Intro to the Jamstack 🍯

Hosting React Applications (and other things) with Netlify

Chris Dalke

https://github.com/chrisdalke/jamstack-lunch-and-learn

WHAT is the Jamstack?

The Jamstack is an architecture for web-based software that is fast, secure, and scalable. The Jamstack is not a single set of tools. It's a **philosophy** around structuring web applications that lowers costs, improves performance, and simplifies development.

1. Javascript

Dynamic content and interactive applications are built with Javascript and run in the browser.

2. APIs

Backend infrastructure is modular and decoupled from any frontend logic. APIs can be your own services, or third-party.

3. Markup

At build time, your frontend is rendered into static Markup and assets, such as HTML, CSS or Javascript files. These are served through a CDN, removing the use of a frontend server.

Note: I'm paraphrasing here from https://jamstack.org/what-is-jamstack/ and https://jamstack.wtf/. These sites are a great introduction to the Jamstack architecture.

WHY the Jamstack?

Simplifies infrastructure

The Jamstack removes the requirement for a frontend server. This reduces points of failure and infrastructure complexity. If outages do occur with other services, the frontend can maintain the user experience by showing loading and placeholder content.

*Infinite scalability**

Your frontend is served directly from the CDN. Focus scaling efforts and infrastructure investment on your API.
(*Most CDNs bill by bandwidth, and your wallet might not be infinite)

Automated builds & improved developer experience

Developers get instant feedback with faster builds, and automatic deployments for branches and PRs. Deployments use the same tooling that developers use locally. "Gitops" means many devops tasks are automatically triggered with git operations.

Faster page loads

Content is served via a CDN, so page load performance is greatly improved globally compared to a single server instance.

and more...

Use Cases

Static Content (Hand-written HTML, Assets, etc)

The Jamstack can be a great alternative to an FTP server or other hosted file server for completely static HTML and assets. You can upload and serve a set of HTML pages from Netlify or another service for free.

Generated Static Sites (Landing Pages, Blogs)

"Static site generator" tools such as Hugo or Gatsby improve developer experience when working on static sites. With Hugo, you specify your content as Markdown files. At build time, Hugo will render Markdown with a set of HTML templates you have chosen. Hugo outputs a folder tree and set of files which can be hosted from a CDN.

React Applications

React frontends are a great candidate for hosting with the Jamstack. Most React applications are *already* built to static files. The frontend bundle is conventionally served from a container with nginx or Apache, but we can remove this dependency entirely by hosting with the Jamstack.

Jamstack Hosting Providers

There are many hosting providers for the Jamstack. Here's a few:

AWS CloudFront + S3

The combination of CloudFront and S3 lets you serve a static site from an S3 bucket. This is useful & cheap for serving very large static assets, but doesn't have many of the features that other Jamstack providers offer.

Netlify

Netlify focuses on making the developer experience extremely smooth, and it's my personal choice for most projects. Automatic SSL certificate, branch deployments, and more are available in a very generous free tier. Netlify works with nearly any toolchain (yarn, npm, hugo, etc.)

Vercel

Another Jamstack hosting provider, focused on Next.js. Next.js is a framework that allows static sites to be generated from React components. With Next.js, you can pull data and render your static content at build time in React, and use the same codebase to provide progressive enhancements in the browser. We won't cover Next.js in this Lunch and Learn; It's an exciting framework but it requires more "buy-in" than deploying an existing stack to Netlify.

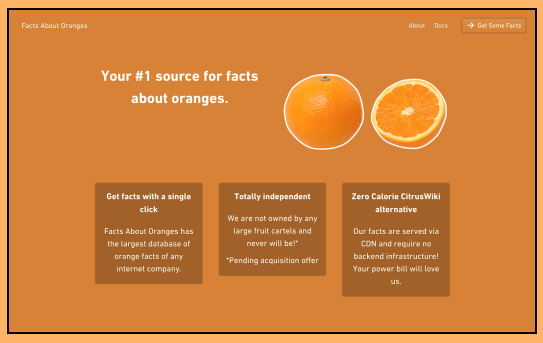
Cloudflare

Just released this week, very closely competes with Netlify. Haven't read much about it but I expect it to be a good competitor to Netlify.

A Practical Example

Let's walk through a practical example. We're going to deploy a simple ecosystem for a fictional product: factsaboutoranges.com.

