tech

Как работают анимации в RN

СберМаркет

«Работаю с RN больше 6 лет. Люблю делать красивые анимации»

Прокопьев Евгений

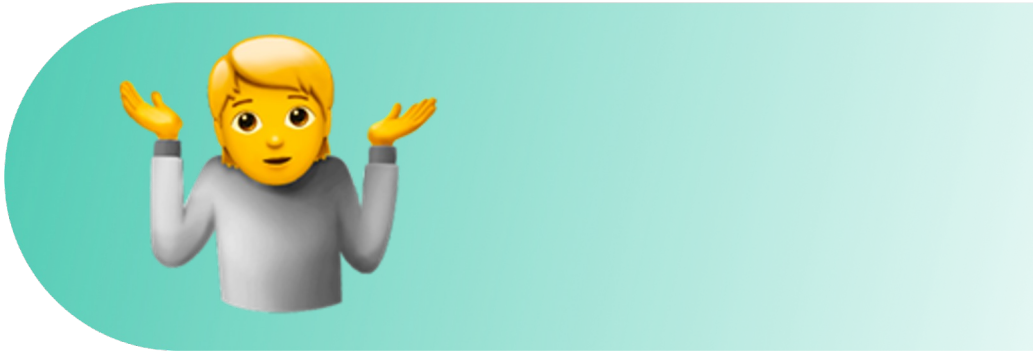
СберМаркет

@PROKOPIEVEVGENI

содержание

tech

кто работал/слышал про RN?



tech

кто делал анимации на RN?

tech

было ли это больно?

tech

“**The biggest mistake** we made as a company **was betting too much on HTML** as opposed to native”

Mark Zuckerberg

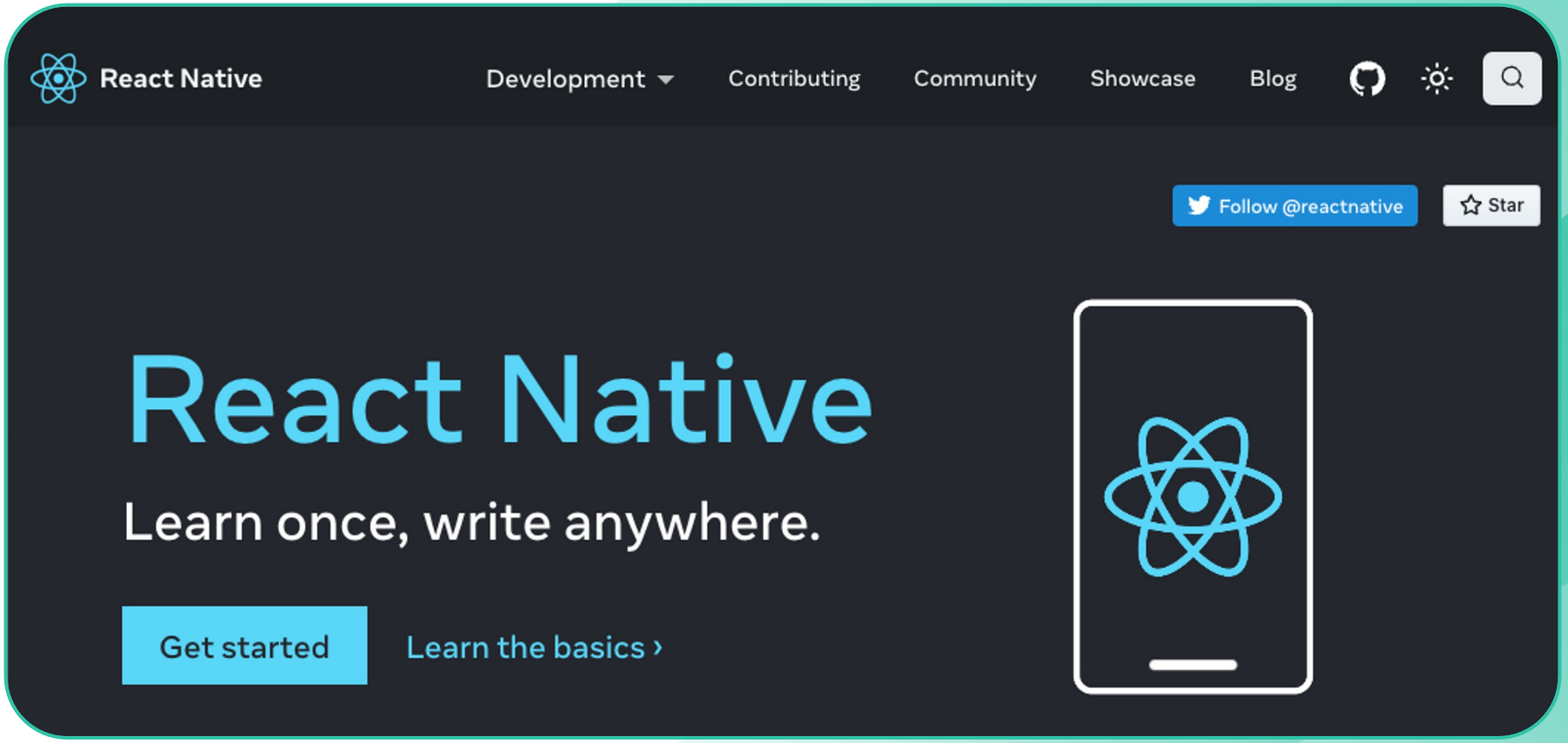
Jordan Walke

Facebook Ads разработчик, создатель React JS

В 2013 придумал способ рендерить нативные iOS элементы из JS

Внутренний хакатон

В 2015 Facebook публикует фреймворк RN, добавив к iOS еще и поддержку Android


React Native
Development
Contributing
Community
Showcase
Blog
Follow @reactnative
Star
React Native
Learn once, write anywhere.
Get started
Learn the basics ›

tech

web

```
1  function MyButton() {
2    return (
3      <button>
4        I'm a button
5      </button>
6    );
7  }
8
9  export default function MyApp() {
10   return (
11     <div>
12       <h1>Welcome to my app</h1>
13       <MyButton />
14     </div>
15   );
16 }
```

native

```
1  function MyButton() {
2    return (
3      <TouchableOpacity>
4        <Text>I'm a button</Text>
5      </TouchableOpacity>
6    );
7  }
8
9  export default function MyApp() {
10   return (
11     <View>
12       <Text>Welcome to my app</Text>
13       <MyButton />
14     </View>
15   );
16 }
17
```

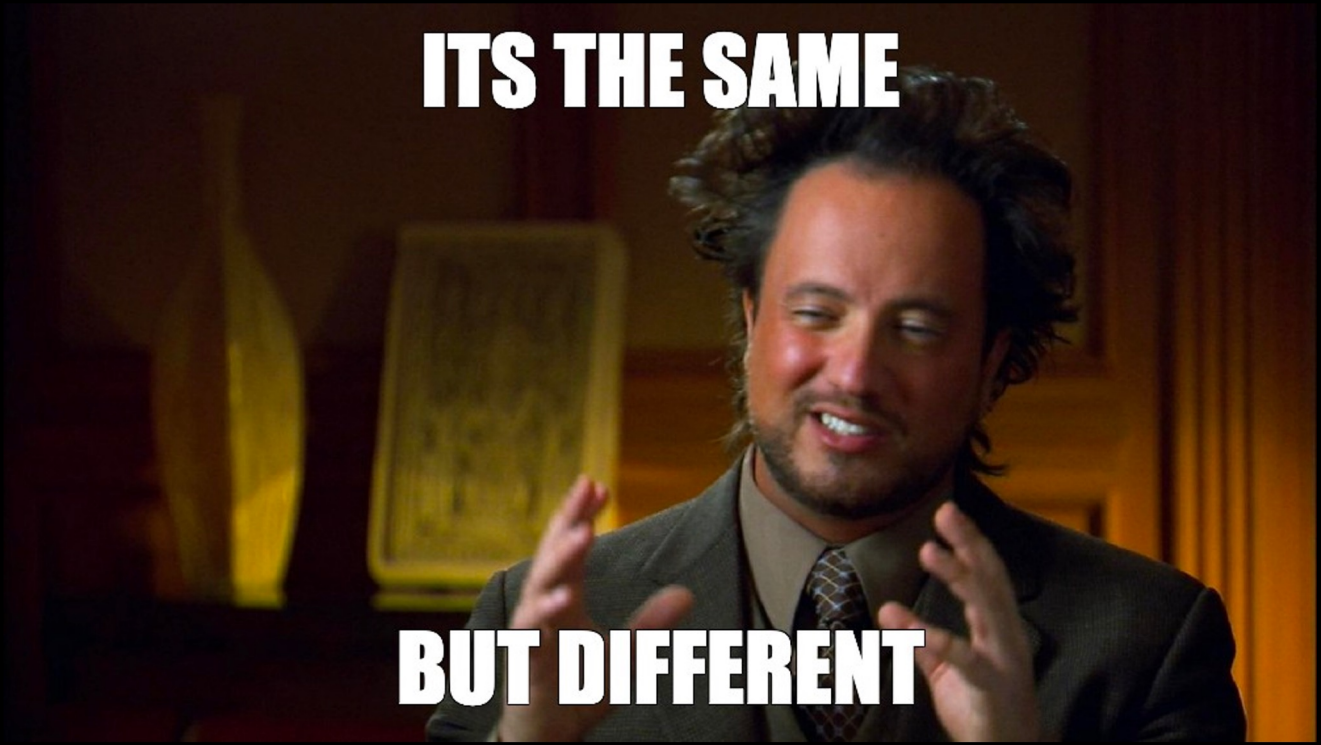
web

```
1  function MyButton() {
2    return (
3      <button>
4        I'm a button
5      </button>
6    );
7  }
8
9  export default function MyApp() {
10   return (
11     <div>
12       <h1>Welcome to my app</h1>
13       <MyButton />
14     </div>
15   );
16 }
```

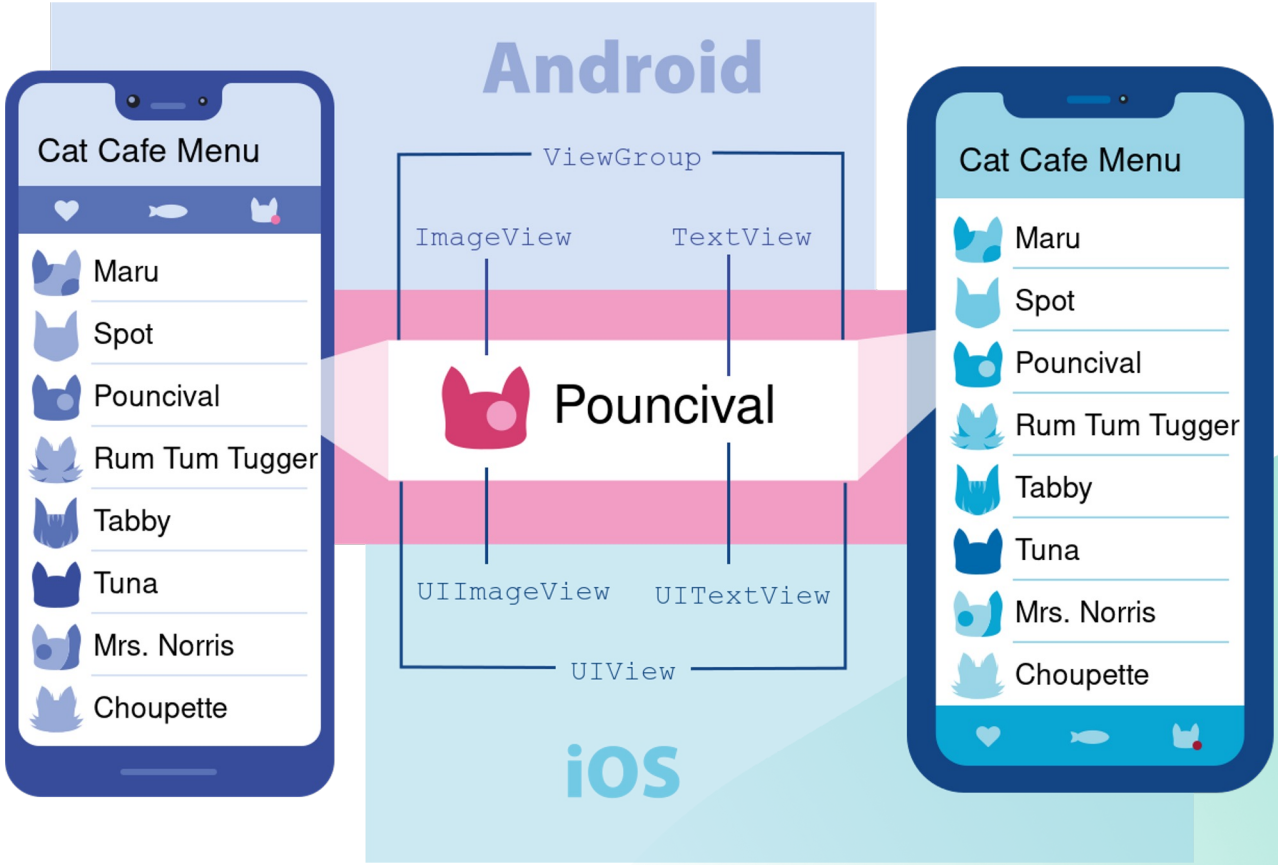
native

```
1  function MyButton() {
2    return (
3      <TouchableOpacity>
4        <Text>I'm a button</Text>
5      </TouchableOpacity>
6    );
7  }
8
9  export default function MyApp() {
10   return (
11     <View>
12       <Text>Welcome to my app</Text>
13       <MyButton />
14     </View>
15   );
16 }
17
```

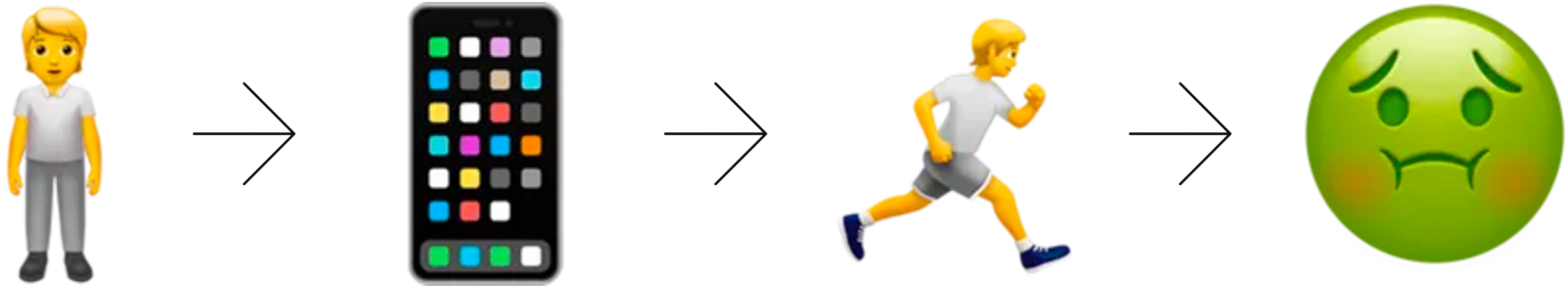
React dom vs React Native



что такое React Native?



