

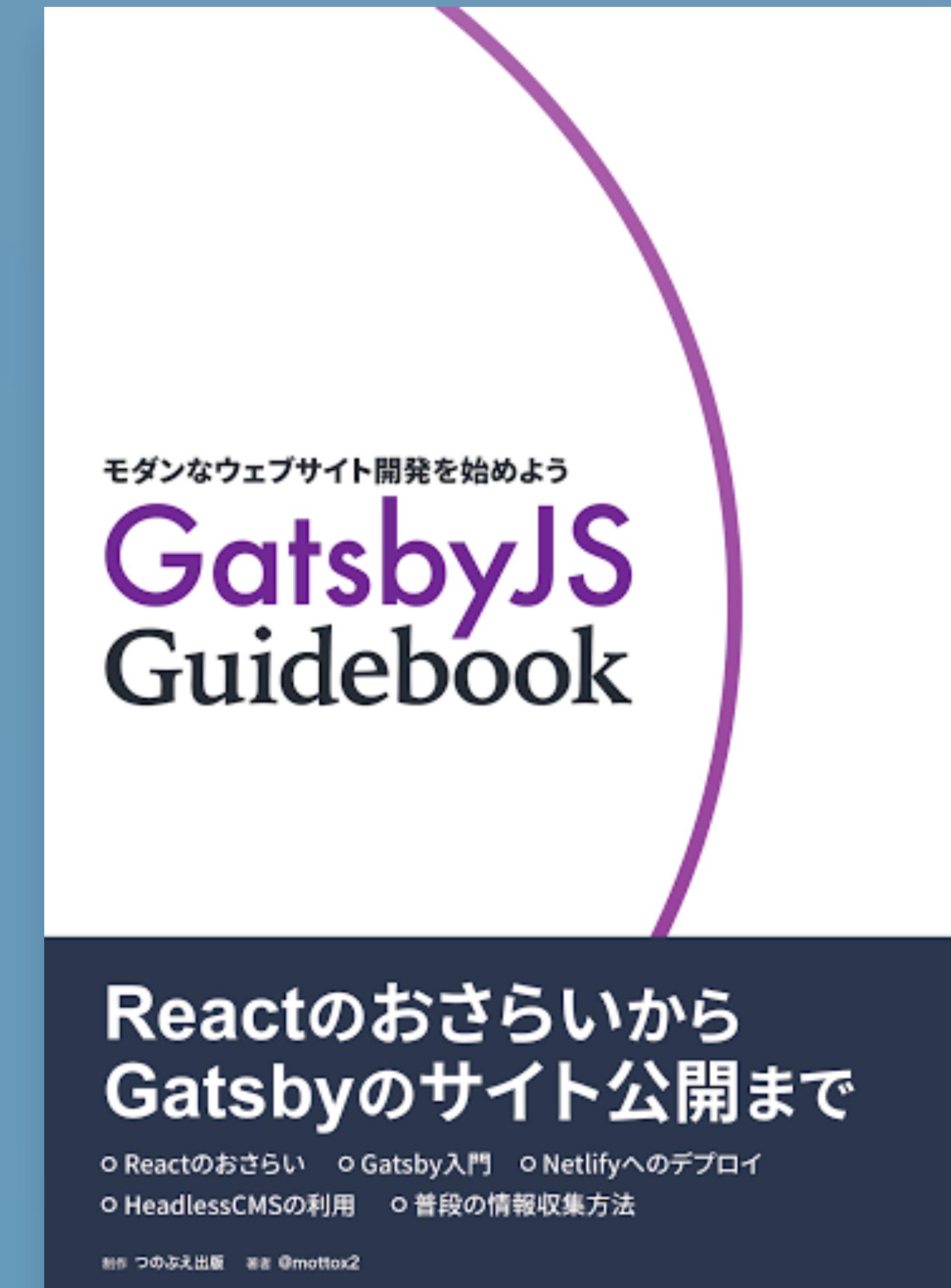
# JSXでつくる宣言的UI

PowerPointプレゼンテーションを宣言的な記述でつくる

## 自己紹介

フリーランスWebエンジニア

@mottox2



お仕事

アプリケーションエンジニア

Watching

Gatsby, Gridsome, Next.js, etc...

