Mobile Applications

2018-2019

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

Options

2007

2008

Options

2007

2008

2010

Options

2007

2008

2010

Options

2007

2008

Native Options

2008

2007

Native Options

2008

2007

Native Options

2008

2007

2014

Native Options

2008

2017

2007

2014

Non-Native Options

2013

Non-Native Options

Non-Native Options

Hybrid App

Non-Native Options

2014

Non-Native Options

2014

2015

Non-Native Options

2014

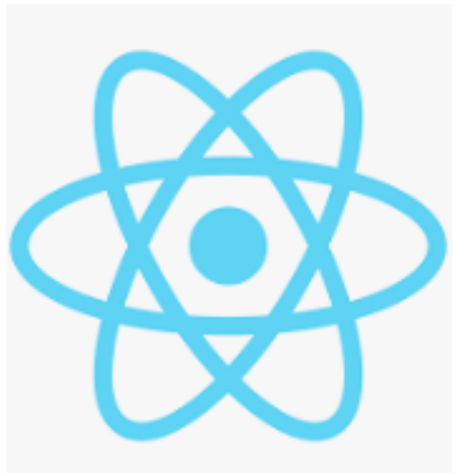
2015

2017

Non-Native Options

2014

2017



2015

Non-Native Options

2014

2017

2015

Non-Native Options

Compiled App

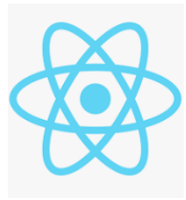
2014

2015

2017

Timeline

Timeline



2007

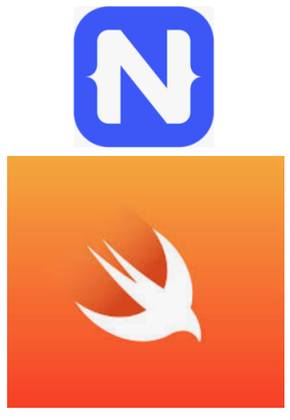
2008

2013

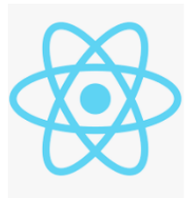
2014

2015

2017



Timeline



2007 2008 2011 2013 2014 2015 2017

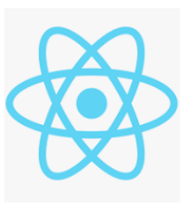
What to learn?

What to learn?

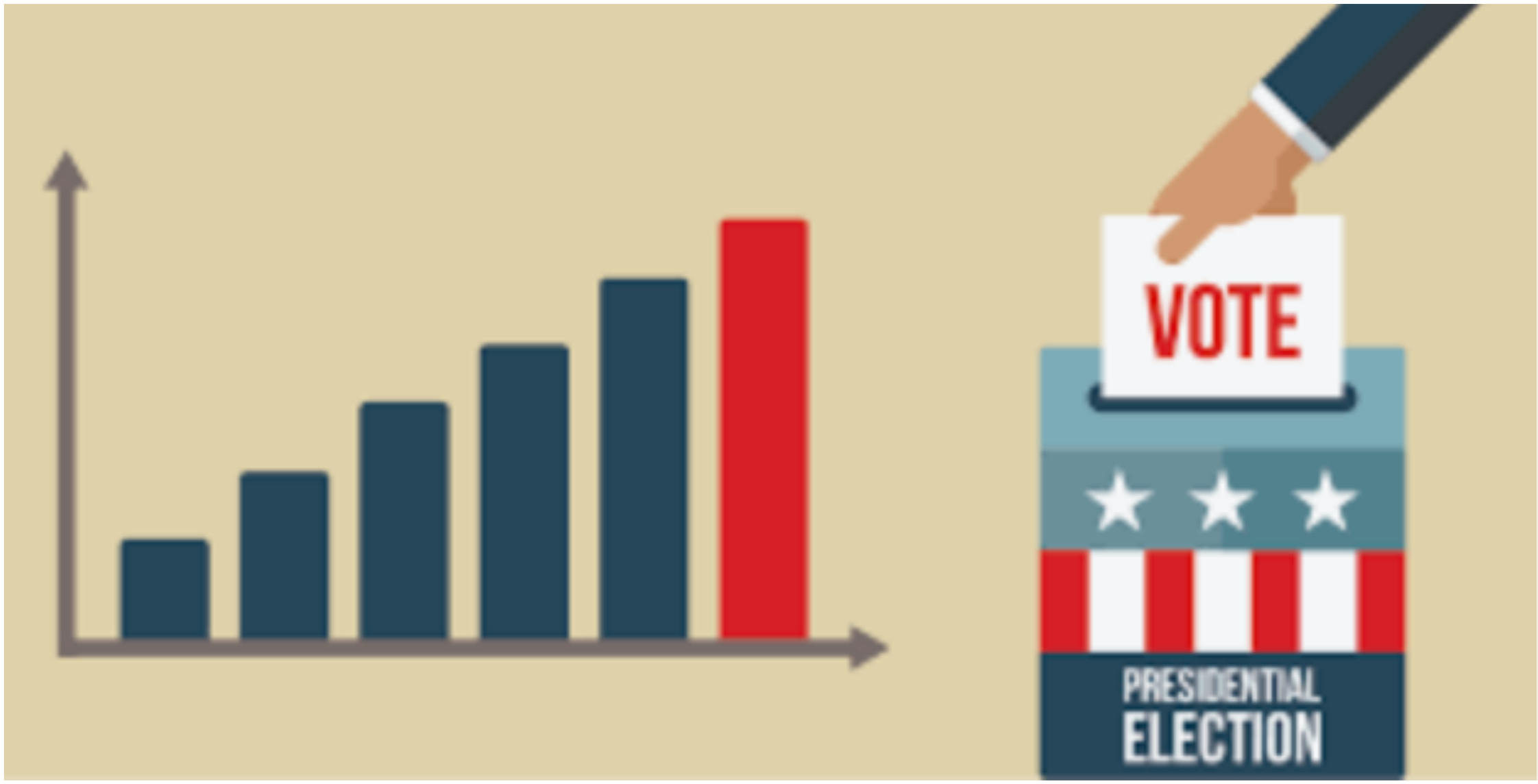
OBJ-C

Java

What to learn?



