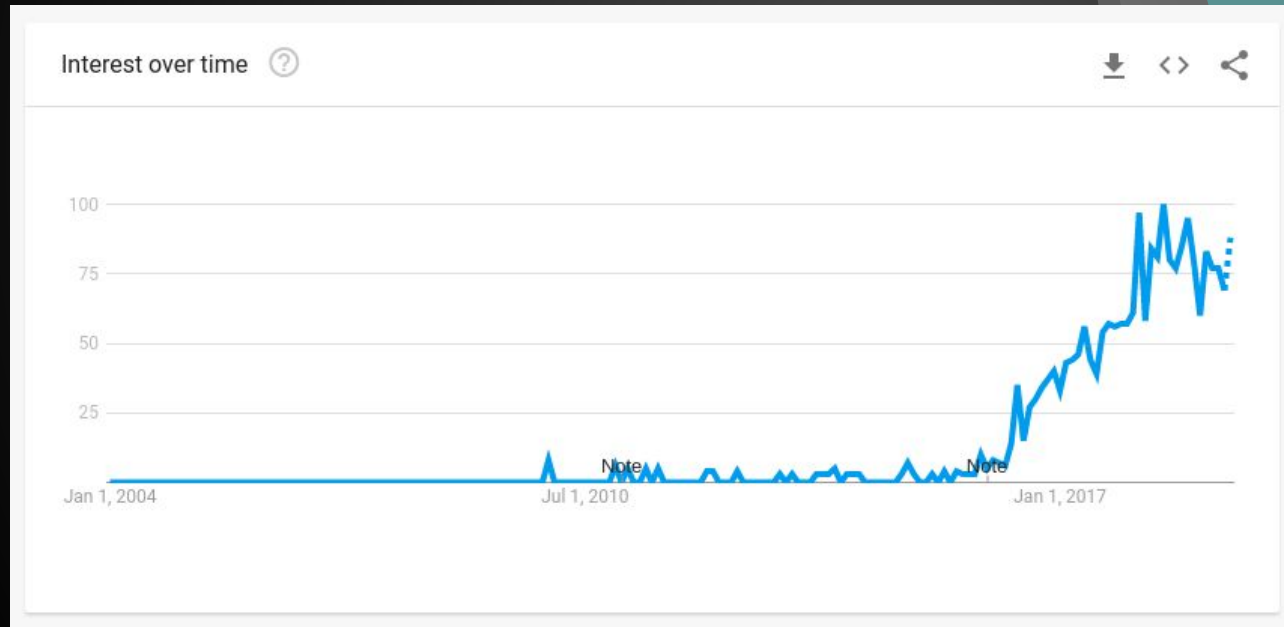
HEADLESS CMS

2004 to present



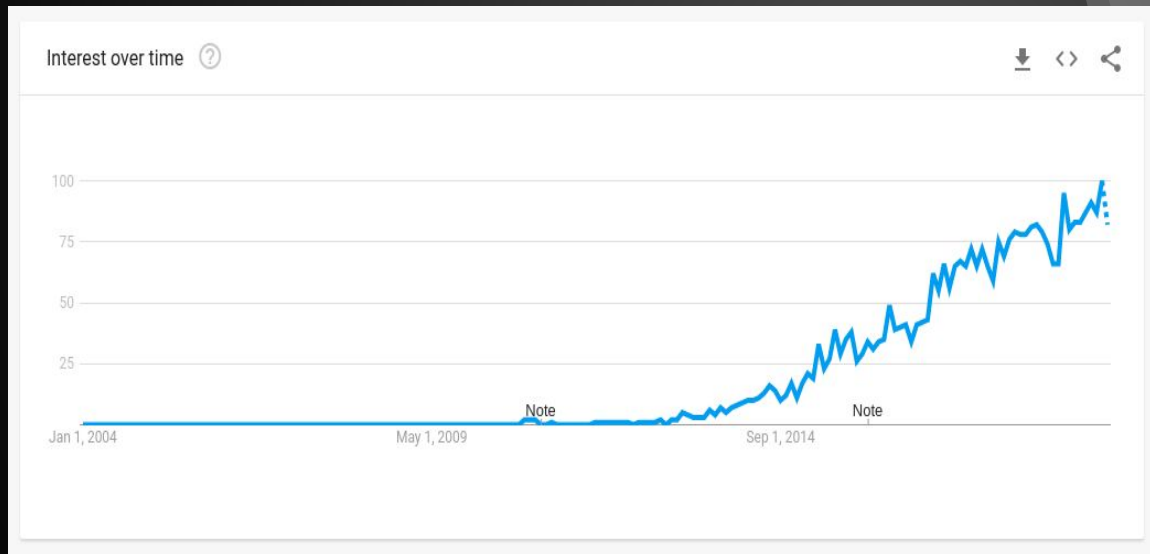
PROGRESSIVE WEB APPS

2004 to present



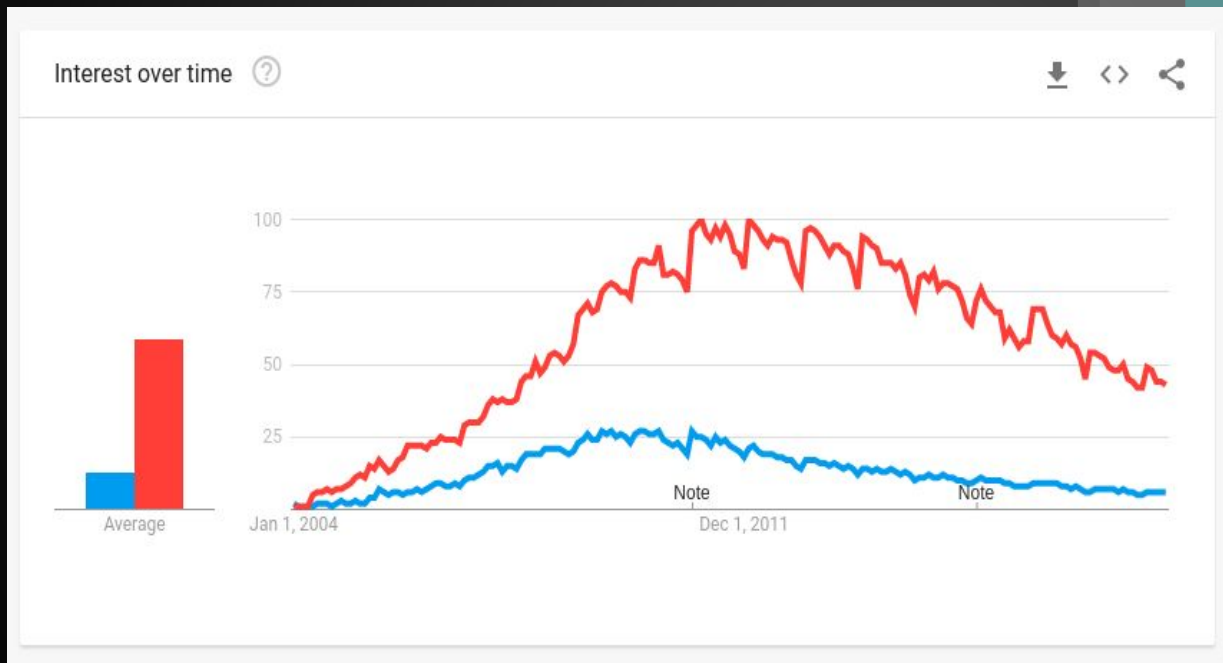
FULL STACK DEVELOPER

2004 to present



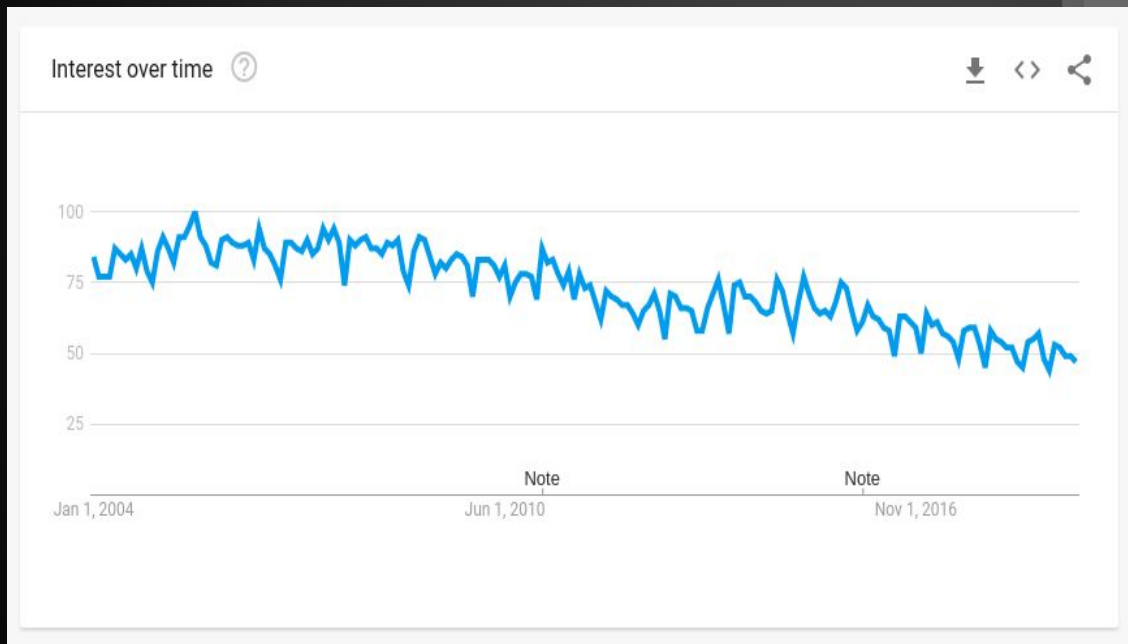
DRUPAL & WORDPRESS

2004 to present



CMS

2004 to present



“We would prefer Python,
Ruby, and/or Javascript
based applications. **We
would steer clear of
Drupal.”**



Jesus Manuel Olivas Retweeted



Nader Dabit

@dabit3

There is a massive paradigm shift happening right now. The front & back end are rapidly moving closer & closer together, assisted by the rise of managed services & serverless tech.

Those who acknowledge & take advantage of this will be the biggest winners in the coming years.

11:21 AM · 21 Apr 19 · [Twitter Web Client](#)

129 Retweets **464** Likes



**Important
Realization #1:**
**We are rapidly
changing how and
what we build.**



Option A
Stay the course.



Option A

Nobody ever got fired
for buying IBM.



Option A



Option B

Learn and adapt.