ひとこと

PowerPointよりKeynote派

# お題のサービス

今日は現場感のある話をしたい



Google Analyticsと連携 したら

PowerPointレポート が出来るサービス

# お題のサービス

今日は現場感のある話をしたい



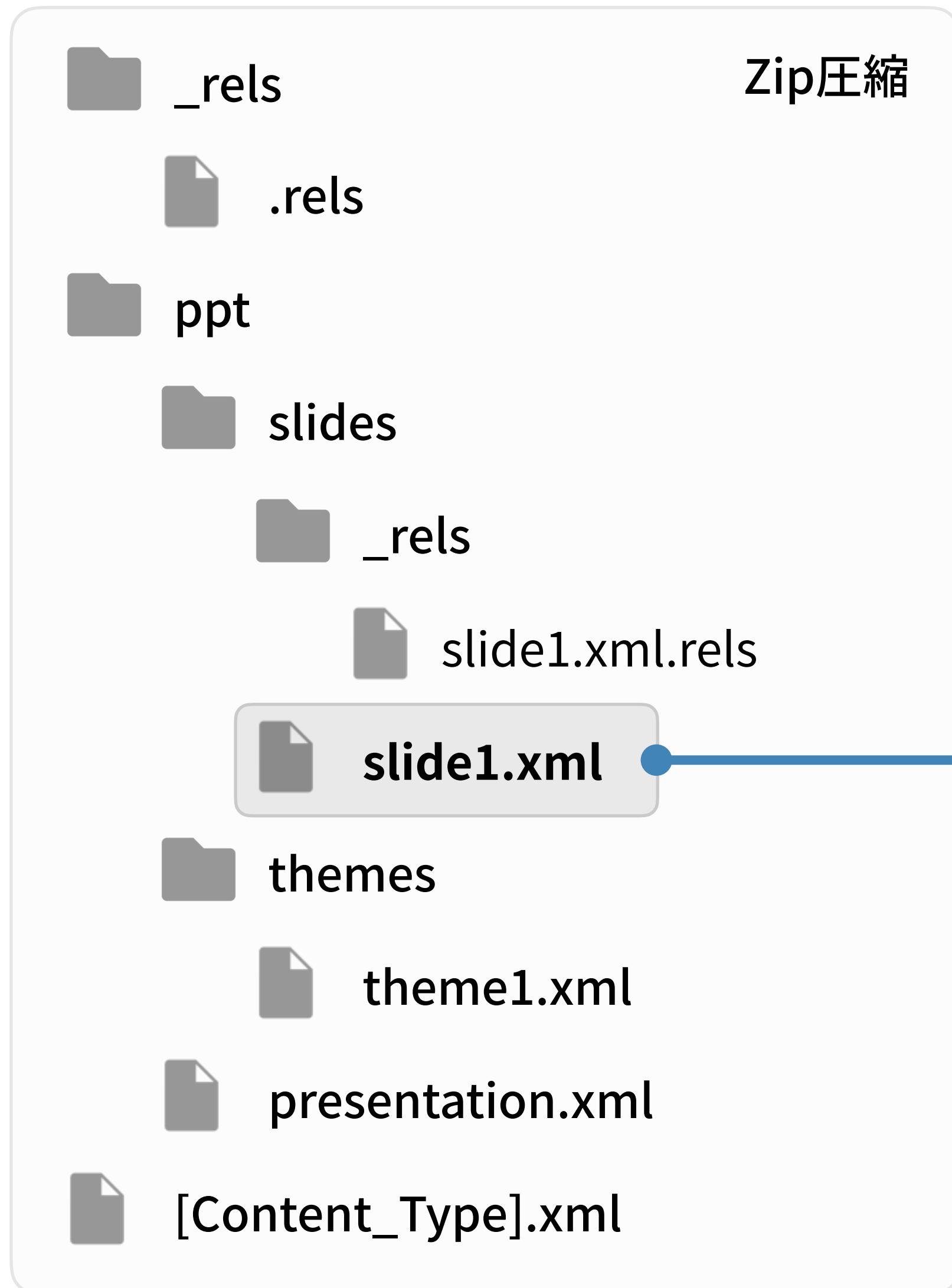
PowerPointファイルってどうやってつくるの？

Google Analyticsと連携 したら

PowerPointレポート が出来るサービス

# Office Open XML(OOXML)

MS OfficeのファイルはXMLで構築されている



```
<p:sp>  
  <p:nvSpPr>  
    <p:cNvPr id="4" name="テキスト ボックス 3"/>  
    <p:cNvSpPr txBox="1"/>  
    <p:nvPr/>  
  </p:nvSpPr>  
  <p:spPr>  
    <a:xfrm>  
      <a:off x="2185086" y="3013501"/>  
      <a:ext cx="7821827" cy="830997"/>  
    </a:xfrm>  
    <a:prstGeom prst="rect">  
      <a:avLst/>  
    </a:prstGeom>  
    <a:noFill/>  
  </p:spPr>  
  <p:txBody>  
    <a:bodyPr wrap="square" rtlCol="0">  
      <a:spAutoFit/>  
    </a:bodyPr>  
    <a:lstStyle/>  
    <a:p>  
      <a:r>  
        <a:rPr lang="en-US" altLang="ja-JP" sz="4800" b="1" dirty="1"/>  
        <a:t>Hello PowerPoint!</a:t>  
      </a:r>  
      <a:endParaRPr kumimoji="1" lang="ja-JP" altLang="en-US" sz="4800" b="1" dirty="1"/>  
    </a:p>  
  </p:txBody>  
  </p:sp>  
</p:spTree>  
</p:cSld>  
<p:clrMap0vr>  
  <a:masterClrMapping/>  
</p:clrMap0vr>  
</p:sld>
```

# 実際のレポート

- ▶ Markdown to PPTXツールでは表現できない複雑さ。
- ▶ レイアウト調整を伴わない変更が好まれる。

## 日別、曜日別、時間別アクセス分析

今月のアクセス数は4,685PVでした。全体の平均で見ると、火曜日にアクセスが増え、土曜日にアクセスが落ち込む傾向にあります(図2)。PV数がピークになる時間帯は17-18時です(図3)。