Facts about Oranges is *the* place to go for facts about oranges. For the launch of the site, we want to deploy a few websites in an ecosystem:
- **A simple landing page** (Hand-written HTML)
- **A blog site**, (A static site generated from Markdown)
- **The core product** (A React application)



We've chosen the Jamstack for this early-stage product for a few reasons:

- It's **cost effective:** All our frontend resources are hosted for free up to a bandwidth limit. The only infrastructure we're paying for is the core API.
- It promotes **quick iteration:** We can instantly deploy a new change to the site by pushing to our repositories. We can very quickly tweak the product and make changes in response to feedback or bugs.

Let's jump right in. We'll be using Netlify to deploy our application. For this tutorial, I've prepared three projects: A static HTML landing page, a Blog site built with Hugo, and a React application. I won't go into too much detail about each, since we will be focusing on deploying the apps with Netlify.

https://github.com/chrisdalke/jamstack-lunch-and-learn

Netlify Setup & Prerequisites

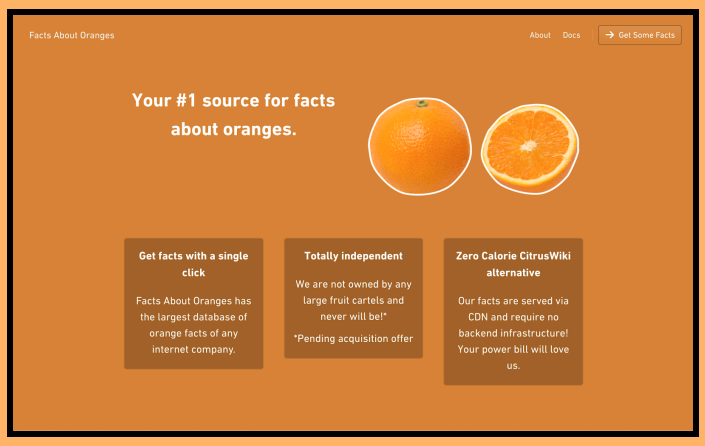
To start, we need to sign up for a free Netlify account on their website: https://www.netlify.com/.

Once we have an account setup, we can get started!

Deploying a Static Site

Our first site will be a static site, with hand-written HTML/CSS.

This is the simplest type of site you could host on Netlify.



Previously, a hand-built HTML site would be deployed with an FTP server or Nginx serving a static directory. Both of these methods require a server instance, and we can use Netlify to deploy to a CDN for free.

Create Site from Git Repository

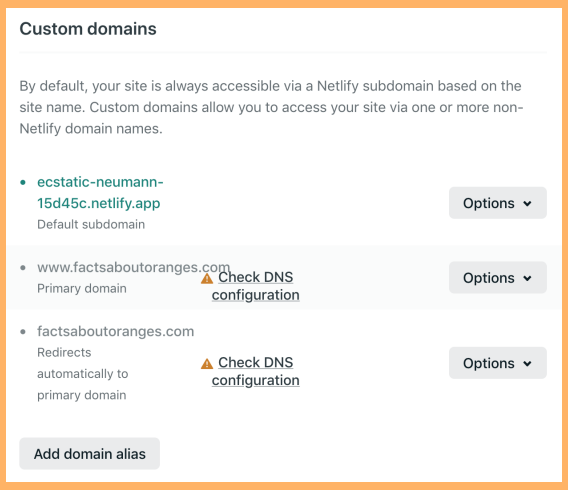
Create a site in the Netlify dashboard, linked to the Git repository for our project. Netlify needs permission to access and manage repositories on your account, so it can add webhooks and trigger in response to commits.
- Branch to deploy: **`main`**
- Build command: **`echo "Done!"`**
- Publish directory: **`static-html`**

Because the static site doesn't need to be built, I've just put in a placeholder build command. In the next few websites, we'll use the build command to trigger a hugo or webpack build.

The site will deploy, and be available at a generated URL like https://ecstatic-neumann-15d45c.netlify.app/.

Setup Custom Domain

We want to point a real domain to the site. Go to the **Site Overview > Set Up a Custom Domain.**



For this website, we'll be routing from www.factsaboutoranges.com. We want any person visiting factsaboutoranges.com to be redirected to the "www" variant, and Netlify will do this automatically.