нельзя писать как под WEB



tech

почему все хейтят RN?

анимации лагают, все лагаеит, люди пишут и не видят лагов, скорее делают вид, что не видят 11:30

Все привет, использую Swipeable из react native gesture handler, для списка с выдвигающимися действиями по бокам как в телеграмме. В итоге анимация лагает. Может подскажите в какую сторону копать? 2 18:05

Втф? Отчего рендер обычного текста может так лагать? 17 16:45

Просто хочется больше примеров получить, так как сл
документации почему-то некоторые анимации получа
тормозами (тестил на iphone se 2, samsung S8)

Всем привет, есть приложение RN+redux+redux-saga. Когда диспатчится экшн в стор тормозит анимация. В чём проблема может быть? Приложена разрослась уже, а вот эти подтормаживала я всё портят, если честно (22:59

Всем привет. У меня есть страница с двумя экранами в больше списки транзакций, для оптимизации я использую VirtualizedList, когда я захожу на страницу ничего не лагает и отрабатывает нормально, но если начинаю переключаться между экранами списки почему то рендерятся полностью, а не виртуально 13:57

Кто знает как это можно пофиксить? 1 13:57

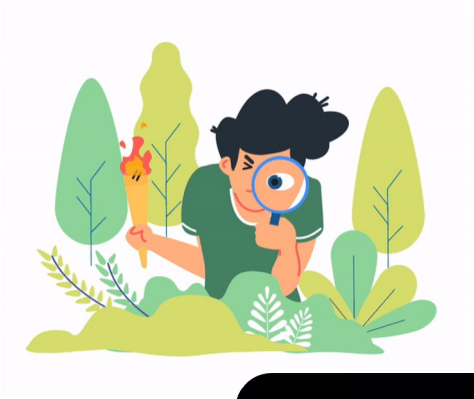
Сейчас гляну, билжу, надеюсь анимации лагать не будут 19:21

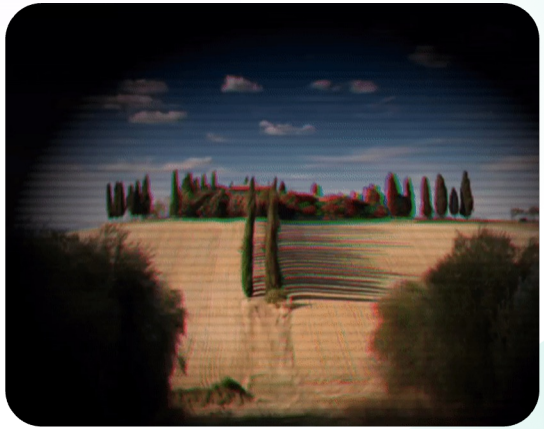
Ребят вопрос такой, чем больше анимаций я закину в `Animated.parallel([...])` тем больше будет лагать?, или я что-то упустил и это можно оптимизировать? 11:52

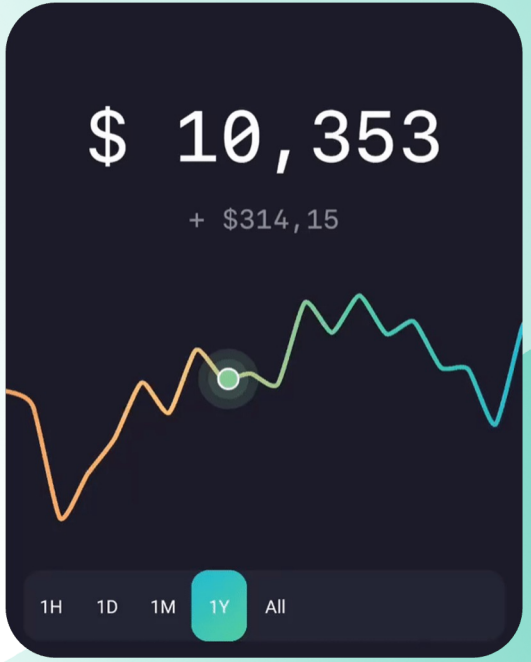
Да вроде норм, использую для прототипирования, потом просто м...
Если бы, этот грёбанный андроид лагает с экспо, именно анимации

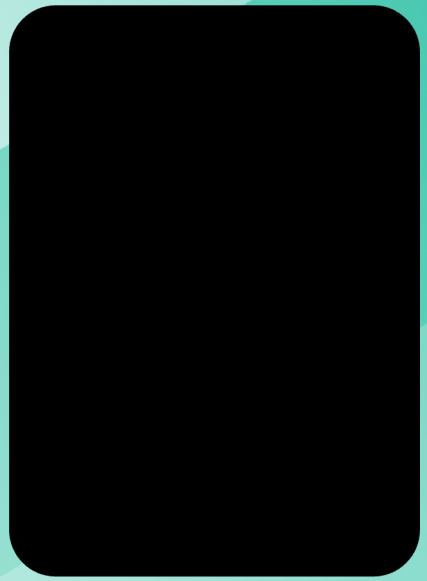
Я разлюбил react native когда мне дали реально трудный проект, где кучу анимаций логики и.т.д

примеры анимаций: все на RN!









tech

анимируем мобилку как на WEB

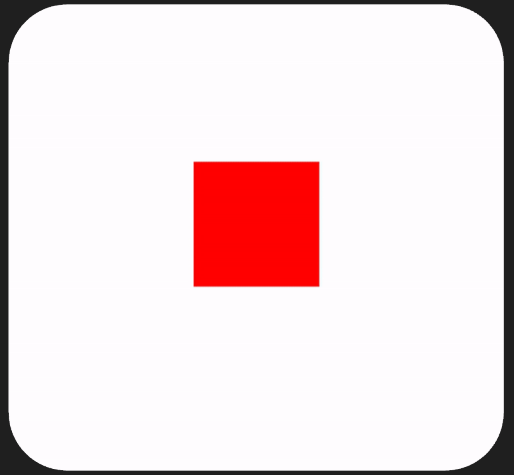
```
3   import {useRequestAnimationFrame} from './animate'

7   const viewRef = useRef<View>(null)
8   const size = useRef(100)
9   const diff = useRef(0.5)
10
11  useRequestAnimationFrame(() => {
12    const newSize = size.current + diff.current
13    viewRef.current?.setNativeProps({
14      width: newSize,
15      height: newSize,
16    })
17    if (newSize >= 100) diff.current = -0.5
18    else if (newSize <= 20) diff.current = 0.5
19  })
```

анимируем мобилку как на WEB

```
3   import {useRequestAnimationFrame} from './animate'

7     const viewRef = useRef<View>(null)
8     const size = useRef(100)
9     const diff = useRef(0.5)
10
11    useRequestAnimationFrame(() => {
12      const newSize = size.current + diff.current
13      viewRef.current?.setNativeProps({
14        width: newSize,
15        height: newSize,
16      })
17      if (newSize >= 100) diff.current = -0.5
18      else if (newSize <= 20) diff.current = 0.5
19    })
```



ставим задачу

- Эффект параллакса картинки на оверскролл списка. Для работы с transform свойствами
- Эффект аккордеона на элементах списка. Для работы со свойствами макета

tech

animated API

Animated

The `Animated` library is designed to make animations fluid, powerful, and painless to build and maintain. `Animated` focuses on declarative relationships between inputs and outputs, configurable transforms in between, and `start` / `stop` methods to control time-based animation execution.

The core workflow for creating an animation is to create an `Animated.Value`, hook it up to one or more style attributes of an animated component, and then drive updates via animations using `Animated.timing()`.

animated

```
13  const scrollY = useRef(new Animated.Value(0)).current
14  const onScroll = Animated.event([{
15    nativeEvent: {
16      contentOffset: {
17        y: scrollY,
18      },
19    },
20  }])
```

animated

```
22    const diffScale = Animated.divide(scrollY, -heightTop)
23    const tmpScale = Animated.add(1, diffScale)
24    const tmpTranslateY = Animated.divide(scrollY, 2)
25    const scale = tmpScale.interpolate({
26      inputRange: [0, 1, 2],
27      outputRange: [1, 1, 2],
28    })
29    const translateY = tmpTranslateY.interpolate({…
32    })
```

animated

```
44          <Animated.Image
45            style={[{
46              transform: [{
47                scale,
48              },{
49                translateY,
50              },
51            ]},
52            styles.view,
53            ]}
```

animated

```
13    const scrollY = useRef(new Animated.Value(0)).current
14 >  const onScroll = Animated.event([{…
20    }])
21    const diffScale = Animated.divide(scrollY, -heightTop)
22    const tmpScale = Animated.add(1, diffScale)
23    const tmpTranslateY = Animated.divide(scrollY, 2)
24 >  const scale = tmpScale.interpolate({…
27    })
28 >  const translateY = tmpTranslateY.interpolate({…
31    })
32    return (
33      <Animated.FlatList
34        onScroll={onScroll}
35        data={data}
36        renderItem={renderItem}
37        keyExtractor={(item) => item.id}
38        scrollEventThrottle={16}
39        contentContainerStyle={styles.contentContainer}
40        ListHeaderComponent={
41          <Animated.Image
42 >          style={[{…
54            }}
55          />
56        }
57      />
58    )
```

animated

```
25  const isHide = useRef(false)
26  const [toggleText, setToggleText] = useState(
27    isHide.current ? 'open' : 'close',
28  )
29  const height = useRef(
30    new Animated.Value(isHide.current ? StartSize : MaxSize),
31  ).current
32
33  const toggleMore = () => {
34    const newIsHeight = !isHide.current
35    isHide.current = newIsHeight
36    setToggleText(newIsHeight ? 'open' : 'close')
37    Animated.timing(height, {
38      toValue: newIsHeight ? StartSize : MaxSize,
39      duration: 600,
40    }).start()
41  }
```

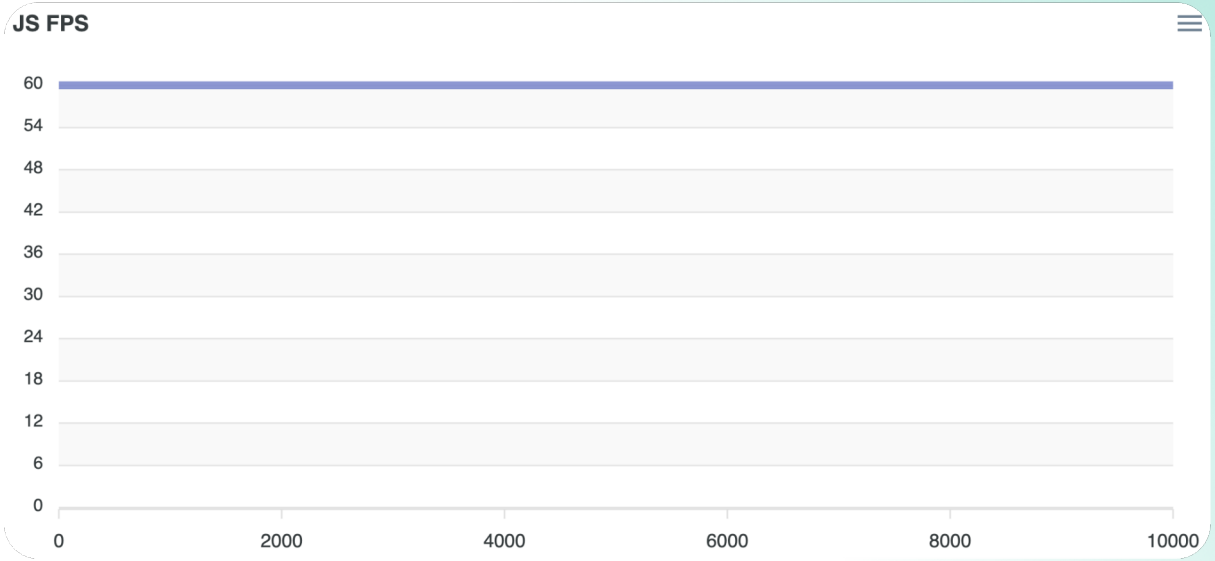
animated

```
25  const isHide = useRef(false)
26  const [toggleText, setToggleText] = useState(
27    isHide.current ? 'open' : 'close',
28  )
29  const height = useRef(
30    new Animated.Value(isHide.current ? StartSize : MaxSize),
31  ).current
32
33  const toggleMore = () => {
34    const newIsHeight = !isHide.current
35    isHide.current = newIsHeight
36    setToggleText(newIsHeight ? 'open' : 'close')
37    Animated.timing(height, {
38      toValue: newIsHeight ? StartSize : MaxSize,
39      duration: 600,
40    }).start()
41  }
```