図1:1日あたりのアクセス数変化

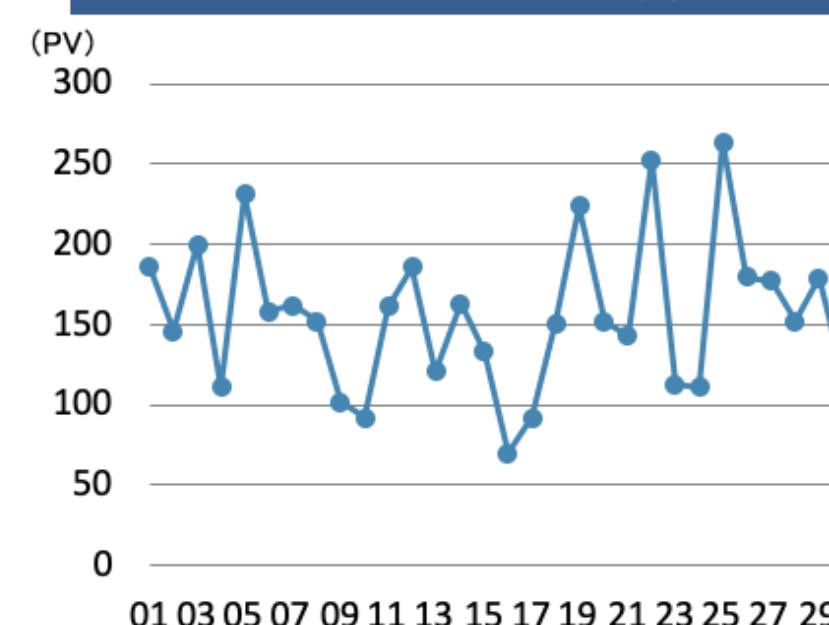


図2:曜日別アクセス数変化

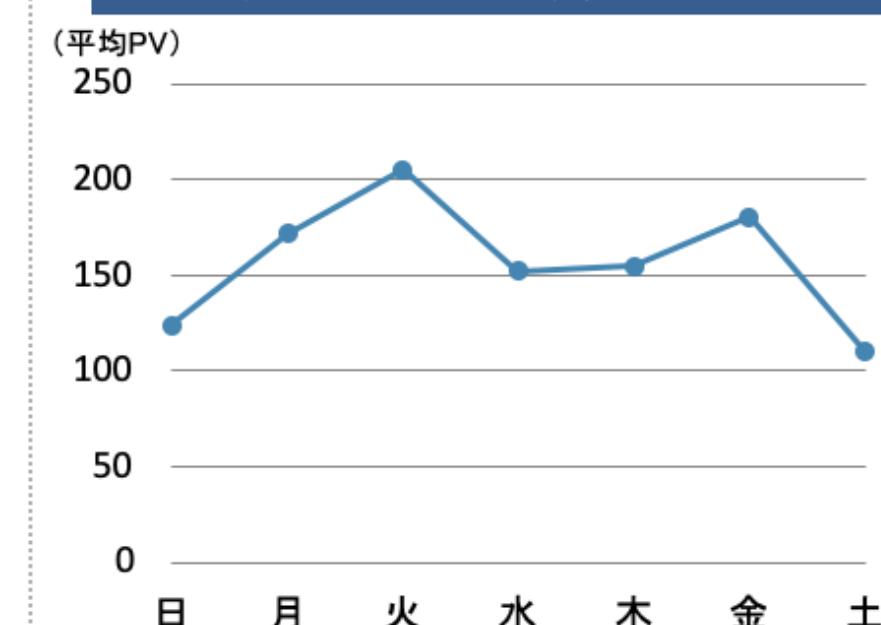
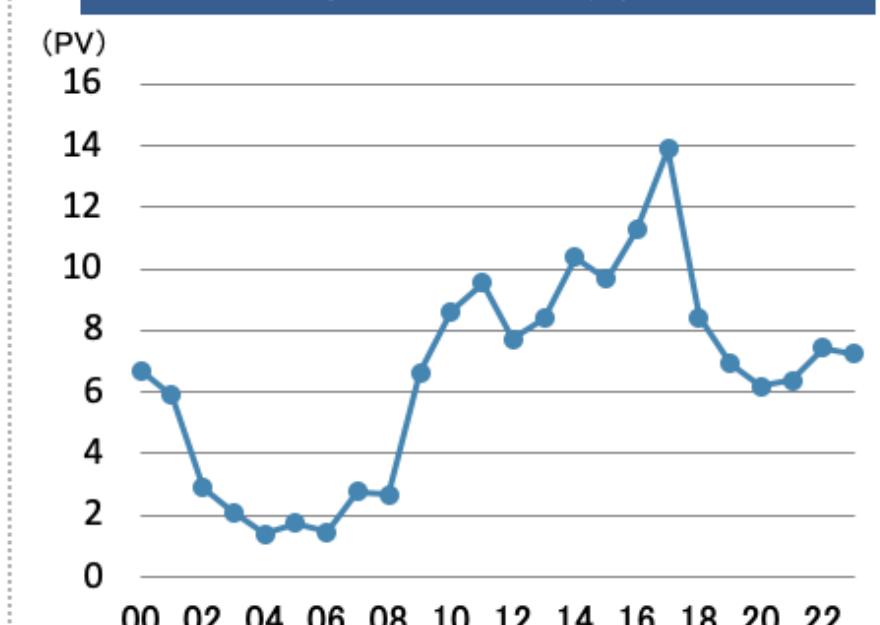


図3:時間帯別アクセス数変化



期間	PV数	期間	PV数
1日目	186	16日目	70
2日目	146	17日目	92
3日目	200	18日目	151
4日目	111	19日目	224
5日目	231	20日目	152
6日目	158	21日目	143
7日目	162	22日目	253
8日目	152	23日目	113
9日目	102	24日目	111
10日目	92	25日目	264
11日目	161	26日目	180
12日目	186	27日目	177
13日目	121	28日目	152
14日目	163	29日目	179
15日目	133	30日目	120

(調査データ:2019年11月1日~11月30日)

# 困っていたこと

一見できる！と思しがちだが…

- ▶ 実装する際に考えることが多く、pptx生成周りの実装が後回しになりがちだった。
- ▶ 下記のコードで雰囲気を掴んでもらえれば…
- ▶ 今日のテーマはこの課題にJavaScriptを使って取り組んでいく話

# 既存の生成ロジック

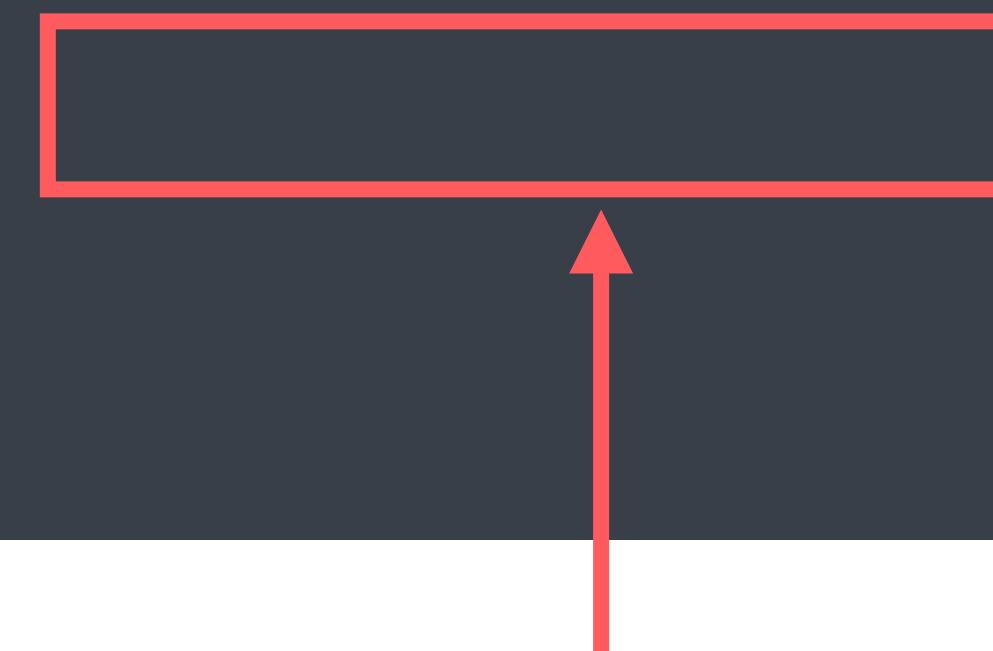
触りたくない気持ちの要因を言語化する

## ① どういう見た目なのか想像しづらい

このコードから生成されるレポートの  
見た目が想像しづらい

## ② 座標の指定が難しい

座標を絶対座標で指定する必要がある



引数はheight, width, x座標, y座標の順

# 課題へのアプローチ

## ① どういう見た目なのか想像しづらい

- ▶ Webの世界で起こった「命令的」→「宣言的」の流れに乗っかる
- ▶ ただしViewが状態を持つことはないので、Webの世界ほどメリットはない

命令的

```
const div = document.createElement('div')
div.className = 'hoge'
div.innerText = 'Hello World'

const wrapper = document.getElementById('app')
wrapper.append(div)
```

