Mobile Applications

2020-2021

Prerequisites

- Modern programming language
- Object oriented
- Statically types
- IDE - IntelliJ/Android Studio or Visual Studio Code

What you should know...

- Basics:
  - Object-oriented programming
  - Classes, methods
  - Exception handling

Bonus

- Functional Programming
- Lambdas
- Higher Order Functions
- Reactive Programming

Options

2007

2008

2010

Native Options

2008

2017

2007

2014

Non-Native Options

2013

Hybrid App

Non-Native Options

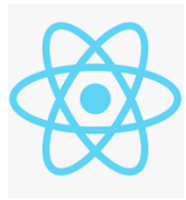
Compiled App

2014

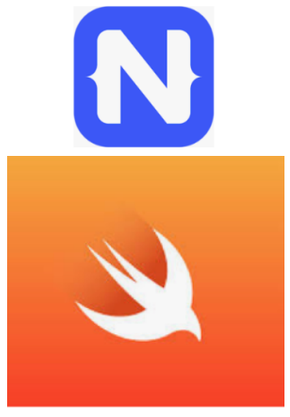
2015

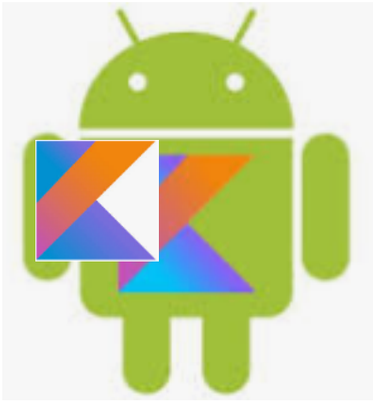
2017

Timeline

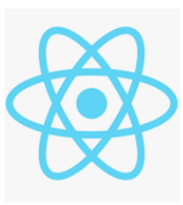


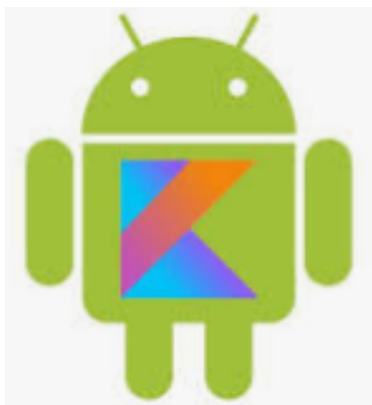
2007 2008 2011 2013 2014 2015 2017



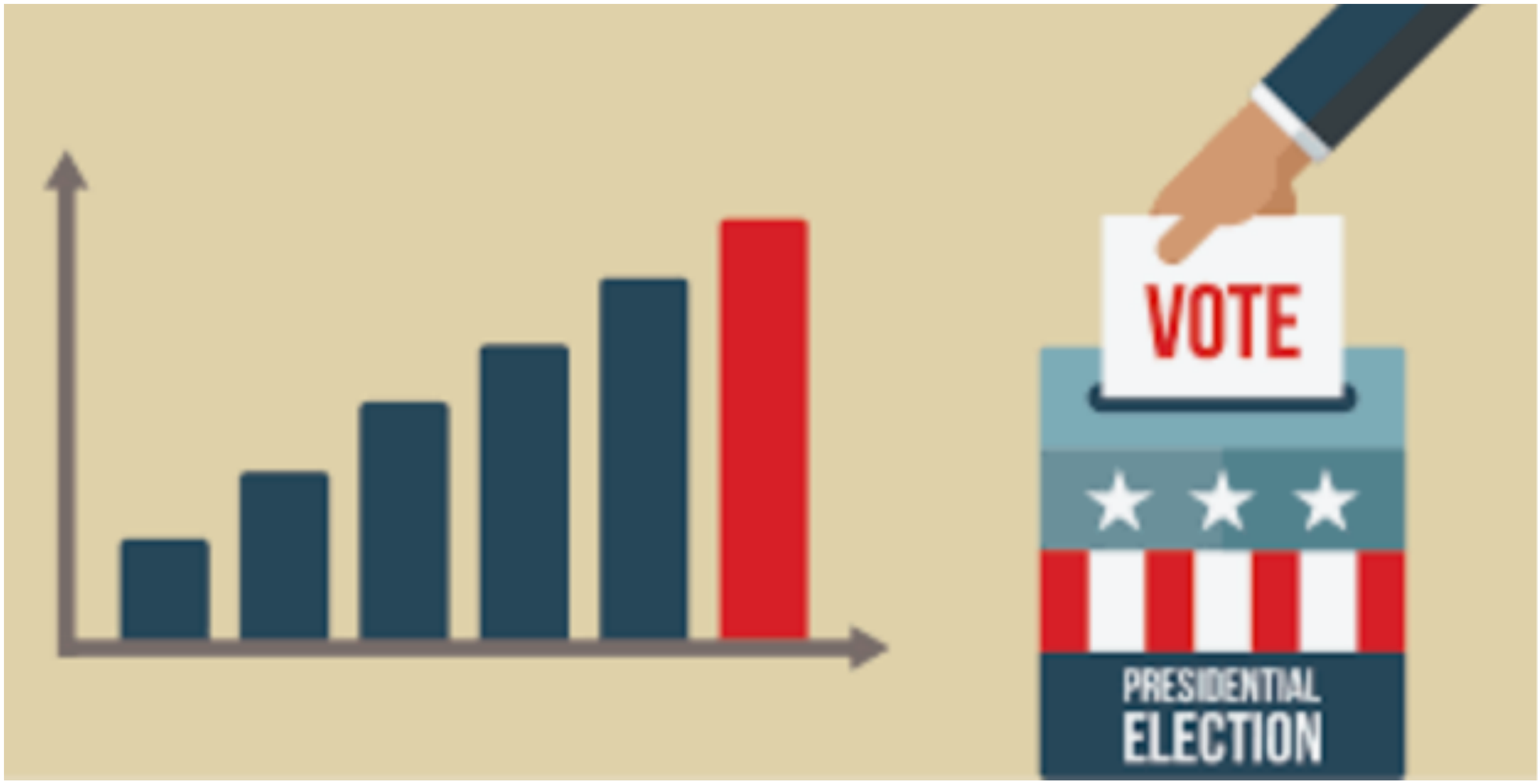


What to learn?