animated

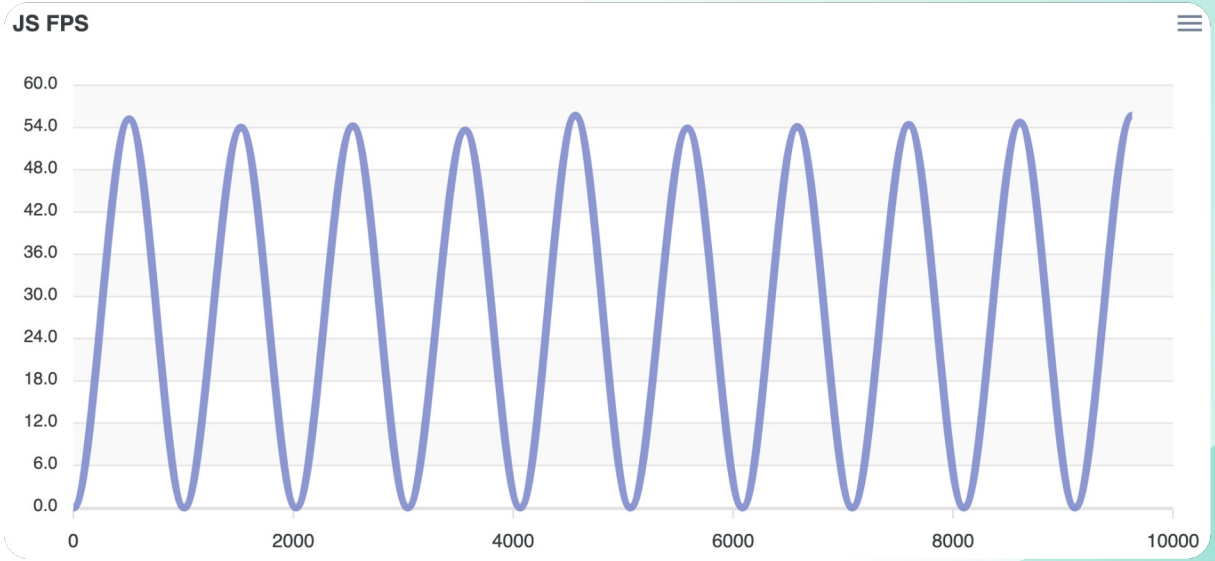

5:12
RAM
219.90
MB
JSC
0.00
MB
Views
579
649
UI
60
JS
60
King Vaughn
Irure commodo id qui consectetur duis in pariatur sunt qui anim voluptate. Ipsum nisi aute consectetur enim sint. Eu laboris exercitation sint et. Occaecat occaecat proident do pariatur enim proident. Sit eiusmod dolor occaecat in in anim dolore et et eu nisi mollit.
close
Mayo Tanner
Irure id magna cupidatat irure reprehenderit amet ex ipsum officia ad dolor. Tempor qui elit eiusmod sunt ex excepteur quis elit cupidatat consectetur. Sint dolor quis voluptate voluptate duis labore laboris magna aliqua voluptate officia aliquip. Anim nisi cupidatat esse voluptate adipisicing proident occaecat magna veniam esse laboris veniam. Proident reprehenderit enim