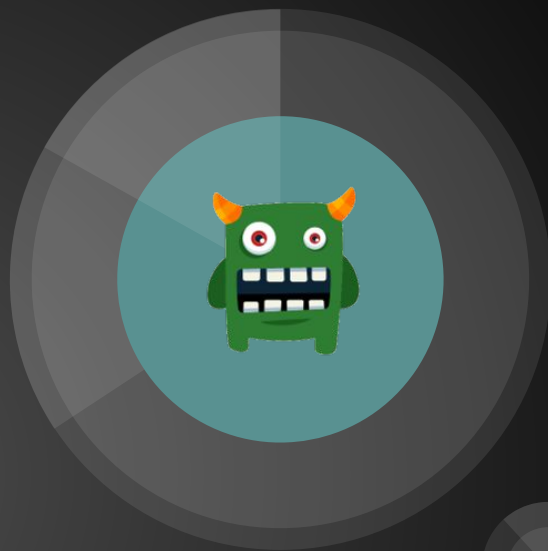
Monolithic CMS vs DECOUPLED FRONTEND

API INTEGRATIONS

MANAGED SERVICES

MICROSERVICES

SERVERLESS



Monolithic CMS vs DECOUPLED FRONTEND

API INTEGRATIONS

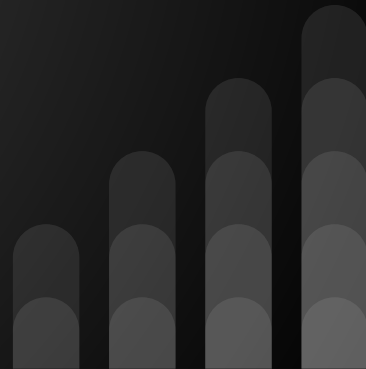
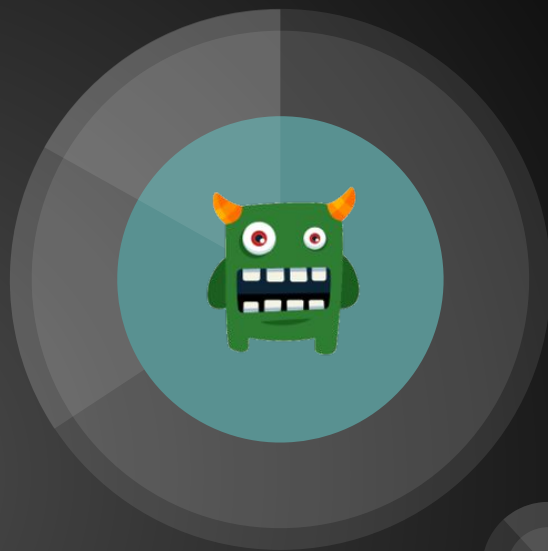
MANAGED SERVICES

MICROSERVICES

SERVERLESS



JavaScript!





Content Management

Landing pages, news, events



Custom applications

Databases, tools



Integrations

APIs, migrations, importers, exporters



Membership/CiviCRM

Customizations, roles, reporting

Monolithic CMS



Web server
Database server
CDN



sitecore



Adobe
Experience
Manager



Frontend website!
HTML, JS, Twig



Customization

Customize all the things



Commerce

Sell, sell, sell!



Search

Search API, Solr



Forums and galleries oh my

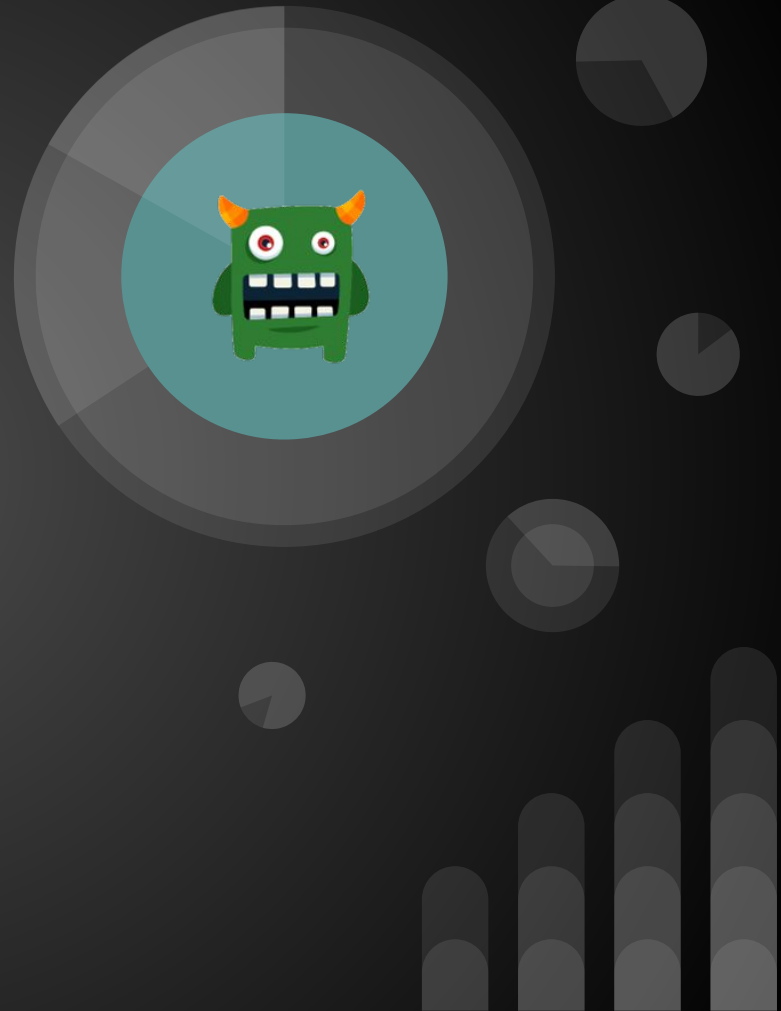
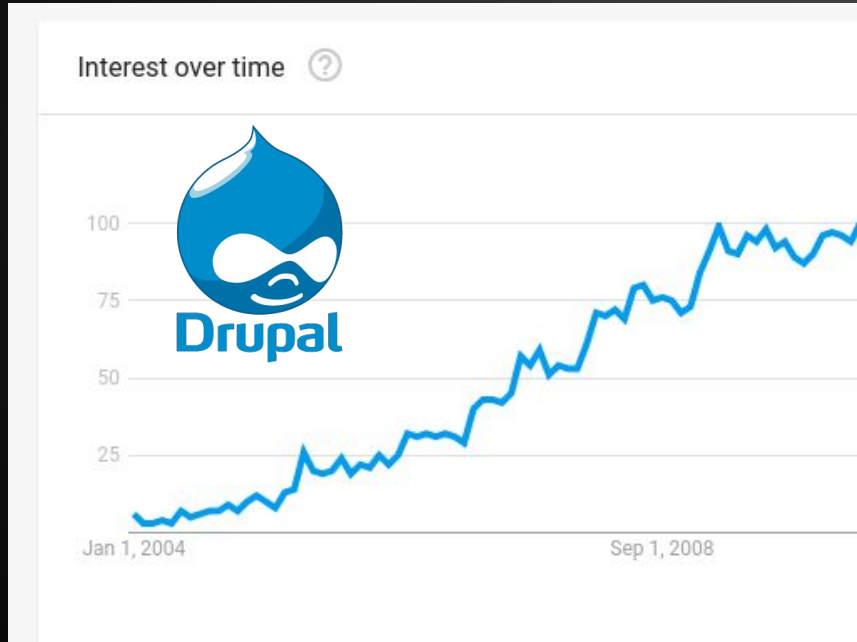
All the things!