## ② 座標の指定が難しい

宣言的

```
const app = () => (
  <div className='hoge'>
    Hello World
  </div>
)

const wrapper = document.getElementById('app')
ReactDOM.render(app, wrapper)
```

# 課題へのアプローチ

## 宣言的UI

- ▶ (雑定義だけど) 最終的なアウトプットを記述するやつ。
- ▶ そもそもXML自体も宣言的と言える。
- ▶ ただ、XMLを構築する際に、命令的な記述を行いたくない。
- ▶ PowerPointで記述するものはUIなのでJSXと親和性が高い。
- ▶ **ReactのJSXに乗っかって宣言的な記述を扱う。**

## Appendix

# 課題へのアプローチ

Web以外でReactを利用している例

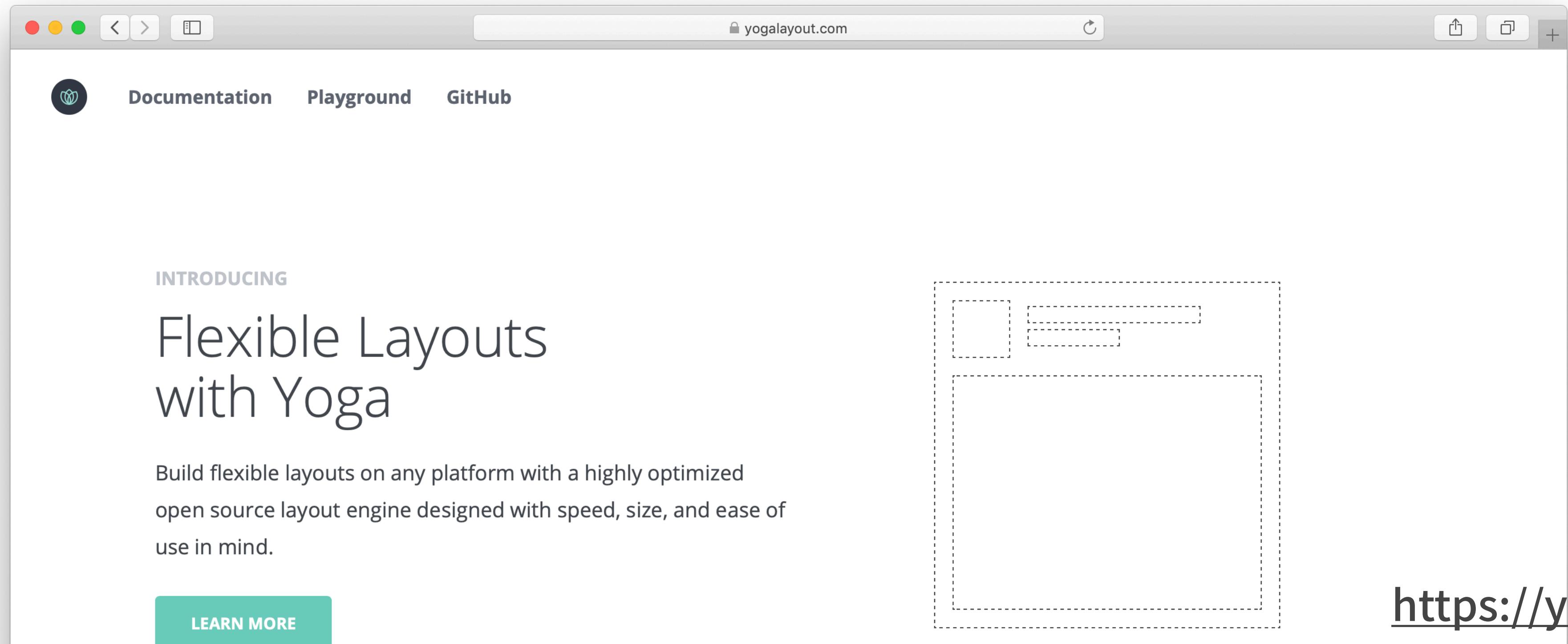
- ▶ ReactNative (For Native App)
- ▶ React360 (For VR)
- ▶ Ink (For CUI)
- ▶ React-PDF (For PDF)
- ▶ React Sketch.app (For Sketch)