animated



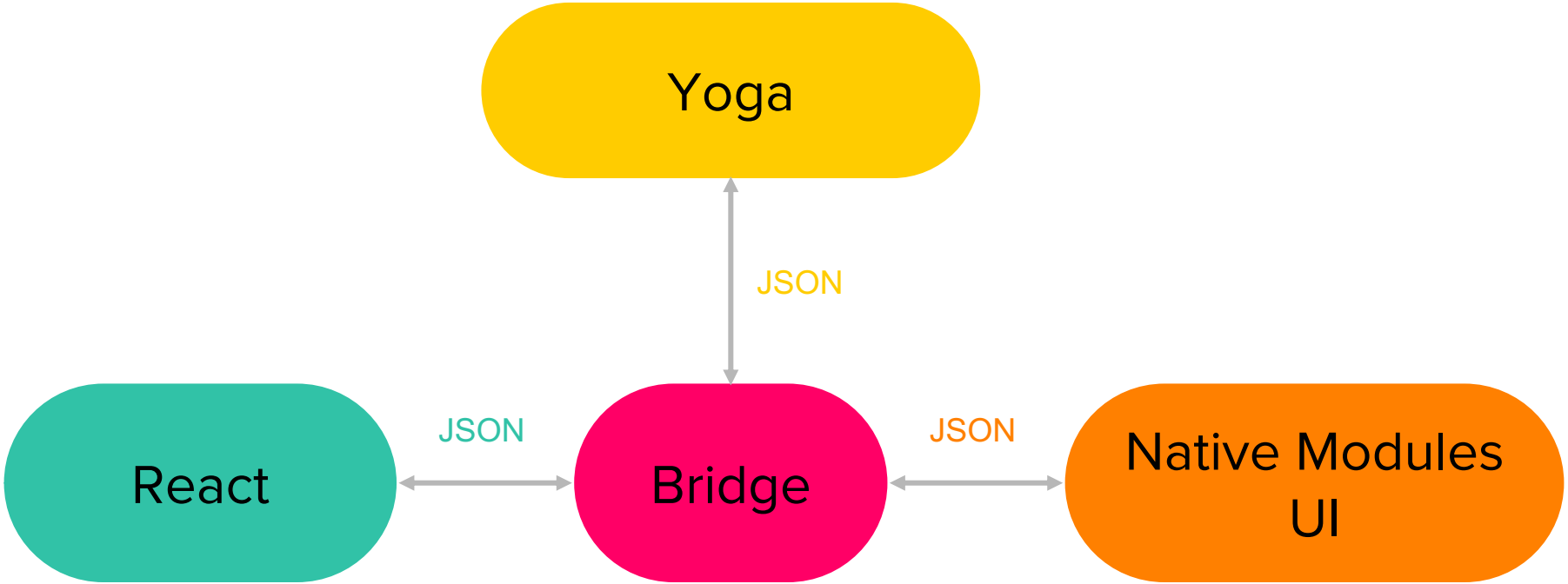
tech

animated

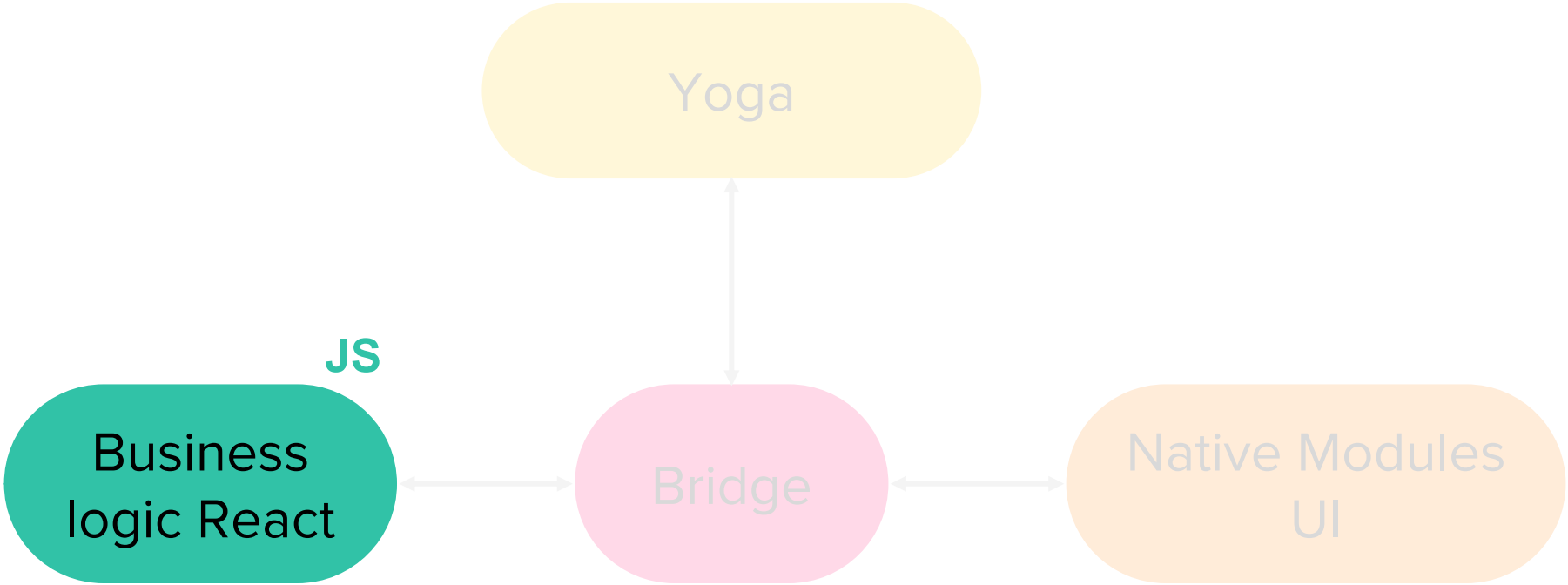


tech

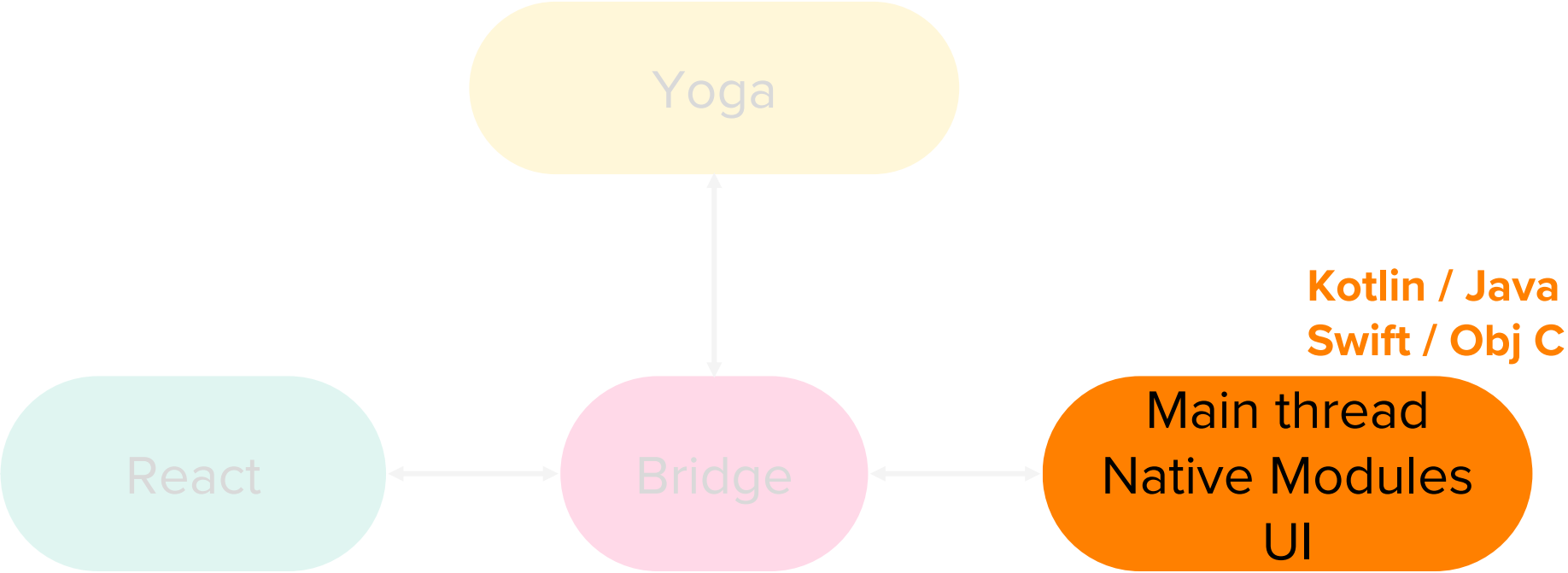
React architecture



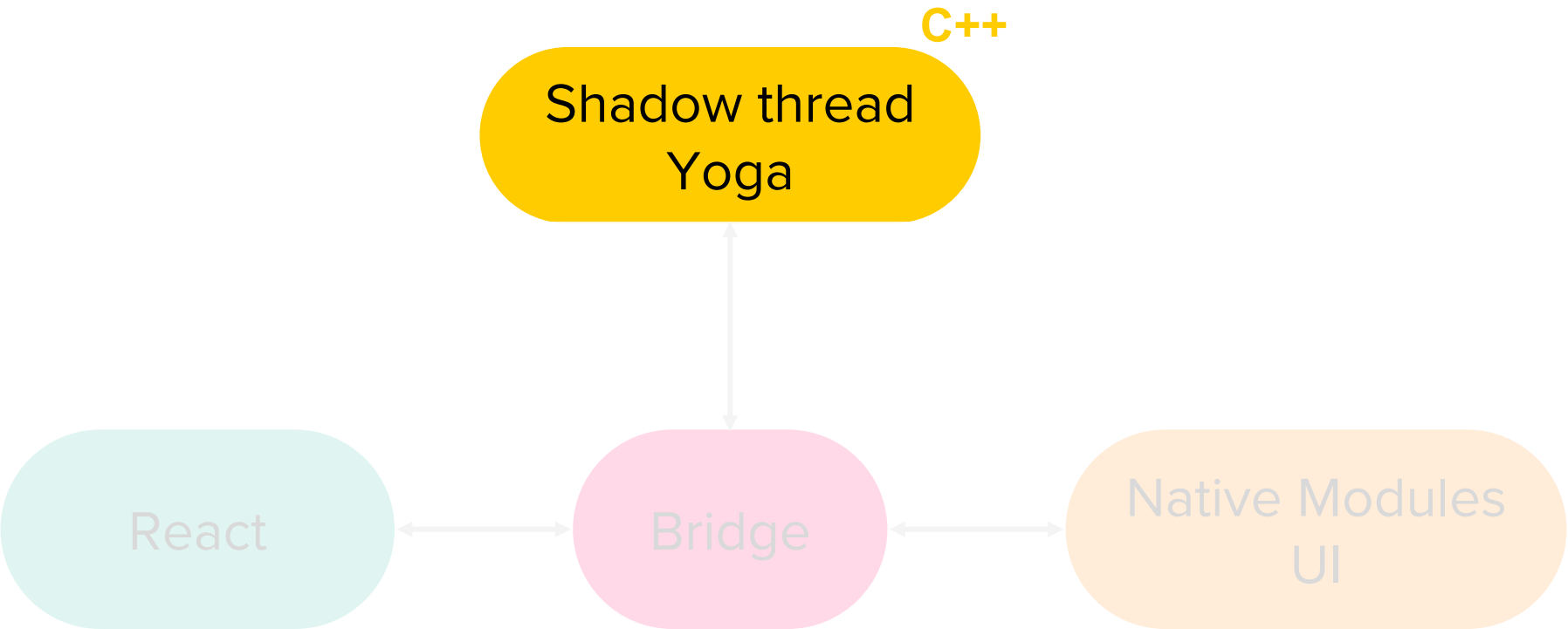
JS thread: react, business logic



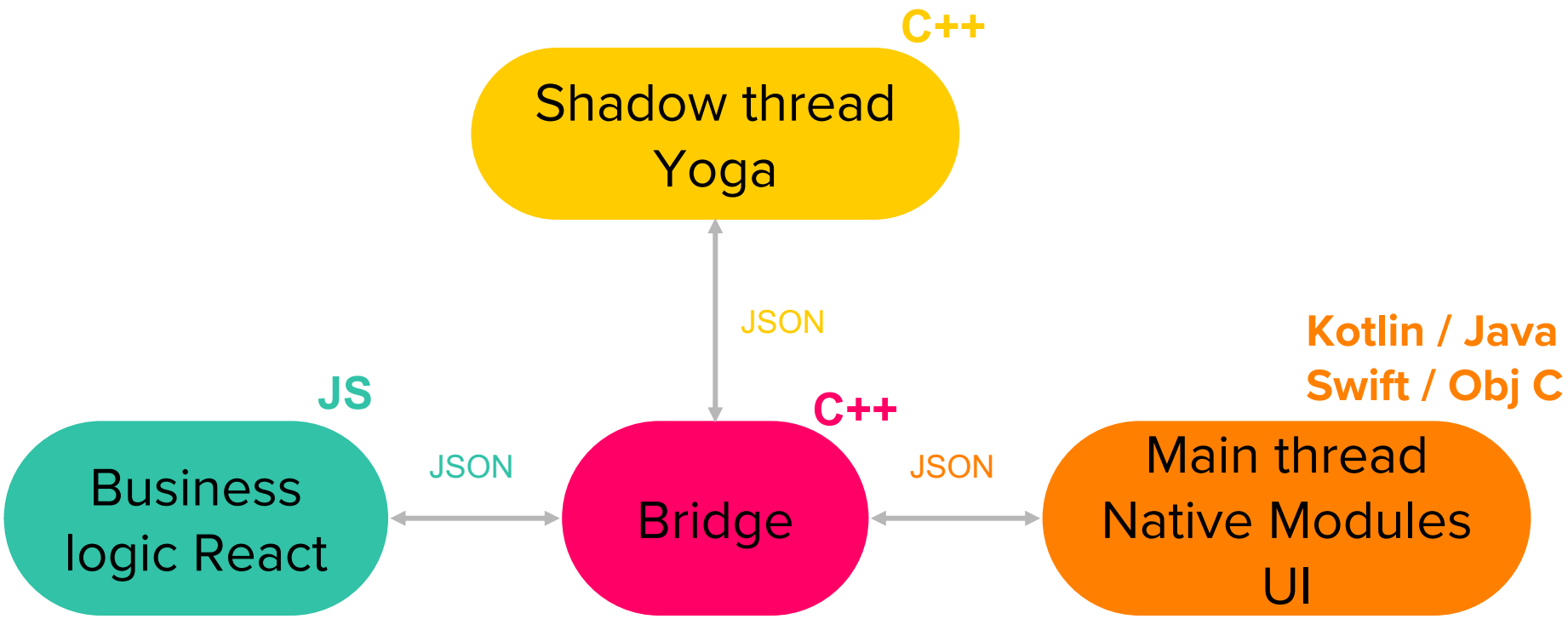
Main thread: UI/Native



Shadow/background thread: Yoga

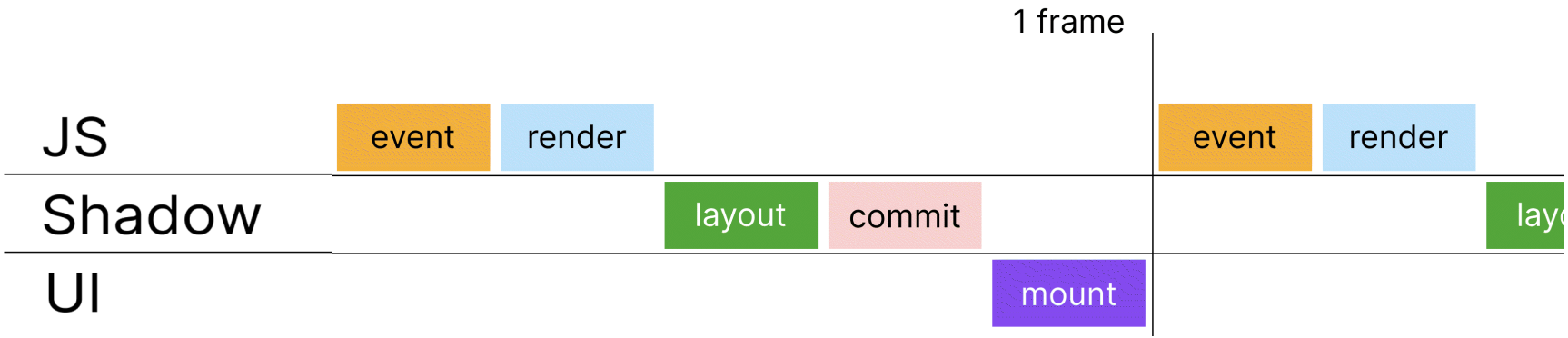


bridge

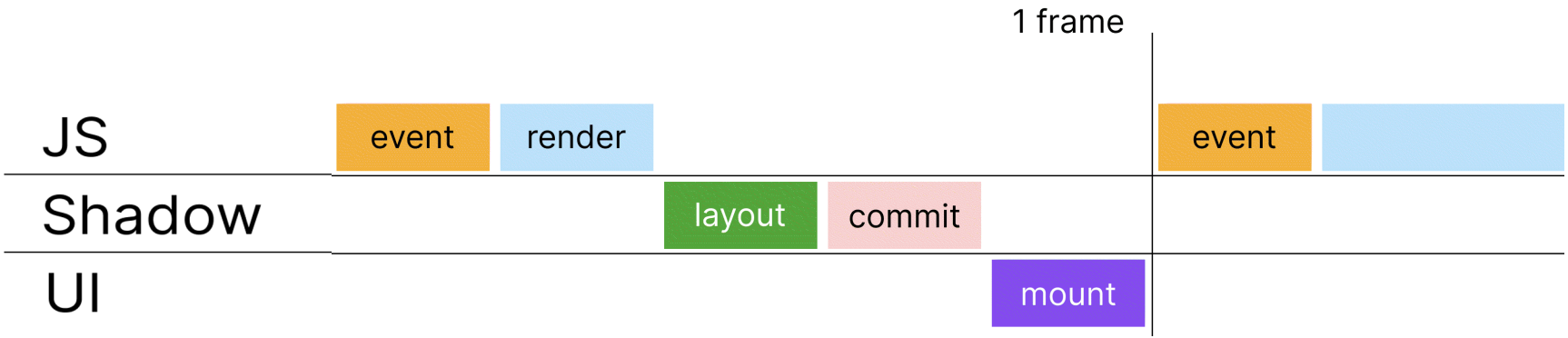


* Каждый блок означает свой поток

Взаимодействие потоков



Взаимодействие потоков



что мы узнали?

- Как анимировать компоненты с помощью Animated
- Animated использует RAF под капотом
- Что такое бридж и зачем он нужен
- Поняли почему такой хейт, в прод такое тащить опасно

tech

что придумал фейсбук?

Using Native Driver for Animated

February 14, 2017 · 7 min read

Janic Duplessis
Software Engineer at App & Flow

For the past year, we've been working on improving performance of animations that use the Animated library. Animations are very important to create a beautiful user experience but can also be hard to do right. We want to make it easy for developers to create performant animations without having to worry about some of their code causing it to lag.

tech

useNativeDriver: true для стабильной работы анимации

Before:

```
<ScrollView
  scrollEventThrottle={16}
  onScroll={Animated.event(
    [{ nativeEvent: { contentOffset: { y: this.state.animatedValue } } }]
  )}
>
  {content}
</ScrollView>
```

After:

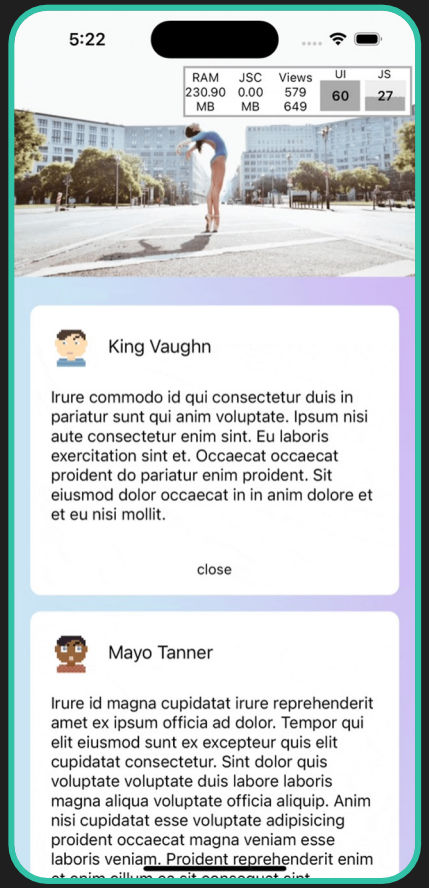
```
<Animated.ScrollView // <-- Use the Animated ScrollView wrapper
  scrollEventThrottle={1} // <-- Use 1 here to make sure no events are ever missed
  onScroll={Animated.event(
    [{ nativeEvent: { contentOffset: { y: this.state.animatedValue } } }],
    { useNativeDriver: true } // <-- Add this
  )}
>
  {content}
</Animated.ScrollView>
```

animated + useNativeDriver: true

```
13  const scrollY = useRef(new Animated.Value(0)).current
14  const onScroll = Animated.event(
15    [
16      {
17        nativeEvent: {
18          contentOffset: {
19            y: scrollY,
20          },
21        },
22      },
23    ],
24    {
25      useNativeDriver: true,
26    },
27  )
```

animated + useNativeDriver: true

```
13  const scrollY = useRef(new Animated.Value(0)).current
14  const onScroll = Animated.event(
15    [
16      {
17        nativeEvent: {
18          contentOffset: {
19            y: scrollY,
20          },
21        },
22      },
23    ],
24    {
25      useNativeDriver: true,
26    },
27  )
```



animated + useNativeDriver: true

```
33  const toggleMore = () => {
34    const newIsHeight = !isHide.current
35    isHide.current = newIsHeight
36    setToggleText(newIsHeight ? 'open' : 'close')
37    Animated.timing(height, {
38      toValue: newIsHeight ? StartSize : MaxSize,
39      duration: 600,
40      useNativeDriver: true,
41    }).start()
42  }
```

animated + useNativeDriver: true

```
33  const toggleMore = () => {
34    const newIsHeight = !isHide.current
35    isHide.current = newIsHeight
36    setToggleText(newIsHeight ? 'open' : 'close')
37    Animated.timing(height, {
38      toValue: newIsHeight ? StartSize : MaxSize,
39      duration: 600,
40      useNativeDriver: true,
41    }).start()
42  }
```

что там внутри?

```
JS->N : NativeAnimatedModule.createAnimatedNode([4,{"inputRange":[0,1,2],"outputRange":
[1,1,2],"extrapolateLeft":"extend","extrapolateRight":"extend","type":"interpolation"}])
JS->N : NativeAnimatedModule.createAnimatedNode([5,{"inputRange":[-1,0,1],"outputRange":
[-1,0,0],"extrapolateLeft":"extend","extrapolateRight":"extend","type":"interpolation"}])
JS->N : NativeAnimatedModule.createAnimatedNode([3,{"type":"transform","transforms":[{"type":"animated","property":"scale","nodeTag":4},
{"type":"animated","property":"translateY","nodeTag":5}]}])
JS->N : NativeAnimatedModule.createAnimatedNode([2,{"type":"style","style":{"transform":3}}])
JS->N : NativeAnimatedModule.createAnimatedNode([1,{"type":"props","props":{"style":2}}])
JS->N : NativeAnimatedModule.connectAnimatedNodeToView([1,3])
JS->N : NativeAnimatedModule.connectAnimatedNodes([2,1])
JS->N : NativeAnimatedModule.connectAnimatedNodes([3,2])
JS->N : NativeAnimatedModule.connectAnimatedNodes([5,3])
JS->N : NativeAnimatedModule.createAnimatedNode([7,{"type":"value","value":0,"offset":0}])
JS->N : NativeAnimatedModule.createAnimatedNode([8,{"type":"value","value":2,"offset":0}])
JS->N : NativeAnimatedModule.createAnimatedNode([6,{"type":"division","input":[7,8]}])
JS->N : NativeAnimatedModule.connectAnimatedNodes([6,5])
JS->N : NativeAnimatedModule.connectAnimatedNodes([8,6])
JS->N : NativeAnimatedModule.connectAnimatedNodes([4,3])
JS->N : NativeAnimatedModule.createAnimatedNode([10,{"type":"value","value":1,"offset":0}])
JS->N : NativeAnimatedModule.createAnimatedNode([12,{"type":"value","value":-262,"offset":0}])
JS->N : NativeAnimatedModule.createAnimatedNode([11,{"type":"division","input":[7,12]}])
JS->N : NativeAnimatedModule.createAnimatedNode([9,{"type":"addition","input":[10,11]}])
JS->N : NativeAnimatedModule.connectAnimatedNodes([9,4])
JS->N : NativeAnimatedModule.connectAnimatedNodes([10,9])
JS->N : NativeAnimatedModule.connectAnimatedNodes([11,9])
JS->N : NativeAnimatedModule.connectAnimatedNodes([12,11])
JS->N : NativeAnimatedModule.connectAnimatedNodes([7,11])
JS->N : NativeAnimatedModule.connectAnimatedNodes([7,6])
JS->N : NativeAnimatedModule.addAnimatedEventToView([509,"onScroll",{"nativeEventPath":["contentOffset","y"],"animatedValueTag":7}])
```

что там внутри?

```
JS->N : NativeAnimatedModule.createAnimatedNode([4,{"inputRange":[0,1,2],"outputRange":
[1,1,2],"extrapolateLeft":"extend","extrapolateRight":"extend","type":"interpolation"}])
JS->N : NativeAnimatedModule.createAnimatedNode([5,{"inputRange":[-1,0,1],"outputRange":
[-1,0,0],"extrapolateLeft":"extend","extrapolateRight":"extend","type":"interpolation"}])
JS->N : NativeAnimatedModule.createAnimatedNode([3,{"type":"transform","transforms":[{"type":"animated","property":"scale","nodeTag":4},
{"type":"animated","property":"translateY","nodeTag":5}]}])
JS->N : NativeAnimatedModule.createAnimatedNode([2,{"type":"style","style":{"transform":3}}])
JS->N : NativeAnimatedModule.createAnimatedNode([1,{"type":"props","props":{"style":2}}])
```

```
32  const scale = tmpScale.interpolate({
33    inputRange: [0, 1, 2],
34    outputRange: [1, 1, 2],
35  })
```

```
36  const translateY = tmpTranslateY.interpolate({
37    inputRange: [-1, 0, 1],
38    outputRange: [-1, 0, 0],
39  })
```