bit.ly/maQuiz



Comparison

Comparison

Write once, use everywhere

Comparison

Write once, use everywhere

Comparison

Write once, use everywhere

Comparison

Write once, use everywhere

Comparison

Write once, use everywhere

Comparison

Write once, use everywhere

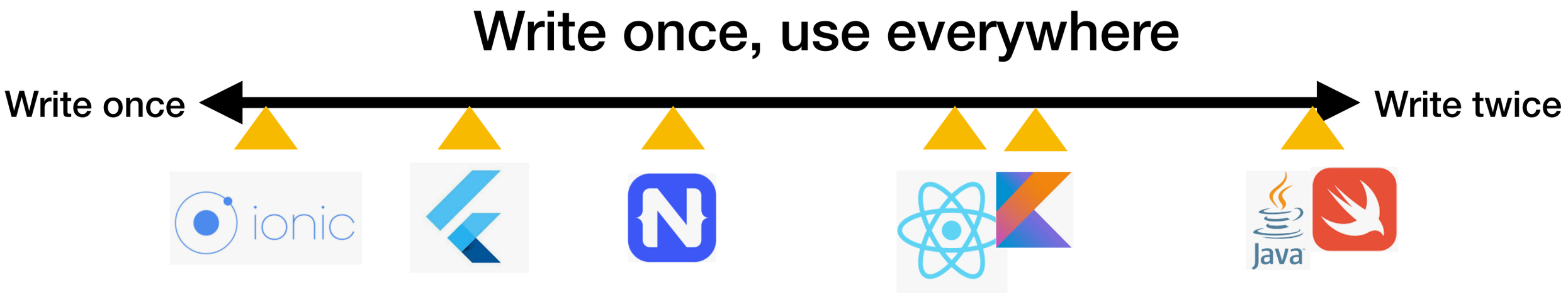
Comparison

Write once, use everywhere

Comparison

Write once, use everywhere

Comparison

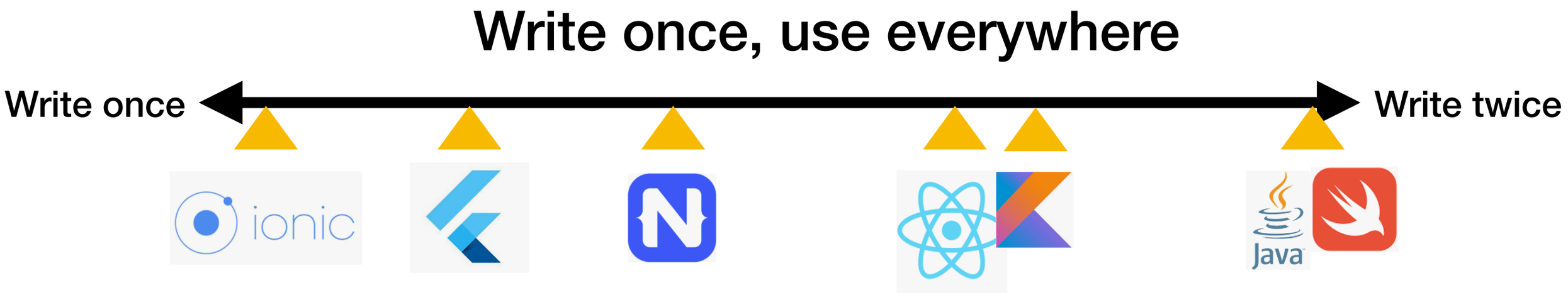

Write once, use everywhere
Write once
Write twice
ionic
Java

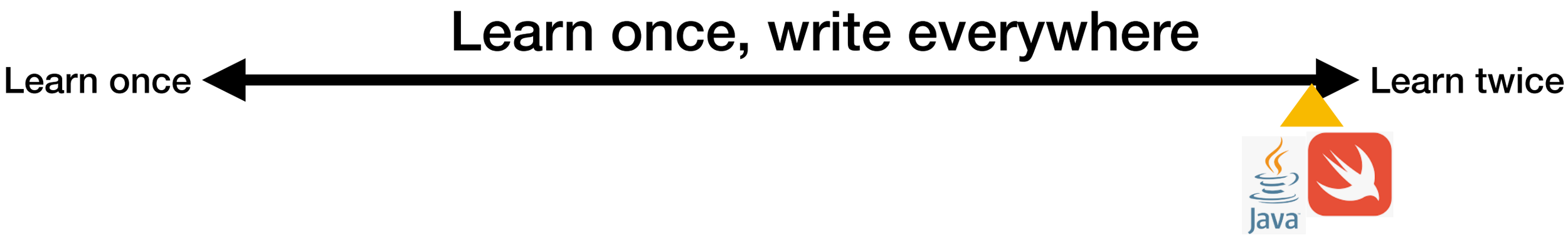
Comparison

Write once, use everywhere

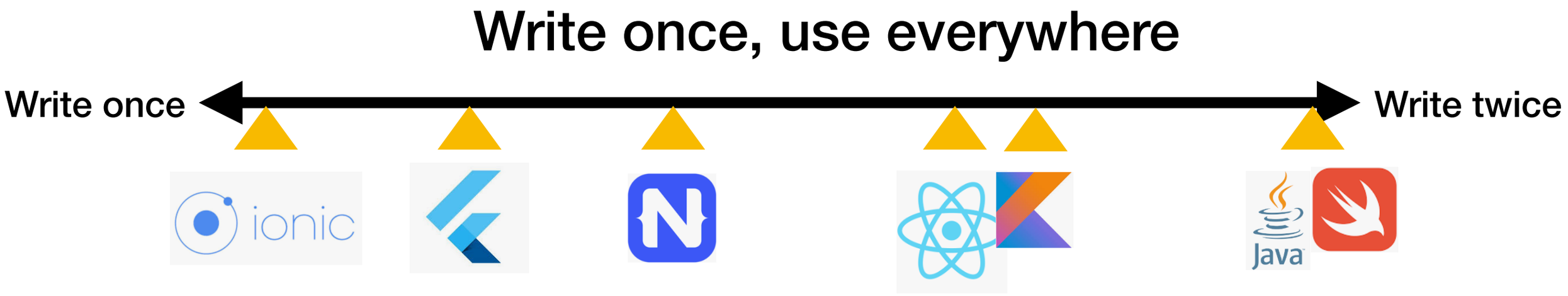
Learn once, write everywhere

Comparison


Write once, use everywhere
Write once
Write twice
ionic
Java

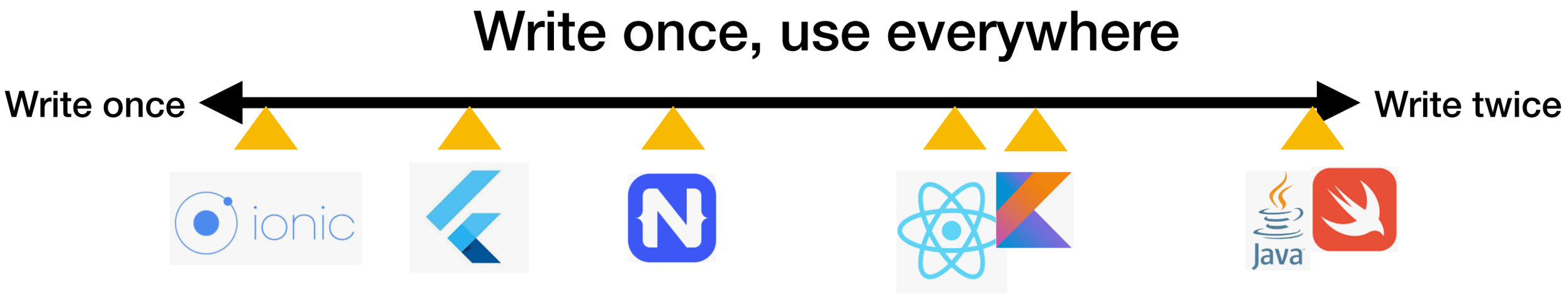

Learn once, write everywhere
Learn once
Learn twice
Java

Comparison


Write once, use everywhere
Write once
Write twice
ionic
Java

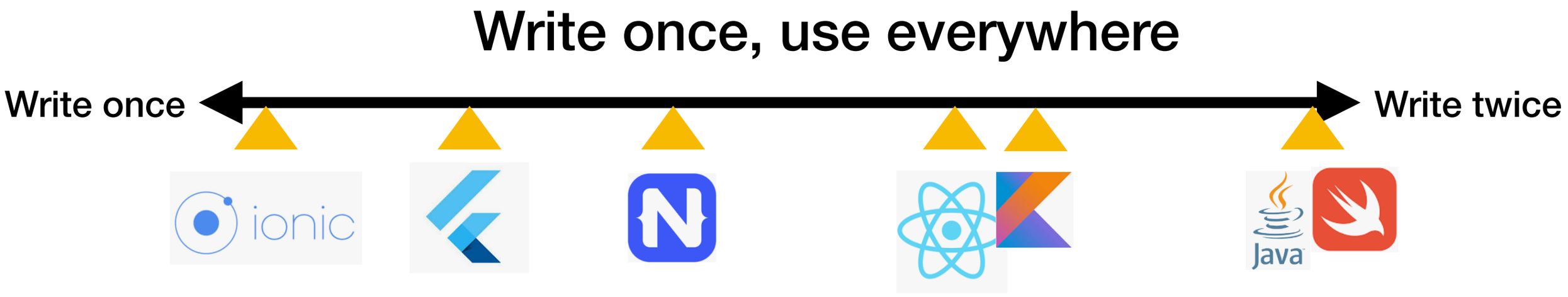

Learn once, write everywhere
Learn once
Learn twice
Java

Comparison


Write once, use everywhere
Write once
Write twice
ionic
Java

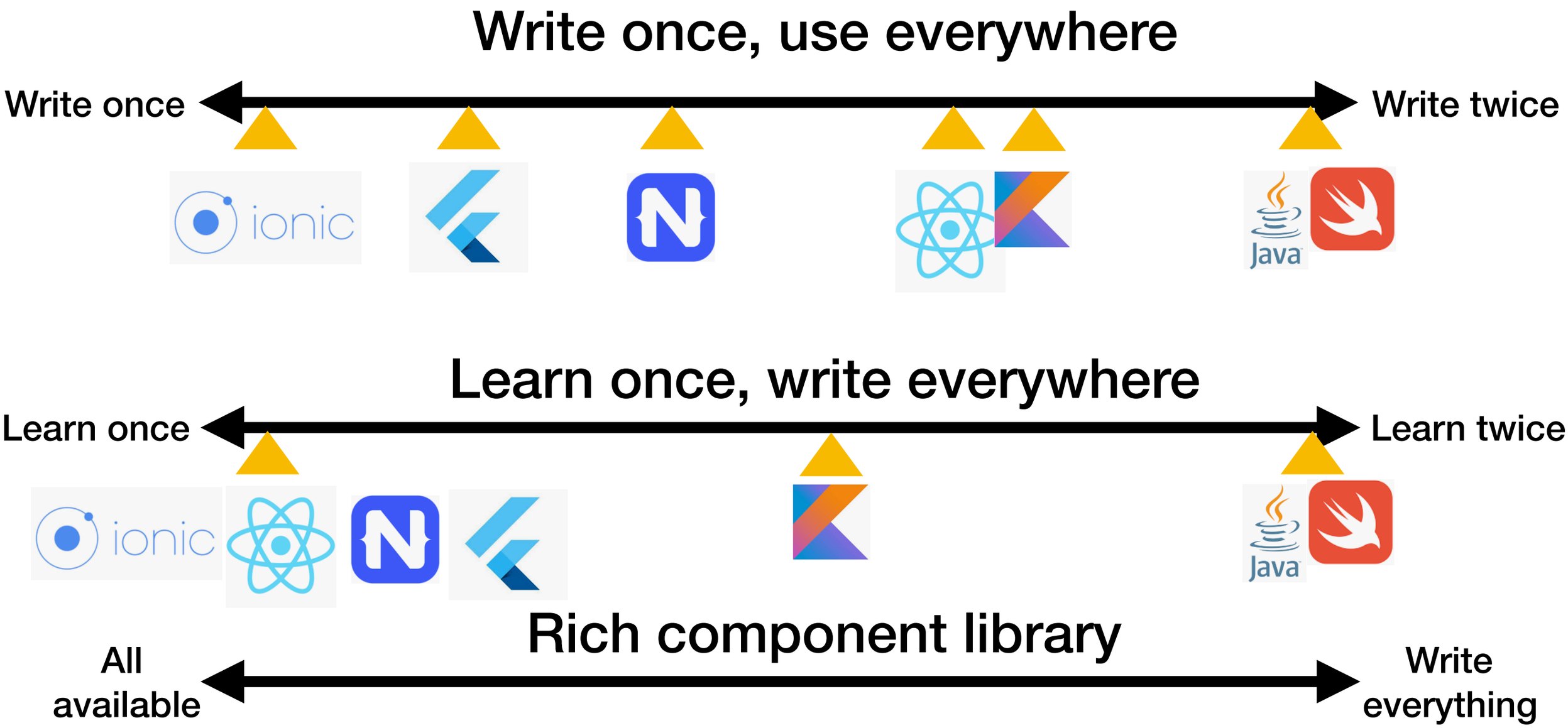

Learn once, write everywhere
Learn once
Learn twice
ionic
Java

Comparison

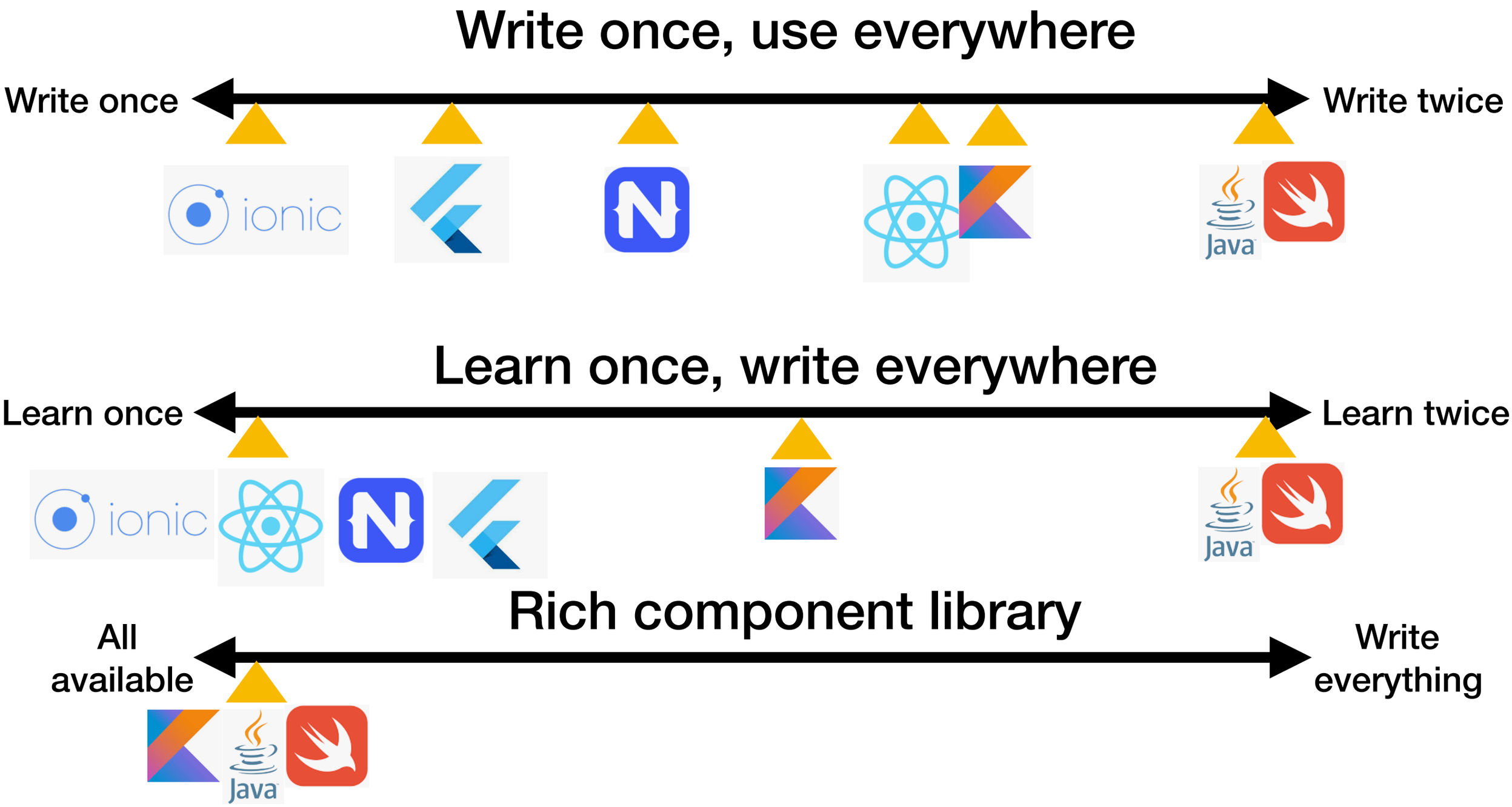

Write once, use everywhere
Write once
Write twice
ionic
Java


Learn once, write everywhere
Learn once
Learn twice
ionic
Java

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

Comparison

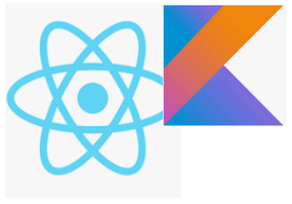

Write once, use everywhere
Write once
Write twice
ionic
Java
Learn once, write everywhere
Learn once
Learn twice
ionic
Java
Rich component library
All available
Write everything
Java