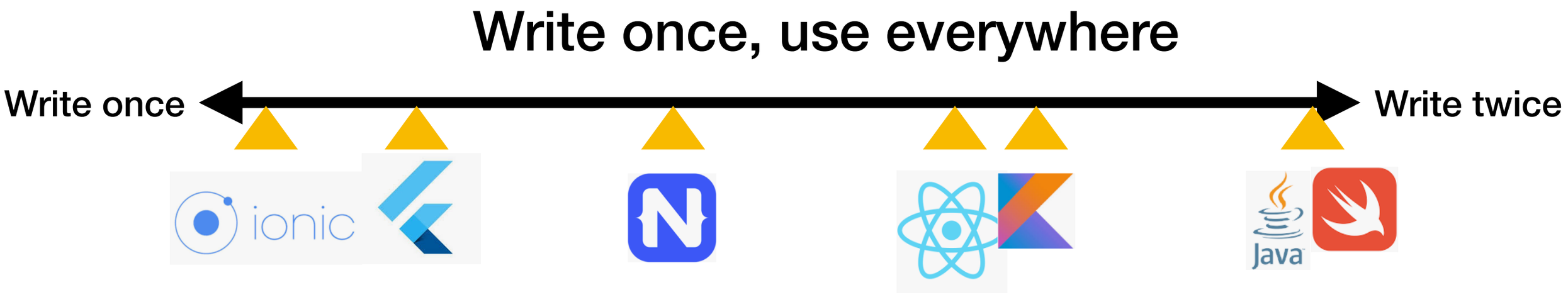


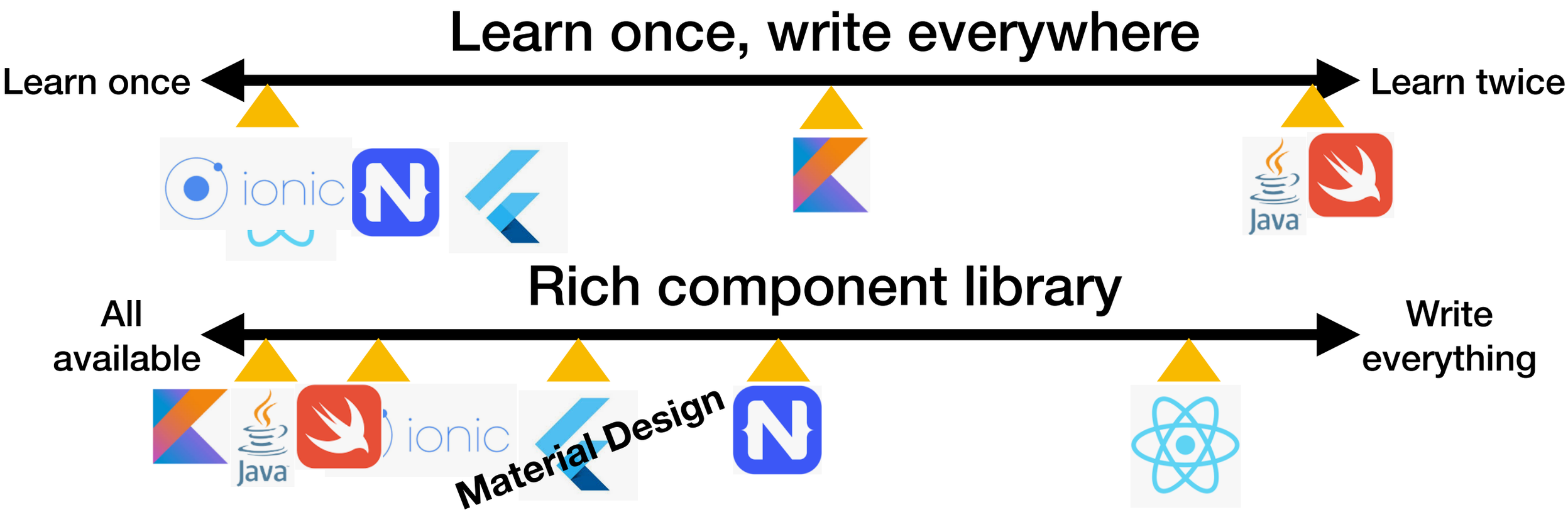


bit.ly/maQuiz2020

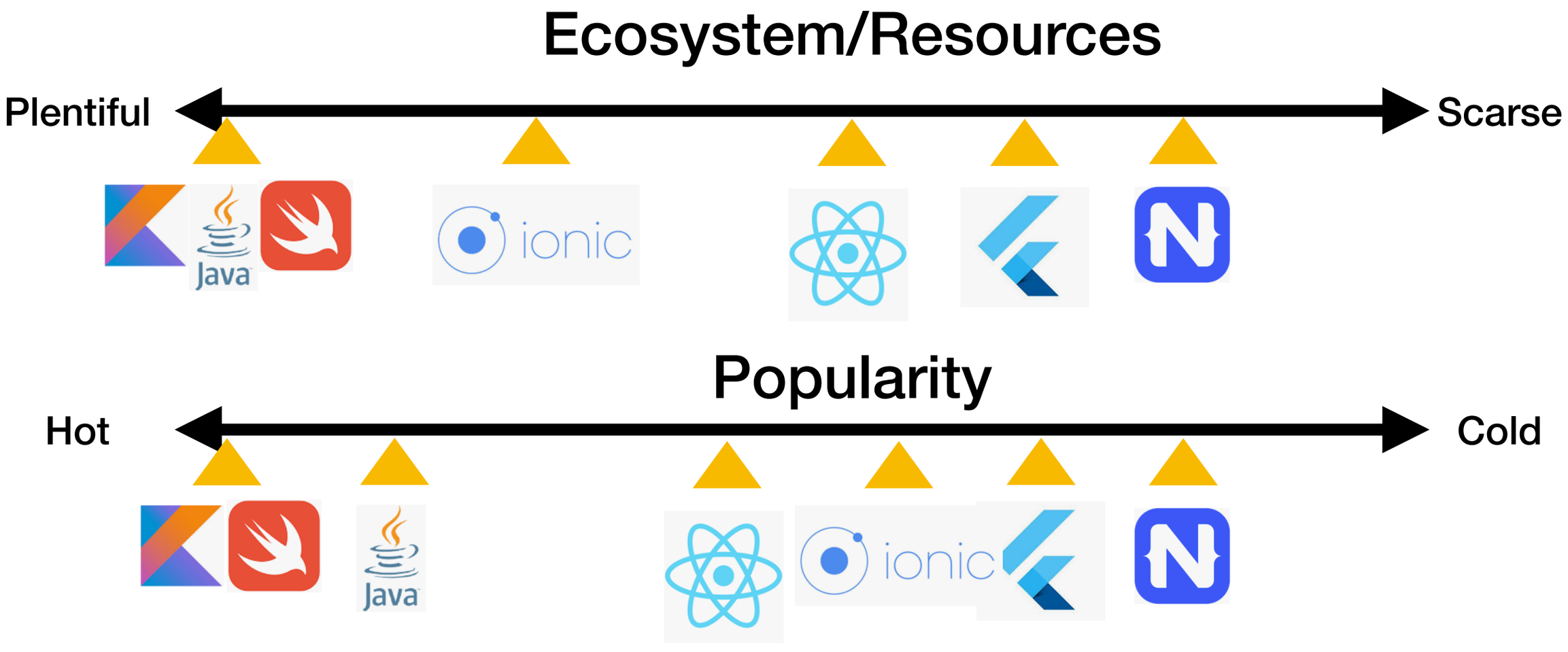


Comparison

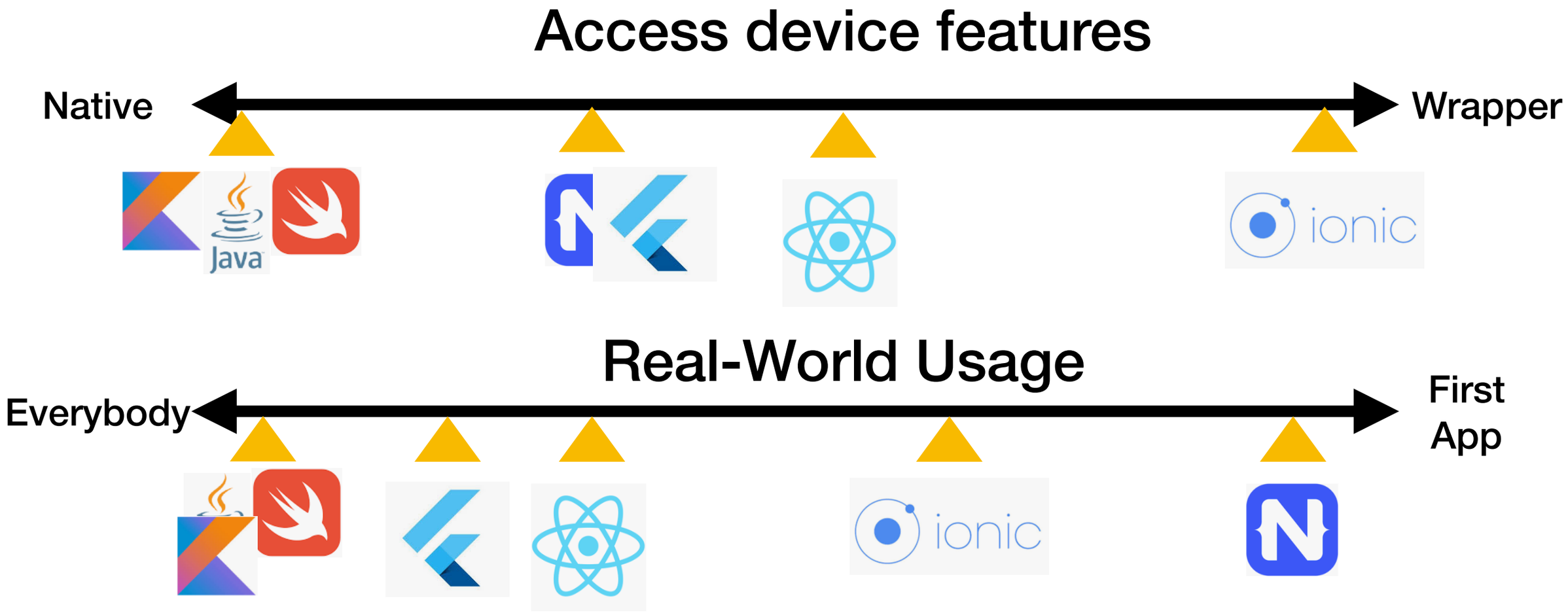

Write once, use everywhere
Write once
Write twice
ionic
Java


Learn once, write everywhere
Learn once
Learn twice
ionic
Java
Rich component library
All available
Write everything
Java
ionic
Material Design

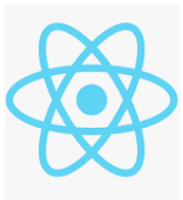
Comparison


Ecosystem/Resources
Plentiful
Scarse
Java
ionic
Popularity
Hot
Cold
Java
ionic
Performance
Native
Wrapper
ionic

Comparison


Access device features
Native
Wrapper
Java
ionic
Real-World Usage
Everybody
First App
ionic

