

# The Ubiquitous nature of Javascript

Stephen White - 20200220  
Göteborg React JS Meetup

# CALLISTA



— ENTERPRISE —

 Recorded Future

*meetup*

## ReactJS Göteborg

 Göteborg, Sweden

 1,025 members · Public group 

 Organized by **Stephen white** and 1 other

# Agenda

- Inspiration
- Dream Stack
- DX
- Backend for the Frontend - Serverless Framework
- Polly Dictate - landscape and code
- Demo

Inspiration

- The 60Fps syndrome- Yehuda Katz
- AWS Certification
- Rick and Morty

# The 60fps syndrome



**Yehuda Katz** 👑 🌟 @wycats · Jan 31

Replying to @wycats

Backend engineers jerk us all around through an endless debate about architectural paradigms while front end engineers are tasked with delivering a real-time, 60fps, race free, portable experience to customers on devices that range from watches to televisions.

💬 10

↻ 17

❤️ 128



# The 60fps syndrome



**Yehuda Katz** 🥞 🌱 @wycats · Jan 31

And backend folks wonder why GraphQL has gained popularity.

While backend teams argue about whether Ruby is better than Scala, whether object oriented programming is a failure, and whether people should use CoreOS or Docker, front-end teams have shit to do.

💬 13

↻ 13

❤️ 138



# The 60fps syndrome

## Different Concerns

- BE - Non Functional Requirements
- FE - Functional Requirements



# The 60fps syndrome

## Backend

- BE - Non Functional Requirements
  - Accessibility
  - Availability
  - Durability
  - Performance
  - Resilience
  - Scalability

# The 60fps syndrome

## Frontend

- FE - Non and Functional Requirements
  - Accessibility
  - Availability
  - Durability
  - Performance
  - Resilience
  - Scalability
  - Get shit done...

# AWS Developer Cert

The screenshot shows the Eclipse IDE interface for a Java project named 'sample.g'. The Package Explorer on the left shows a project structure with 'src' and 'ANTLR\_HOME/antlr.jar'. The main editor displays the source code for 'sample.g', which defines a grammar for parsing expressions. The code includes a package declaration, a class hierarchy, and grammar rules for 'expression' and 'VARIABLE\_NAME'. The 'Problems' window at the bottom shows three errors related to the 'term' rule not being defined.

```
header {  
    package a.b.c;  
}  
  
class SampleParser extends Parser;  
  
expression  
    : term (PLUS term)*  
    ;  
  
class SampleLexer extends Lexer;  
  
VARIABLE_NAME  
    : ('a'..'z')+  
    ;
```

Description	Resource	In Folder	Location
Exiting due to errors.	sample.g	Sample/src/a/b/c	
Rule 'term' is not defined	sample.g	Sample/src/a/b/c	line 8

The screenshot shows the AWS Management Console home page. It features a grid of service tiles categorized into 'Compute & Networking', 'Database', 'App Services', 'Storage & Content Delivery', 'Deployment & Management', and 'Additional Resources'. The 'Service Health' section indicates that all services are operating normally.

**Additional Resources**

- Getting Started: See our documentation to get started and learn more about how to use our services.
- Trust Advisor: Get product recommendations to save money, improve fault tolerance, improve performance, and close security gaps.
- Service Health: All services operating normally.
- Support: Submit a support case or contact our 24/7 support.
- Get Started Page: [Create Here]

[as]



# Two Brothers in a Van Hindrance to 60fps

[https://www.youtube.com/watch?v=lrc\\_dew1eYw](https://www.youtube.com/watch?v=lrc_dew1eYw)