The Golden Hammer

“I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail.”

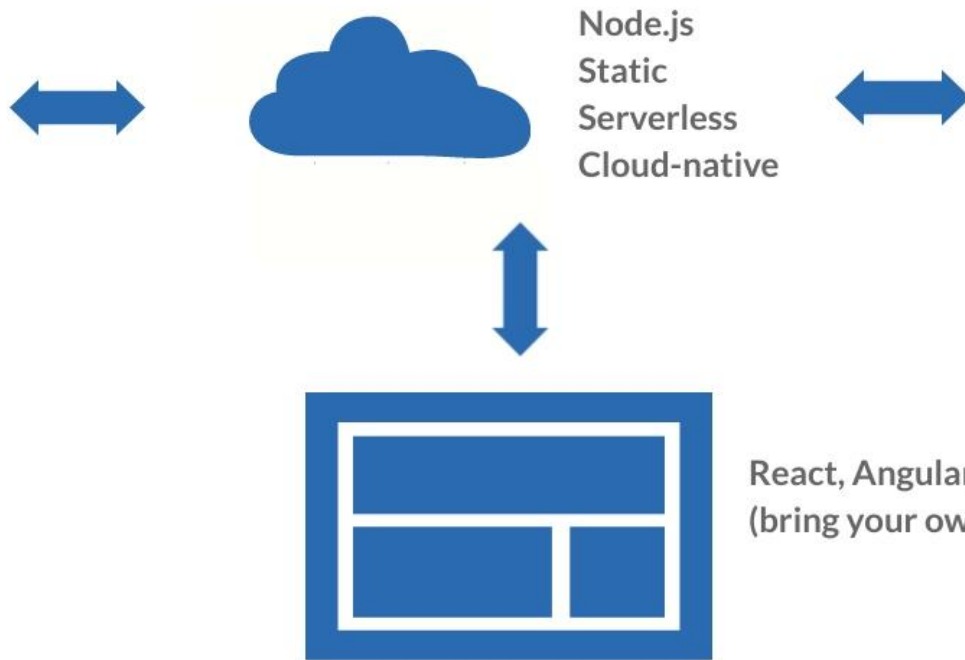


Once upon a time you dressed so fine.





Modern Web



Static files
Markdown



Search (Algolia)
Custom APIs
SaaS APIs



Custom DBs
Mongo/SQLite



JSON
XML/CSV

React, Angular, Vue
(bring your own!)

**Important
Realization #2:**
**We are no longer
building websites.**





**Everything else has
been solved and
commoditized.**
So what then is our
job?



Static files
Markdown



Custom APIs
SaaS APIs



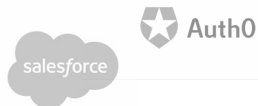
Custom DBs
Mongo/SQLite



JSON
XML/CSV



This is what really matters, right?



