Welcome to the Space JAM 🎶 🎶

@tpiros

Hi 👋 I am

Tamas Piros

Developer Evangelist
Cloudinary

Director
Full Stack Training

Google Developer Expert
Web Technologies

leggo 🚶

ya tu sabe’ como va 😎

dembow 🥁

papi 🕺

a fuego 🔥

mami 💃

janguear 🥴

pa’ la calle 🛣️

pana 👨🤝👨

socio 😎

pega’o 🤑

real hasta la muerte ✝️

Fresh body, fit mind. 🏋️🧠

Thank you for the standing ovation. 👏

We will talk about the **JAMstack** 🚨🍯

JAMstack?

- JavaScript
- API
- Markup

“A modern architecture: Create fast, secure sites and dynamic apps with JavaScript, APIs and pre-rendered Markup, served without web servers.”

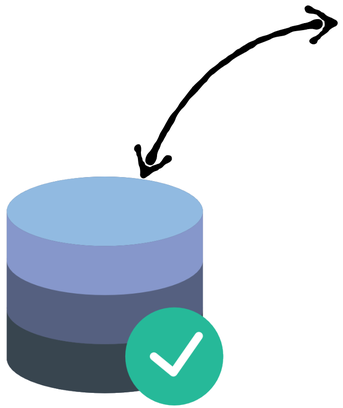
–Phil Hawksworth (Netlify)

A modern architecture: Create fast, secure sites and dynamic apps with JavaScript, APIs and pre-rendered Markup, served without web servers.

An origin server processes and responds to requests arriving from clients. Typically it's a computer that has an application running on it, and it's capable of serving up content - such as a website.

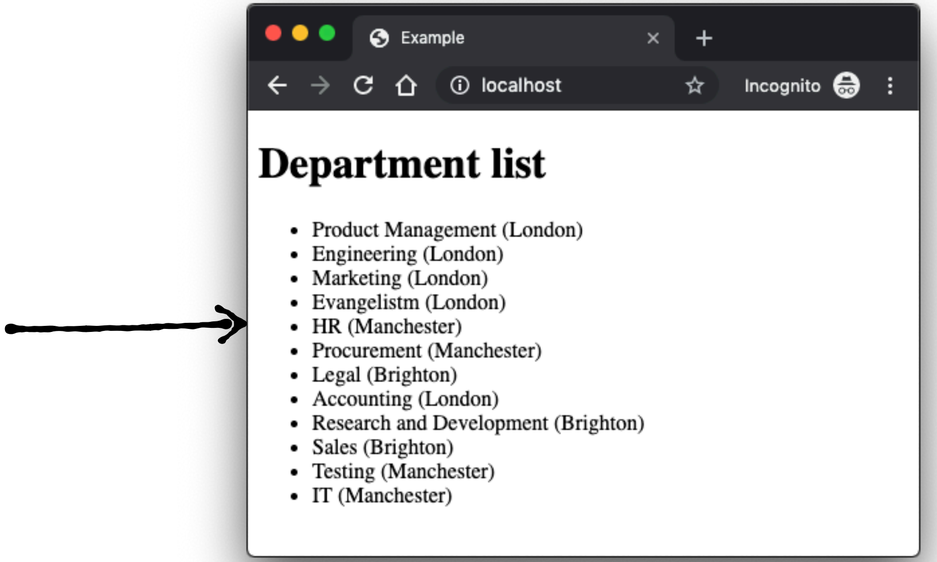
Origin Server

LAMP



```php
<div>
  <h1>Department list</h1>
  <?php
    $database_link = mysqli_connect("127.0.0.1", "root", "marina", "project");
    if($database_link === false){
      die("ERROR: Could not connect. " . mysqli_connect_error());
    }

    $sql = "SELECT name, location FROM departments";
    if ($result = mysqli_query($database_link, $sql)) {
      if (mysqli_num_rows($result) > 0) {
        echo "<ul>";
        while ($row = mysqli_fetch_array($result)) {
          echo "<li>" . $row['name'] . " (" . $row['location'] . ")</li>";
        }
        mysqli_free_result($result);
      } else{
        echo "<p>No departments found</p>";
      }
    } else{
      echo "ERROR: Could not able to execute $sql. " . mysqli_error($database_link);
    }
    mysqli_close($database_link);
  ?>
</div>
```