# 課題へのアプローチ

① どういう見た目なのか想像しづらい

- ▶ ReactNativeの内部でも使われているレイアウトエンジン『Yoga』を用いる
- ▶ 大きさ・座標計算を任せると

② 座標の指定が難しい

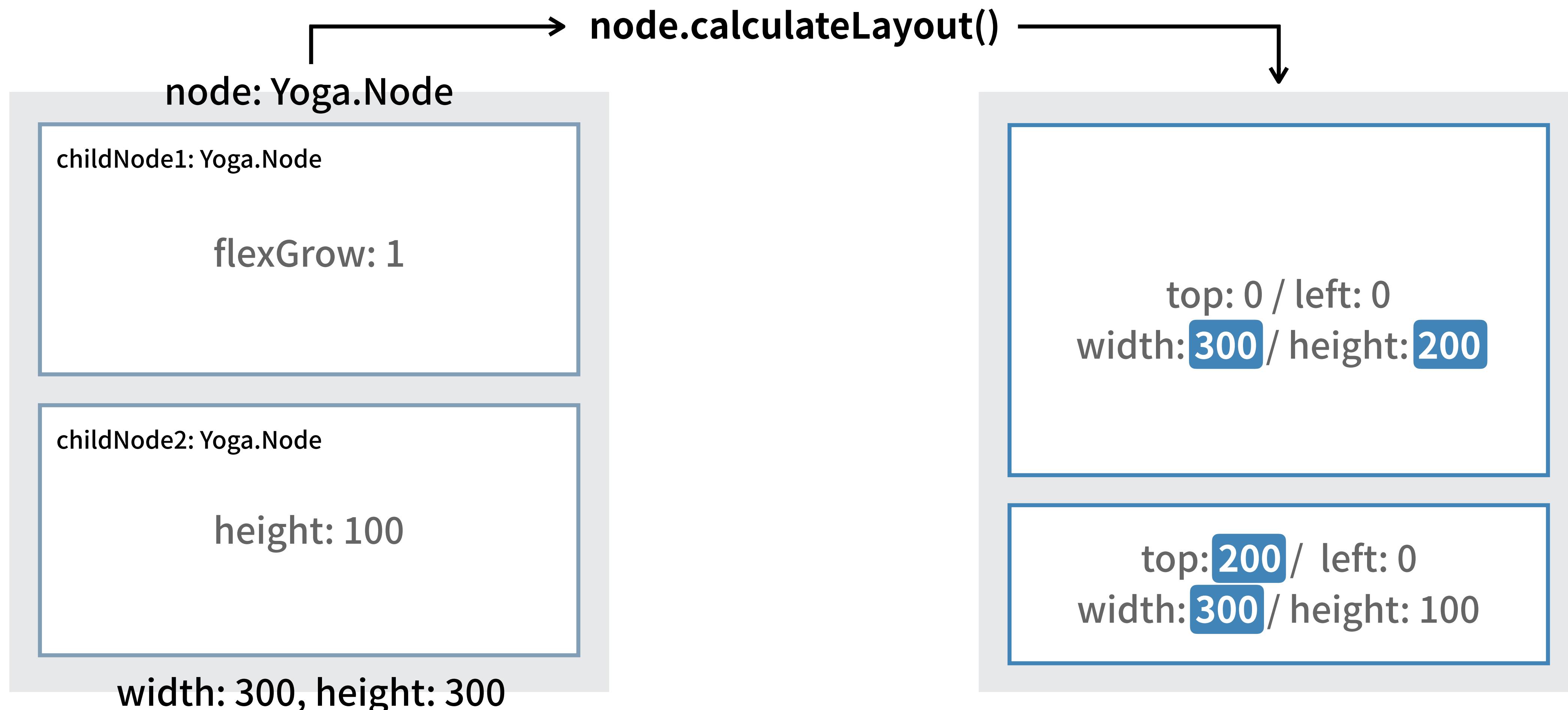


<https://yogalayout.com/>

# 課題へのアプローチ

Yogaを用いたレイアウト計算

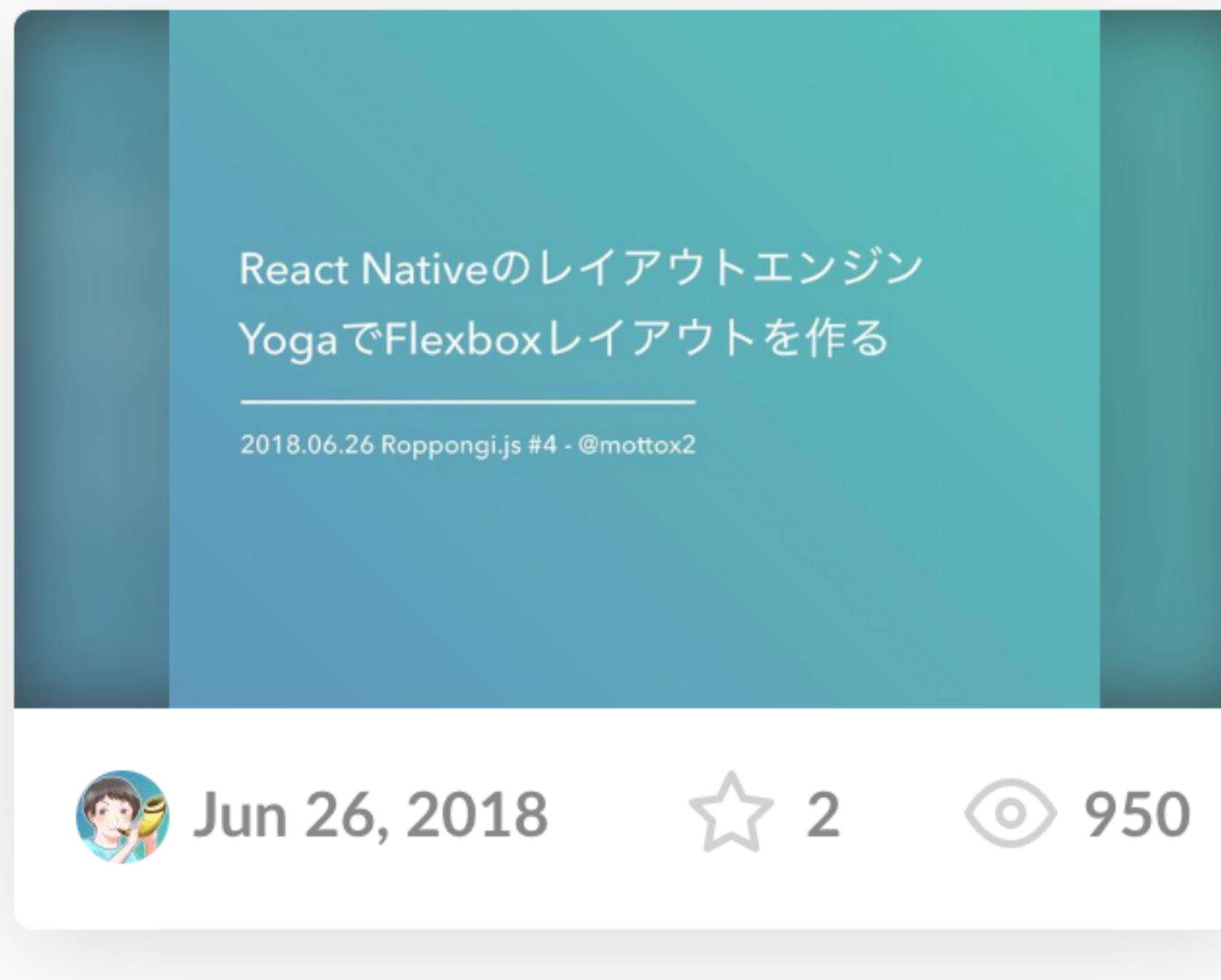
- ▶ width, height, top, leftといった座標や大きさを計算してくれる。