что там внутри?

```
JS->N : NativeAnimatedModule.createAnimatedNode([7,{"type":"value","value":0,"offset":0}])
JS->N : NativeAnimatedModule.createAnimatedNode([8,{"type":"value","value":2,"offset":0}])
JS->N : NativeAnimatedModule.createAnimatedNode([6,{"type":"division","input":[7,8]}])
JS->N : NativeAnimatedModule.connectAnimatedNodes([6,5])
JS->N : NativeAnimatedModule.connectAnimatedNodes([8,6])
```

```
const scrollY = useRef(new Animated.Value(0)).current;
```

```
const translateY = divide(scrollY, 2);
```

tech

что там внутри?

```
JS->N : NativeAnimatedModule.createAnimatedNode([10,{"type":"value","value":1,"offset":0}])
JS->N : NativeAnimatedModule.createAnimatedNode([12,{"type":"value","value":-262,"offset":0}])
JS->N : NativeAnimatedModule.createAnimatedNode([11,{"type":"division","input":[7,12]}])
JS->N : NativeAnimatedModule.createAnimatedNode([9,{"type":"addition","input":[10,11]}])
JS->N : NativeAnimatedModule.connectAnimatedNodes([9,4])
JS->N : NativeAnimatedModule.connectAnimatedNodes([10,9])
```

```
const diffScale = divide(scrollY, -heightTop);
const scale = add(1, diffScale);
const onSc
```

tech

что там внутри?

```
JS->N : NativeAnimatedModule.connectAnimatedNodes([9,4])
JS->N : NativeAnimatedModule.connectAnimatedNodes([10,9])
JS->N : NativeAnimatedModule.connectAnimatedNodes([11,9])
JS->N : NativeAnimatedModule.connectAnimatedNodes([12,11])
JS->N : NativeAnimatedModule.connectAnimatedNodes([7,11])
JS->N : NativeAnimatedModule.connectAnimatedNodes([7,6])
JS->N : NativeAnimatedModule.addAnimatedEventToView([509,"onScroll",{"nativeEventPath":["contentOffset","y"],"animatedValueTag":7}])
JS->N : UIManager.createView([509,"RCTScrollView",1,
{"scrollEventThrottle":16,"onScroll":true,"collapsable":false,"flexGrow":1,"flexShrink":1,"flexDirection":"column","overflow":"scroll","rem
oveClippedSubviews":false,"onLayout":true,"onScrollBeginDrag":true,"onScrollEndDrag":true,"onMomentumScrollBegin":true,"onMomentumScrollEnd
":true,"alwaysBounceVertical":true,"onResponderGrant":true,"onResponderReject":true,"onResponderRelease":true,"onResponderTerminationReques
t":true,"onStartShouldSetResponder":true,"onStartShouldSetResponderCapture":true,"onTouchEnd":true,"onTouchMove":true,"onTouchStart":true,"
onTouchCancel":true,"snapToStart":true,"snapToEnd":true,"pagingEnabled":false}])
```

```
const scale = add(1, diffScale);
```

```
const onScroll = Animated.event([{
  nativeEvent: {
    contentOffset: {
      y: scrollY,
    },
  }}],
  {
    useNativeDriver: true,
  },
);
<Animated.FlatList
  onScroll={onScroll}
```

tech

что там внутри?

```
JS->N : NativeAnimatedModule.createAnimatedNode([4,{"inputRange":[0,1,2],"outputRange":[1,1,2],"extrapolateLeft":"extend","extrapolateRight":"extend","type":"interpolation"}])
JS->N : NativeAnimatedModule.createAnimatedNode([5,{"inputRange":[-1,0,1],"outputRange":[-1,0,0],"extrapolateLeft":"extend","extrapolateRight":"extend","type":"interpolation"}])
JS->N : NativeAnimatedModule.createAnimatedNode([3,{"type":"transform","transforms":[{"type":"animated","property":"scale","nodeTag":4},{"type":"animated","property":"translateY","nodeTag":5}]}])
JS->N : NativeAnimatedModule.createAnimatedNode([2,{"type":"style","style":{"transform":3}}])
JS->N : NativeAnimatedModule.createAnimatedNode([1,{"type":"props","props":{"style":2}}])
JS->N : NativeAnimatedModule.connectAnimatedNodeToView([1,3])
JS->N : NativeAnimatedModule.connectAnimatedNodes([2,1])
JS->N : NativeAnimatedModule.connectAnimatedNodes([3,2])
JS->N : NativeAnimatedModule.connectAnimatedNodes([5,3])
JS->N : NativeAnimatedModule.createAnimatedNode([7,{"type":"value","value":0,"offset":0}])
JS->N : NativeAnimatedModule.createAnimatedNode([8,{"type":"value","value":2,"offset":0}])
JS->N : NativeAnimatedModule.createAnimatedNode([6,{"type":"division","input":[7,8]}])
JS->N : NativeAnimatedModule.connectAnimatedNodes([6,5])
JS->N : NativeAnimatedModule.connectAnimatedNodes([8,6])
JS->N : NativeAnimatedModule.connectAnimatedNodes([4,3])
JS->N : NativeAnimatedModule.createAnimatedNode([10,{"type":"value","value":1,"offset":0}])
JS->N : NativeAnimatedModule.createAnimatedNode([12,{"type":"value","value":-262,"offset":0}])
JS->N : NativeAnimatedModule.createAnimatedNode([11,{"type":"division","input":[7,12]}])
JS->N : NativeAnimatedModule.createAnimatedNode([9,{"type":"addition","input":[10,11]}])
JS->N : NativeAnimatedModule.connectAnimatedNodes([9,4])
JS->N : NativeAnimatedModule.connectAnimatedNodes([10,9])
JS->N : NativeAnimatedModule.connectAnimatedNodes([11,9])
JS->N : NativeAnimatedModule.connectAnimatedNodes([12,11])
JS->N : NativeAnimatedModule.connectAnimatedNodes([7,11])
JS->N : NativeAnimatedModule.connectAnimatedNodes([7,6])
JS->N : NativeAnimatedModule.addAnimatedEventToView([509,"onScroll",{"nativeEventPath":["contentOffset","y"],"animatedValueTag":7}])
JS->N : UIManager.createView([509,"RCTScrollView",1,{"scrollEventThrottle":16,"onScroll":true,"collapsable":false,"flexGrow":1,"flexShrink":1,"flexDirection":"column","overflow":"scroll","removeClippedSubviews":false,"onLayout":true,"onScrollBeginDrag":true,"onScrollEndDrag":true,"onMomentumScrollBegin":true,"onMomentumScrollEnd":true,"alwaysBounceVertical":true,"onResponderGrant":true,"onResponderReject":true,"onResponderRelease":true,"onResponderTerminationRequest":true,"onStartShouldSetResponder":true,"onStartShouldSetResponderCapture":true,"onTouchEnd":true,"onTouchMove":true,"onTouchStart":true,"onTouchCancel":true,"snapToStart":true,"snapToEnd":true,"pagingEnabled":false}])
```

```
transform: [
  {
    scale: scale.interpolate({
      inputRange: [0, 1, 2],
      outputRange: [1, 1, 2],
    }),
  },
  {
    translateY: translateY.interpolate({
      inputRange: [-1, 0, 1],
      outputRange: [-1, 0, 0],
    }),
  },
],
```

```
const scrollY = useRef(new Animated.Value(0)).current;
```

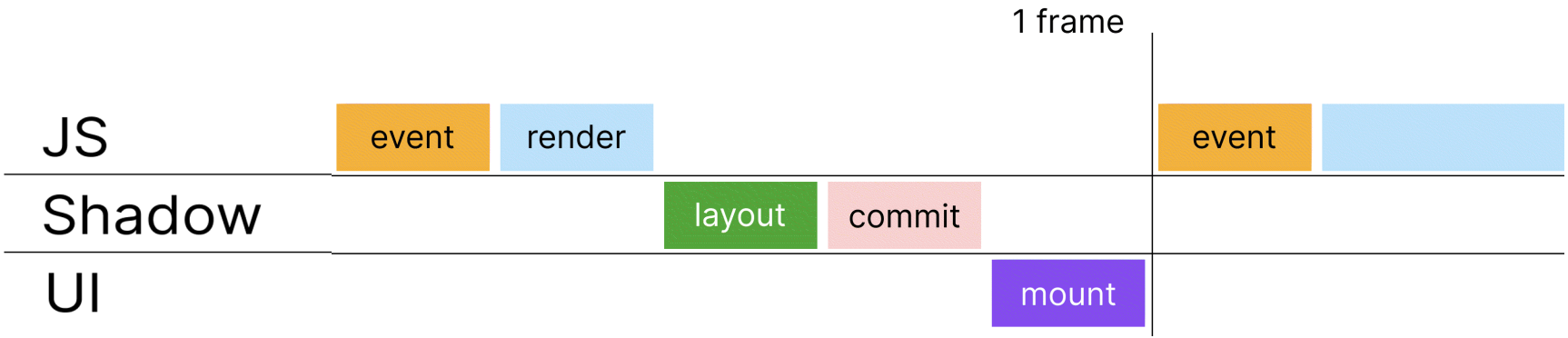
```
const translateY = divide(scrollY, 2);
```

```
const diffScale = divide(scrollY, -heightTop);
```

```
const scale = add(1, diffScale);
```

```
const onScroll = Animated.event([{
  nativeEvent: {
    contentOffset: {
      y: scrollY,
    },
  }}],
  {
    useNativeDriver: true,
  },
);
<Animated.FlatList
  onScroll={onScroll}
```

Взаимодействие потоков



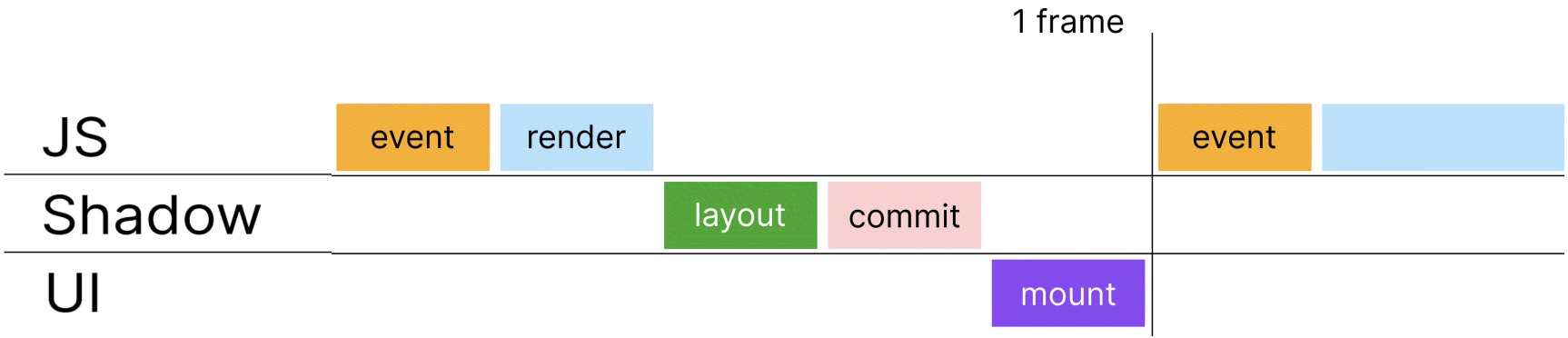
ограничения

Caveats

Not everything you can do with Animated is currently supported in Native Animated. The main limitation is that you can only animate non-layout properties, things like `transform` and `opacity` will work but Flexbox and position properties won't. Another one is with `Animated.event`, it will only work with direct events and not bubbling events. This means it does not work with `PanResponder` but does work with things like `ScrollView#onScroll`.

Native Animated has also been part of React Native for quite a while but has never been documented because it was considered experimental. Because of that make sure you are using a recent version (0.40+) of React Native if you want to use this feature.

Взаимодействие потоков



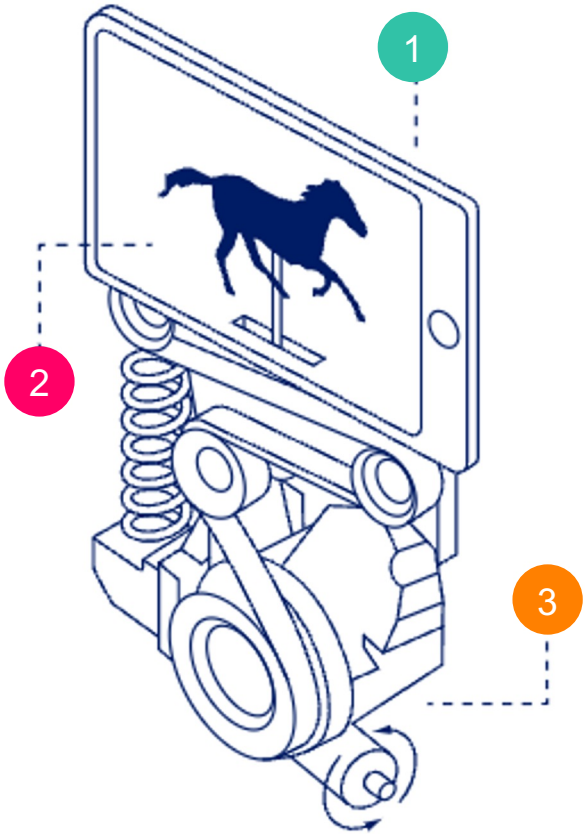
animated useNativeDriver: true

- Научились анимировать, даже когда JS лагает
- Сделали как для мобилки
- Кажется, тут мы уже не такие злые
- Но этого недостаточно

reanimated

React Native Reanimated

React Native's Animated library reimplemented



tech

почему Reanimated?