# The Backend!



# API's - Rest vs GraphQL



# BE Strikes back





# Context Switching



# The Process!



# Technical Dept



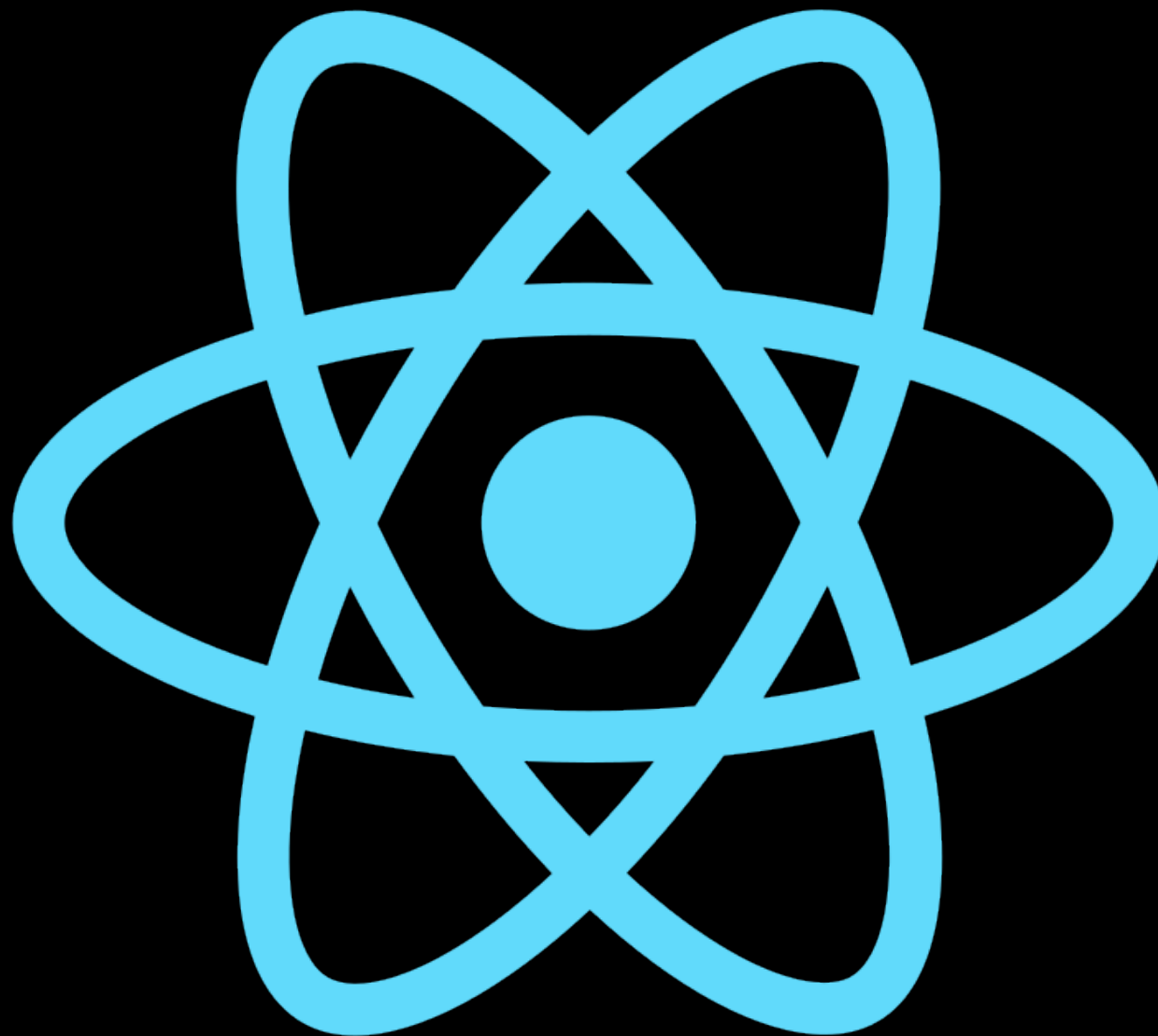
If only we could use JS in  
the BE



# Dream Stack



serverLess  
framework



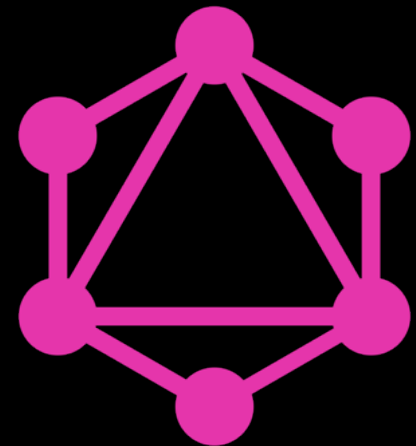
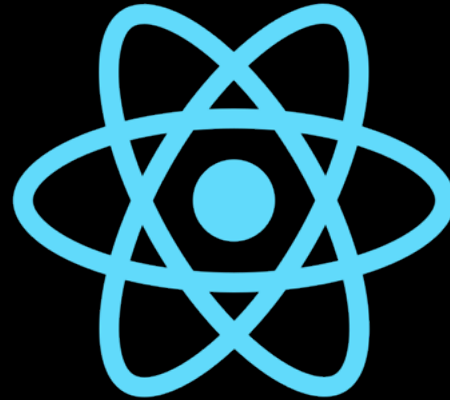
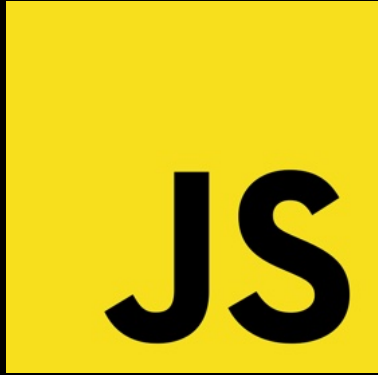
**DX**