# 課題へのアプローチ

アプローチにあたって

- ▶ JSXからSketchファイルを生成する「React Sketch.app」の存在を知っていた。
- ▶ 以前に勉強会でLTをしていた。
- ▶ プロトタイプを作成し、実現可能か確認を行った。



React Nativeのレイアウトエンジン  
YogaでFlexboxレイアウトを作る

2018.06.26 Roppongi.js #4 - @mottox2

Jun 26, 2018    ☆ 2    ⚡ 950

React Nativeのレイアウトエンジン Yogaで  
Flexboxレイアウトを作る

<https://speakerdeck.com/mottox2/yoga-with-sketch>

# 課題へのアプローチ

## One More Thing

- ▶ 2019年のJSXらしいアプローチを取りたかった => TSX
- ▶ **VSCode as Service**
  - ▶ TypeScriptを採用し型による補完/補助が効くように
  - ▶ JSDocも利用し簡単な説明が表示されるように

The screenshot shows a dark-themed code editor in VS Code. On the left, there's a file containing JSX code. A tooltip is open over the word 'D' in the first line, showing 'Click to show 2 definitions.' Below the code, a JSDoc-style comment block is visible:

```
const data = [
  ...
]
```

PageHeader component for title and description

export default () => (
 <Slide padding="8">
 <Page>
 ...
 <Character>
 ...
 <Table flexGrow={1}>
 ...
 </Table>
 </Page>
)

The right side of the image shows the code completion interface for the 'padding' prop of the 'Slide' component. It lists 'padding?: number;' and '(JSX attribute) LayoutProps.padding?: number'. At the bottom, a red error message is displayed: 'Type 'string' is not assignable to type 'number''. The status bar at the bottom of the screen shows 'VS Code (2322)'.

# デモ



**jsx-presentation-starter**

利用者向けのテンプレート

<https://github.com/kobit-develop/jsx-presentation-starter>



実装の流れ

# JSXを利用したXMLの生成

react-test-renderer

- ▶ react-test-rendererを利用して扱いやすいオブジェクトに変換できる

JSX

```
<Slide>
  <Text fontSize={24} height={10}>
    Page Title
  </Text>
  <Table flexGrow={1}>
    {/* 略 */}
  </Table>
</Slide>
```

Object

```
{
  "type": "slide",
  "props": {},
  "children": [
    {
      "type": "text",
      "props": { "fontSize": 24, "height": 10 },
      "children": [
        "Page Title"
      ]
    },
    {
      "type": "table",
      "props": { "flexGrow": 1 },
      "children": []
    }
  ]
}
```

→ `testRenderer.create(jsx).toJSON()`

# Yogaを用いたレイアウト計算

- 一部レイアウト指定が必要だが、いい感じに計算してもらう
- 後はいいかんじにXMLを生成すればOK

## Object

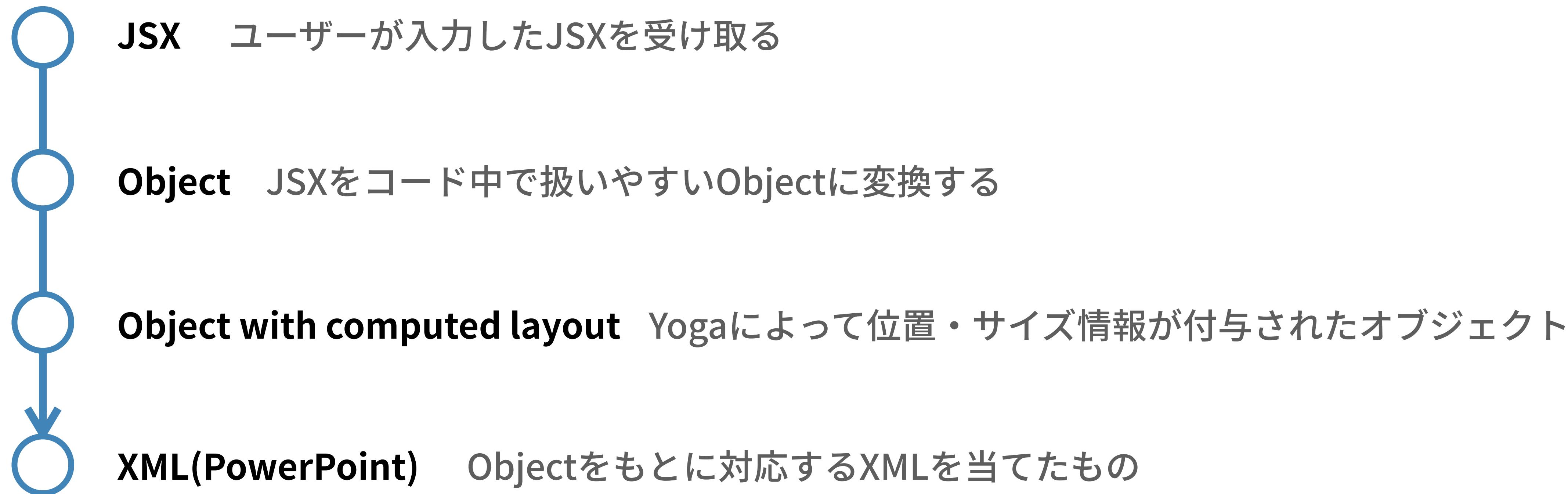
```
{  
  "type": "slide",  
  "props": {},  
  "children": [  
    {  
      "type": "text",  
      "props": { "fontSize": 24, "height": 10 },  
      "children": [  
        "Page Title"  
      ]  
    },  
    {  
      "type": "table",  
      "props": { "flexGrow": 1 },  
      "children": []  
    }  
  ]  
}
```

