slidenumbers: true

Zoom in modern web/mobile development

[.footer: ##### @JamalBelilet (🧿 🐙 📸 👍)]

Let's be on the same page

Web and mobile applications are built with at least two separate components*

1. **backend** side (server business logic, databases...)
2. and **frontend** side (ui, cache, client business logic...)

Communicating through **HTTP** requests.

Backend - server

[.code-highlight: 4-11]

```
const express = require('express')
const server = express()
const port = 3000
server.get(
    '/articles',
    (req, res) => {
        // verify authenticity
        // select **articles** from database
        res.json(articles)
    }
)
server.listen(port,() => console.log(`Running on port ${port}!`))
```

Front - client

```
const response = await fetch(`server-address:3000/articles`)
const articles = await response.json()

// transform the list of articles into
// a list of UI elements to show
```

What's the trend?

- Front: **REACT**, GATSBY, VUE
- MOBILE: **FLUTTER**, SWIFT/**SWIFTUI**, ANDROID/**COMPOSE**
- BACKEND: ?

History

1. **React**: 2013/**2015**
2. **Flutter**: 2017/**December 2018**
3. **Jetpack Compose** (Kotlin): **2019**
4. **SwiftUI** (Swift): **2019**

The trend

REACT isn't just taking the web development by storm, it's **changing the way we do code** using almost all the major front solutions (**iOS Swift, Android Kotlin, Flutter, JS**)

Everything went Declarative

Here's **React in a nutshell**

UI = f(state)

Everything went Declarative

Actually it is **React, Flutter, Jetpack Compose and SwiftUI in a nutshell**

UI = f(state)

F

f is simply **the answer to what the app should look like?**

It **renders** a predefined **structure of UI elements**

Composed the same way we do when **playing lego-games**.

F (React)

[.code-highlight: 1-7]

```
const App = () => (
  <Layout>
    <Header title={"IWD"} />
    <Articles limit={10} />
    <Footer />
  </Layout>
);

const Header = ({ title }) => (
  <header>
    <BigLogo />
    <h1>{title}</h1>
  </header>
);
```

State

It is the **set of variables** needed to fulfill the placeholders.

[.code-highlight: 2, 6-9]

```
const Ranking = ({title}) => (
  const [rank, setRank] = useState(0)

  <Col>
    <h3>{title}</h3>
    <Button onClick={() => setRank(rank+1)}>
      <Icon icon={'clap'}/>
      <p>{rank}</p>
    </Button>
  </Col>
)
```

UI

UI is **the result of re-running** the **f** function **on the state** in **response** to any routine that **changes the state**.

The end result is always **a tree** of the composed elements fulfilled with the provided state.

right fit

Code

Let's see some examples using Flutter, Jetpack Compose, and SwiftUI

Flutter

```
class Preview extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        Column(
          children: [
            Text("Title")
            Text("subtitle")
          ]
        ),
        Icon(Icons.star)
      ],
    );
  }
}
```

Jetpack Compose (Kotlin)(Android)

```
@Composable
fun NewsStory() {
  val image = +imageResource(R.drawable.header)
  Column(
    modifier = Spacing(16.dp)
    ) {
    Clip(shape = RoundedCornerShape(8.dp)) {
      DrawImage(image)
    }
    HeightSpacer(16.dp)
    Text("A day in Shark Fin Cove")
    Text("Davenport, California")
    Text("December 2018")
  }
}
```

SwiftUI (Swift)(iOS)

```
struct LandmarksScreen: View {
  var landmarks: [Landmark]

  var body: some View {
    List(landmarks) { landmark in
      HStack {
        Image(landmark.thumbnail)
        Text(landmark.name)
        Spacer()

        if landmark.isFavorite {
          Image(systemName: "star.fill")
              .foregroundColor(.yellow)
        }
      }
    }
```

```
  }
}
```

What's happening on the server side?

fit inline

GraphQL

A. Describe your data (declarative!)

```
type Project {
  name: String,
  tagline: String,
  contributors: [User]
}
```

B. Ask for what you want

C. Get predictable results

Learn any*

jaml.tech/learn

Thank you!

###
Slides: jaml.tech/iwd
###
@JamalBelilet (🐳 🐙 📸 👍)