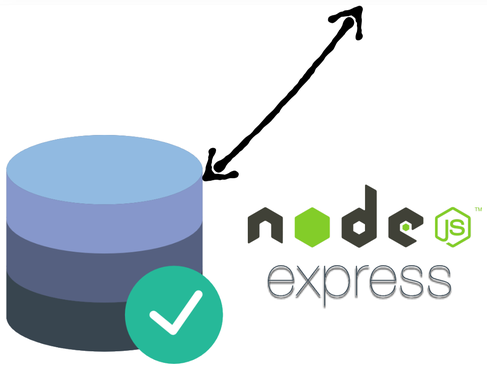


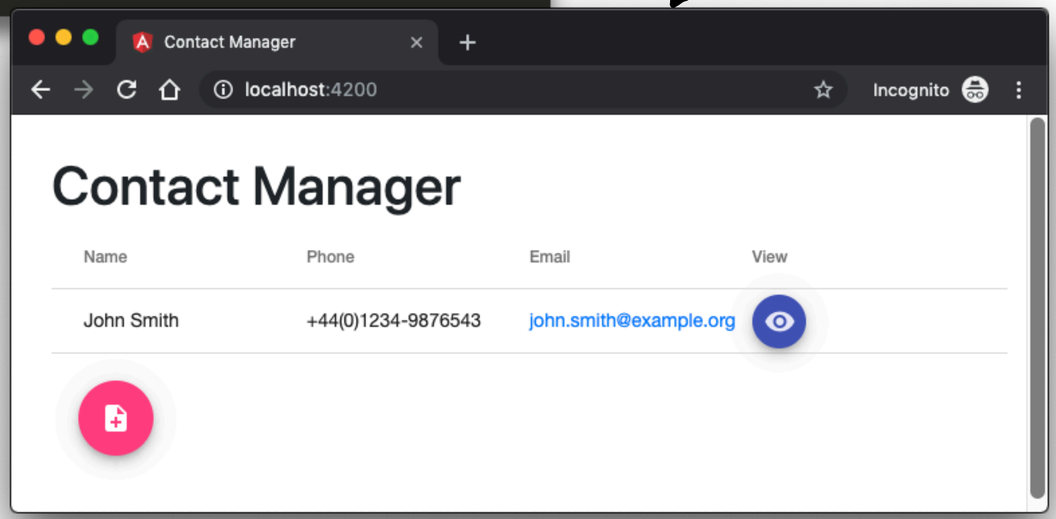
MEAN

```ts
// some-service.ts
export class ContactService {
  constructor(private http: Http) { }
  list(): Observable<any> {
    return this.http.get('http://localhost:3000/api/contacts');
  }
}

// some-component.ts
export class ContactListComponent {
  displayedColumns: string[] = ['name', 'phone', 'email', 'view'];
  dataSource: DataSource<any> = new ContactDataSource(this.contactService);
  // ...
}
```

```html
<!-- some-component.html -->
<div class="display-table">
  <h1>Contact Manager</h1>
  <md-table #table [dataSource]="dataSource">
    <ng-container mdColumnDef="name">
      <md-header-cell *mdHeaderCellDef> Name </md-header-cell>
      <md-cell *mdCellDef="let element">{{ element.name }} </md-cell>
    </ng-container>
  </md-table>
  <!-- ... -->
</div>
```





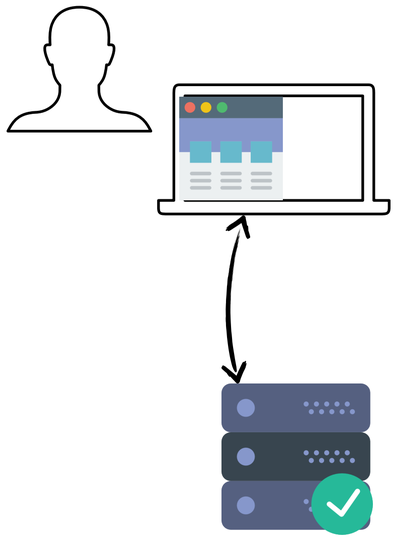
An edge server is a set of computers distributed globally, with a task of delivering content as fast as possible to the closest requesting user.