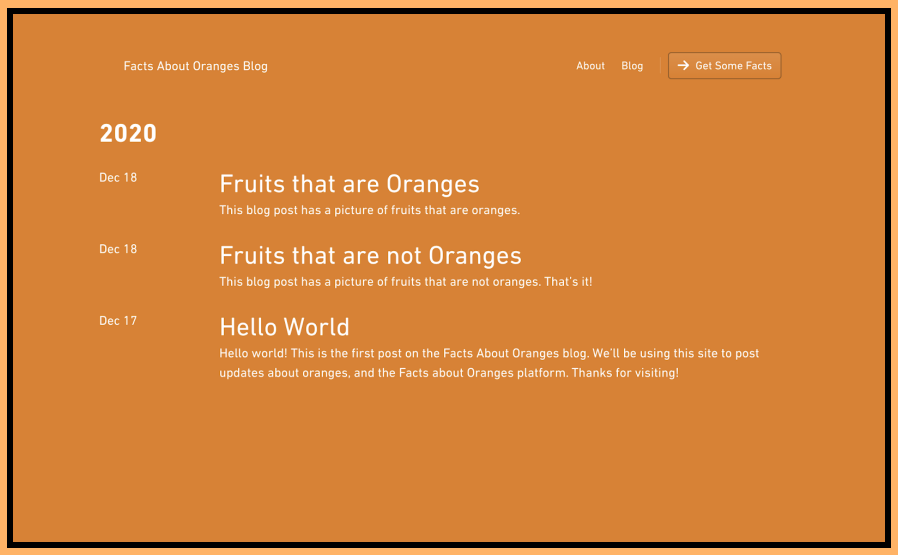
Netlify will provide you instructions to configure the DNS settings for your site to redirect to its servers. We'll use Route 53 to add the necessary A and CNAME records to our site.

Advanced Features

Asset optimization - The images and resources on the page are unnecessarily large. Netlify can automatically optimize our page. To enable this setting, go to **Site Settings > Build & Deploy > Asset optimization.**

Form detection - We can tell Netlify to handle form entries on our site, to set up a very simple signup form. Go to **Site Settings > Build & Deploy > Form detection,** and enable the setting.
Any form entries will now be handled in the Forms section. We can add an outgoing notification, to email us when someone uses the form.

Deploying a Site built with a Static Site Generator



Now that we've deployed the basic landing site, let's try deploying something that requires files to be built. We'll deploy a blog site, so we can give our customers updates about the software.

The blog is built with Hugo, which is a **static site generator.** Static site generators convert content, usually in Markdown or another semantic markup format, into HTML pages which can be hosted statically.

I won't go over the nuances of Hugo in this lunch and learn. In order to build a static site with Hugo, we run the **`hugo`** command. This will output a built site to the public/ folder.

Let's go through the same process in Netlify, this time adding the following settings:
- Branch to deploy: **`main`**
- Build command: **`hugo`**
- Base directory: **`hugo-blog`**
- Publish directory: **`public`**

https://github.com/chrisdalke/jamstack-lunch-and-learn

Pushing Changes

If we make a change to the git repository, the site will rebuild and deploy automatically. Let's add a new blog post, and push it to the repo. We can watch Netlify build the site in the **Deploys** tab.

✨Netlify Nuance: Caching

Netlify has put a lot of thought into their caching strategy. When the Netlify CDN serves content, it needs to set caching headers that allow clients to efficiently cache assets, while also letting you quickly push new content. Netlify uses two headers:

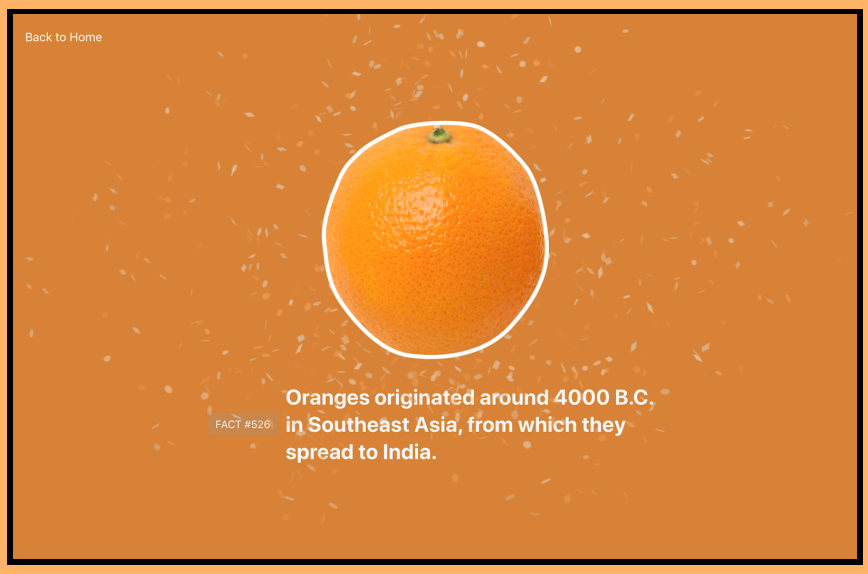
`max-age=0, must-revalidate, public:` Client browsers should cache content with no timeout, but they must revalidate the content with the CDN server on every request.