- Архитектура RN плохо подходит для взаимодействий с пользователем
- Анимация отстают минимум на 1 кадр (usenAtiveDriver:false)
- Нельзя анимировать свойства макета (useNativeDriver:true)

tech

‘Worklet’

```
function someWorklet(greeting) {
  'worklet';
  console.log("Hey I'm running on the UI thread");
}
```

```
const style = useAnimatedStyle(() => {
  console.log("Running on the UI thread");
  return {
    opacity: 0.5
  };
});
```

reanimated

```
7   import {useSharedValue, useAnimatedScrollHandler} from 'react-native-reanimated'

20    const scrollY = useSharedValue(0)
21    const onScroll = useAnimatedScrollHandler((event) => {
22      scrollY.value = event.contentOffset.y
23    })
```

reanimated

```
8   import {useDerivedValue} from 'react-native-reanimated'
```

```
24    const scale = useDerivedValue(() => {
25      let diff = scrollY.value / heightTop
26      if (diff > 0) {
27        diff = 0
28      }
29      return 1 - diff
30    })
31  > const translateY = useDerivedValue(() => {…
37    })
```

reanimated

```
9   import Animated, {useAnimatedStyle} from 'react-native-reanimated'
```

```
38  const style = useAnimatedStyle(() => {
39    return {
40      transform: [
41        {
42          translateY: translateY.value,
43        },
44        {
45          scale: scale.value,
46        },
47      ],
48    }
49  })
```

reanimated

```
20    const scrollY = useSharedValue(0)
21 >  const onScroll = useAnimatedScrollHandler((event) => {…
23    })
24 >  const scale = useDerivedValue(() => {…
30    })
31 >  const translateY = useDerivedValue(() => {…
37    })
38 >  const style = useAnimatedStyle(() => {…
49    })
50
51    return (
52      <Animated.FlatList
53        data={data}
54        renderItem={renderItem}
55        keyExtractor={(item) => item.id}
56        onScroll={onScroll}
57        scrollEventThrottle={16}
58        contentContainerStyle={styles.contentContainer}
59        ListHeaderComponent={
60          <Animated.Image
61            style={[styles.view, style]}
62 >          source={{…
64            }}
65          />
66        }
67      />
68    )
69  }
```

reanimated

```
2  import {withTiming} from 'react-native-reanimated'
```

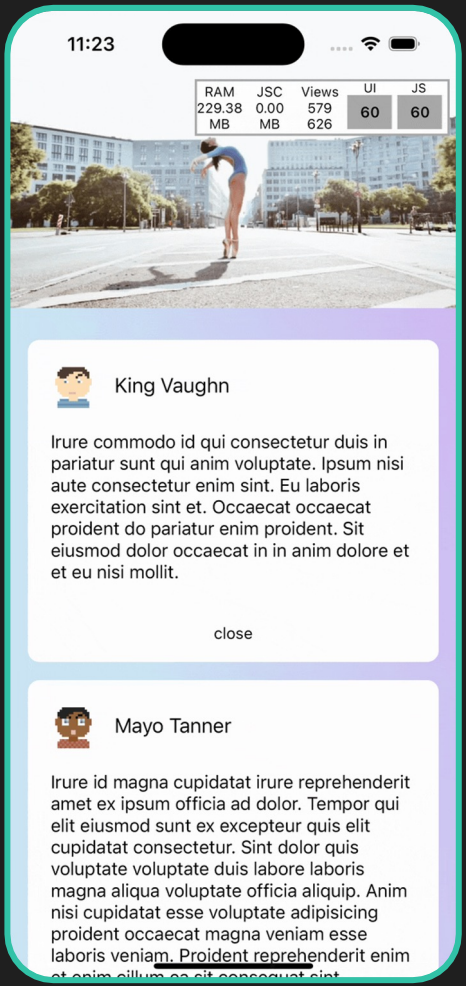
```
25   const isHide = useSharedValue(false)
26   const toggleText = useSharedValue(isHide.value ? 'open' : 'close')
27   const maxHeight = useSharedValue(isHide.value ? StartSize : MaxSize)
28   const abooutStyle = useAnimatedStyle(() => ({
29     maxHeight: withTiming(maxHeight.value, {
30       duration: 500,
31     }),
32   }))
```

reanimated

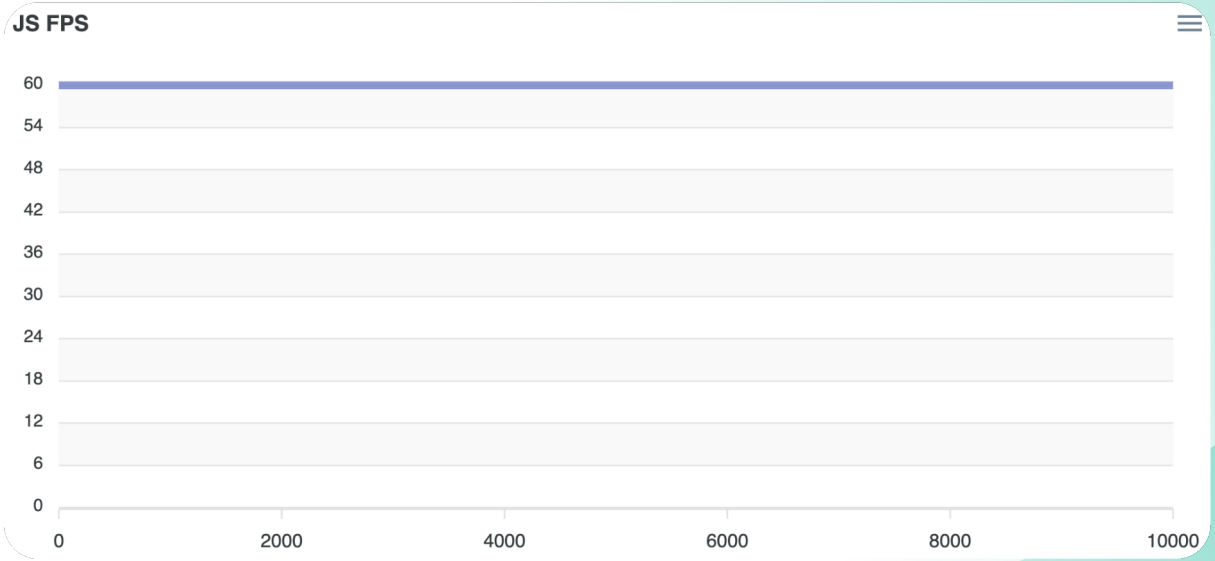
```
34  const toggleMore = () => {
35    'worklet'
36    const newIsHeight = !isHide.value
37    isHide.value = newIsHeight
38    maxHeight.value = newIsHeight ? StartSize : MaxSize
39    toggleText.value = newIsHeight ? 'open' : 'close'
40  }
```

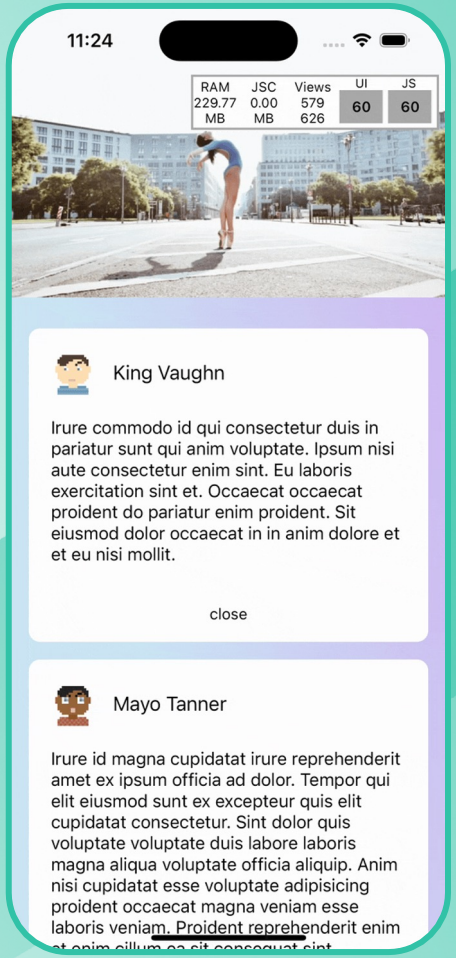
reanimated

```
25  const isHide = useSharedValue(false)
26  const toggleText = useSharedValue(isHide.value ? 'open' : 'close')
27  const maxHeight = useSharedValue(isHide.value ? StartSize : MaxSize)
28  const abooutStyle = useAnimatedStyle(() => ({…
32  }))
33  const toggleMore = () => {…
39  }
40  return (
41    <Animated.View style={[styles.container, abooutStyle]}>
42      <View style={styles.item}>
43        <Avatar seed={name} />
44        <Text style={styles.name}>{name}</Text>
45      </View>
46      <Text style={[styles.text]}>{about}</Text>
47      <GestureDetector gesture={Gesture.Tap().onEnd(toggleMore)}>
48        <View style={styles.button}>
49          <AnimatedText style={{color: 'black'}} text={toggleText} />
50        </View>
51      </GestureDetector>
52    </Animated.View>
53  )
```



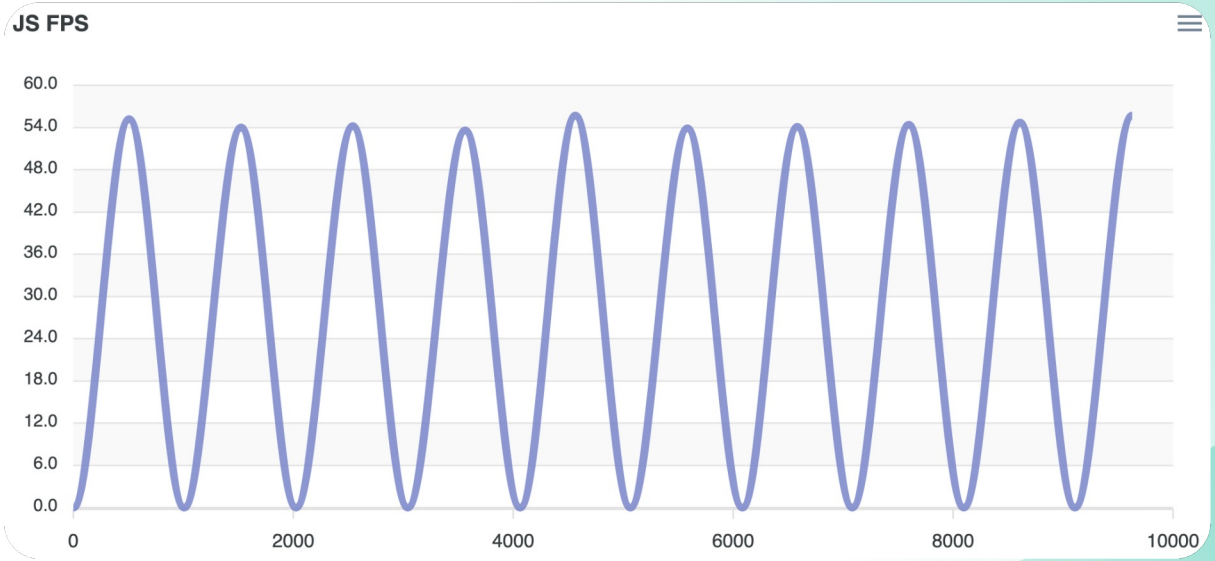
замеры


JS FPS
60
54
48
42
36
30
24
18
12
6
0
0
2000
4000
6000
8000
10000


11:24
RAM
229.77 MB
JSC
0.00 MB
Views
579
626
UI
60
JS
60
King Vaughn
Irure commodo id qui consectetur duis in pariatur sunt qui anim voluptate. Ipsum nisi aute consectetur enim sint. Eu laboris exercitation sint et. Occaecat occaecat proident do pariatur enim proident. Sit eiusmod dolor occaecat in in anim dolore et et eu nisi mollit.
close
Mayo Tanner
Irure id magna cupidatat irure reprehenderit amet ex ipsum officia ad dolor. Tempor qui elit eiusmod sunt ex excepteur quis elit cupidatat consectetur. Sint dolor quis voluptate voluptate duis labore laboris magna aliqua voluptate officia aliquip. Anim nisi cupidatat esse voluptate adipisicing proident occaecat magna veniam esse laboris veniam. Proident reprehenderit enim

tech

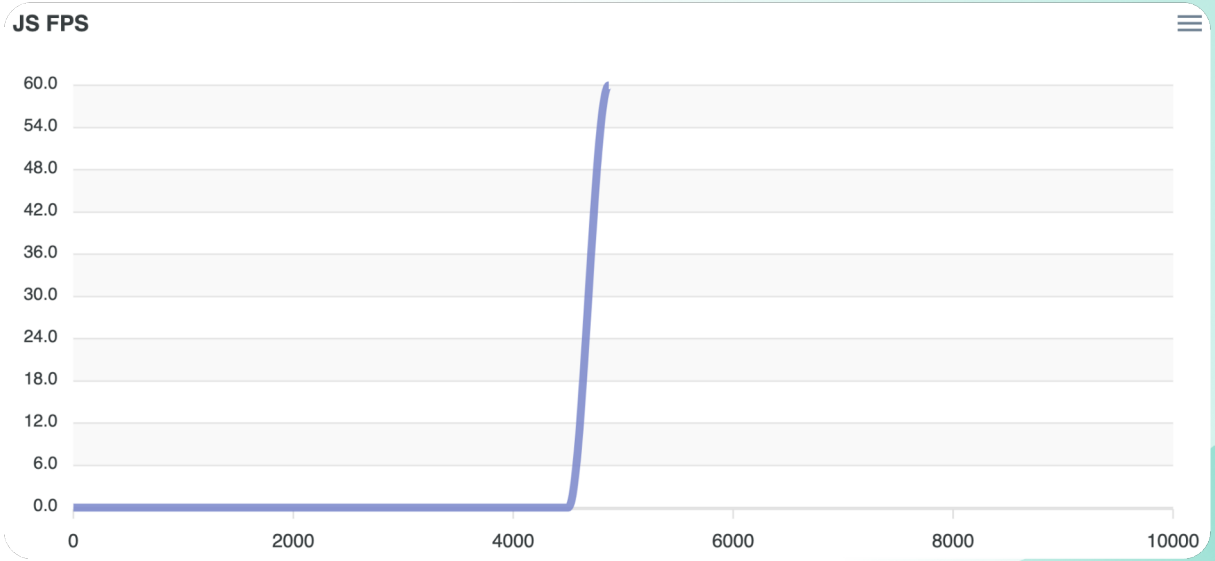
замеры

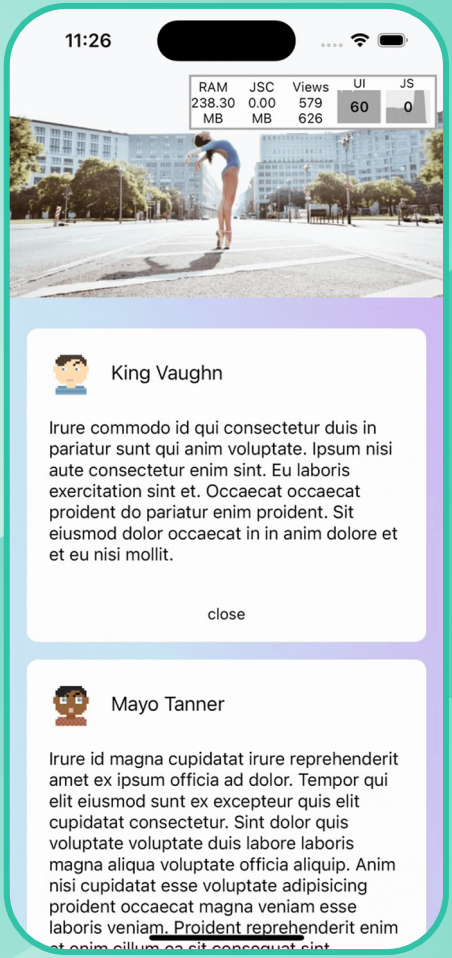

JS FPS
60.0
54.0
48.0
42.0
36.0
30.0
24.0
18.0
12.0
6.0
0.0
0
2000
4000
6000
8000
10000


11:25
RAM
233.93 MB
JSC
0.00 MB
Views
579
626
UI
60
JS
28
King Vaughn
Irure commodo id qui consectetur duis in pariatur sunt qui anim voluptate. Ipsum nisi aute consectetur enim sint. Eu laboris exercitation sint et. Occaecat occaecat proident do pariatur enim proident. Sit eiusmod dolor occaecat in in anim dolore et et eu nisi mollit.
close
Mayo Tanner
Irure id magna cupidatat irure reprehenderit amet ex ipsum officia ad dolor. Tempor qui elit eiusmod sunt ex excepteur quis elit cupidatat consectetur. Sint dolor quis voluptate voluptate duis labore laboris magna aliqua voluptate officia aliquip. Anim nisi cupidatat esse voluptate adipisicing proident occaecat magna veniam esse laboris veniam. Proident reprehenderit enim