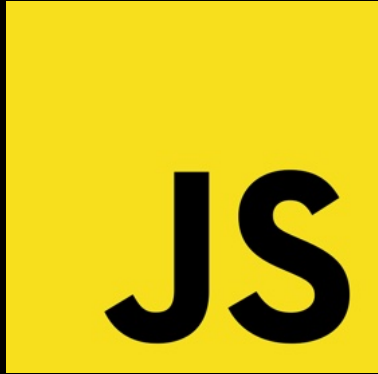
**DX**

**DX**

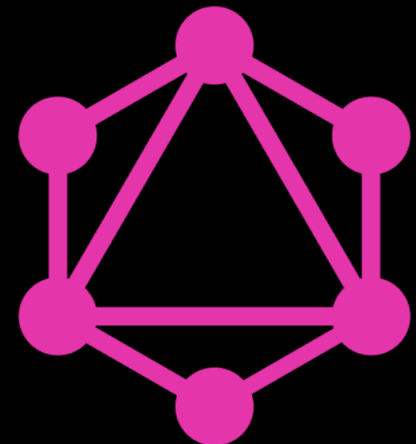
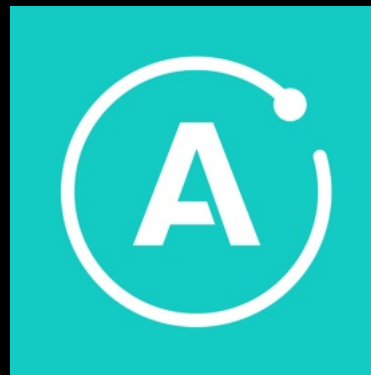
# Frontend Patterns

















# Backend Patterns



Almost the same



# Although where did all those AWS Services come from!!

 <b>Compute</b> EC2 Lightsail <a href="#">↗</a> ECR ECS EKS Lambda Batch Elastic Beanstalk Serverless Application Repository AWS Outposts EC2 Image Builder	 <b>Customer Enablement</b> AWS IQ <a href="#">↗</a> Support Managed Services	 <b>Machine Learning</b> Amazon SageMaker Amazon CodeGuru Amazon Comprehend Amazon Forecast Amazon Fraud Detector Amazon Kendra Amazon Lex Amazon Machine Learning Amazon Personalize Amazon Polly Amazon Rekognition Amazon Textract Amazon Transcribe Amazon Translate AWS DeepLens AWS DeepRacer Amazon Augmented AI	 <b>Application Integration</b> Step Functions Amazon EventBridge Amazon MQ Simple Notification Service Simple Queue Service SWF
 <b>Storage</b> S3 EFS FSx S3 Glacier Storage Gateway AWS Backup	 <b>Blockchain</b> Amazon Managed Blockchain	 <b>Satellite</b> Ground Station	 <b>AWS Cost Management</b> AWS Cost Explorer AWS Budgets AWS Marketplace Subscriptions
 <b>Database</b> RDS DynamoDB	 <b>Quantum Technologies</b> Amazon Braket <a href="#">↗</a>	 <b>Management &amp; Governance</b> AWS Organizations CloudWatch AWS Auto Scaling CloudFormation CloudTrail Config OpsWorks	 <b>Customer Engagement</b> Amazon Connect Pinpoint Simple Email Service
		 <b>Analytics</b> Athena EMR CloudSearch Elasticsearch Service Kinesis	 <b>Business Applications</b> Alexa for Business Amazon Chime <a href="#">↗</a> WorkMail

# Serverless and JS gives us

- DX
- Minimal Context switch
- Tooling
- Code as Infrastructure
- Patterns and Conventions
  - Functional Programming

Move towards Cross  
Functional Teams

# Serverless Framework, what it offers

- JS is king!
  - JVM based languages, Kotlin, Java ... Scala
  - GO
  - Rust
- Code as Infrastructure
- Plugin Model
  - Offline
  - Webpack - hot reload
  - SNS / SQS - Offline!!
  - ....
- Deploy to multiple clouds ( Although most support for AWS)



# Lambda

```
export const get = async event => await ns.get(event.path.id)
```



Thin as possible



Most of your code



Notes DB



# Code As Infrastructure

```
notes-get:  
  handler: src/notes.get  
  environment:  
    TABLE: ${self:custom.pollynotesDb}  
  events:  
    - http:  
      path: notes/{id}  
      method: GET  
      integration: lambda  
      cors: true  
      request:  
        parameters:  
          paths:  
            id: true  
      authorizer: aws_iam
```