Comparison

Write once, use everywhere

Write once

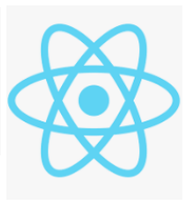
Write twice



Learn once, write everywhere

Learn once

Learn twice

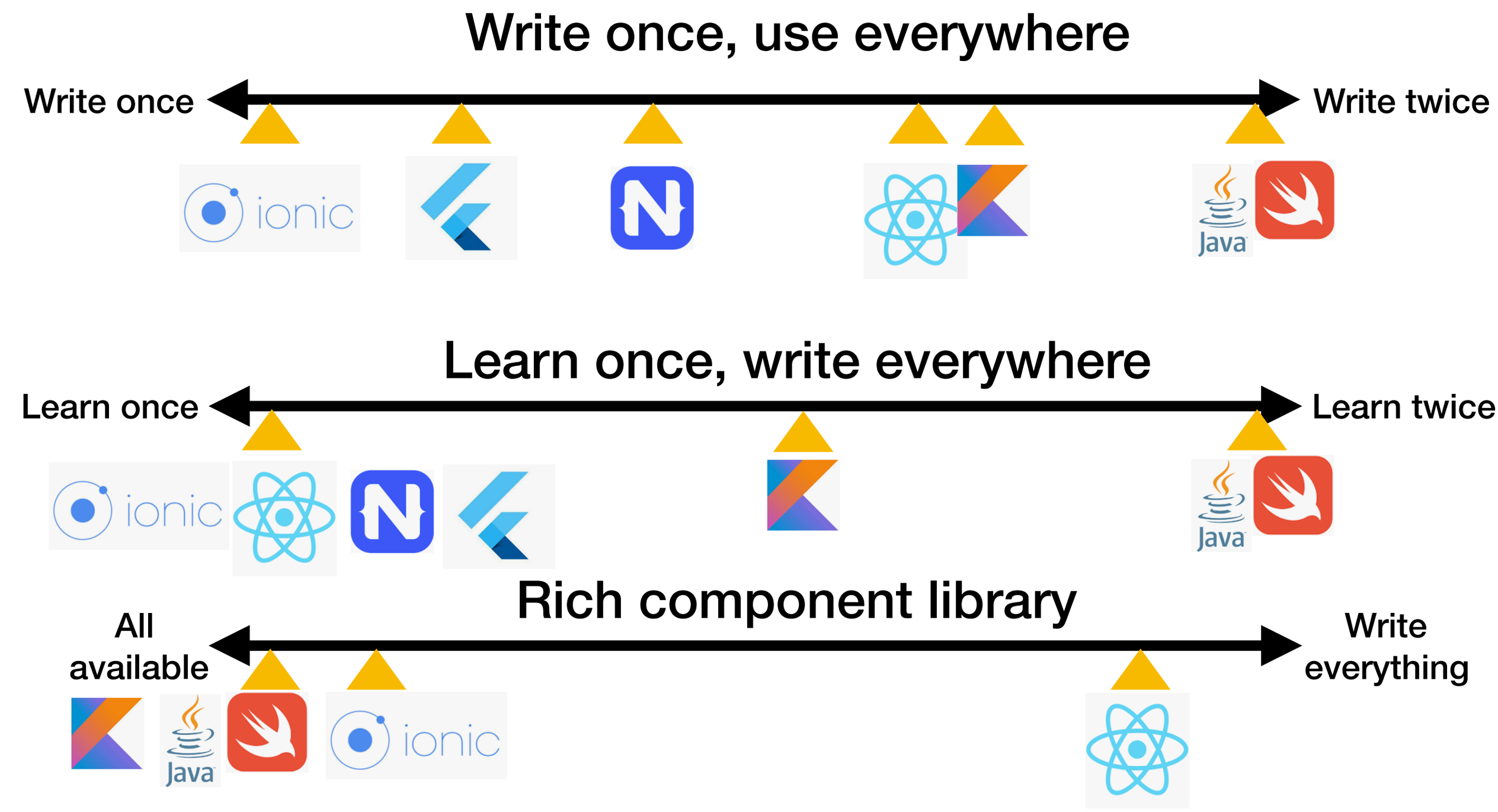


Rich component library

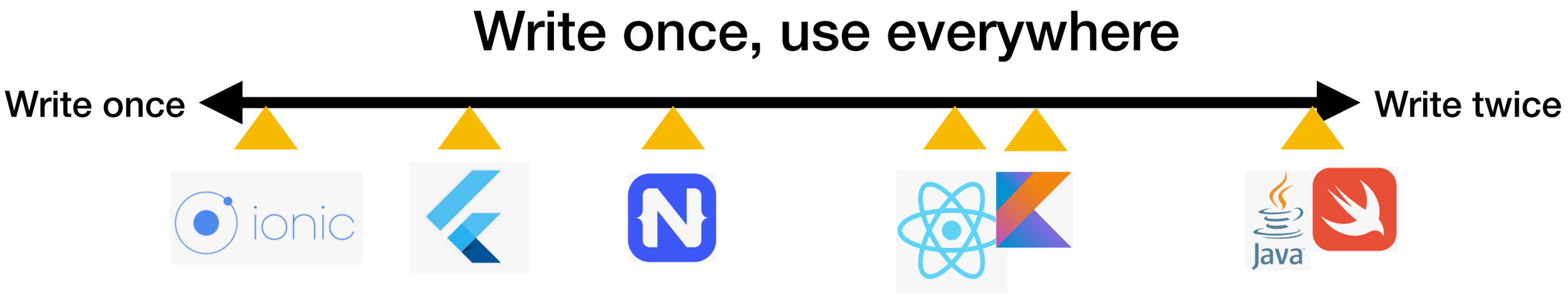
All available

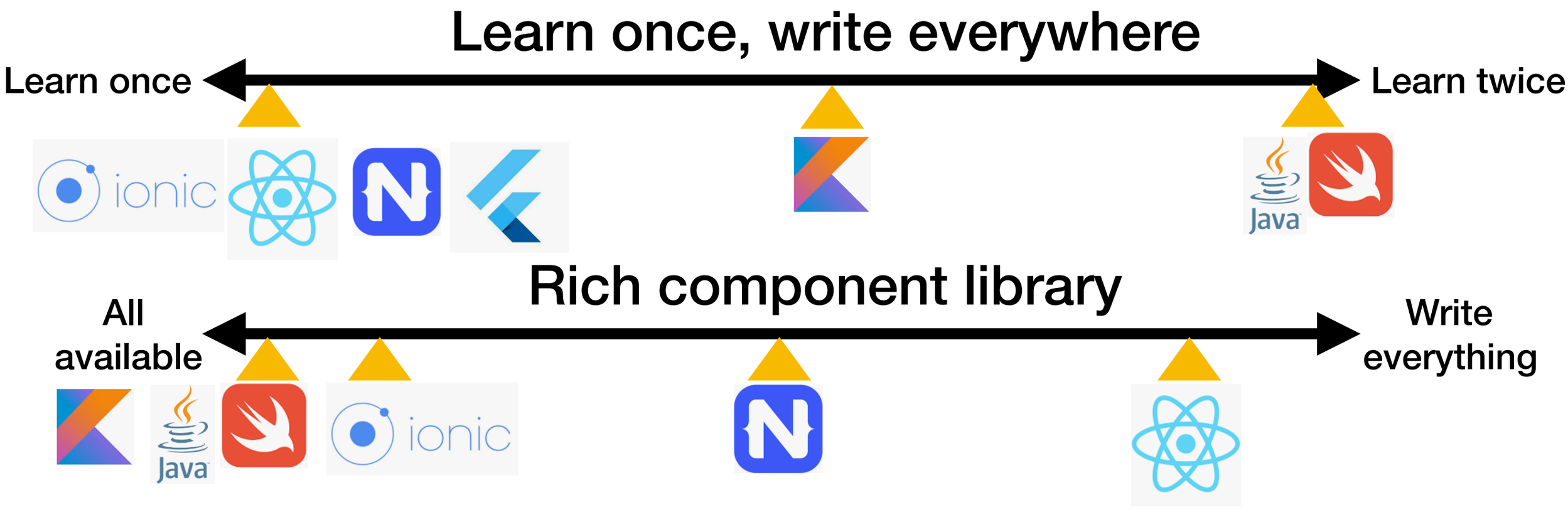
Write everything

Comparison

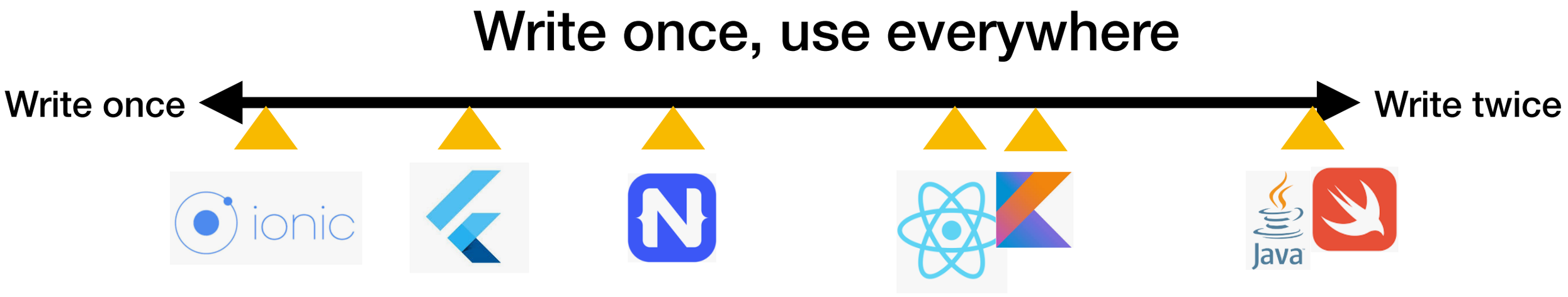

Write once, use everywhere
Write once
Write twice
ionic
Java
Learn once, write everywhere
Learn once
Learn twice
ionic
Java
Rich component library
All available
Write everything
Java
ionic