## Object

```
{  
  "type": "slide",  
  "props": {},  
  "layout": { "width": 60, "height": 50 },  
  "children": [  
    {  
      "type": "text",  
      "props": { "fontSize": 24, "height": 10 },  
      "layout": { "top": 0, "height": 10, "width": 60 },  
      "children": [  
        "Page Title"  
      ]  
    },  
    {  
      "type": "table",  
      "props": { "flexGrow": 1 },  
      "layout": { "top": 10, "height": 40, "width": 60 },  
      "children": []  
    }  
  ]  
}
```

# データの流れ

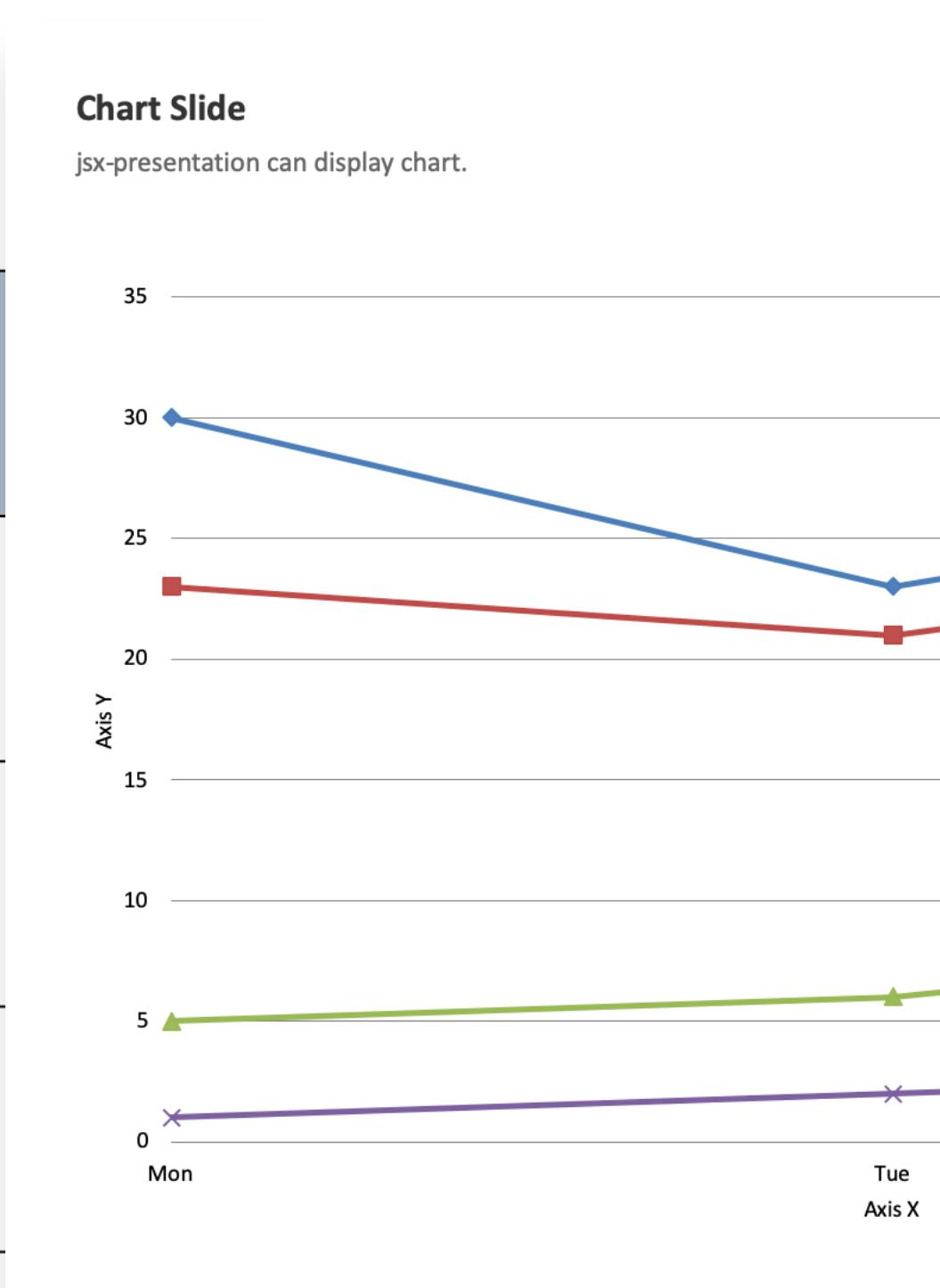
- ▶ JSXを段階的に加工して最終的なXMLを得る
- ▶ 中間状態を持つことでテストが書きやすい