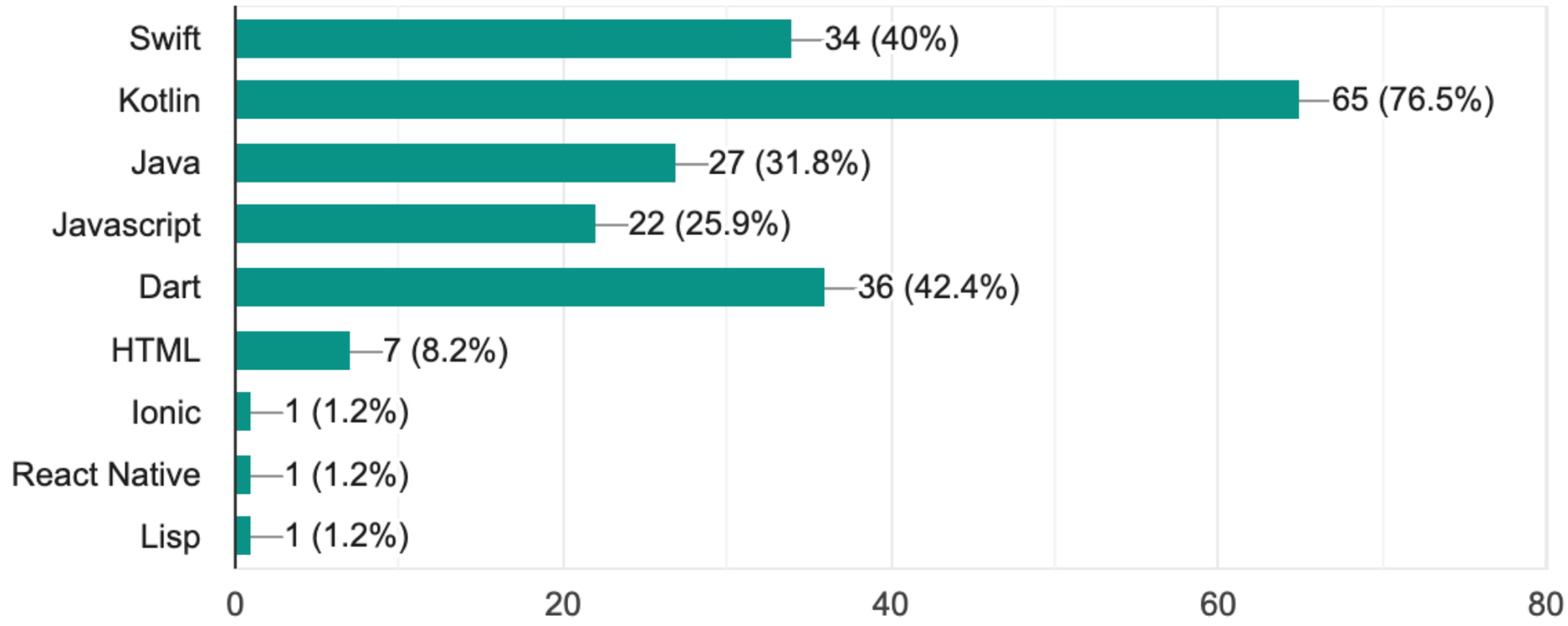
Previous years



Responses from last year

What language would you like to use/learn?

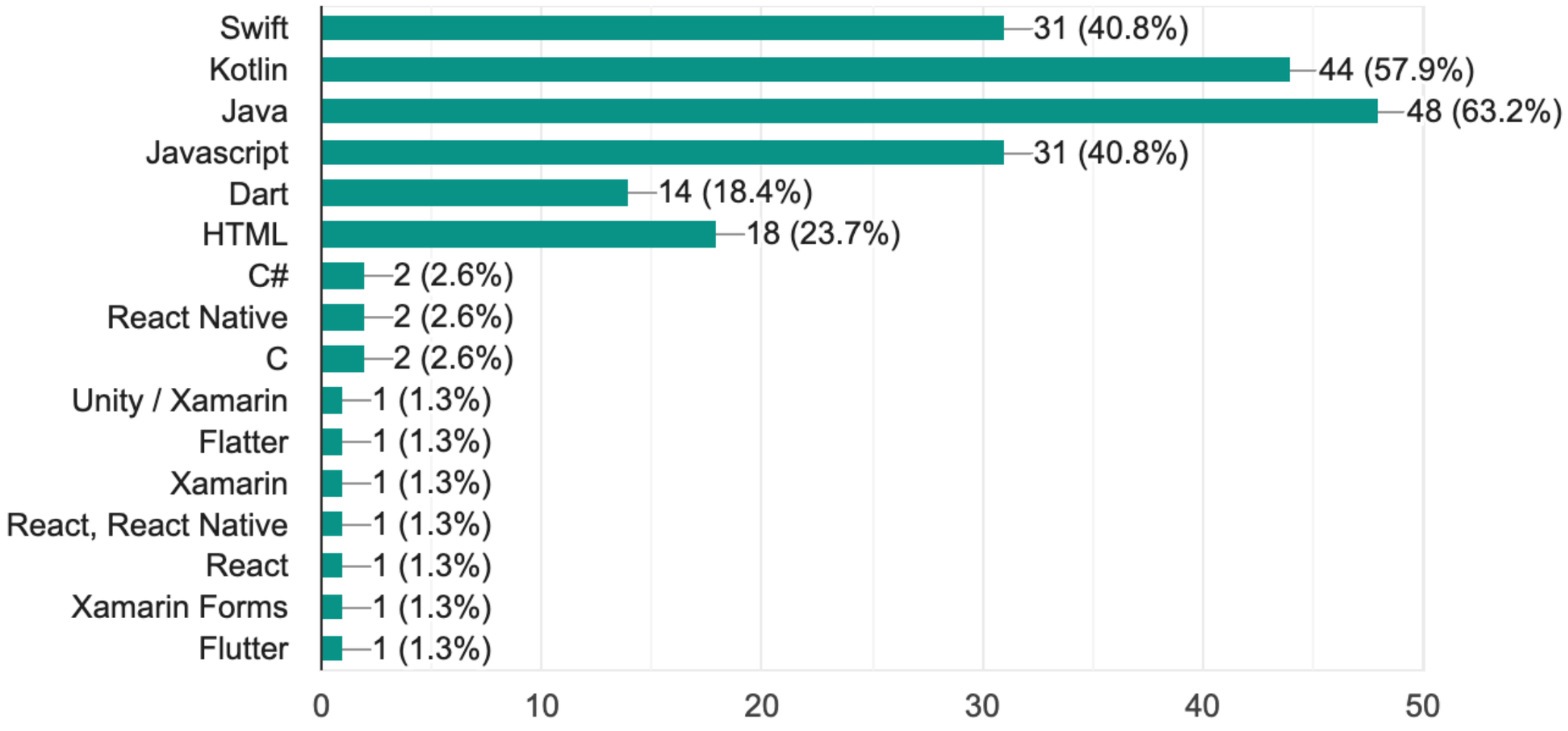
85 responses



Responses from 2018

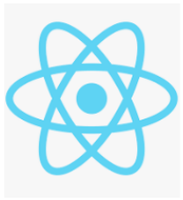
What language would you like to use/learn?