Static files
Markdown



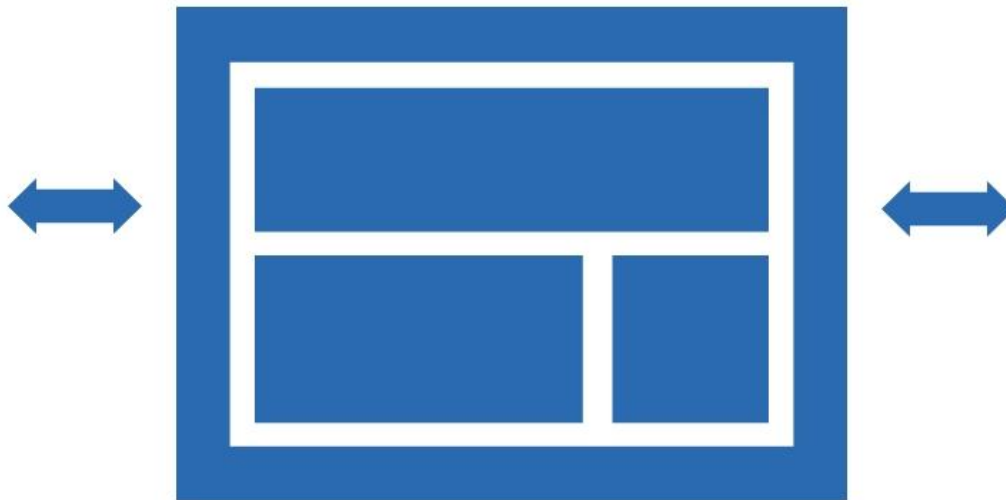
Custom APIs
SaaS APIs



Custom DBs
Mongo/SQLite

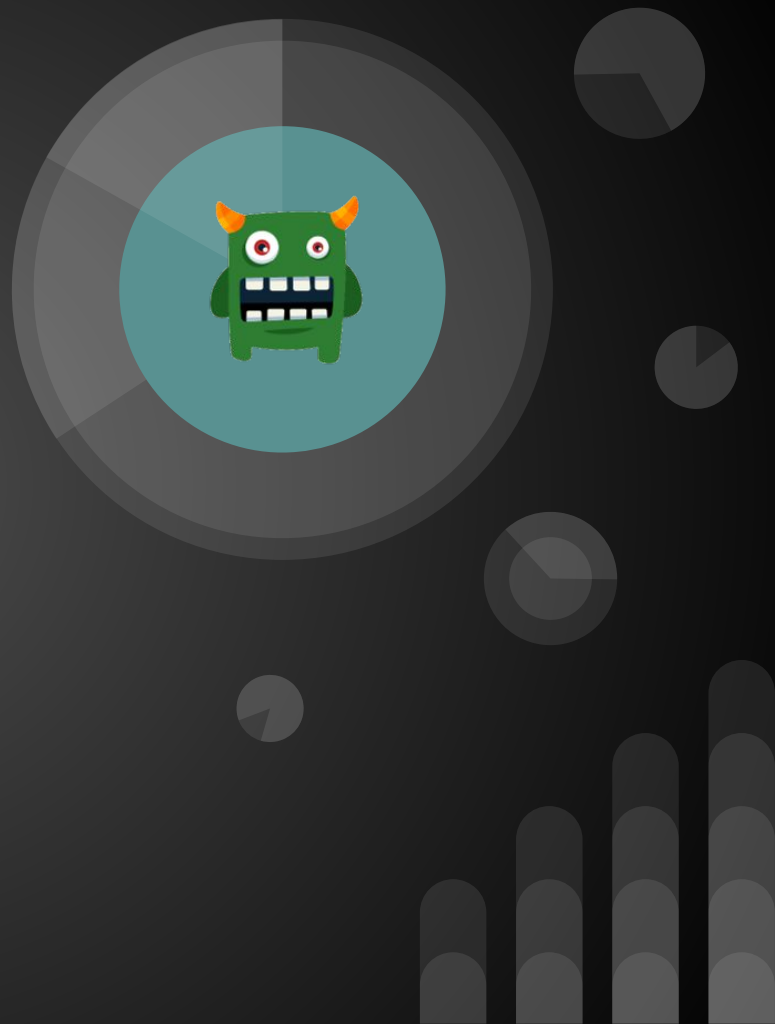
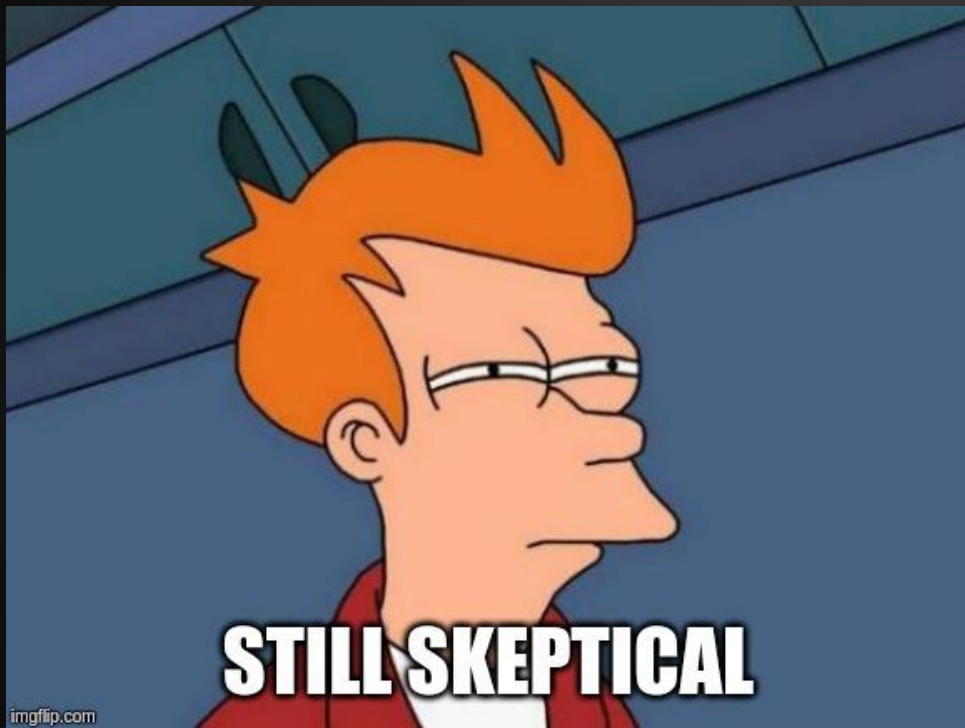


JSON
XML/CSV



React, Angular, Vue
(bring your own!)





Can't Drupal also provide a good user experience?

Why would I give up all of its out-of-box features?

Why add another layer of complication and expense?



From the dawn of time,
web developers hath
sought to build better
websites, and to build
them faster.



What if there is a better way to build the web?



SPEED AND PERFORMANCE



FLEXIBILITY



USER EXPERIENCE



DEVELOPER EXPERIENCE



OVERALL COST



TRADITIONAL
CMS

MODERN
WEB



	TRADITIONAL CMS	MODERN WEB
SPEED AND PERFORMANCE	<ul style="list-style-type: none">• Critical Rendering• Code splitting and pre-renders• Server Side Rendering (SSR)• Node (Chrome V8)• Static Rendering	
FLEXIBILITY		
USER EXPERIENCE		
DEVELOPER EXPERIENCE	✗	✓
OVERALL COST	✗	✓



	TRADITIONAL CMS	MODERN WEB
SPEED AND PERFORMANCE	✗	✓
FLEXIBILITY	<ul style="list-style-type: none">• Data agnostic• Frontend agnostic• Right tool for the right job	
USER EXPERIENCE		
DEVELOPER EXPERIENCE		
OVERALL COST	✗	✓



	TRADITIONAL CMS	MODERN WEB
SPEED AND PERFORMANCE	✗	✓
FLEXIBILITY	✗	✓
USER EXPERIENCE	<ul style="list-style-type: none">• Faster• More effective• More delightful	
DEVELOPER EXPERIENCE		
OVERALL COST		



	TRADITIONAL CMS	MODERN WEB
SPEED AND PERFORMANCE	✗	✓
FLEXIBILITY		
USER EXPERIENCE		
DEVELOPER EXPERIENCE		
OVERALL COST	✗	✓

- One stack = good feelings
- Faster (and more fun)
- Components/NPM/hot reloading



	TRADITIONAL CMS	MODERN WEB
SPEED AND PERFORMANCE	✗	✓
FLEXIBILITY	✗	✓
USER EXPERIENCE		
DEVELOPER EXPERIENCE		
OVERALL COST		