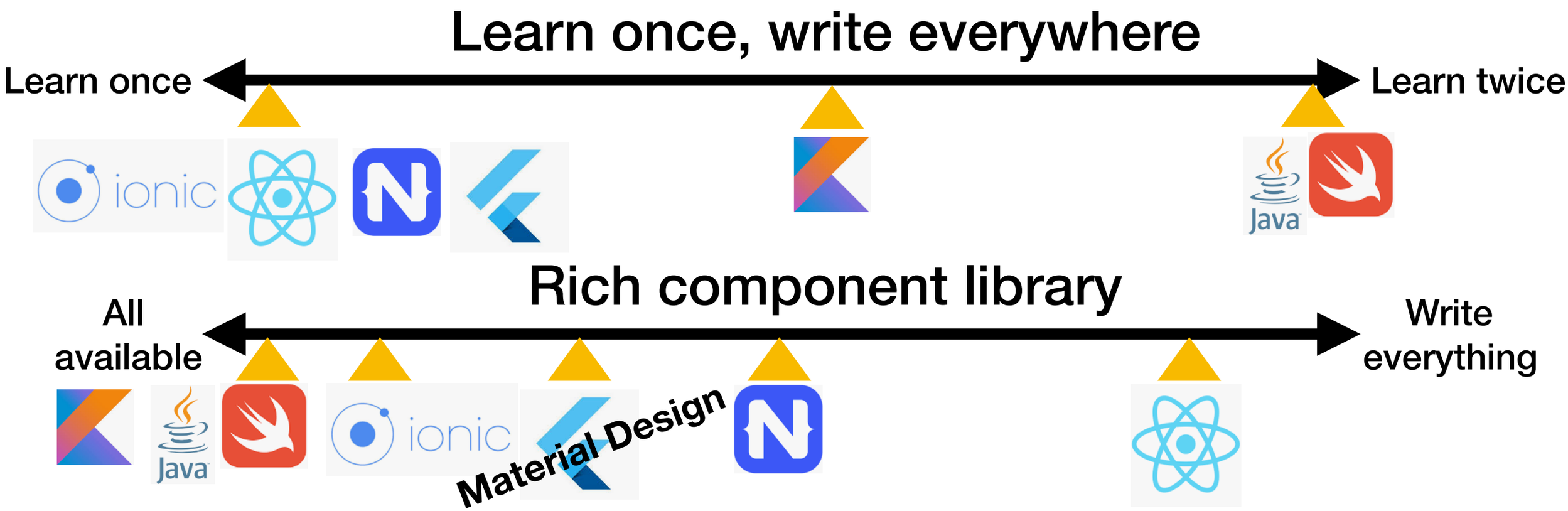
Comparison


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

Comparison

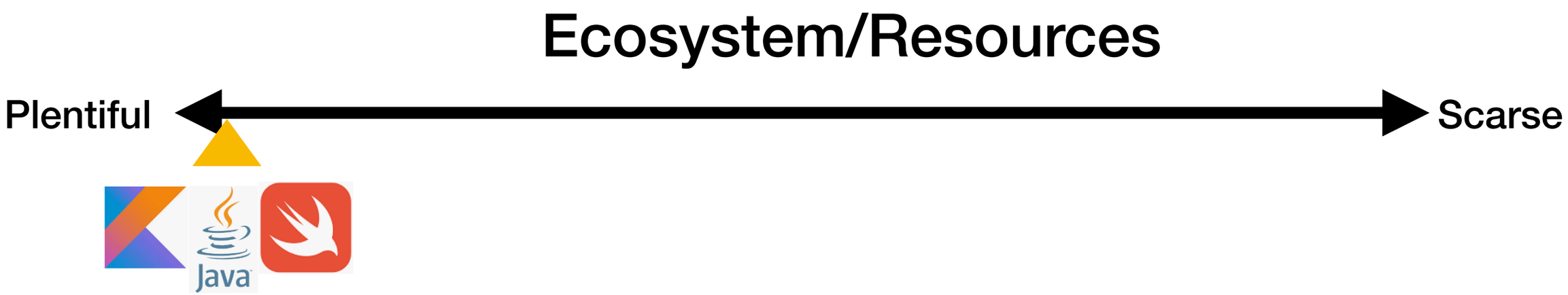

Write once, use everywhere
Write once
Write twice
ionic
Java


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

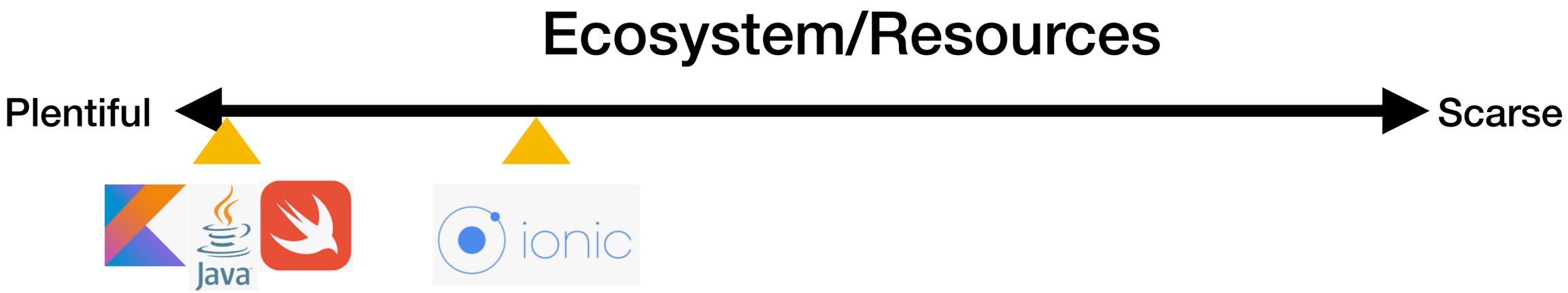
Comparison

Comparison

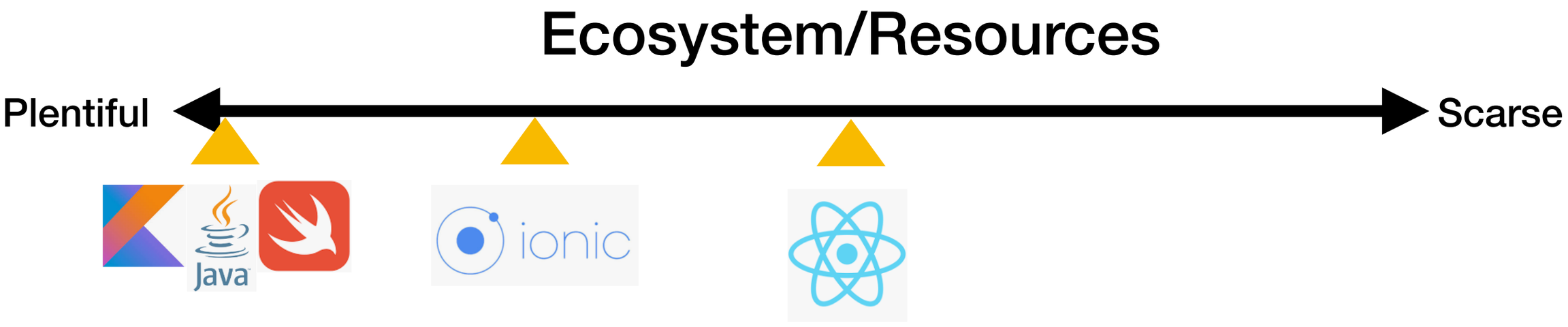
Comparison



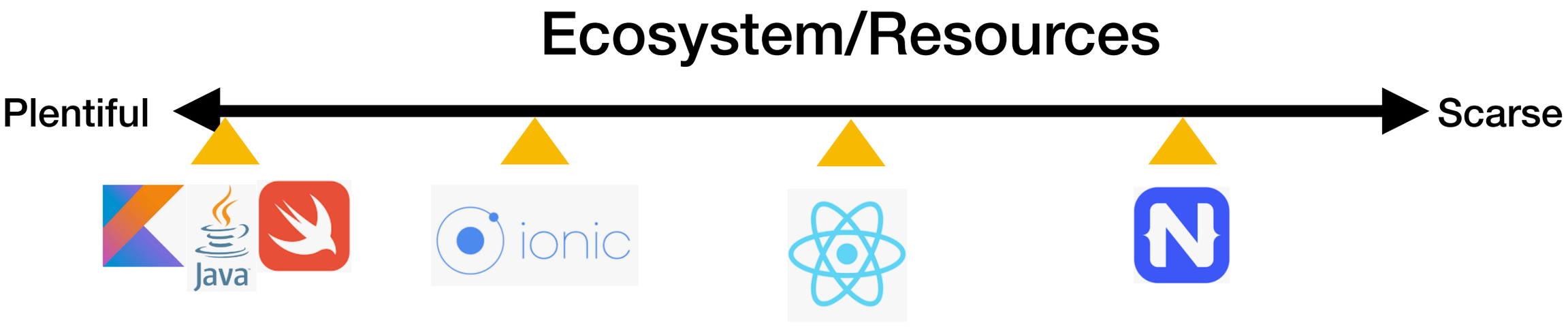
Comparison



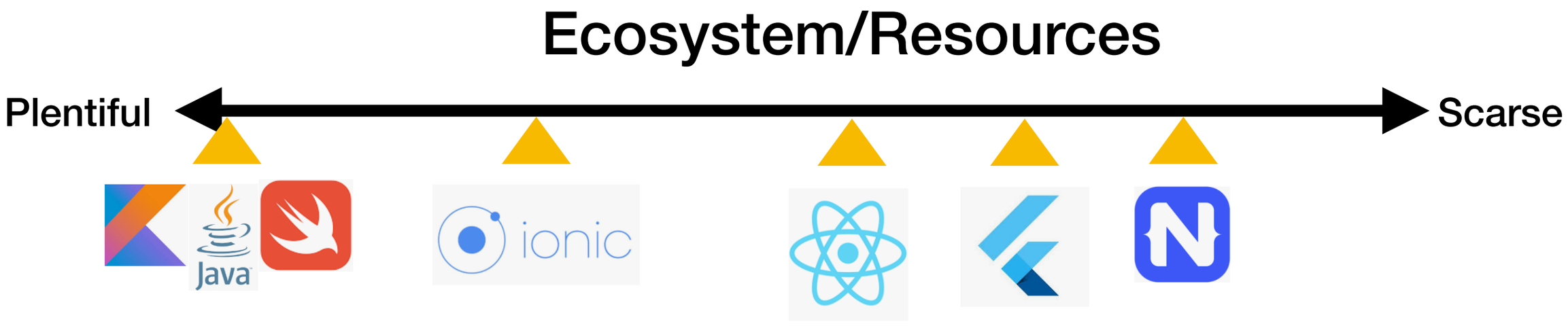
Comparison



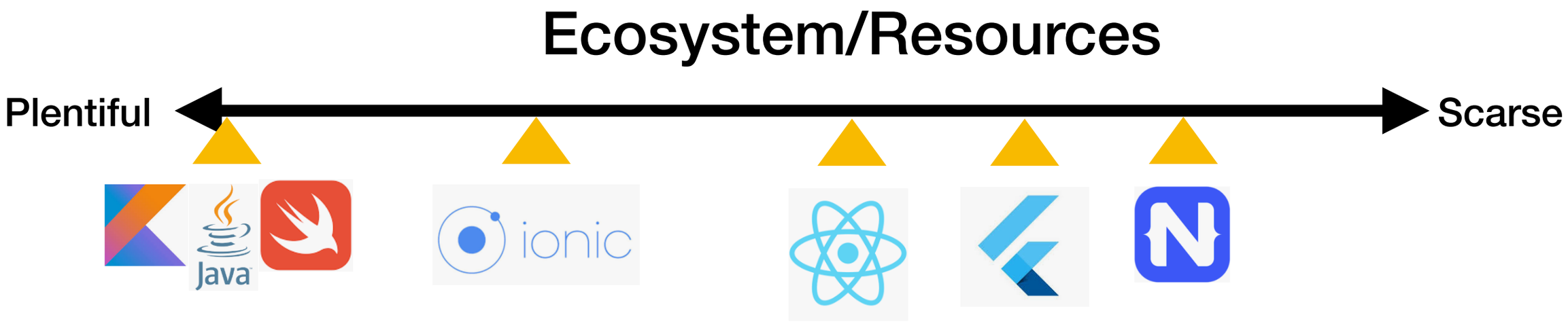
Comparison

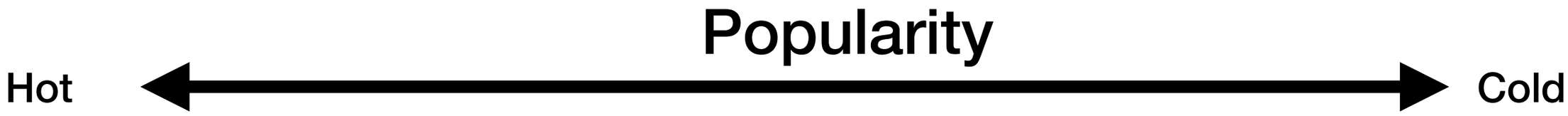


Comparison

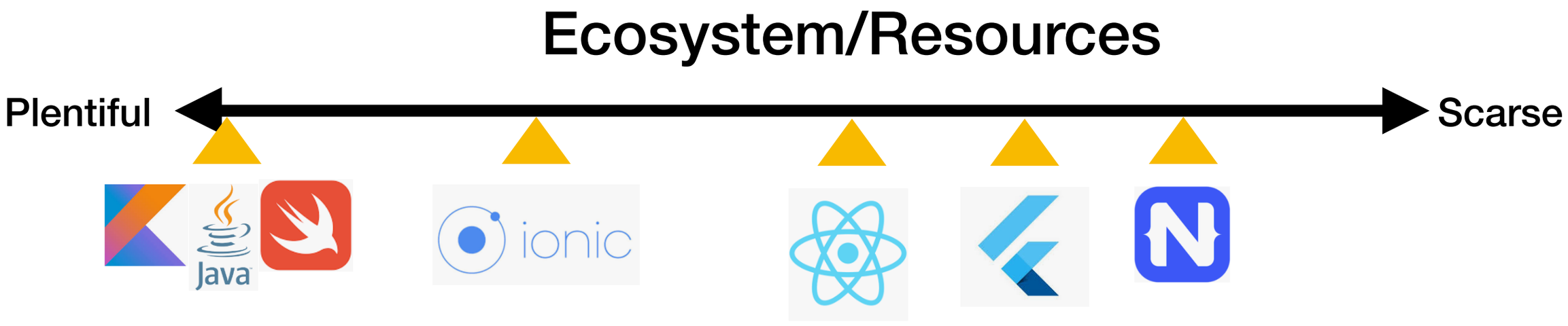


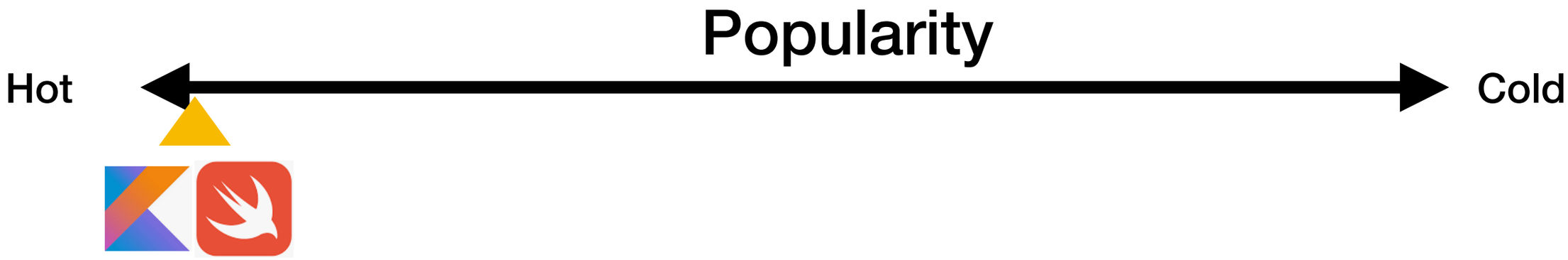
Comparison


Ecosystem/Resources
Plentiful
Scarse
Java
ionic

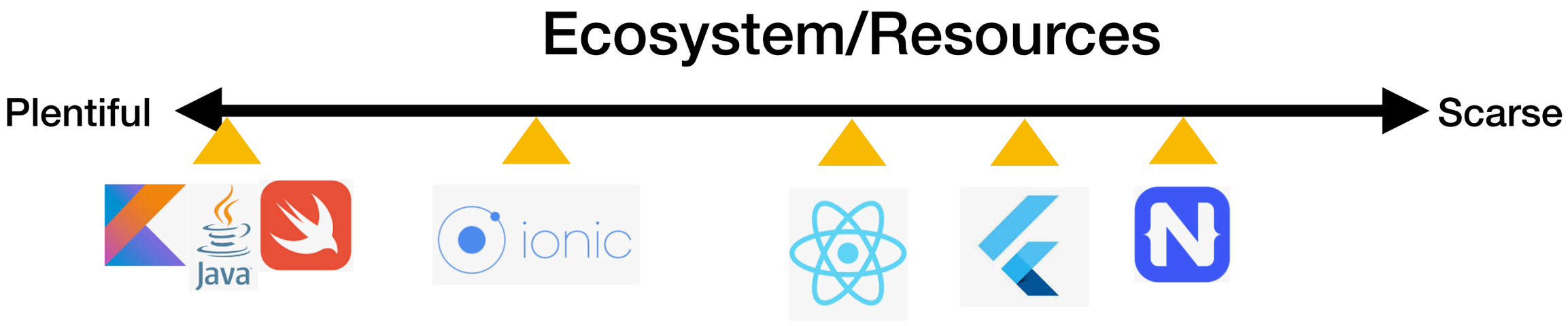

Popularity
Hot
Cold

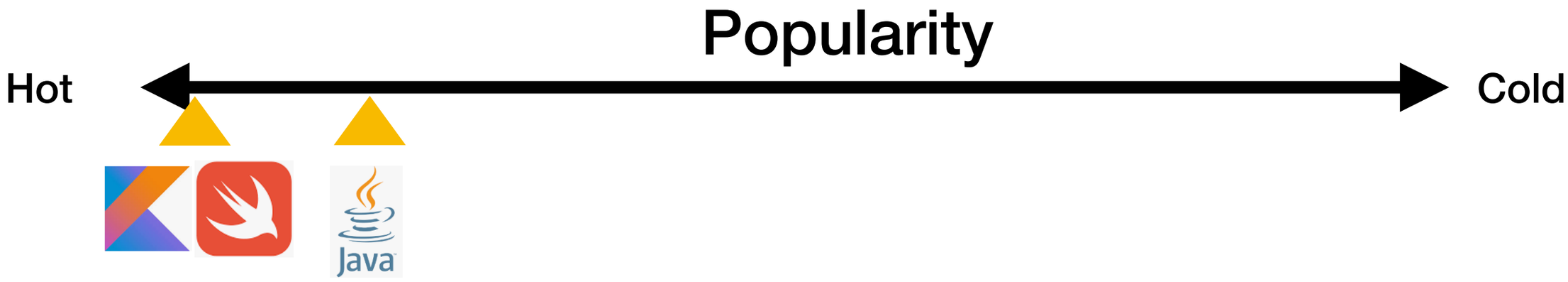
Comparison


Ecosystem/Resources
Plentiful
Scarse
Java
ionic

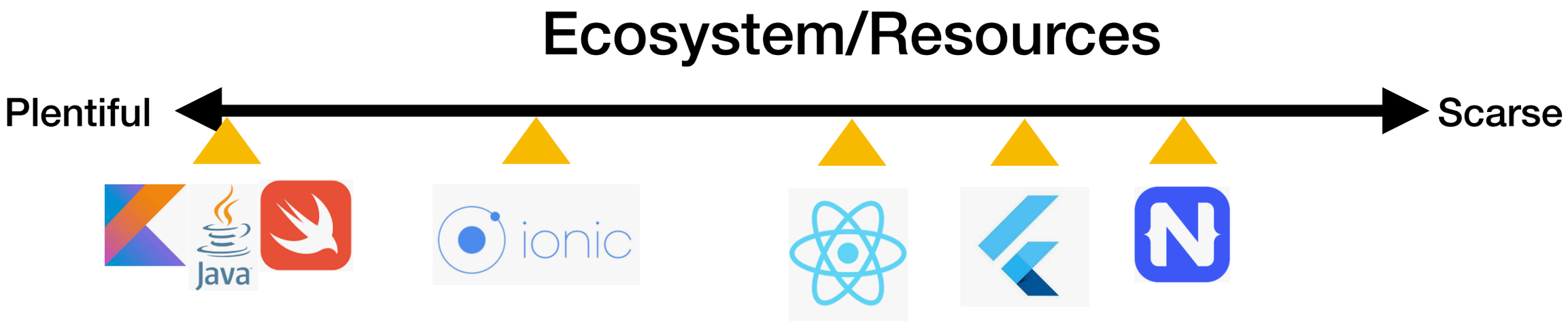

Popularity
Hot
Cold

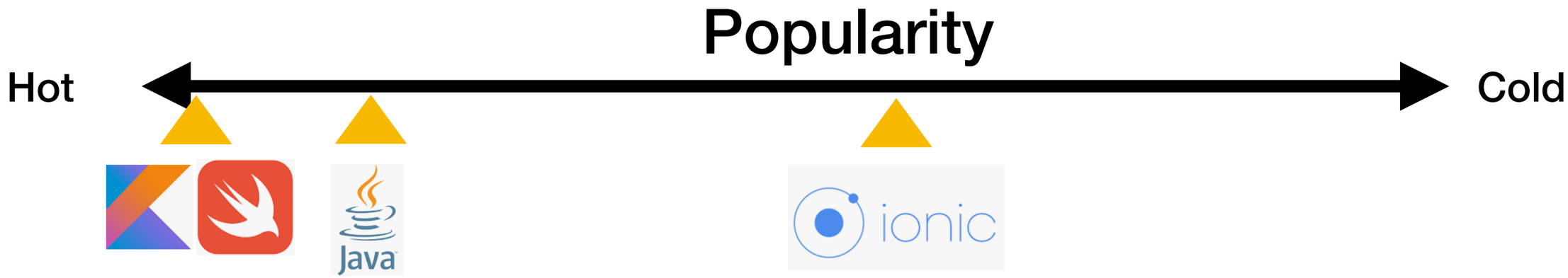
Comparison


Ecosystem/Resources