tech

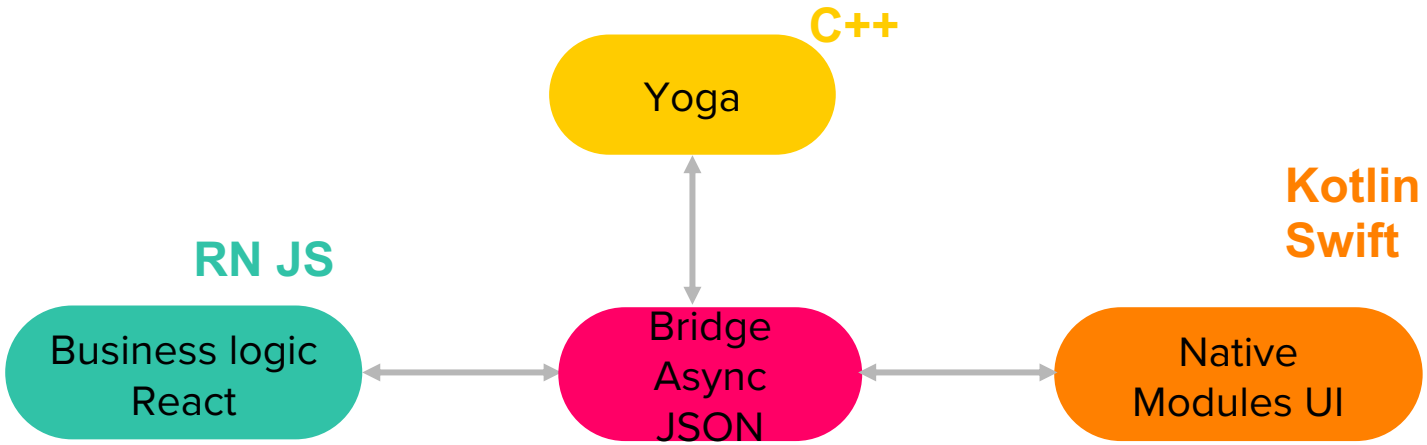
замеры


JS FPS
60.0
54.0
48.0
42.0
36.0
30.0
24.0
18.0
12.0
6.0
0.0
0
2000
4000
6000
8000
10000


11:26
RAM
238.30 MB
JSC
0.00 MB
Views
579
626
UI
60
JS
0
King Vaughn
Irure commodo id qui consectetur duis in pariatur sunt qui anim voluptate. Ipsum nisi aute consectetur enim sint. Eu laboris exercitation sint et. Occaecat occaecat proident do pariatur enim proident. Sit eiusmod dolor occaecat in in anim dolore et et eu nisi mollit.
close
Mayo Tanner
Irure id magna cupidatat irure reprehenderit amet ex ipsum officia ad dolor. Tempor qui elit eiusmod sunt ex excepteur quis elit cupidatat consectetur. Sint dolor quis voluptate voluptate duis labore laboris magna aliqua voluptate officia aliquip. Anim nisi cupidatat esse voluptate adipisicing proident occaecat magna veniam esse laboris veniam. Proident reprehenderit enim

tech

архитектура RN



почему синхронность — это важно?

- Скорость
- Удобство
- Потребление памяти
- Прямое управление UI

tech

Пример синхронного вызова методов в Web

Например, в нативке реализован метод log(), а в JS есть ссылка на этот метод, с помощью console.log мы вызываем этот метод по ссылке

```
> console.log
<· ƒ log() { [native code] }
```

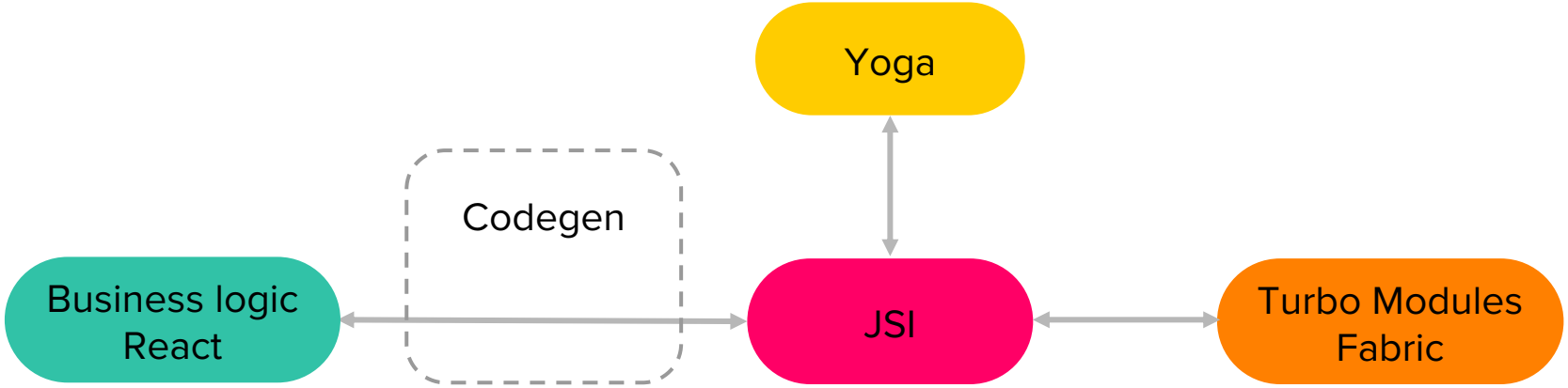
```
> fetch
<· ƒ fetch() { [native code] }
```

JSI делает то же самое для RN

tech

Javascript Interface (JSI)

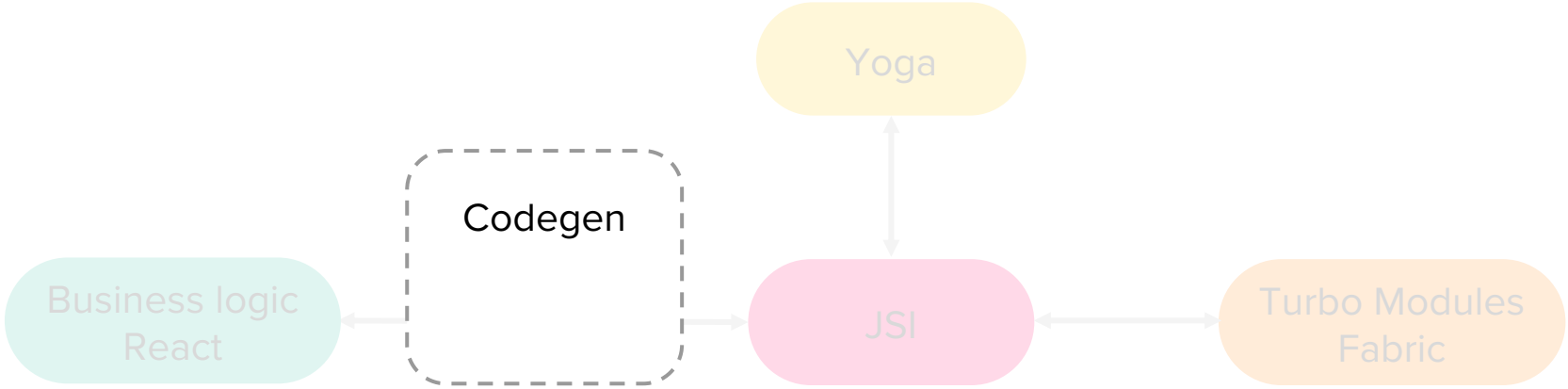
Разработан для устранения некоторых недостатков старого механизма бриджа



Все взаимодействия между потоками **синхронные** в отличие от старого бриджа

Codegen

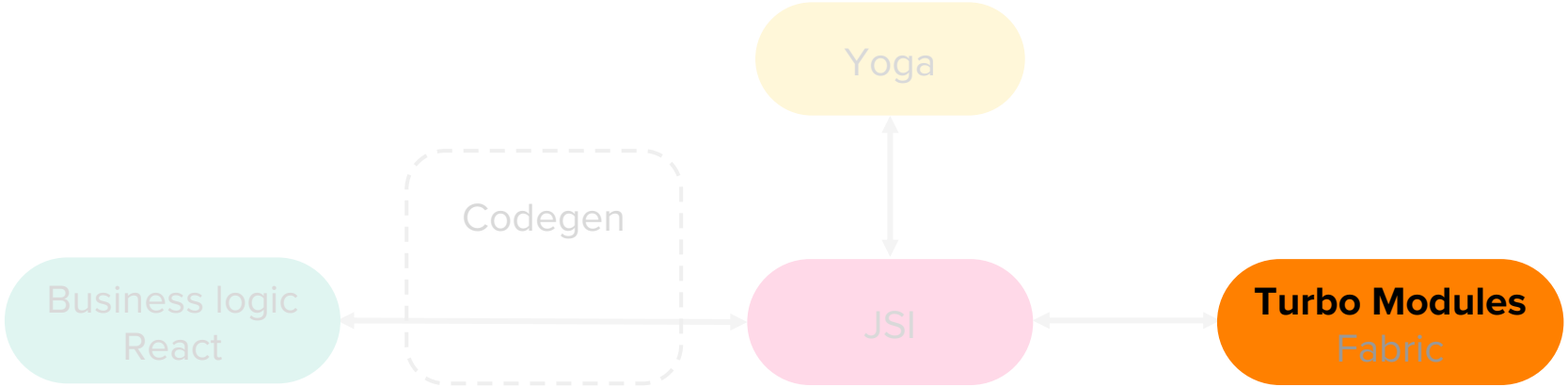
Автоматизирует создание нативных интерфейсов для работы с Turbo Modules и Fabric



Генерируют C++ код для создания типизированных интерфейсов для JS модулей, обеспечивая безопасность и полную совместимость с типами в Turbo Modules

про turbo modules

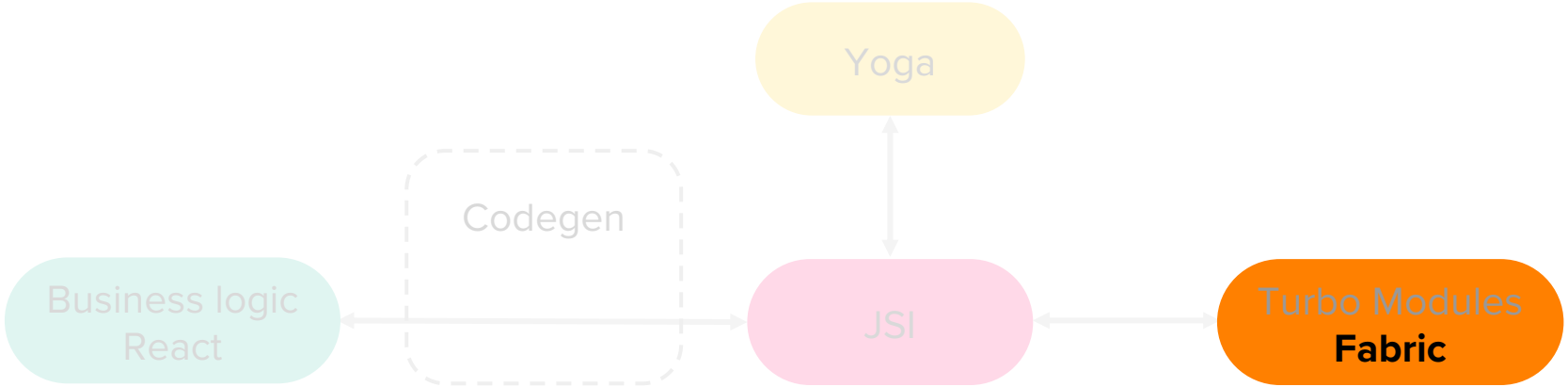
Поддерживают «ленивую загрузку» нативных модулей, чтобы не замедлять запуск апп,
и синхронные вызовы методов из JS



Можно писать кросс-платформенные модули на C++

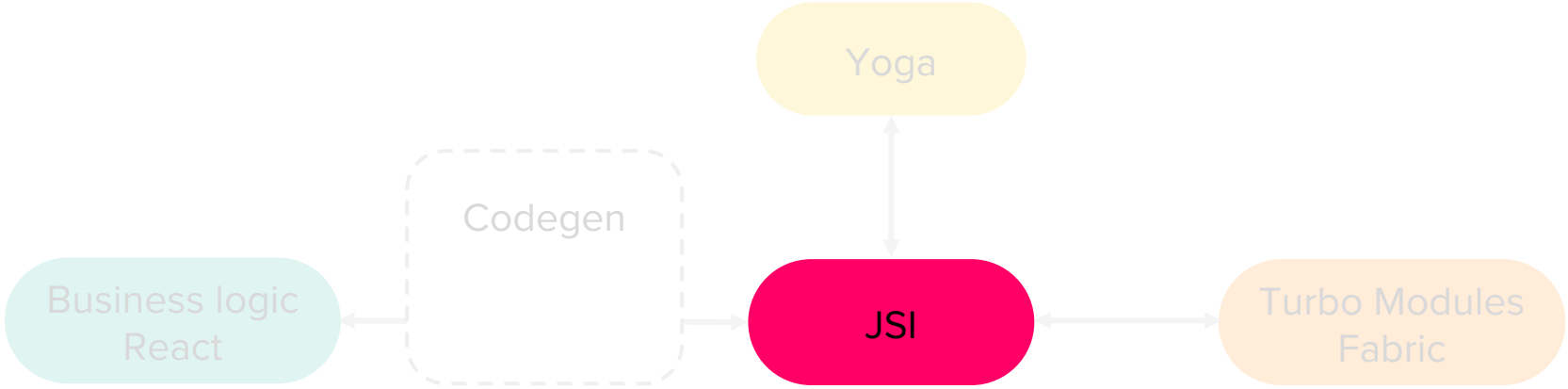
про fabric

Fabric делает работу с UI синхронной и приоритезирует UI задачи над асинхронными вызовами