- Development
- Support and maintenance
- Hosting and infrastructure

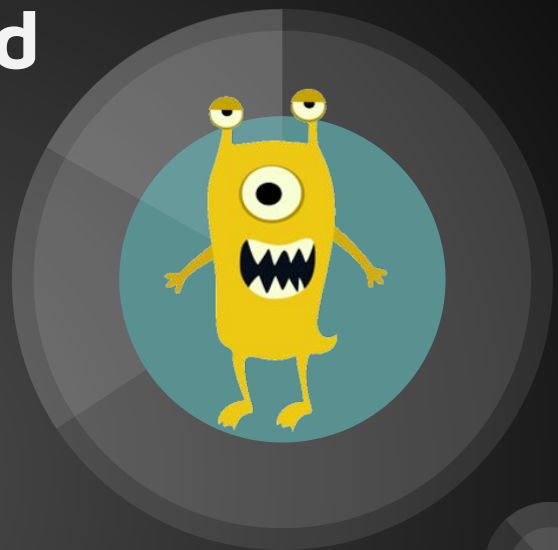


**Important
Realization #3:**
**You probably still
need a CMS.**



24 Headless CMS That Should Be On Your Radar in 2019

CMSWire.com



- Butter CMS
- Contentstack
- **Contentful**
- dotCMS
- Mura
- Cloud CMS
- Cockpit CMS
- Core dna
- **Craft CMS**
- Zesty.io
- Directus
- StoryblokWe
- **GraphQL CMS**
- Gentic Mesh
- Cosmic JS
- Kentico Cloud
- **Prismic.io**
- Quintype
- **Sanity.io**
- Scrivito
- Squidex
- DNN Evoq Content
- Strapi
- Superdesk

We offer spaces for all types of use cases, from single page apps, to campaign websites, mobile apps, etc. Below are some of our more popular ones.



MICRO SPACE

For teams working on single-purpose sites, mobile apps and IoT

1 sandbox environment, plus

1 role

2 locales

24 content types

5,000 records

\$39 / month



LARGE SPACE

For teams shipping complex apps and working with multilingual content

5 sandbox environments,
GraphQL API, plus

4 roles

10 locales

48 content types

50,000 records

\$879 / month



ENTERPRISE-GRADE SPACES

For organization-wide use and teams producing content at scale



GRAPHCMS

Up to
cu

More

More con



BASIC

Value pack for small apps or static sites

\$49

per project / month

[START A FREE TRIAL](#)

Users	10
Stages	1
Locales	1
Roles	4
Records	∞

Basic includes:



STANDARD

Perfect for complex apps and multilingual content

\$149

per project / month

[START A FREE TRIAL](#)

Users	20
Stages	2
Locales	4
Roles	4
Records	∞

All from Basic, plus:



PROFESSIONAL

For organizations that want to be in full control

\$499

per project / month

[START A FREE TRIAL](#)

Users	50
Stages	3
Locales	10
Roles	∞
Records	∞

All from Standard, plus:



ENTERPRISE

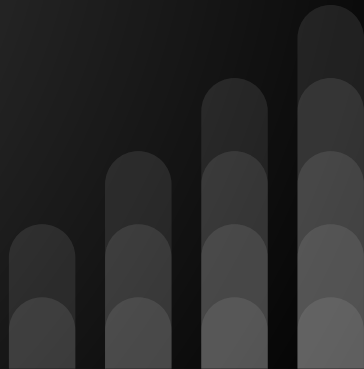
For teams with big ideas and paramount expectations

Custom pricing tailored to
your requirements

[TALK TO SALES!](#)

- ✓ All Features
- ✓ Custom Limits
Be it the number of users or API operations. We've got you covered.
- ✓ Unparalleled Performance
Enterprise-grade infrastructure managed by our team of experts.
- ✓ SLA-Guaranteed Uptime
Availability is monitored by our engineers and we guarantee a minimum of 99.9% uptime.

- Plus a million free tiers
- And plenty of open source





State of Drupal presentation (May 2016)

📅 May 12, 2016

🕒 51 sec read time

🔗 Permalink

🏷️ Drupal

State of Drupal

DrupalCon

New Orleans

DrupalCon New Orleans comes at an important time in the history of Drupal. Now that Drupal 8 has launched, we have a lot of work to do to accelerate Drupal 8's adoption as well as plan what is next.

In my keynote presentation, I shared my thoughts on where we should focus our efforts in order for Drupal to continue its path to become the leading platform for assembling the world's best digital experiences.

Based on recent survey data, I proposed key initiatives for Drupal, as well as shared my vision for building cross-channel customer experiences that span various devices, including conversational technologies like Amazon Echo.



The best JSON:API implementation in existence

The JSON:API module for Drupal is almost certainly the most feature-complete and easiest-to-use JSON:API implementation in existence.

The Drupal JSON:API implementation supports *every* feature of the JSON:API 1.0 specification out-of-the-box. Every Drupal entity (a *resource object* in JSON:API terminology) is automatically made available through JSON:API. Existing access controls for both reading and writing are respected. Both translations and revisions of entities are also made available. Furthermore, querying entities (*filtering resource collections* in JSON:API terminology) is possible without any configuration (e.g. setting up a "Drupal View"), which means front-end developers can get started on their work right away.




```

1 // 20190606144854
2 // http://dev-mtcv.pantheonsite.io/jsonapi/node/page
3
4 {
5   "jsonapi": {
6     "version": "1.0",
7     "meta": {
8       "links": {
9         "self": {
10           "href": "http://jsonapi.org/format/1.0/"
11         }
12       }
13     }
14   },
15   "data": [
16     {
17       "type": "node--page",
18       "id": "4e4028d5-8b4e-458b-8320-22fb2a01771a",
19       "attributes": {
20         "drupal_internal__nid": 2,
21         "drupal_internal__vid": 52,
22         "langcode": "en",
23         "revision_timestamp": "2019-05-28T20:27:28+00:00",
24         "revision_log": null,
25         "status": true,
26         "title": "Safe Browsing",
27         "created": "2019-05-12T17:41:59+00:00",
28         "changed": "2019-05-28T20:27:28+00:00",
29         "promote": false,
30         "sticky": false,
31         "default_langcode": true,
32         "revision_translation_affected": null,
33         "path": {
34           "alias": "/safe-browsing",
35           "pid": 1,
36           "langcode": "en"
37         }
38       },

```

/jsonapi



Important Realization #4:

Drupal solved most of the hard problems 10 years ago or more.



- Content architecture and entities
- Nodes and taxonomies
- Entity References
- Workflow and Permissions
- Content scheduling
- Media and File Handling
- Battle tested goodness



Important Realization #5: Drupal is still the bomb.