# Starter Templateの用意

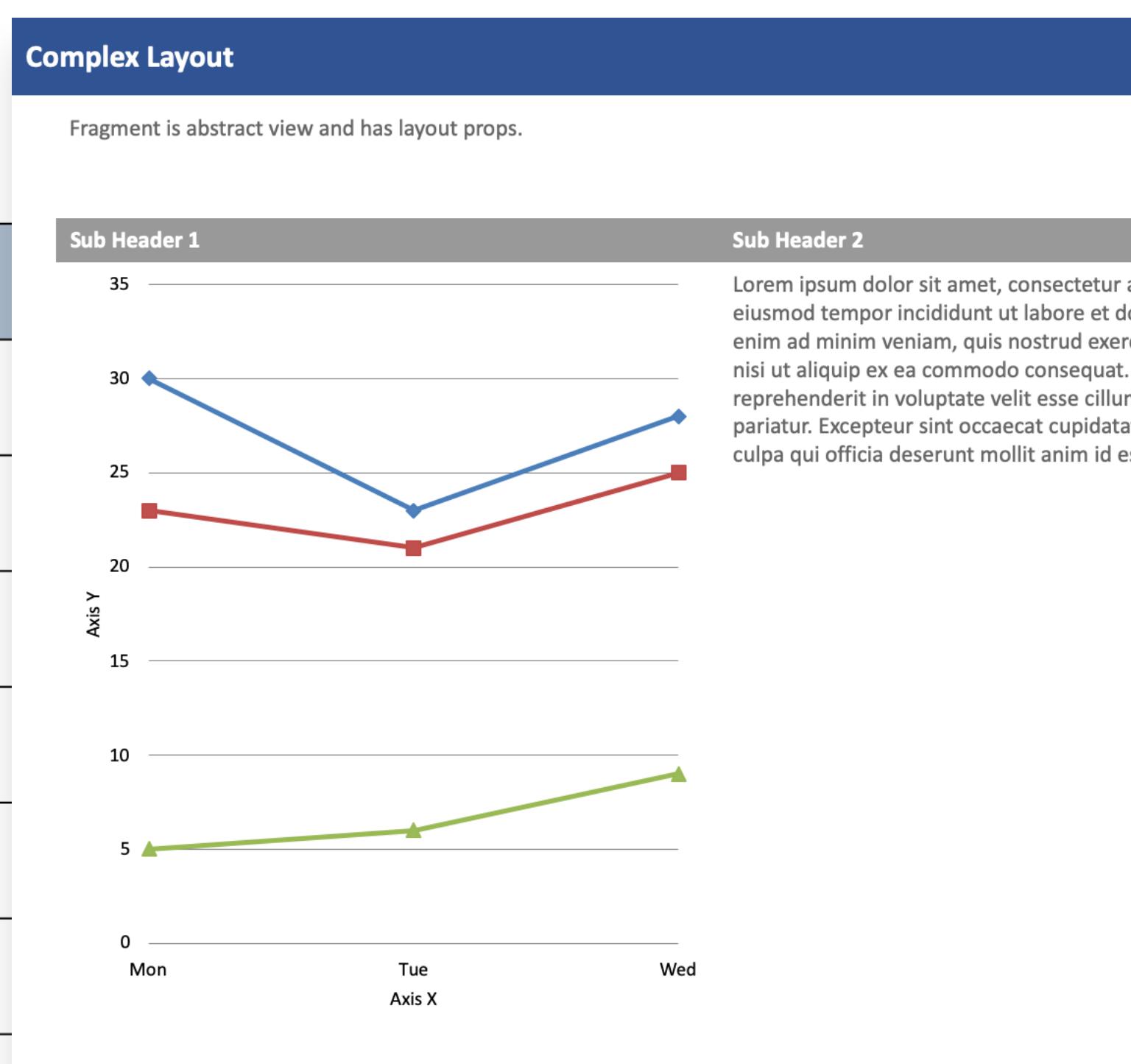
- ▶ Babel + React + TypeScriptの環境が必要なのでセットアップ難易度が高い。
- ▶ 作るだけでは使われないので、セットアップ済みのテンプレートを用意した。
- ▶ 社内的に便利なCSVがそのままテーブルになるパターンなどを用意してある。

	PV
	123123
	98329
	98329



**Table from csv**  
These data derived from csv

date	pv
1/1	123
1/2	234
1/3	423
1/4	973
1/5	812
1/6	443



# 困ったこと・困っていること

- ▶ react-domで名前空間付きのタグをrenderしようとすると警告が出る。
  - ▶ 自作して対応した。 [mottox2/react2xml](#)
- ▶ APIの設計思想が不十分。
  - ▶ PowerPointの経験を想定したAPI or HTMLライクに書きたいAPI
  - ▶ Fill or backgroundColor
  - ▶ Simple or Easyの話
  - ▶ 他のReact-XXX事例を見る限りSimpleに寄せるとと思う。

# 今後の課題とまとめ

# OSSとして開発

- ▶ 会社のMissionに「働きがいも、経済成長も」とあるため、OSSとして開発している。
  - ▶ 国連サミットで採択された持続可能な開発目標（SDGs）における17のゴールのひとつ。
- ▶ 現在v0.0.20なので安定はしていない。



## **jsx-presentation**

本体のコード

<https://github.com/kobit-develop/jsx-presentation>



## **jsx-presentation-starter**

利用者向けのテンプレート

<https://github.com/kobit-develop/jsx-presentation-starter>

# 現状と今後の課題

OSSとして

- ▶ まだExperimentalなプロジェクト。
  - ▶ API設計の思想を固める。
  - ▶ 今のところ、異常系の動作を無視している。
  - ▶ 設計を見直す必要がある。
- ▶ 名前が長いので、短くかっこいい名前がほしい。
- ▶ v0.1がリリースしたらツイートします。よろしく。

# 現状と今後の課題

会社のプロダクトとして

- ▶ 本ライブラリを用いた MVP作成を試しているところ
  - ▶ かなり開発者体験がよくなつた
  - ▶ 難しいレイアウトや要素はライブラリ側が未対応
- ▶ 今後の方針
  - ▶ Production投入を目指す
  - ▶ 最終的にはLambdaで動く関数として動作させたい
  - ▶ これが達成されればサービスの進化が早くなるはず

# まとめ

- ▶ JavaScriptのエコシステムに乗っかって、命令的な実装を宣言的な記法で行えるようにした。
  - ▶ 結局はOOXMLを書くためのDSLではある。
  - ▶ 既存の技術の組み合わせでも十分。
- ▶ 事例の引き出しは課題に向き合う際の武器になる。
  - ▶ 新しいモノ好きと揶揄されることもある。
  - ▶ 表面的になぞるのではなく、どういった応用が効くのか意識しながら調査するとよい。

# お気持ち

- ▶ JavaScriptの世界は広がっている。
  - ▶ Web以外のフォーマットに取り組むのも面白い。
  - ▶ JavaScriptらしいアプローチで取り組みたい。
  - ▶ JSXの応用は色々できそう。MDXあたりを使ったなにかの到来を期待している。
  - ▶ パラダイムシフトを別の分野で適用できないか考える。
- ▶ ライブラリ・フレームワーク作者の気持ちが理解できる。
- ▶ すぐに役にたたないものを作るのは楽しい。
- ▶ Keynote派の自分が、LT資料制作に使い始めたら勝利です。
- ▶ Productionに入ったらどこかでLTしたい。

## Appendix

# 詳しく知りたい人のためのリンク集

- ▶ スライド: 宣言的UI @sonatard (builderscon Tokyo 2019)
  - ▶ <https://speakerdeck.com/sonatard/xuan-yan-de-ui>
  - ▶ 宣言的UIについてまとめたスライド
- ▶ 書籍: Office Open XMLフォーマットガイド (技術の泉シリーズ)
  - ▶ <https://www.amazon.co.jp/gp/product/B07ZJ4ZZZB>
  - ▶ Web上で情報が得にくいOOXMLについて書かれた本



# Thank you!

興味を持った方がいれば懇親会で話しかけてください