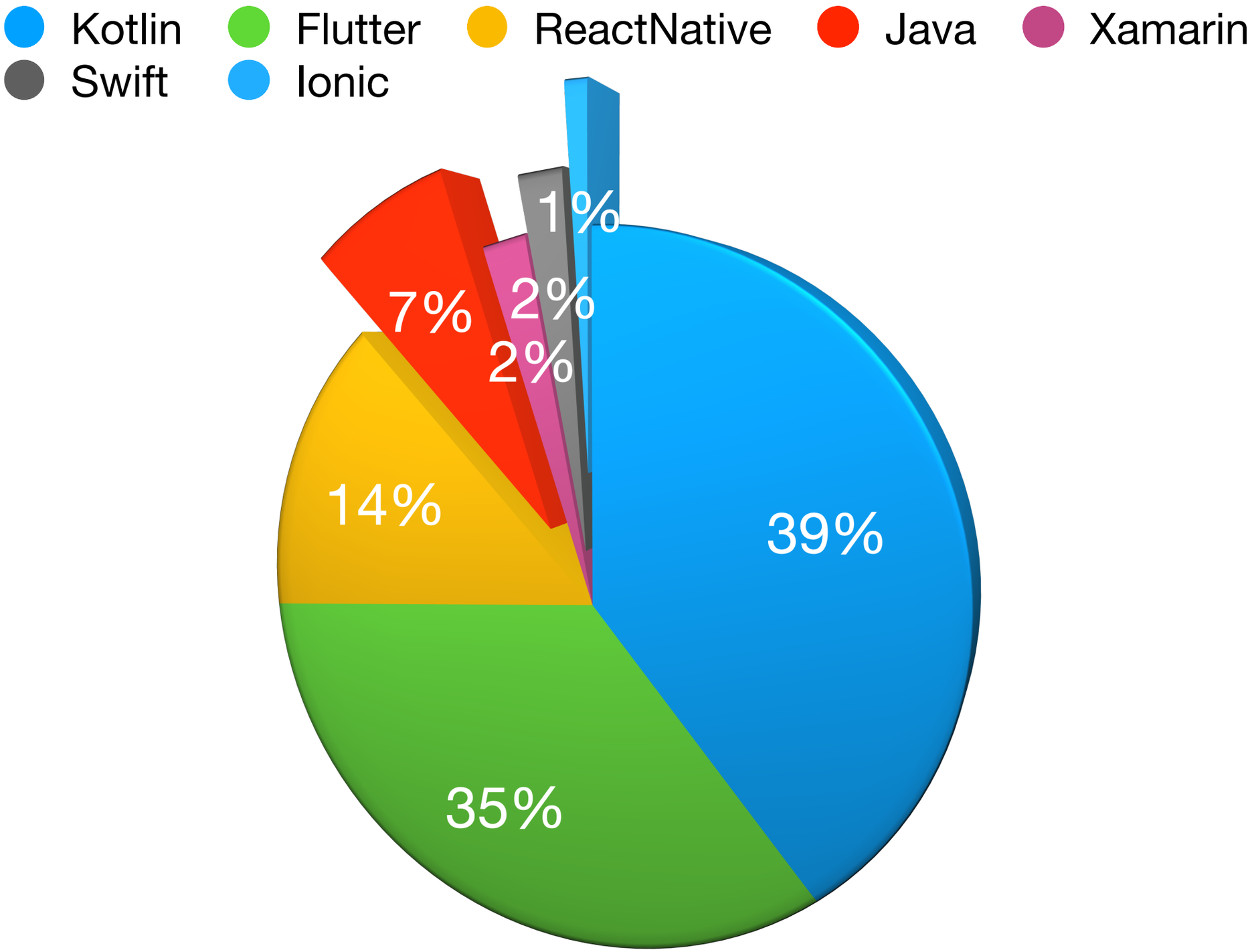
76 responses

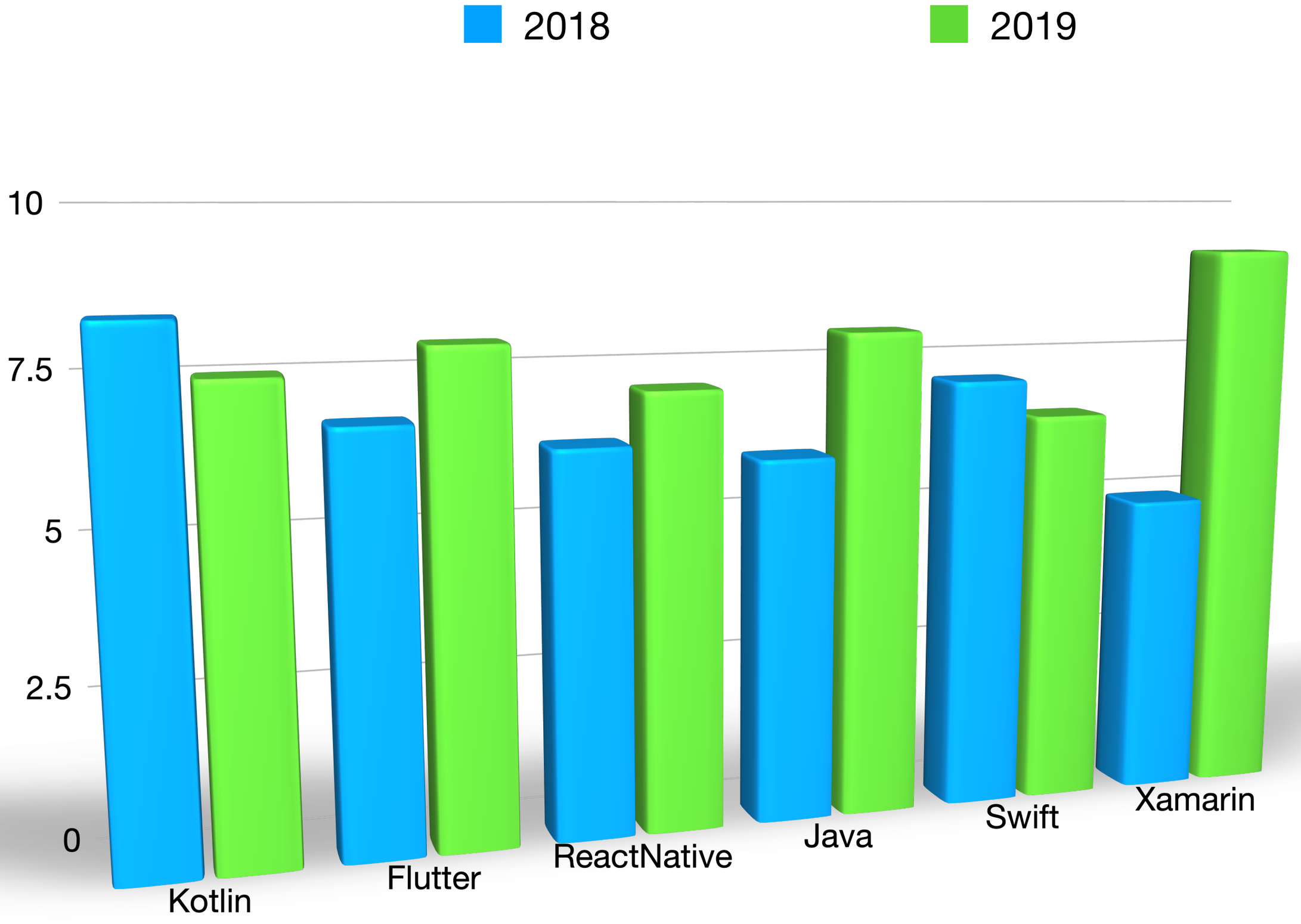


Verify the quiz

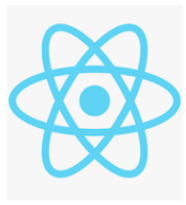
Previous Year




Kotlin
Flutter
ReactNative
Java
Xamarin
Swift
Ionic
1%
2%
2%
7%
14%
39%
35%


2018
2019
10
7.5
5
2.5
0
Kotlin
Flutter
ReactNative
Java
Swift
Xamarin

This year



Why Kotlin

- Modern programming language
- Object oriented
- Lambdas, Coroutines, Properties
- Since 2011
- Open Sources 2012
- Official First Class Android Citizen since 2017
- IntelliJ and Android Studio 3.0+

Why Kotlin

```java
public class Aquarium {

    private int mTemperature;

    public Aquarium() { }

    public int getTemperature() {
        return mTemperature;
    }

    public void setTemperature(int mTemperature) {
        this.mTemperature = mTemperature;
    }

    @Override
    public String toString() {
        return "Aquarium{" +
                "mTemperature=" + mTemperature +
                '}';
    }
}
```