- Free and Open source
- API-first
- No “record” limitations
- No “rate” limitations
- Arguably better at the fundamentals of CMS
- Right tool for this job



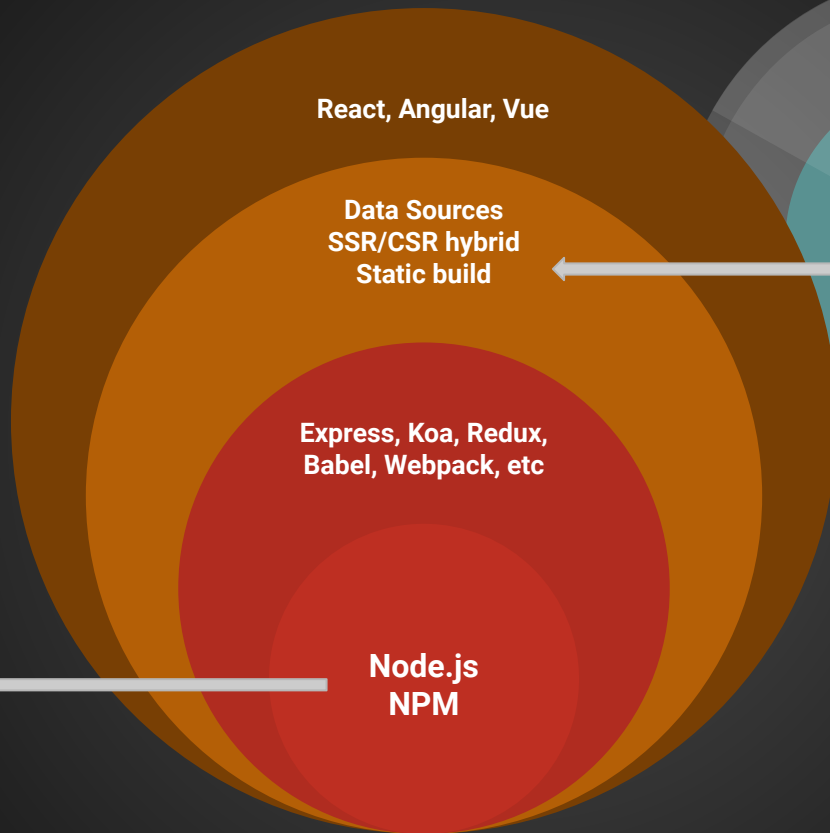




JavaScript
TypeScript
ECMAScript/ES



NODE
SERVER



React, Angular, Vue

Data Sources
SSR/CSR hybrid
Static build

Express, Koa, Redux,
Babel, Webpack, etc

Node.js
NPM

SaaS
Drupal/WordPress
Managed Services (APIs)
XML/JSON/CSV
Microservices



DB



DB



DB



SERVER



SERVER



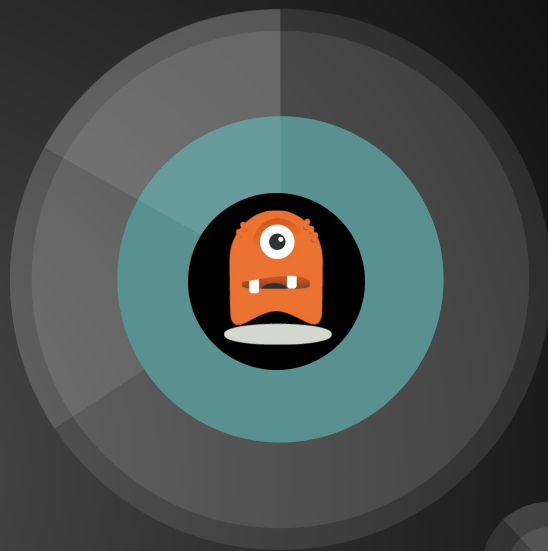
SERVER

React

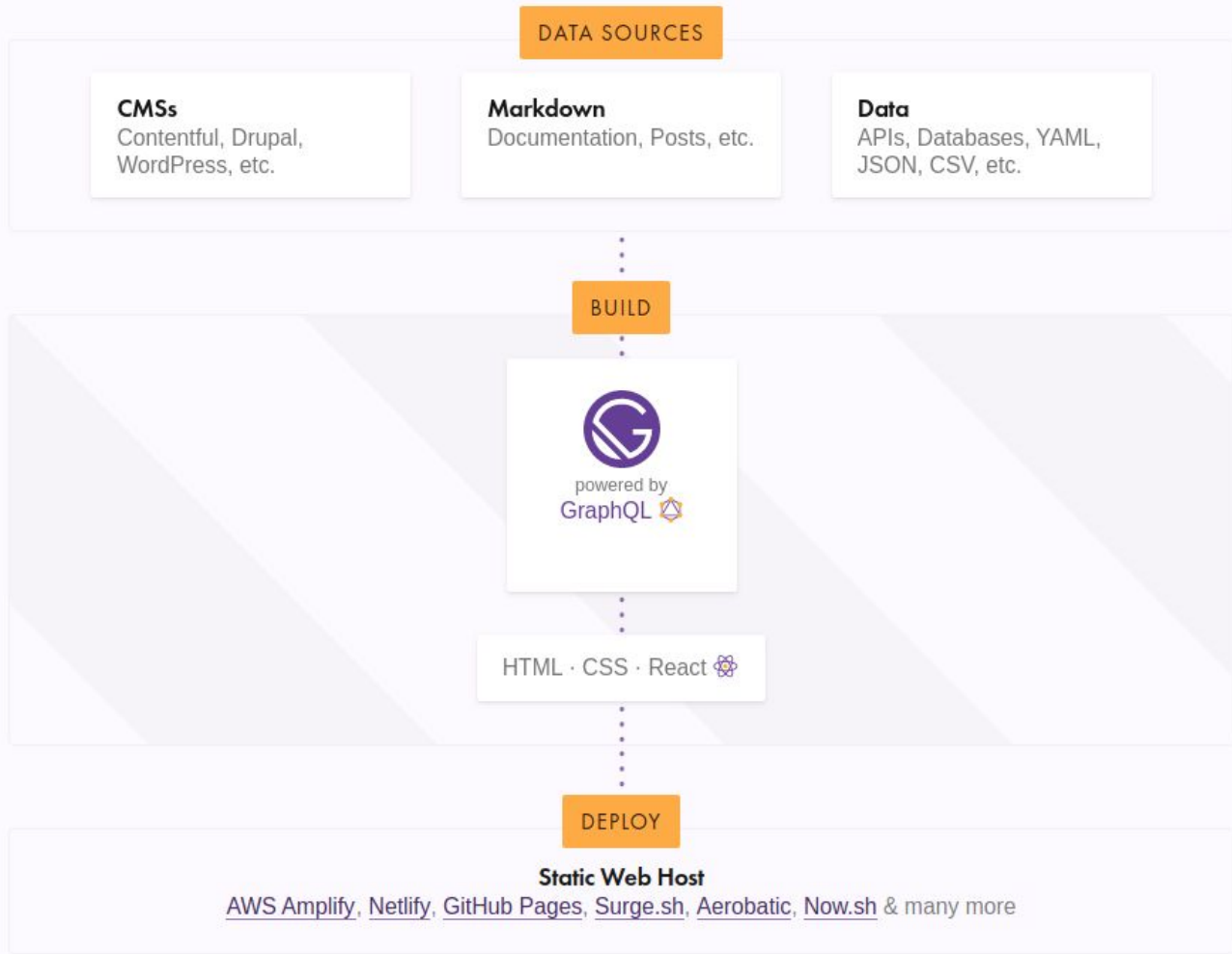
A JavaScript library for building user interfaces