Plentiful
Scarse
Java
ionic

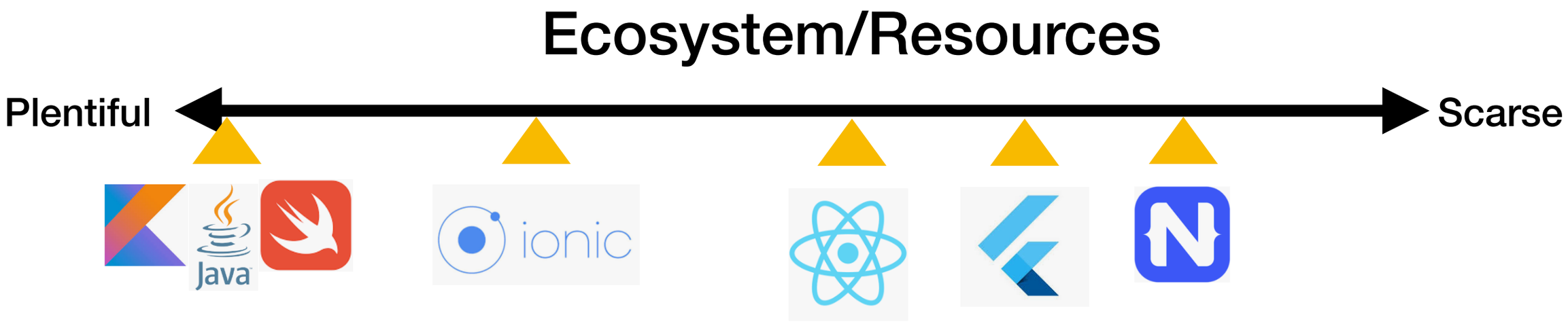

Popularity
Hot
Cold
Java

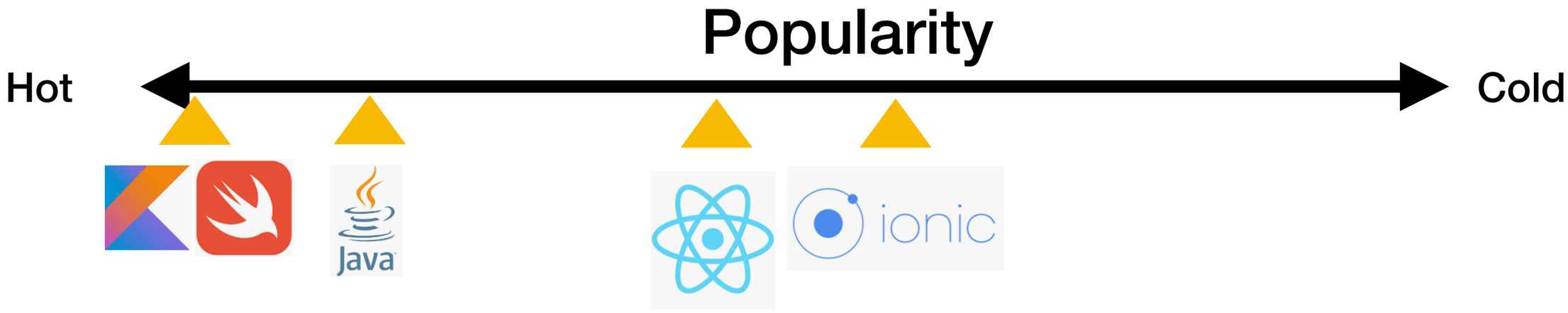
Comparison


Ecosystem/Resources
Plentiful
Scarse
Java
ionic

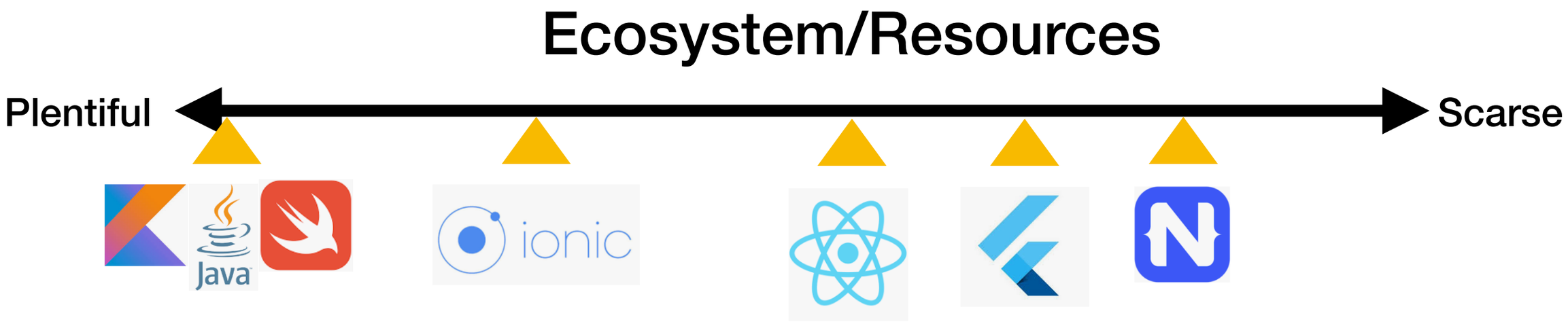

Popularity
Hot
Cold
Java
ionic

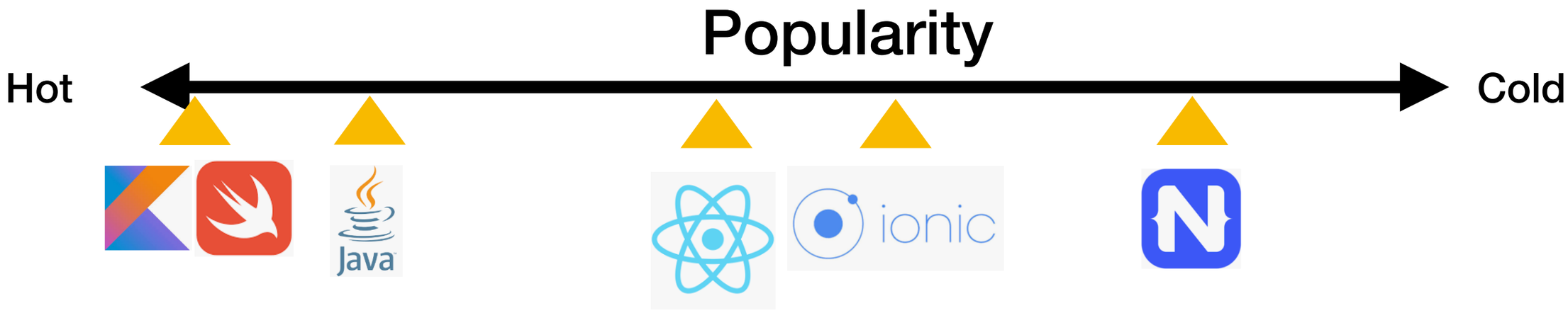
Comparison


Ecosystem/Resources
Plentiful
Scarse
Java
ionic

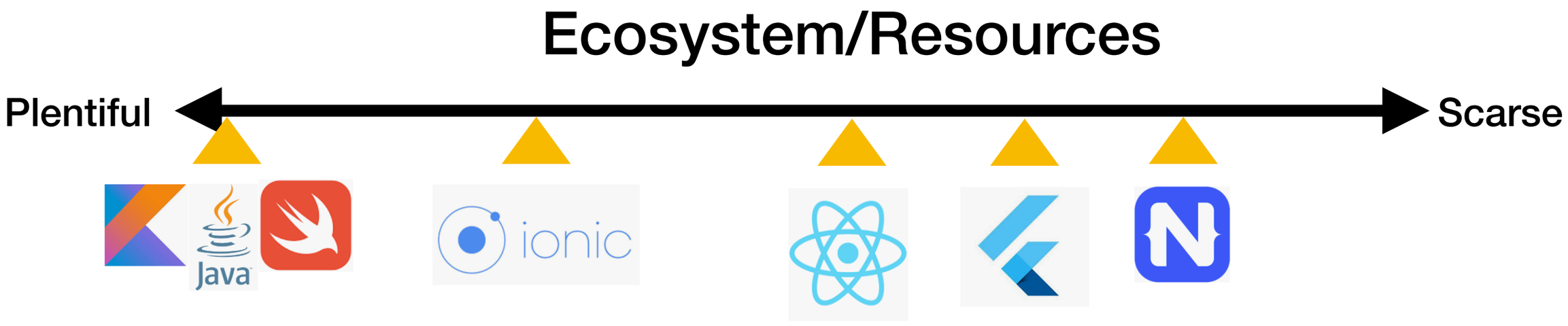

Popularity
Hot
Cold
Java
ionic

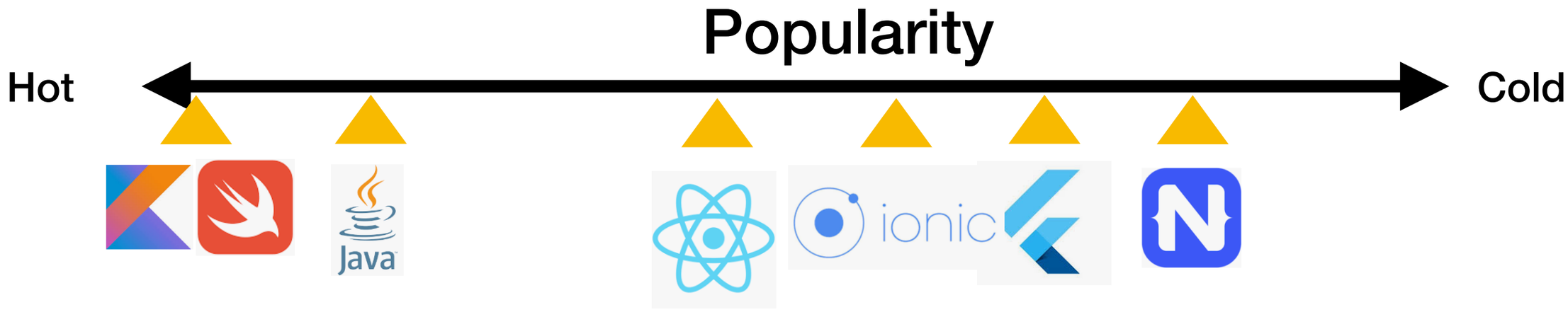
Comparison


Ecosystem/Resources
Plentiful
Scarse
Java
ionic

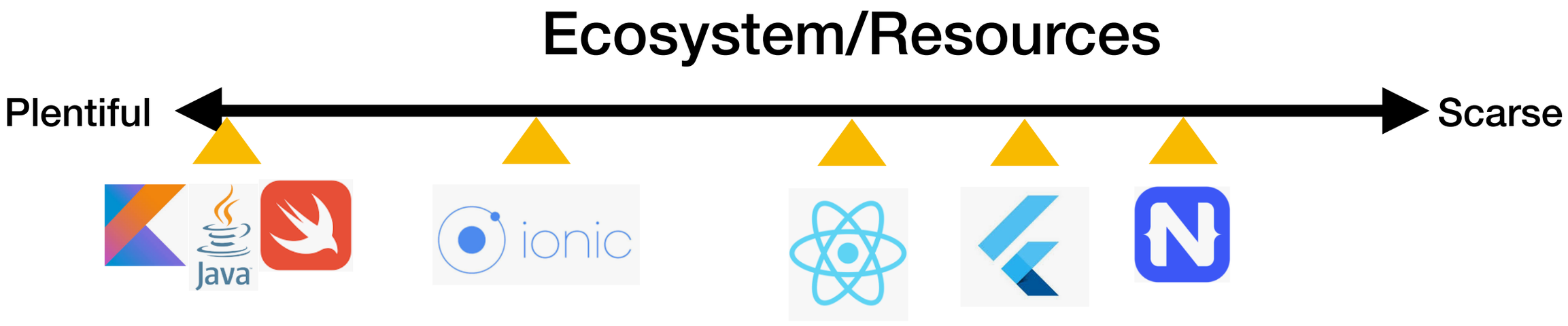

Popularity
Hot
Cold
Java
ionic

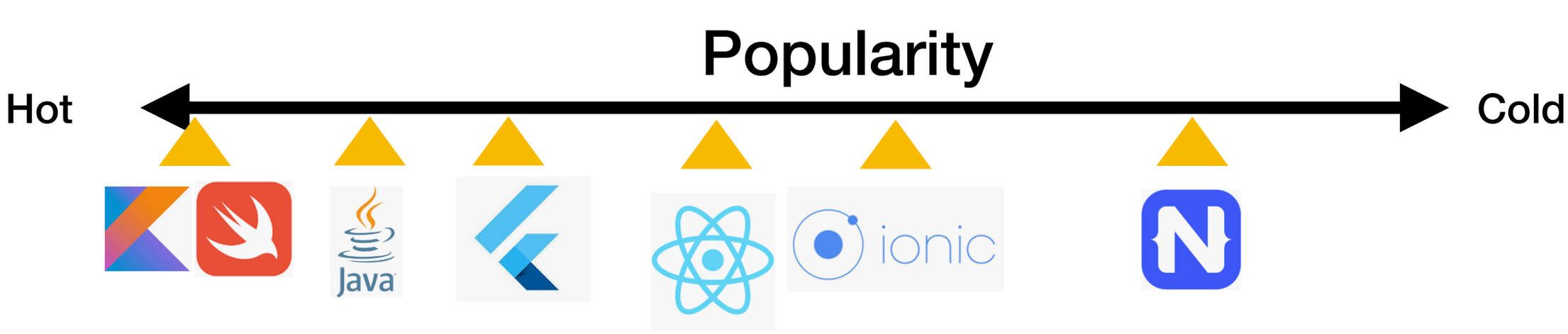
Comparison


Ecosystem/Resources
Plentiful
Scarse
Java
ionic

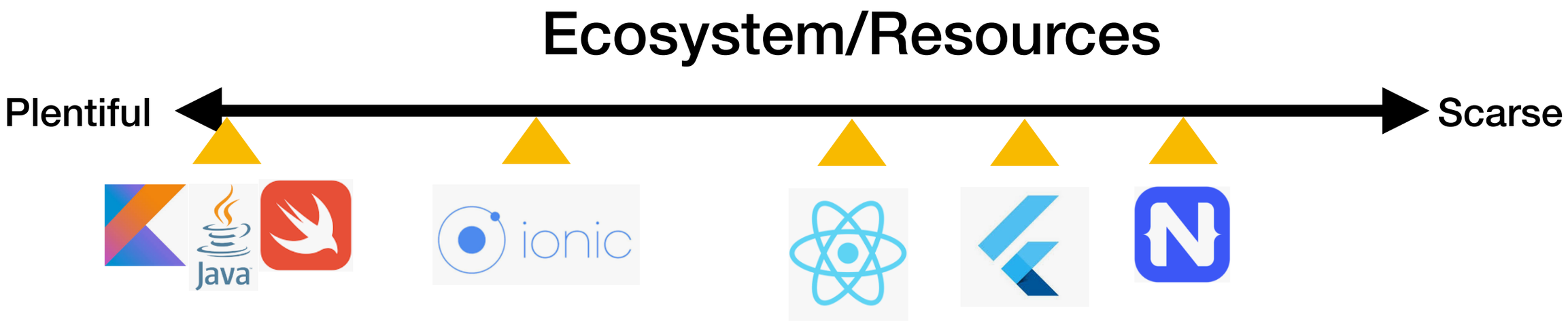

Popularity
Hot
Cold
Java
ionic

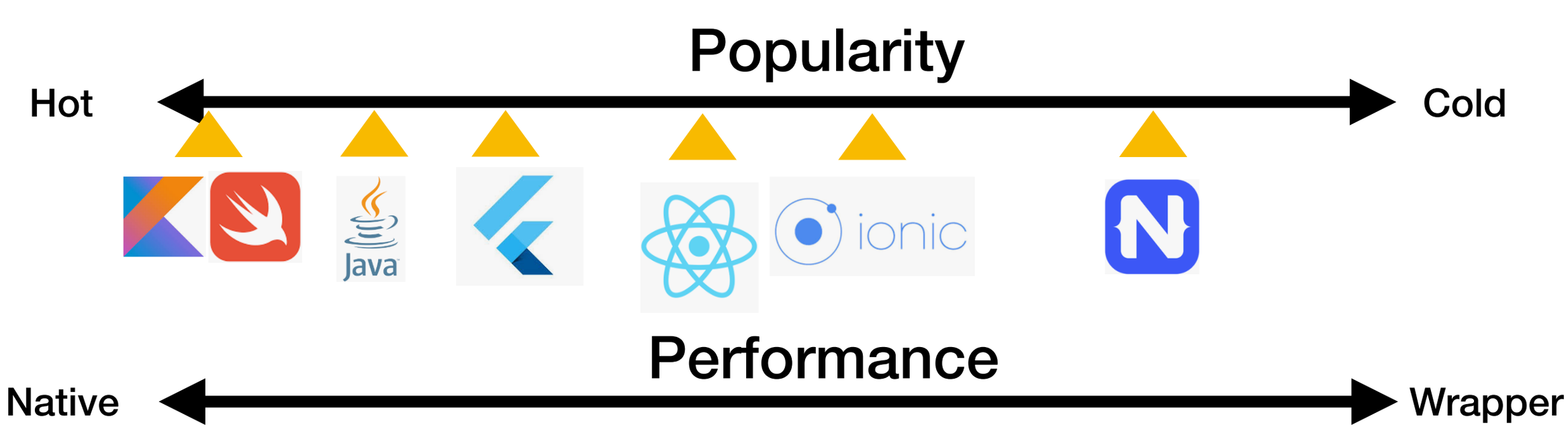
Comparison


Ecosystem/Resources
Plentiful
Scarse
Java
ionic


Popularity
Hot
Cold
Java
ionic

Comparison

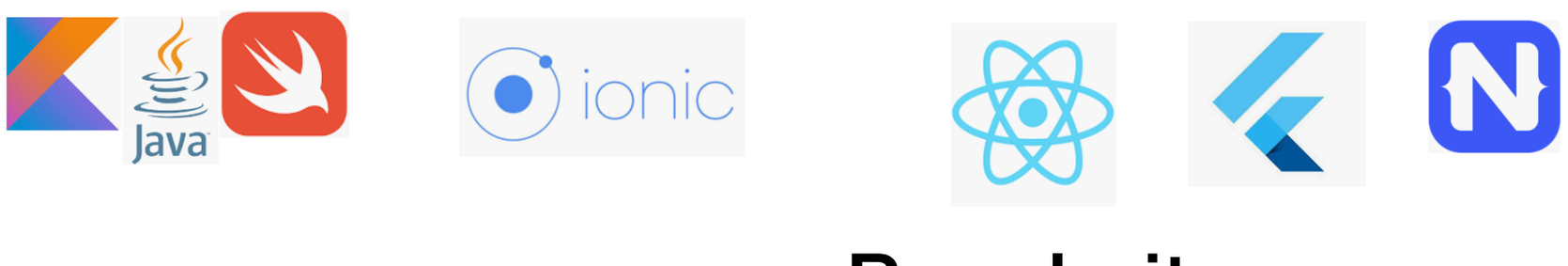

Ecosystem/Resources
Plentiful
Scarse
Java
ionic


Popularity
Hot
Cold
Java
ionic
