Serverless Server-Side Rendering

Natalie Qabazard / @natqab

Senior Software Engineer
@trulia
React, GraphQL, Typescript, AWS

@home
Pasta “a mano”
Fact:

53% of mobile site visits are abandoned if a page takes longer than 3 seconds to load

https://www.doubleclickbygoogle.com/articles/mobile-speed-matters/
Client Rendering
Source: https://medium.com/walmartlabs/the-benefits-of-server-side-rendering-over-client-side-rendering-5d07ff2cefe8
Server Rendering
Server Sending Ready to be rendered HTML Response to Browser

Browser Renders the page, Now Viewable, and Browser Downloads JS

Browser executes React

Page Now Interactable

Source: https://medium.com/walmartlabs/the-benefits-of-server-side-rendering-over-client-side-rendering-5d07ff2cefe8
**Pros**

- SEO
- Improved Perceived Performance

**Cons**

- Added Complexity
- TTFB slower than client rendering
Express
import React from 'react';

class HelloWorld extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      buttonClicked: false
    };
  }

  render() {
    const backgroundStyles = {
      backgroundColor: 'black',
      height: '100%',
      width: '100%
    };
    const textStyles = {
      color: 'white',
      textAlign: 'center',
      paddingTop: '25%',
      paddingBottom: '25%
    };

    return (
      <div style={backgroundStyles}>
        <h1 style={textStyles}>hello world!</h1>
      </div>
    );
  }
}

export default HelloWorld;
import { createElement } from "react";
import { renderToString } from "react-dom/server";
import HelloWorld from "../components/HelloWorld";
import { Router, Request, Response, NextFunction } from "express";
const MarkupHandler = Router();

MarkupHandler.get('/', function(request, response, next) {
  const html = renderToString(createElement(HelloWorld));

  response.send(html);
  next();
});

export { MarkupHandler };
Next.js 6
A Little About Next.js

- Framework for server-rendered components developed by Zeit
- Next.js 6:
  - Typescript support
  - Babel 7
  - Flow types
  - Much more...
Serverless
Why Go Serverless?

- Function as a Service (FaaS)
- Many options, like AWS Lambda
- Pay for what you use
- Supports C#, Go, Java, Python, Node
- Autoscaling
API Gateway
Why API Gateway?

- Endpoints
- Methods
- Event Object
- Proxy to Lambda
How To:
1

Serverless Function

1

API Endpoint

1

Web App
1. Create Lambda

2. Upload .zip to Lambda

3. Create API using API Gateway
   a. **Tip:** enable Lambda proxy integration

4. Create endpoint with GET method

5. Deploy API

6. Invoke URL
Takeaways

- Serverless could save you a ton of money
- SSR improves user experience
- Next.js
  - SSR
  - Typescript
- Terraform: “infrastructure as code”
Thank you!

Natalie Qabazard / @natqab

Senior Software Engineer