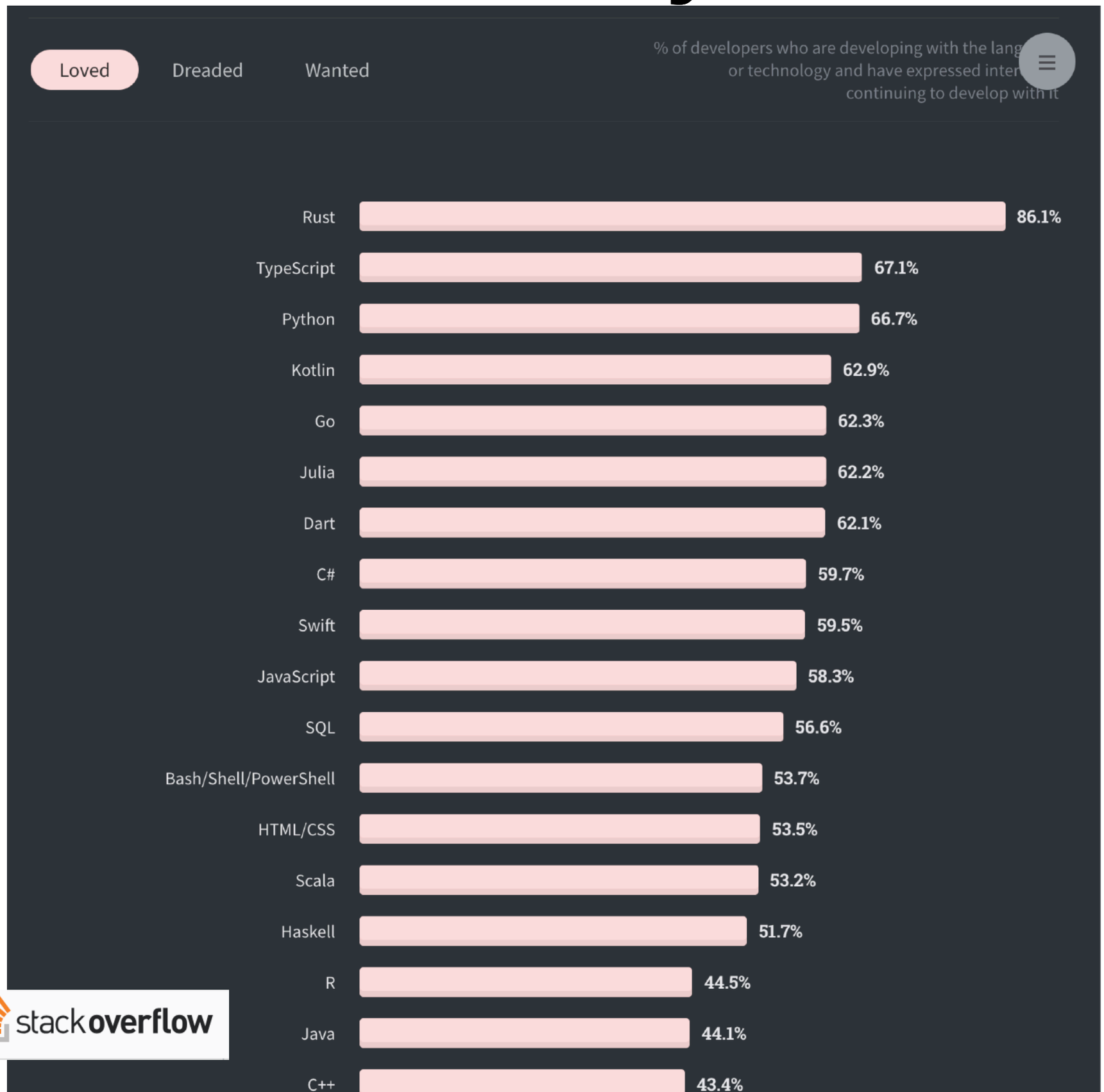
Java

Why Kotlin

```
class Aquarium (var temperature: Int = 0)
```

Kotlin equivalent

Why Kotlin



stackoverflow

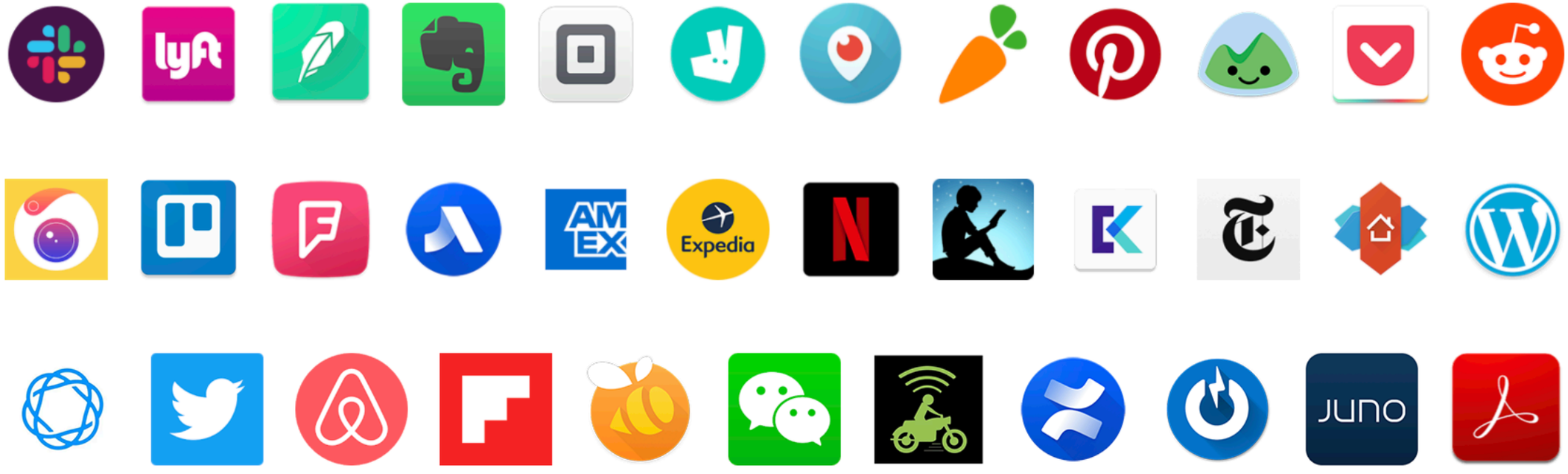
Why Kotlin


Kotlin
LEARN
COMMUNITY
TRY ONLINE
Reference
Tutorials
Books
More resources
Overview
Kotlin for Server Side
Kotlin for Android
Kotlin for JavaScript
Kotlin for Native
Coroutines
Multiplatform
What's New in 1.1
What's New in 1.2
What's New in 1.3
Getting Started
Basics
Classes and Objects
Functions and Lambdas
Multiplatform Programming
Edit Page
Multiplatform projects are an experimental feature in Kotlin 1.2 and 1.3. All of the language and tooling features described in this document are subject to change in future Kotlin versions.
Working on all platforms is an explicit goal for Kotlin, but we see it as a premise to a much more important goal: sharing code between platforms. With support for JVM, Android, JavaScript, iOS, Linux, Windows, Mac and even embedded systems like STM32, Kotlin can handle any and all components of a modern application. And this brings the invaluable benefit of reuse for code and expertise, saving the effort for tasks more challenging than implementing everything twice or multiple times.
How it works
Overall, multiplatform is not about compiling all code for all platforms. This model has its obvious limitations, and we understand that modern applications need access to unique features of the platforms they are running on. Kotlin doesn't limit you to the common subset of all APIs in the world. Every component can share as much code as needed with others but can access platform APIs at any time through the expect/actual mechanism provided by the language.