Performance
Native
Wrapper

Comparison

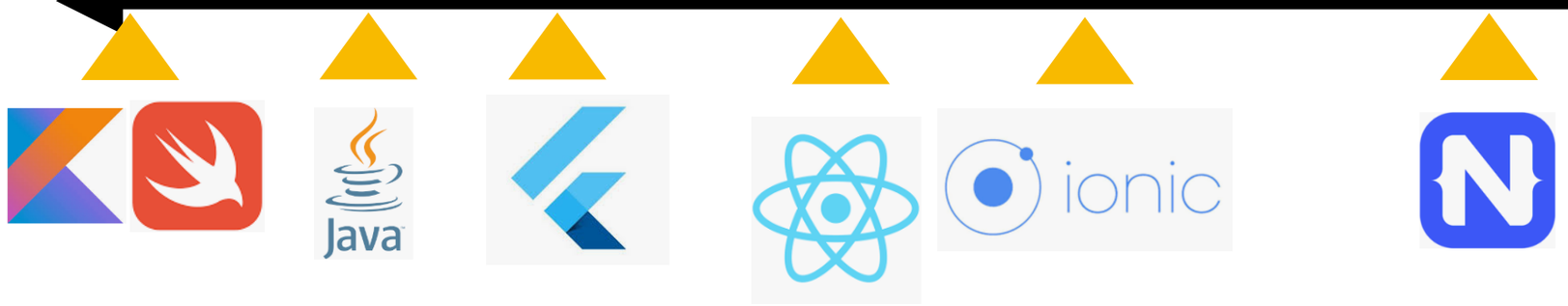
Ecosystem/Resources

Plentiful ⟵ ⟶ Scarse



Popularity

Hot ⟵ ⟶ Cold



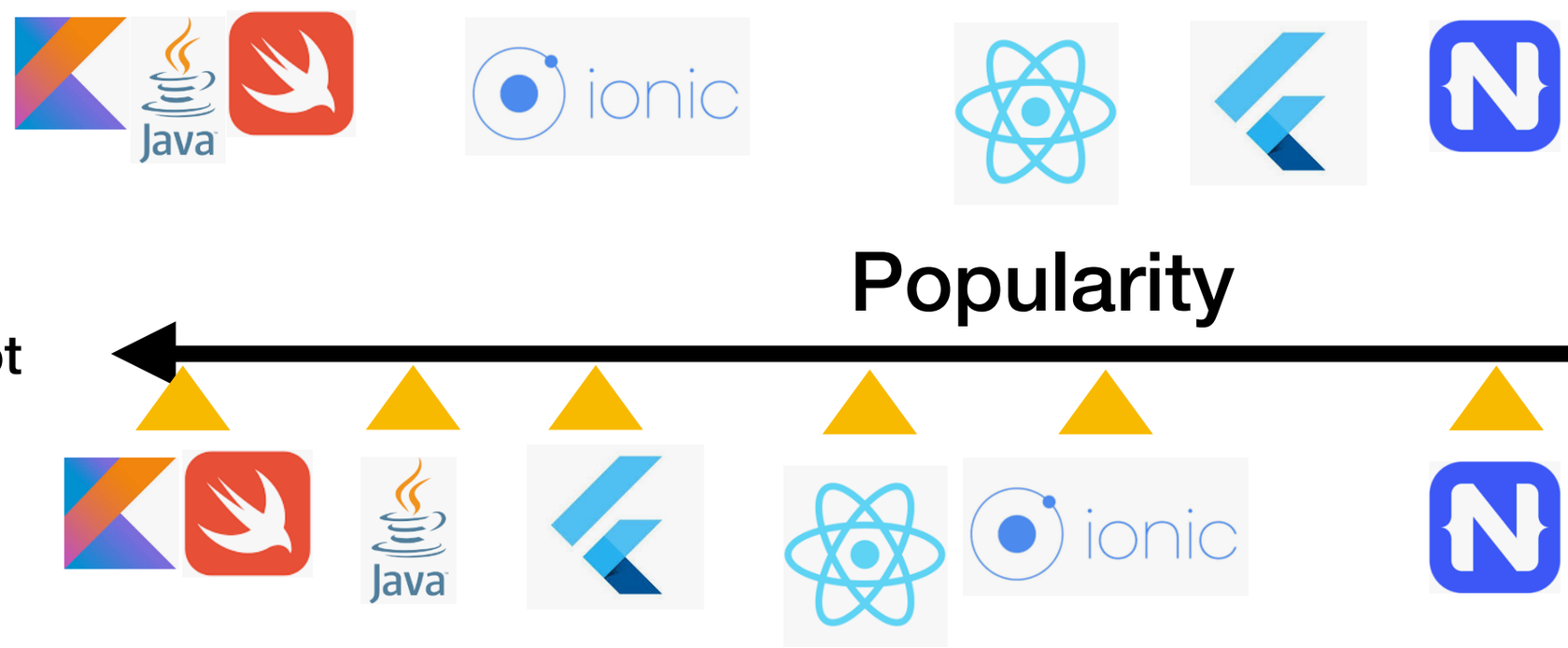
Performance

Native ⟵ ⟶ Wrapper

Comparison

Ecosystem/Resources

Plentiful ⟵ ⟶ Scarse



Popularity

Hot ⟵ ⟶ Cold

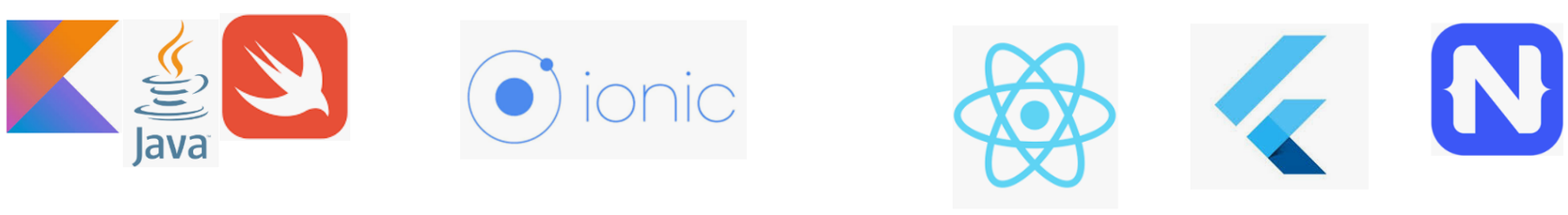
Performance

Native ⟵ ⟶ Wrapper

Comparison

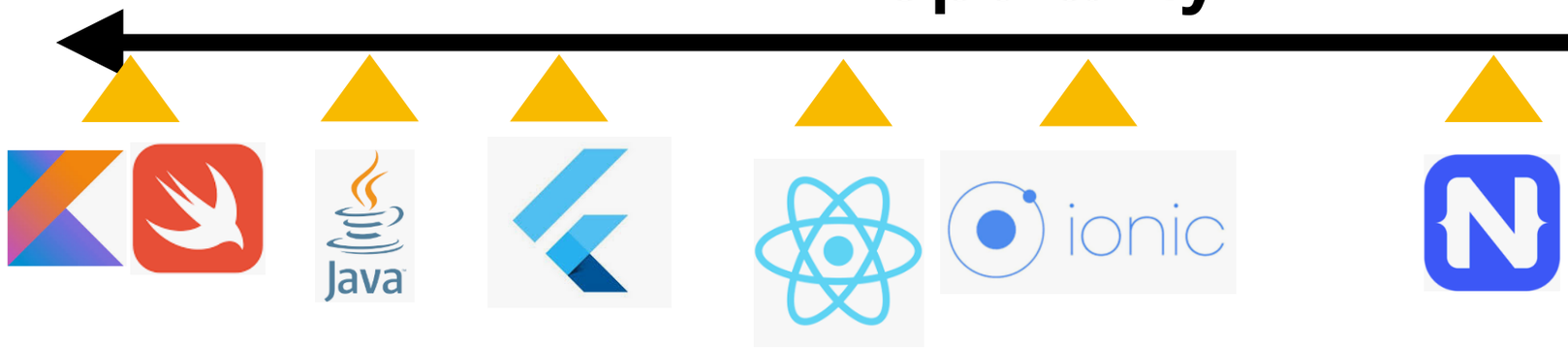
Ecosystem/Resources

Plentiful ⟵ ⟶ Scarse



Popularity

Hot ⟵ ⟶ Cold



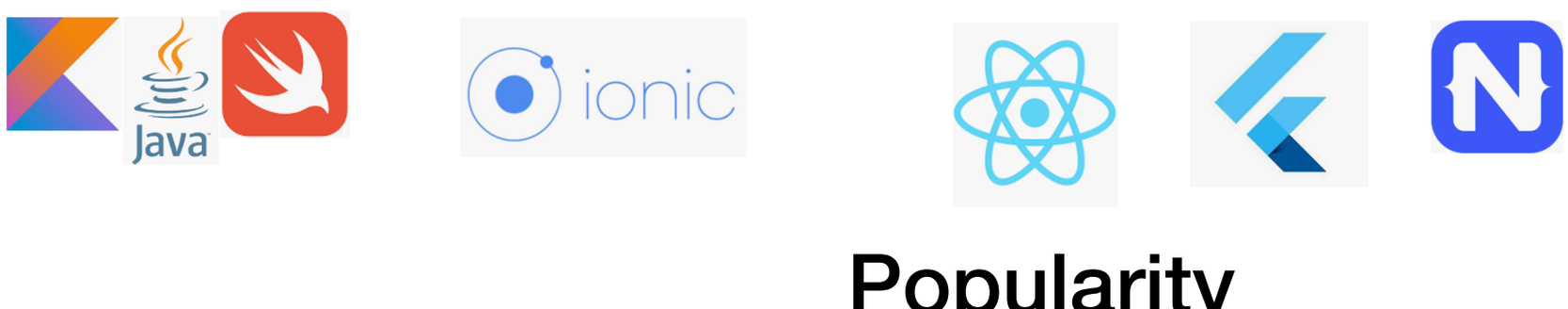
Performance

Native ⟵ ⟶ Wrapper

Comparison

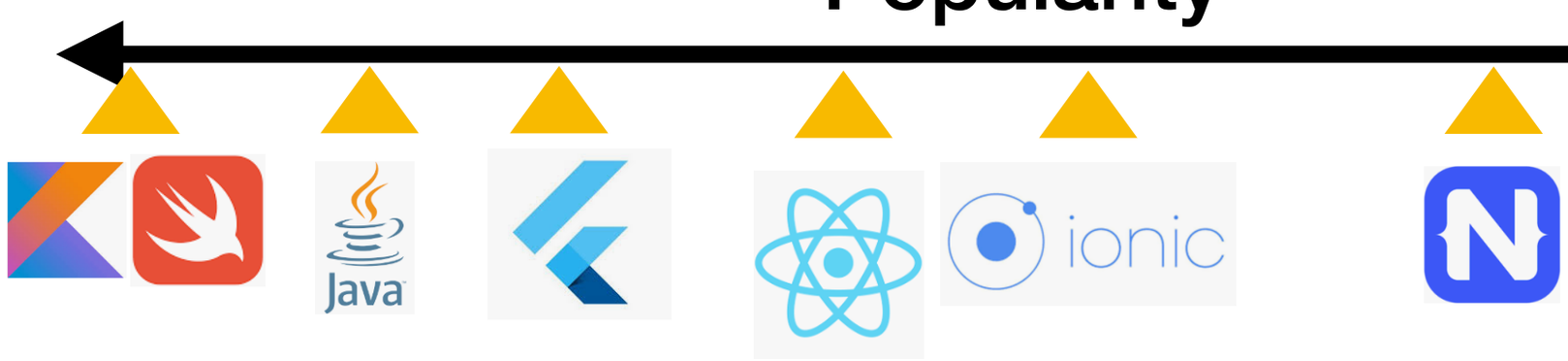
Ecosystem/Resources

Plentiful ⟵ ⟶ Scarse



Popularity

Hot ⟵ ⟶ Cold



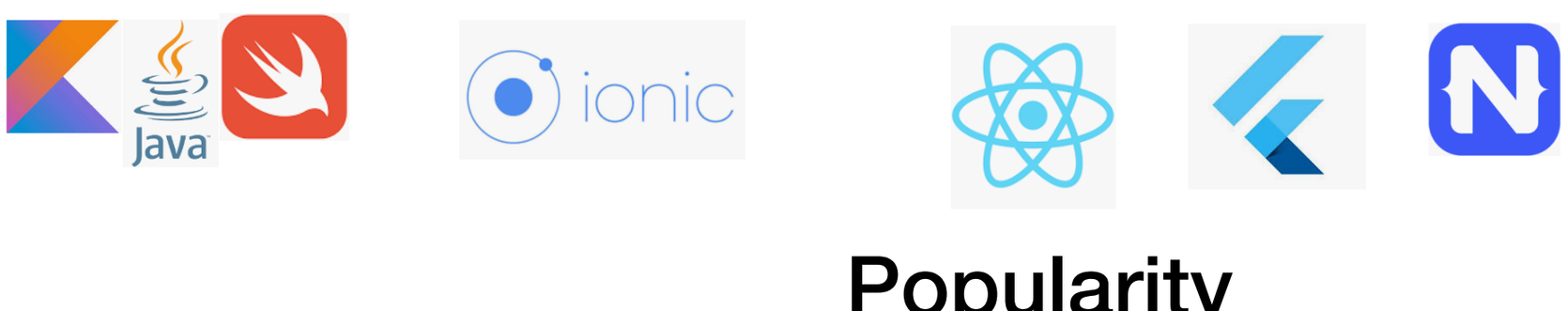
Performance

Native ⟵ ⟶ Wrapper

Comparison

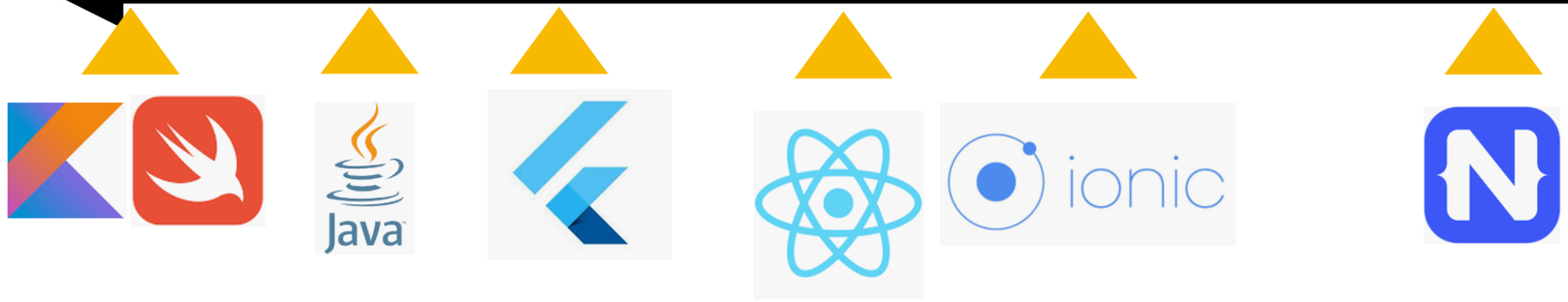
Ecosystem/Resources

Plentiful ⟵ ⟶ Scarse



Popularity

Hot ⟵ ⟶ Cold



Performance

Native ⟵ ⟶ Wrapper

Comparison

Comparison

Comparison

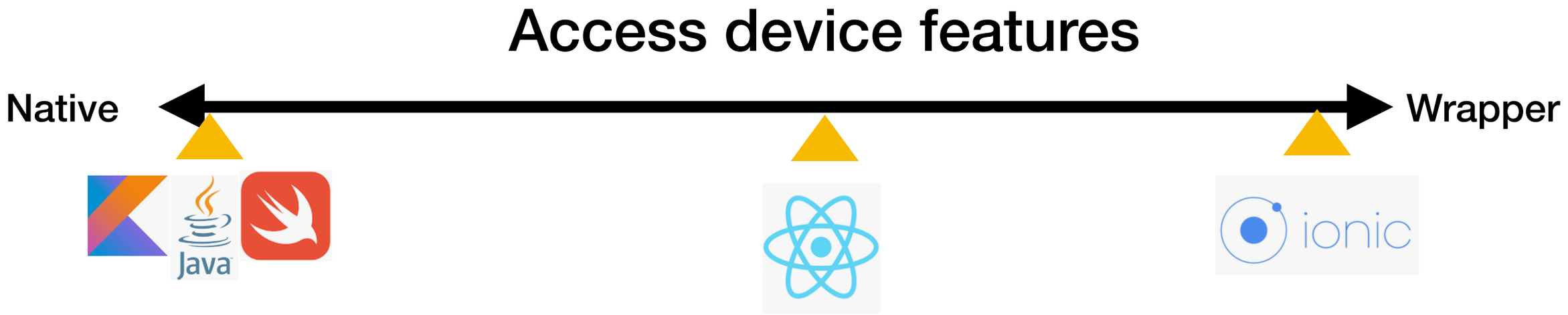
Comparison

Access device features