Get Started

Take the Tutorial >



Meet Gatsby!





The React Framework for Server-Rendered Apps

See Showcase

License: MIT [View Docs](#) [GitHub](#)



“I would never build
anything in React
**without choosing
Next or Gatsby.**”

Wes Bos, Feb. 2019



JAMstack: noun \ 'jam-stak' \

Modern web development architecture based on client-side JavaScript, reusable APIs, and prebuilt Markup.



The World is your Playground:

- **Netlify and Now**
- **Codepen/Sandbox/Glitch**
- **One click demos**
- **Free tiers for miles**
- **Excellent documentation**



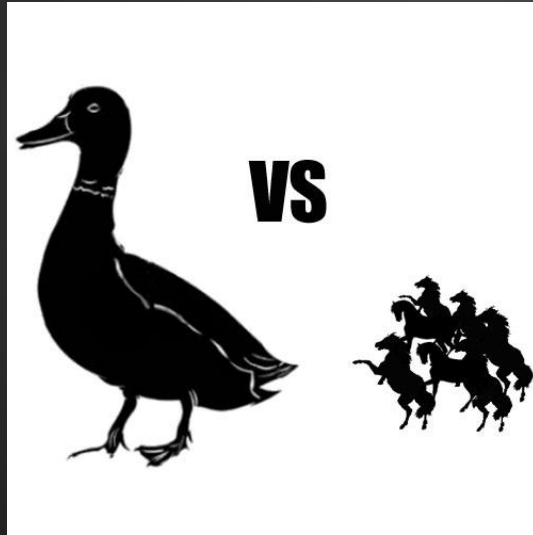
Important Realization #6:

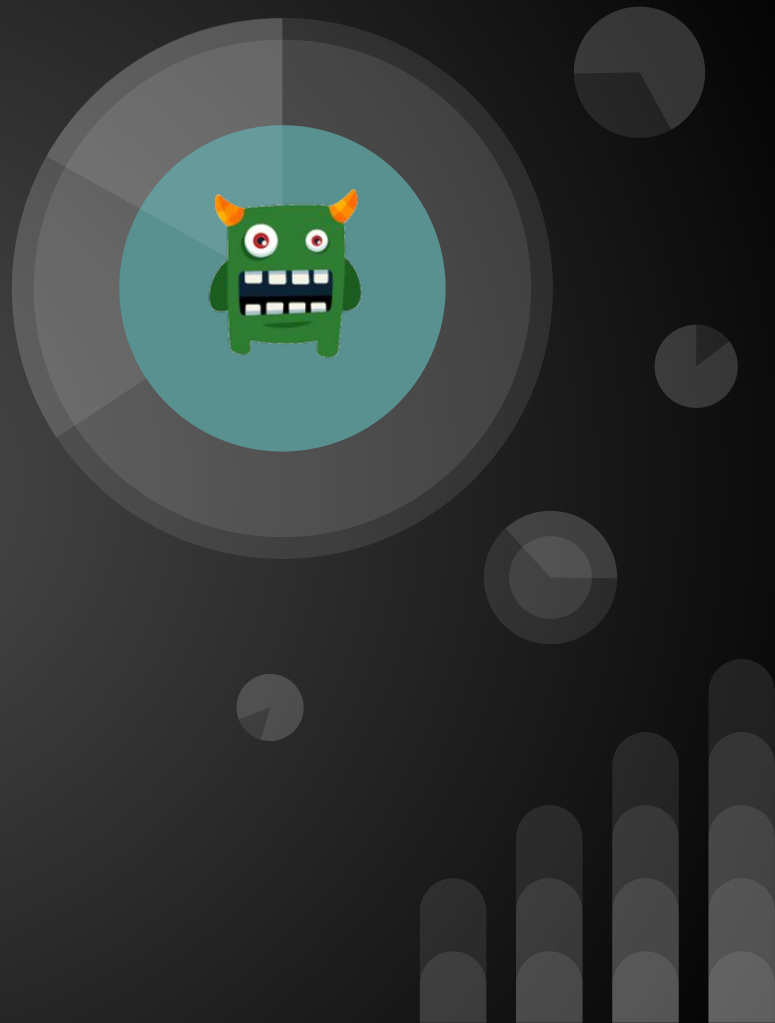
Things that were super
hard 5 years ago are now
easy (and even fun!)



Important Realization #7:

100 duck sized horses vs one horse sized duck.







Thank you!



Citizen Tim

Electric Citizen | October 2019