Apps built with Kotlin

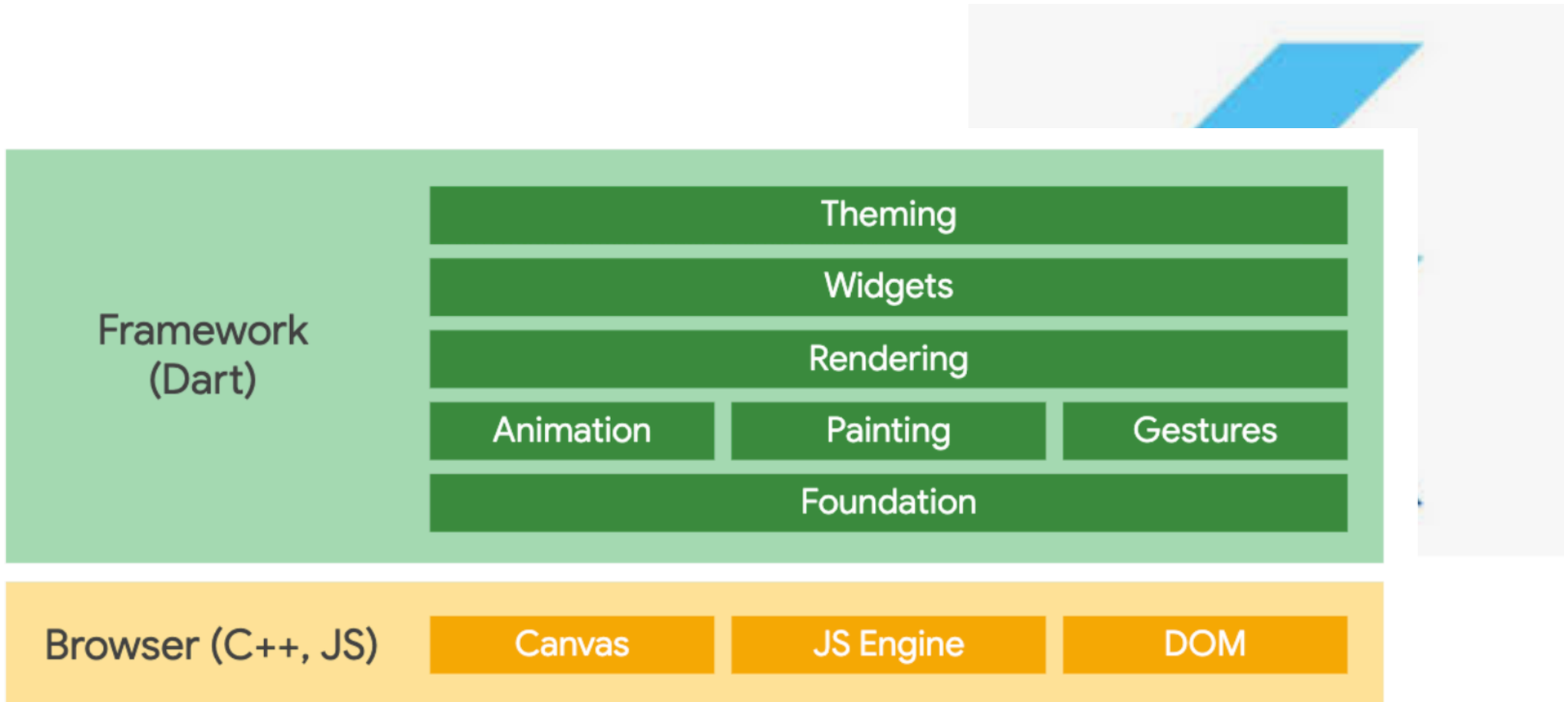
Many apps are already built with Kotlin—from the hottest startups to Fortune 500 companies. Learn how Kotlin has helped their teams become more productive and write higher quality apps.

SEE DEVELOPER STORIES



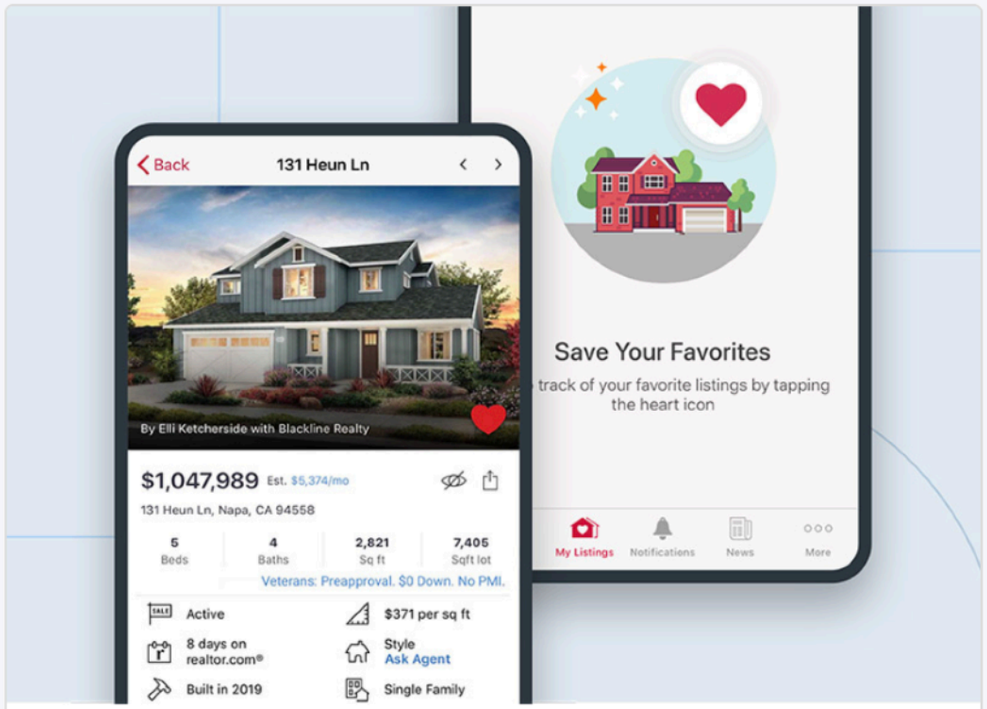
Why Flutter

#1 Hot reload
#2 Full set of (Material Design) widgets
#3 Everything is a Widget
#4 Different themes for Android/iOS
#5 Many packages



Apps take flight with Flutter

See how customers are using Flutter to make beautiful apps in record time



realtor.com®

With listings updated in real time, Realtor.com's award-winning app helps users find the home of their dreams.

Download

Tencent 腾讯

Tencent uses Flutter throughout the company for several apps including AITeacher, Now Live, K12, Mr. Translator, QiDian, and DingDang.

Watch video

Course Goals

- Knowledge of key base concepts for developing mobile applications.
- Learn the Android platform.
- Learn a framework to develop multi-platform applications (Android&iOS)

Lecture outcomes

- Understand the generated artifacts
- Lifecycle of applications, activities and fragments.
- Use logs to debug and study the behavior.