Edge Server

CDN

- Content Delivery Network
- Globally available servers for serving static content
- Used heavily to distribute CSS / JS today

The JAMstack approach



- No need for an origin server
- Any static file hosting can serve files
- Can be served from a CDN
  - “Edge server”
  - Cached version
  - Globally available

Key Terms

- Server-Side Rendering (SSR)
- Pre-rendering & (Re)-Hydration
- Static Site Generator (SSG)
- Client-Side Rendering (CSR)
- Headless CMS

Server-Side Rendering

- HTML generated on-demand at server-side
- Data queries and structures are done before the render
- Server is the main execution engine
- Templates drive how HTML is generated
- Final HTML is sent to the browser, on request

Pre-Rendering

- Pre-render (or pre-build) HTML files
- Happens at build time, not at request time
- Dynamic events are added via “hydration”
- Pre-rendering is a mix of SSR and CSR

Static Site Generator

- Generates static HTML based on templates
- Produces HTML per each route, ahead of time
- Frontend framework specific SSGs
- Vanilla JS SSGs
- Non-JS based SSGs

Static Site Generator (Examples)

- Gatsby 
- Nuxt.js 
- Next.js 
- VuePress 
- Gridsome 
- 11ty 
- Jekyll 
- Hugo 

Pre-Rendering vs SSG

- Both generate static HTML ahead of time?
- Pre-rendered sites normally depend on JavaScript for hydration
- SSG works without JavaScript

Client-Side Rendering

- Browser is the main execution engine
- All operations happen in the browser
  - templating, routing, data fetch, render
- No full page reload
- Heavy JavaScript execution

Rendering Overview

Server ⟷ Client

	Server-side Rendering	Static Site Generator	Pre-Rendering with Hydration	Client Side Rendering
Overview	Each page reload generates a new page, where content is fetched and generated entirely on the server.	All pages are pre-generated at build time	Server generates the pages, but the browser also executes JavaScript to “boot” / “hydrate” the application.	Everything is done by the client - including the rendering and the booting of the application.
HTML generated by / when	Server / on request	A build process / ahead of time	Server & Client (hydration) / ahead of time & on request	Client / on request
Pros / Cons	Good for SEO 👍 Page loads may be slow 👎	Good for SEO 👍 Fast performance 👍 Multiple frontend frameworks 👍 Other languages 👍	JavaScript for further DOM manipulation 👍 Could have misleading UX 👎	No need to request HTML from server on route change 👍 Bad SEO 👎

Headless CMS

- Content Mana
- Coupled vs d
- API and git ba

Headless CMS (Examples)

- Strapi - API based
- Netlify CMS - git based (YAML & frontmatter)
- Contentful - API based
- Cosmic JS - API based
- Appernetic - git based