API GW

/notes/{id}  
GET



GetNote

# Test

```
it("can create a note", async () => {  
  const note = await api.post({ note: "the cat on the mat" });  
  expect(note.note).toEqual("the cat on the mat");  
  await api.delete(note.id);  
});
```

Local Remote

Demo

# Polly Dictate



IOS



Android



Web



Cognito

Web Socket



API GW



Publish

Dynamo Stream



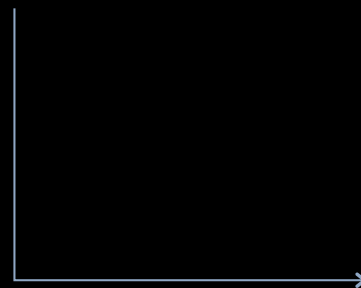
Subscriptions



GraphQL



Notes DB



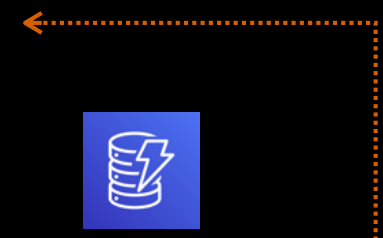
Dictate



Mp3 Bucket



Amazon Polly



# Expo - deploy

```
App.json
"expo": {
  "bundleIdentifier": "polly-native-
expo",
  "updates": {
    "enabled": true
  }
  "checkAutomatically": true
}
```



Expo App

```
App.json
"expo": {
  "bundleIdentifier": "polly-native-dev",
  "updates": {
    "enabled": true
  }
  "checkAutomatically": true
}
```



Dev App

```
App.json
"expo": {
  "bundleIdentifier": "polly-native",
  "updates": {
    "enabled": false
  }
  "checkAutomatically": false
}
```



Prod App

Channels

Default

Develop

Production

```
expo publish --release-channel <channelName>
```

# Conclusions

- Two Bros
  - The backend, move away from functional teams to cross functional with a common language... JS
  - API's Rest vs GQL, let the api be driven by the needs of the client not by what the BE have time to provide. Make it flexible and let it evolve.
  - BE Strikes back! Talk the same language have the same concerns ( i.e the customer )
  - Context Switching - mitigated with the use of JS in all layers and the use of a decided programming style, be it functional or OOP.
  - The Process - with greater efficiency and productive, trust and empowerment of the developer follows, less trust generally means more process, more trust infers that the devs can focus on getting stuff done!
  - Technical Dept - will decrease if more devs take responsibility for the code base. With greater DX devs have more time to fine tune and polish their code.
  - JS in all levels!

# Conclusions

- Two Bros - The Process
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
  - <https://agilemanifesto.org>



# Conclusions

- React Native
  - gives a native experience.
  - Web skills for native apps.
  - Great community
  - Modern standards

# Demo List

- Show the app in IOS and Android Sims
- Show the Web App
- Run the BE locally
  - Show GraphQL
  - Add a gql resolver
    - SearchNotes
  - Back to GraphQL

# Polly Dictate



IOS



Android



Web



aws



Cognito



API GW



Mp3 Bucket



CreateNote



UpdateNote



GetNote



ListNotes



RemoveNote



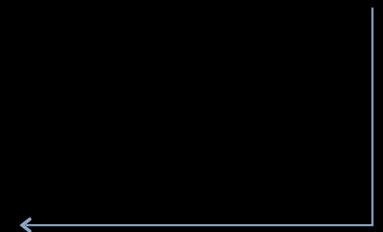
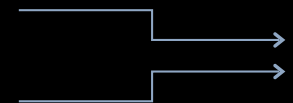
Dictate



GraphQL



Notes DB





Yehuda Katz  

@wycats



This back and forth about microservices vs monoliths is hopelessly stuck in a backend-focused world.

What else should we expect? Front-end engineers are given the incredibly hard tasks of caring for the end user. In too many places, backend engineers still control everything.

2:14 AM · Jan 31, 2020 · [Twitter for Android](#)



**Yehuda Katz** 🥨 🌱 @wycats · Jan 31

And front-end engineers have to fight for crumbs of support from the hallowed halls of the backend team.

I once watched a front-end team spend years asking their backend team for a single field to store arbitrary data in. Sorry, the big brains in the backend team know best.

💬 11

↻ 4

❤️ 78





**Yehuda Katz** 🥨 🌸 @wycats · Jan 31

We're far overdue for a conversation about web application architecture where front-end developers took a seat at the table, and where our concerns mattered as much as the latest iteration of the argument about monoliths vs microservices.

💬 15

↻ 15

❤️ 123



# Enterprise Damage