`etag:` A hash of the resource. The browser sends this hash to the CDN server on *every request*. The browser returns a **304 NOT MODIFIED** status if the hash is valid and doesn't send any data, which allows the client to use it's cached copy.

The CDN servers are extremely fast and geographically distributed, so there is a low overhead associated with the hash check. This allows resources to be instantly updated, while also maintaining client-side caching.

See https://www.netlify.com/blog/2017/02/23/better-living-through-caching/ for more information on their caching strategy.

Deploying a React Application



Deploying a React application is not too different from the previous examples. We'll go through the previous steps with our react app, deploying the site to app.factsaboutoranges.com.
- Branch to deploy: **`main`**
- Build command: **`yarn build`**
- Base directory: **`react-app`**
- Publish directory: **`build`**

Redirect All Pages to the Bundle

We've deployed the site, but there's an issue -- The React application is only present on /index.html, so if we reload a page a client side router won't handle the URL.

To fix this, we'll add a **netlify.toml** file, which configures the Netlify deployment. We can add the following lines to the file:

```
[[redirects]]
  from = "/*"
```

```
to = "/index.html"
status = 200
```

This will redirect all paths that are not already a valid URL to the index.html page, meaning React will now be present at all URLs on the domain.

We can also use this feature to proxy any request: The **"to"** path can be any internal or external URL. We could redirect **/api** to proxy to a third-party API server, for example.

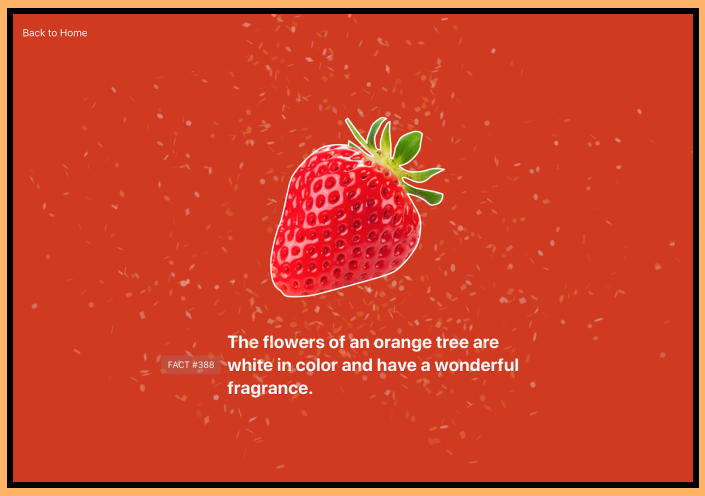
Branch & PR Deployments

Up until now, we've only selected the **"main"** branch to deploy from -- But you could have multiple branches, such as **develop** for dev builds and **main/master** for production builds.

We can configure netlify to build a copy of the site for all branches and PRs. This gives developers an instant preview of their changes, deployed in a real environment.

Let's try this out: Go to **Site Settings > Build & deploy > Deploy contexts**, and turn on Branch Deploys for All branches and Pull Requests. Netlify will generate a preview deployment for any branch in the repository.

We have a branch in the repository with a potential change to the Oranges application, so let's visit the deploy preview for the **strawberry** branch.



Hmm, maybe we should consider the impact to customers before merging this. Branch deployments and fast builds create a rapid feedback cycle that improves testing and reliability of new features.

... That's it!

We've deployed a full static of frontends on the Jamstack, from a static landing website all the way to a fully interactive react application. Along the way, we dived into the Jamstack philosophy and Netlify as a hosting provider.

Netlify is a powerful tool to implement the Jamstack architecture for frontends, and I hope everyone learned something new!

https://github.com/chrisdalke/jamstack-lunch-and-learn

What's Next

One conclusion to take away from this: You can all use this. Deploying side projects is easier, cheaper and more powerful than ever!

What's next? Some ideas:

… Caching wordpress sites to reduce infrastructure cost?
… Fully static dashboard applications which pull data at build time?
… A profitable SaaS project with zero infrastructure costs?
… Other fruit-facts-as-a-service?

https://github.com/chrisdalke/jamstack-lunch-and-learn