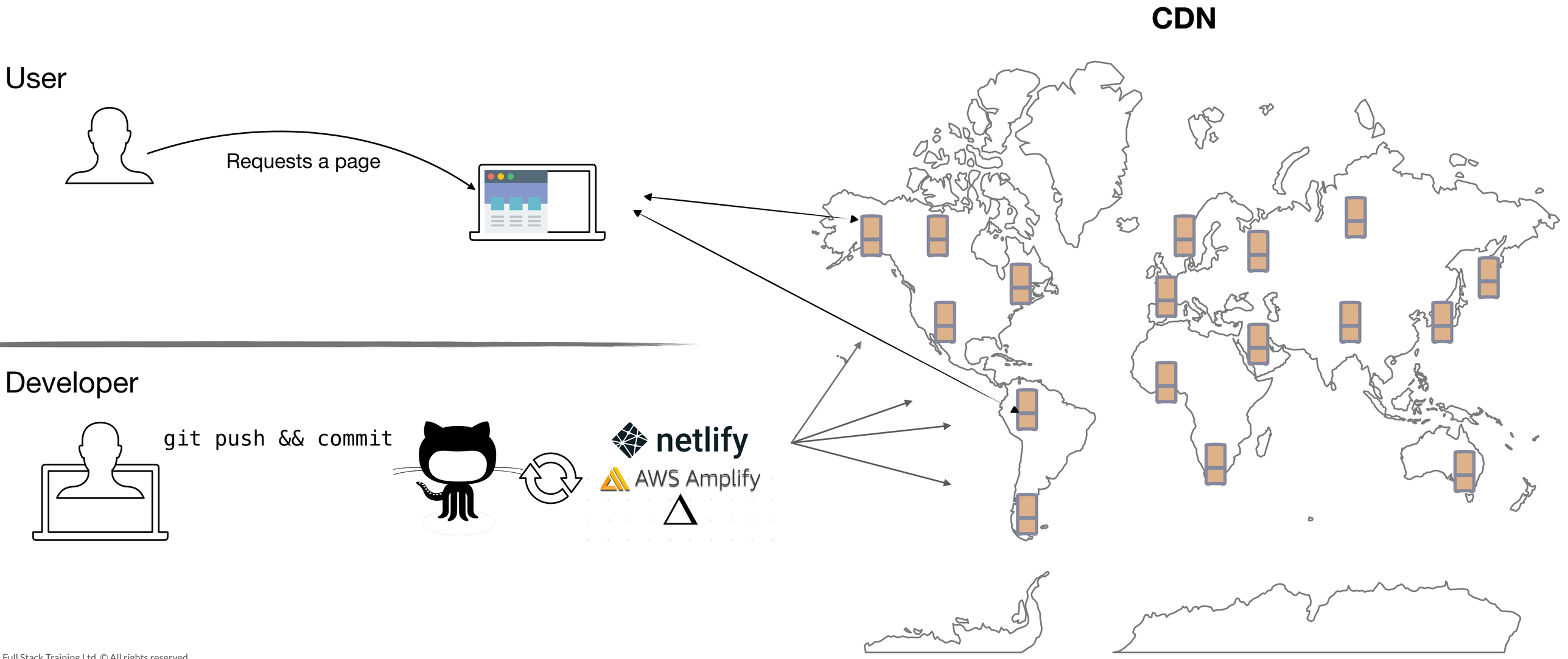
Static != Static

- Make apps dynamic by utilising APIs
- “Lean on the shoulder of giants”
- “There’s an API for that”
- FaaS / Serverless

API (Examples)

- Payments? Stripe
- Images? Cloudinary
- SMS Messaging? Twilio
- Forms? Formspree
- E-Commerce? Snipcart
- Authentication? Auth0
- Custom built API (FaaS)

High Level Overview



Additional Benefits

- Performance
- Cost
- Planning
- Maintenance
- Development
- Deployment
- Security

What will we build?

- 11ty (SSG)
- Netlify: CMS (Content, Headless CMS), Identity (Authentication), Deployment
- Cloudinary (Images)
- Formspree (Forms)
- Algolia (Search)
- Firebase (“likes” / “claps”)

Getting Ready

- Have a Netlify Account (https://app.netlify.com/signup)
- Have a Cloudinary Account (https://cloudinary.rocks/cdc)
- Have Node.js installed (v10.x.x)
- Open your favourite code editor

- If you get stuck: https://github.com/tpiros/11ty-nunjucks-blog

🥁 https://jamstack.training

📯 https://blog.fullstacktraining.com/introduction-to-the-jamstack/

@tpiros

🚨Media Developer Experts wanted🚨

https://cloudinary.com/partners/media-developers

🇩🇴 Muchas gracias! 🇩🇴