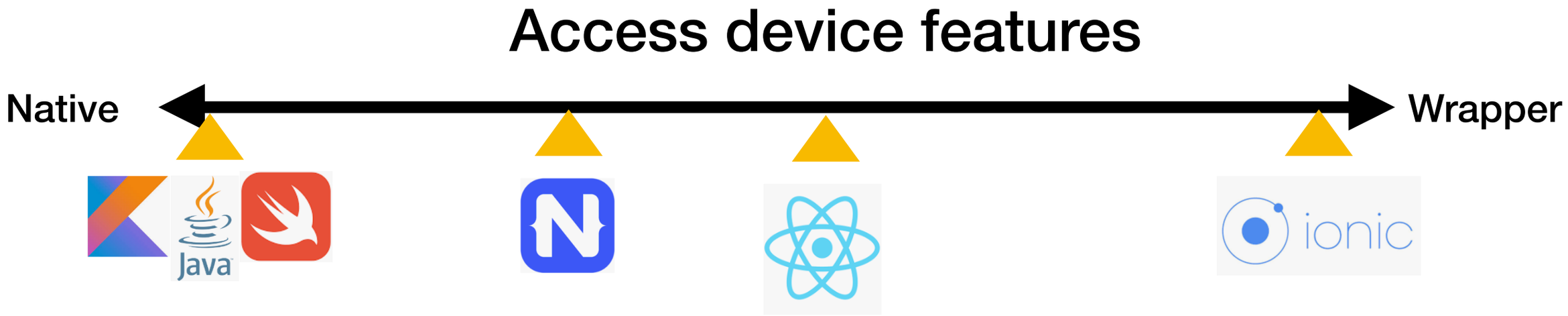
Native

Wrapper

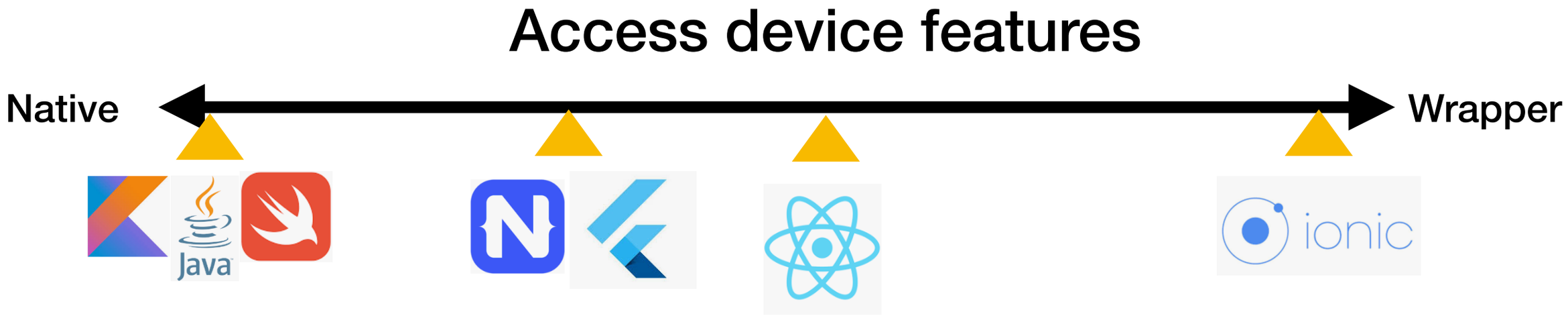
Comparison



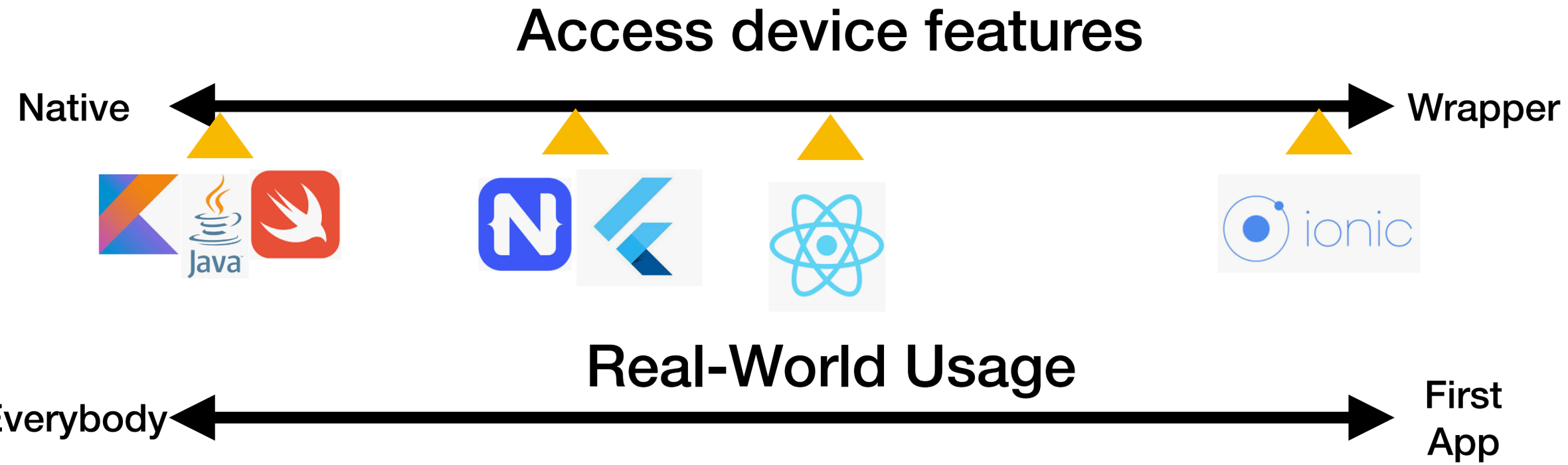
Comparison



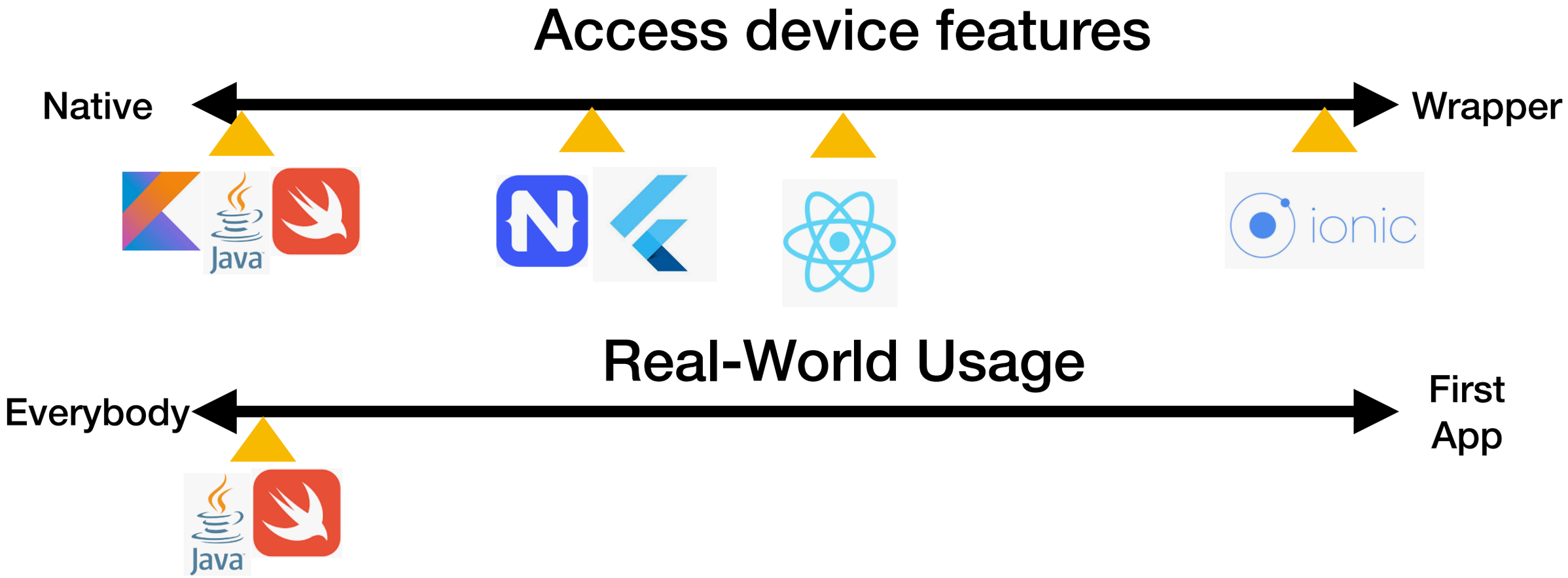
Comparison



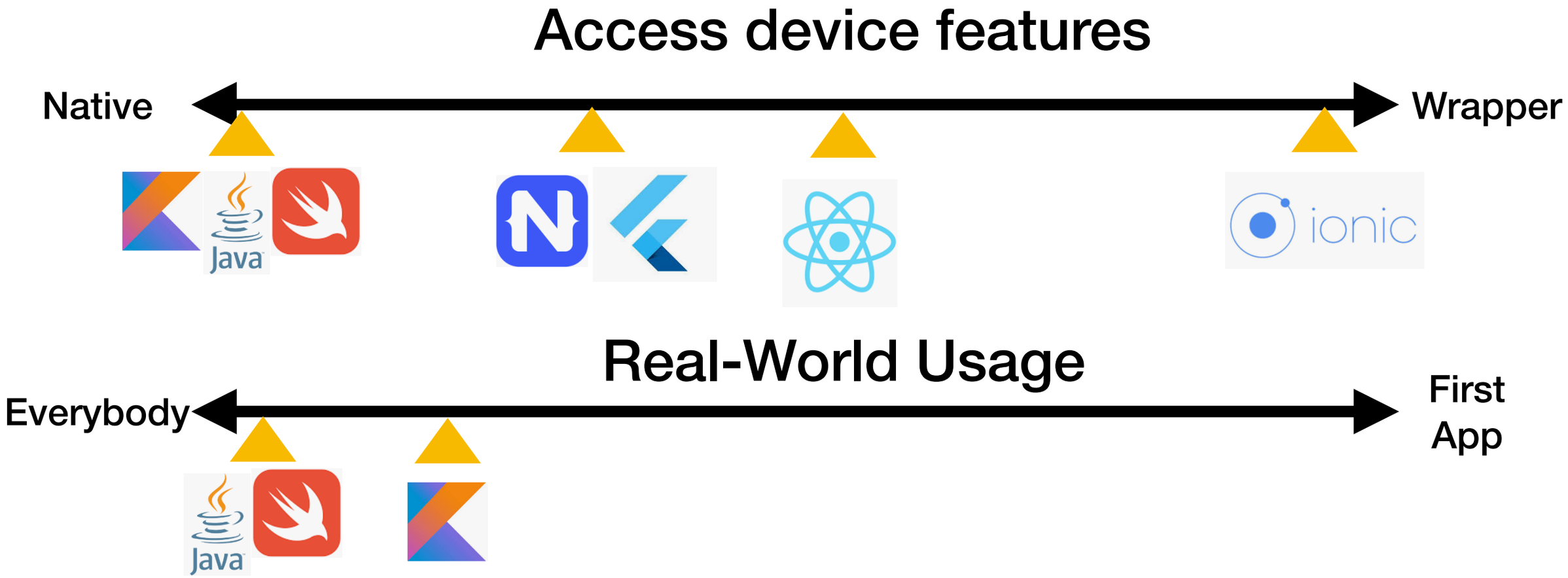
Comparison


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

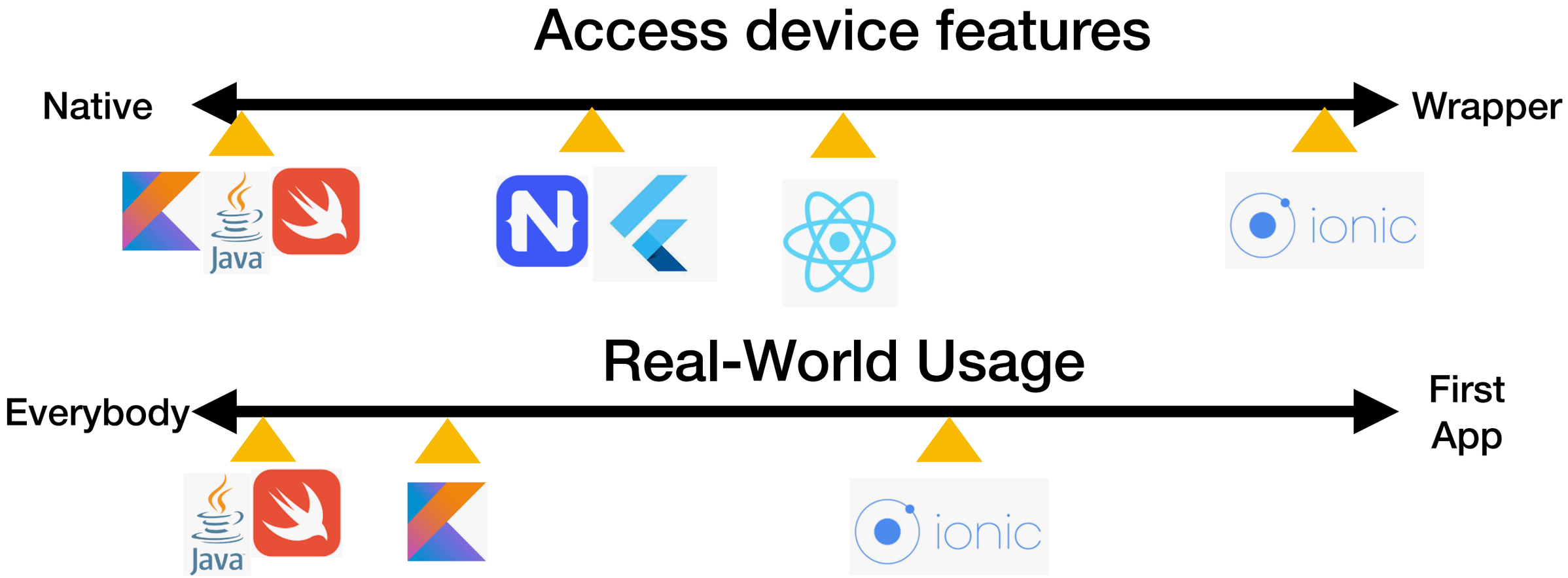
Comparison


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

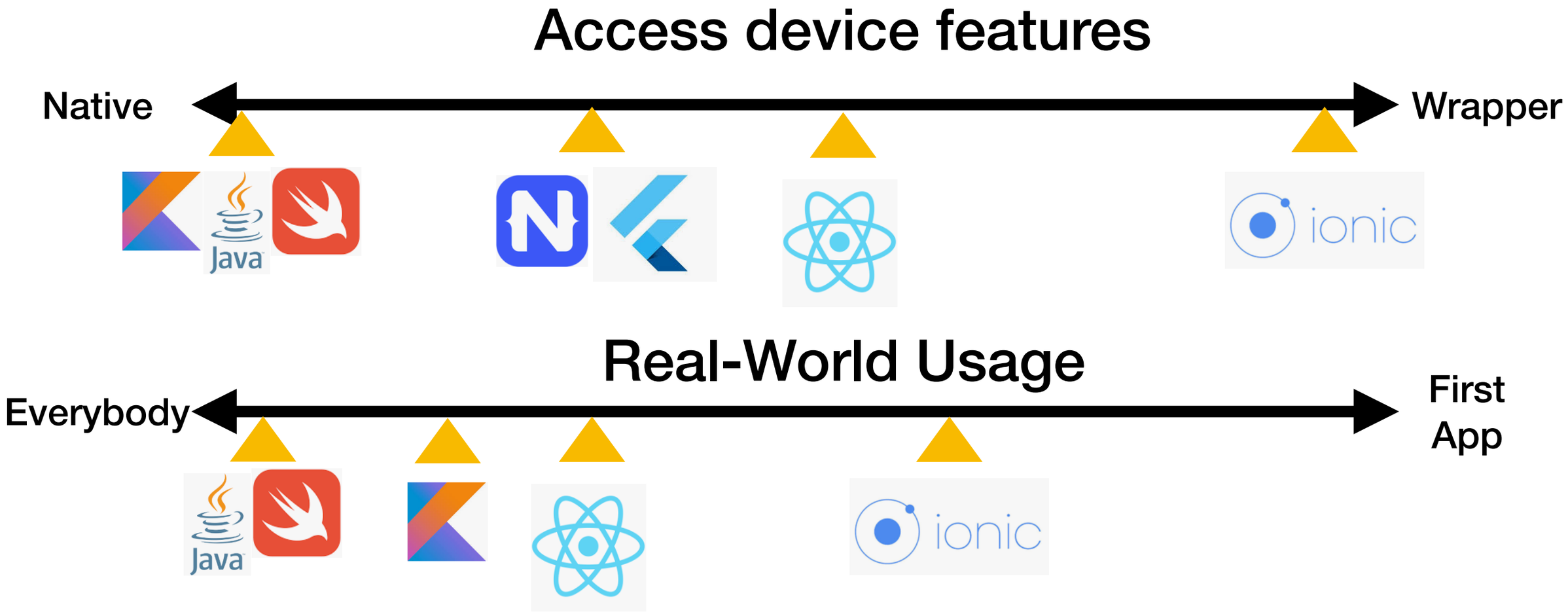
Comparison


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

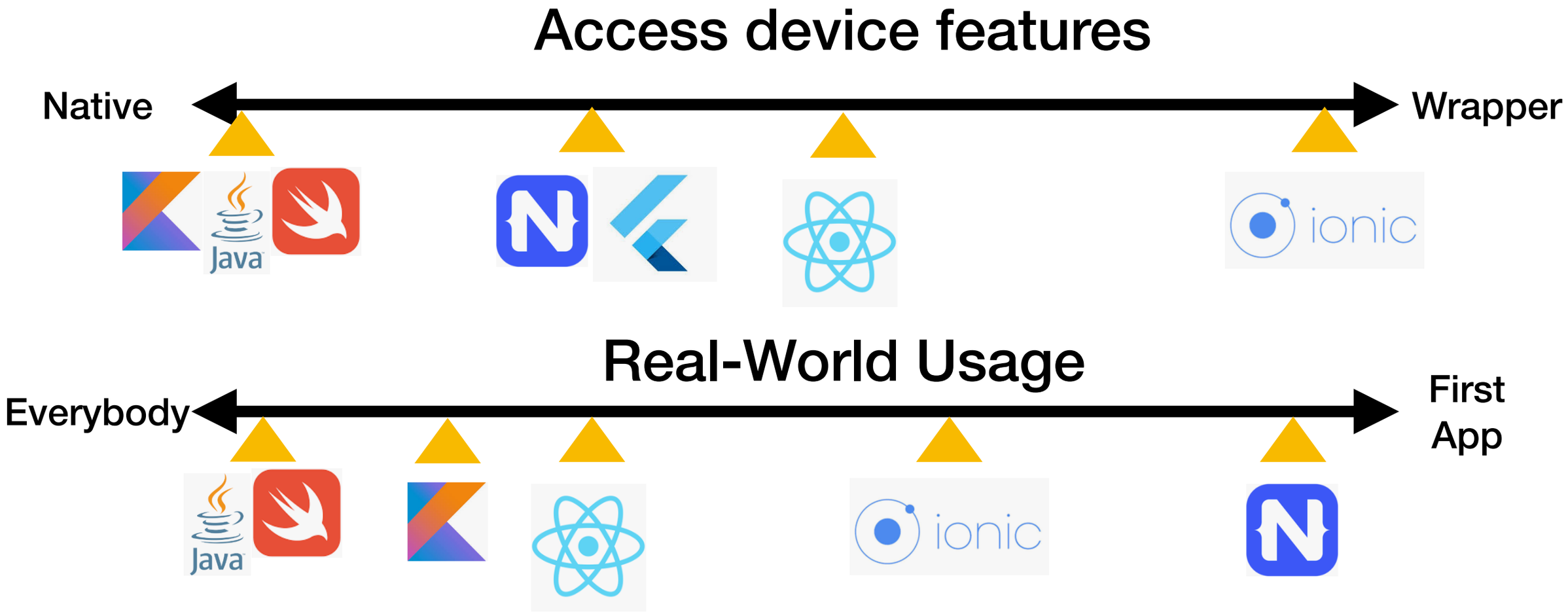
Comparison


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

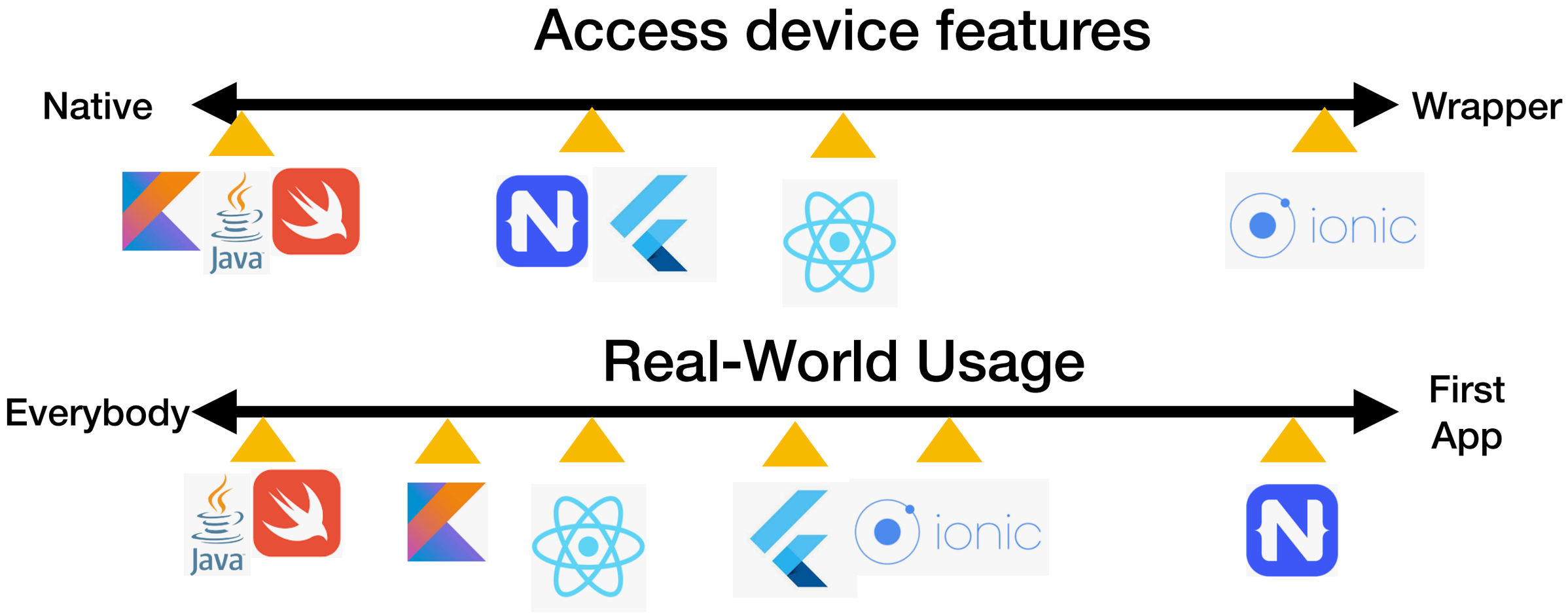
Comparison


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

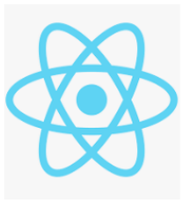
Comparison


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

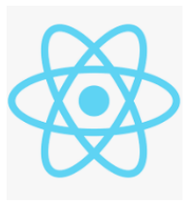
Comparison


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

Previous years

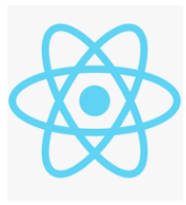


Previous years

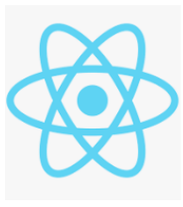


Verify the quiz

This year



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)
```

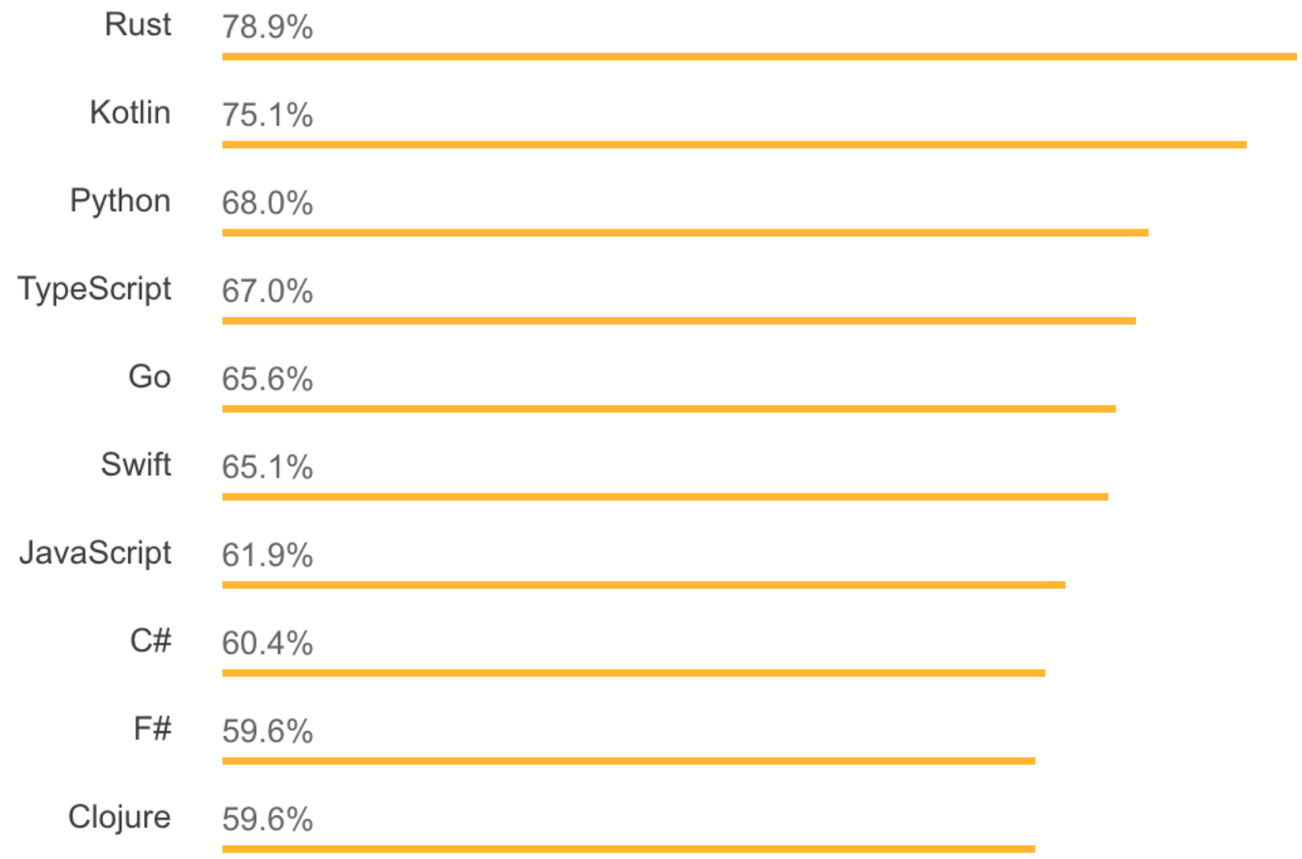
Kotlin equivalent

Why Kotlin

Most Loved, Dreaded, and Wanted

Most Loved, Dreaded, and Wanted Languages

Loved | Dreaded | Wanted



stackoverflow

Why Kotlin

Kotlin LEARN COMMUNITY TRY ONLINE

Reference | Tutorials | Books | More resources

- Getting Started
- Android
- Java Interop
- JavaScript
- Multiplatform Projects
  - iOS and Android
  - Multiplatform Library

Multiplatform Project: iOS and Android

Edit Page

Last Updated 4 October 2018

Sharing Kotlin code between iOS and Android

In this tutorial we will create an iOS and Android application, by making use of Kotlin's code sharing features. For Android we'll be using Kotlin/JVM, while for iOS it will be Kotlin/Native.

We'll learn how to:

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, many, many packages

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, many, many packages

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.