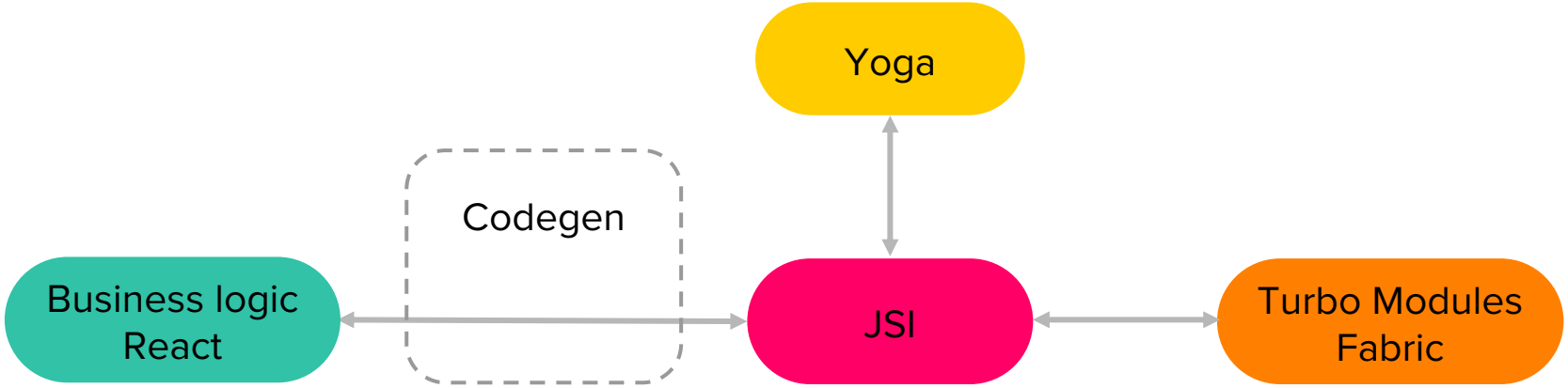
про JSI

Обеспечивает соединение новых модулей и JS за счет создания Host объектов

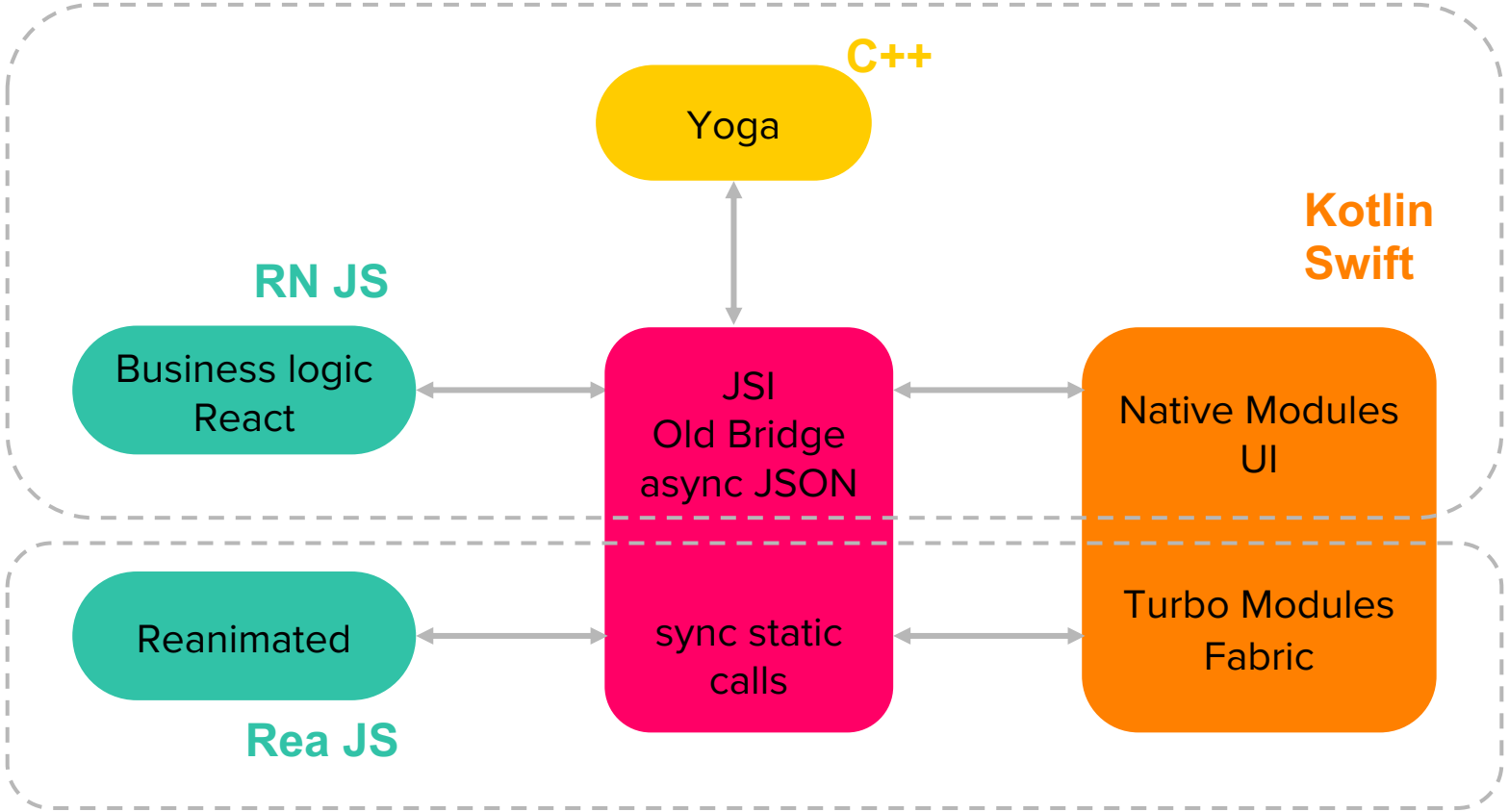


новая архитектура

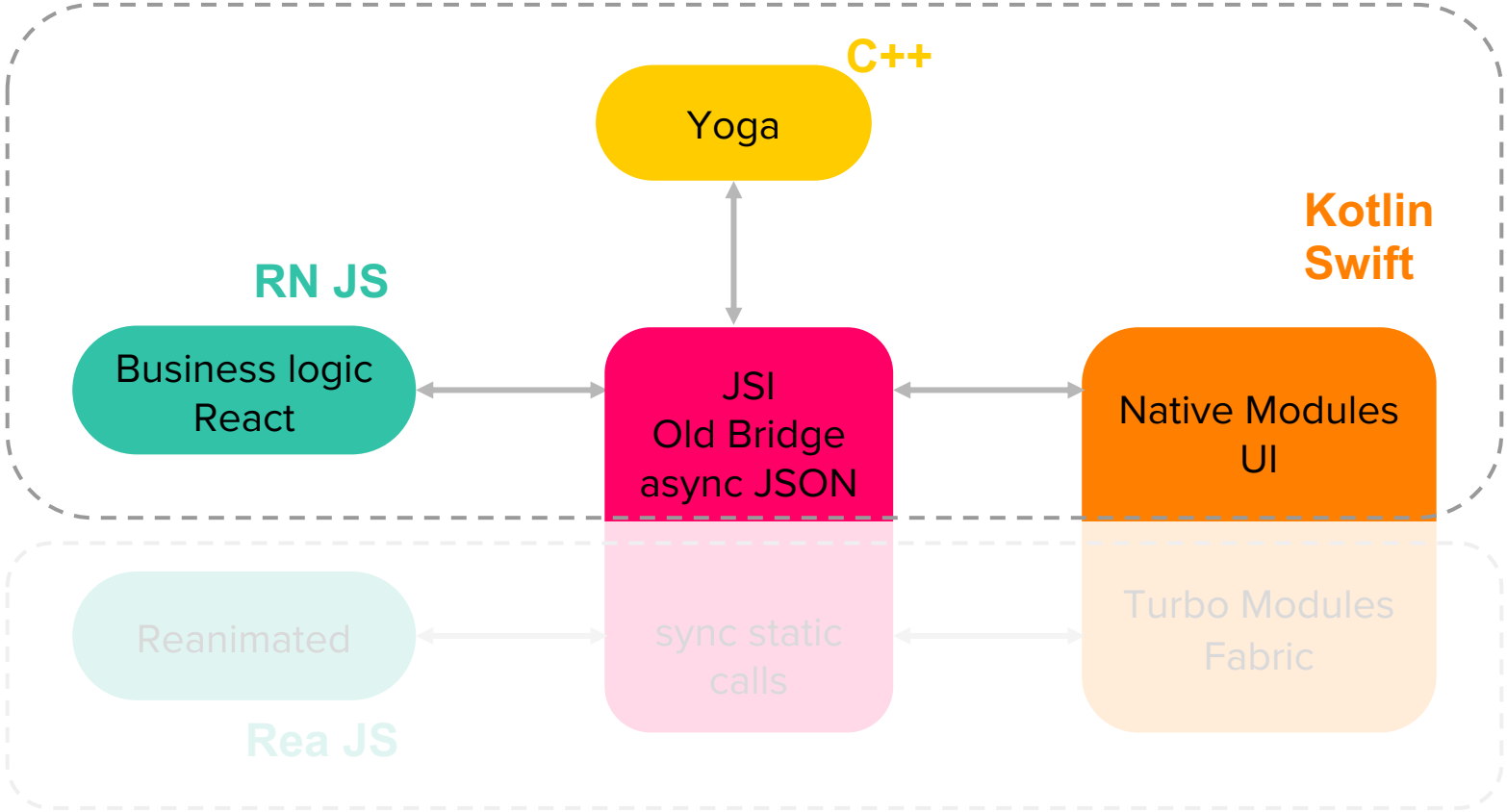
Вызываем нативные методы синхронно. Разные потоки благодаря типам теперь знают друг о друге. Перестали посылать сериализованные строки



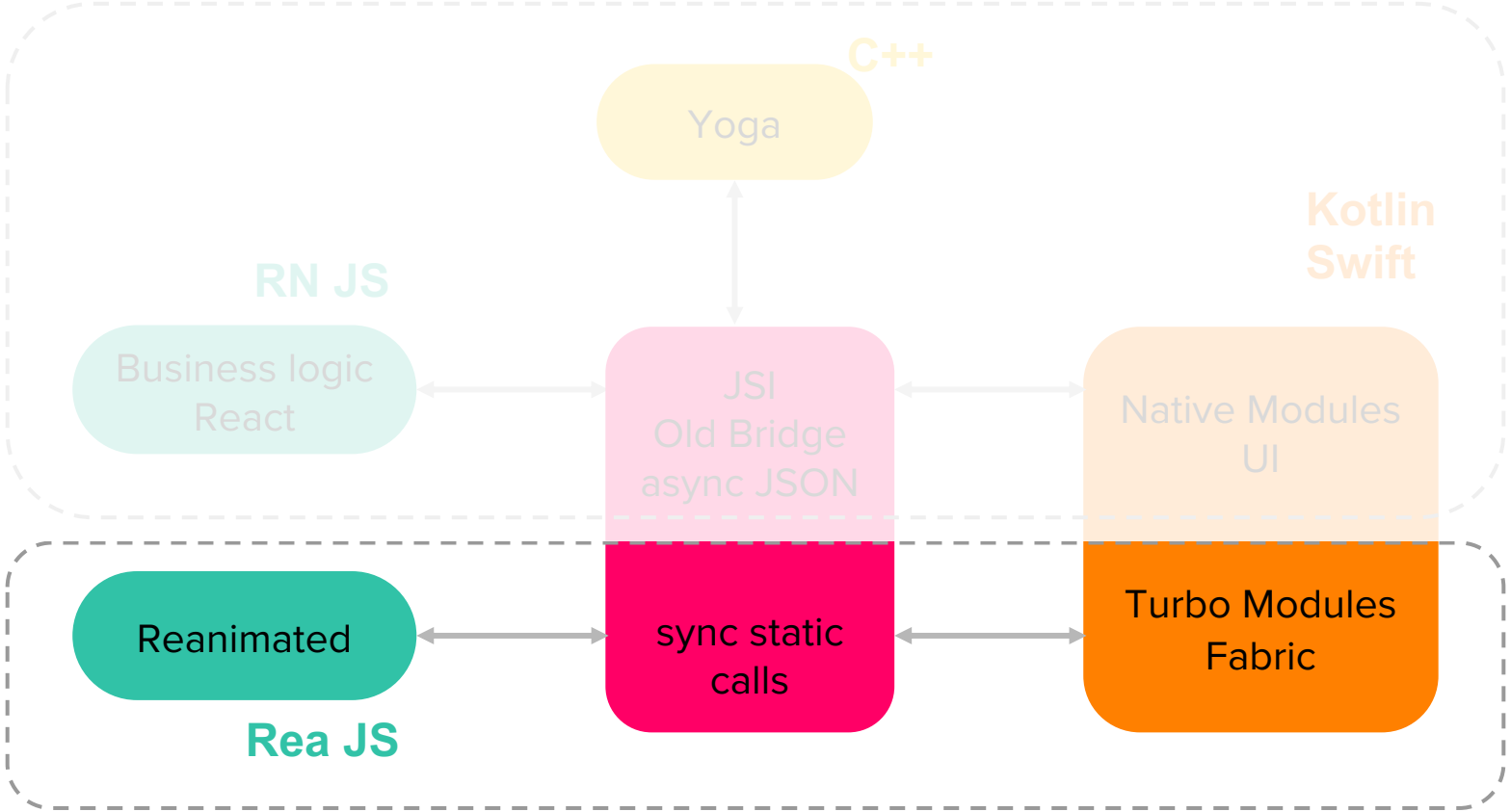
архитектура RN с Reanimated



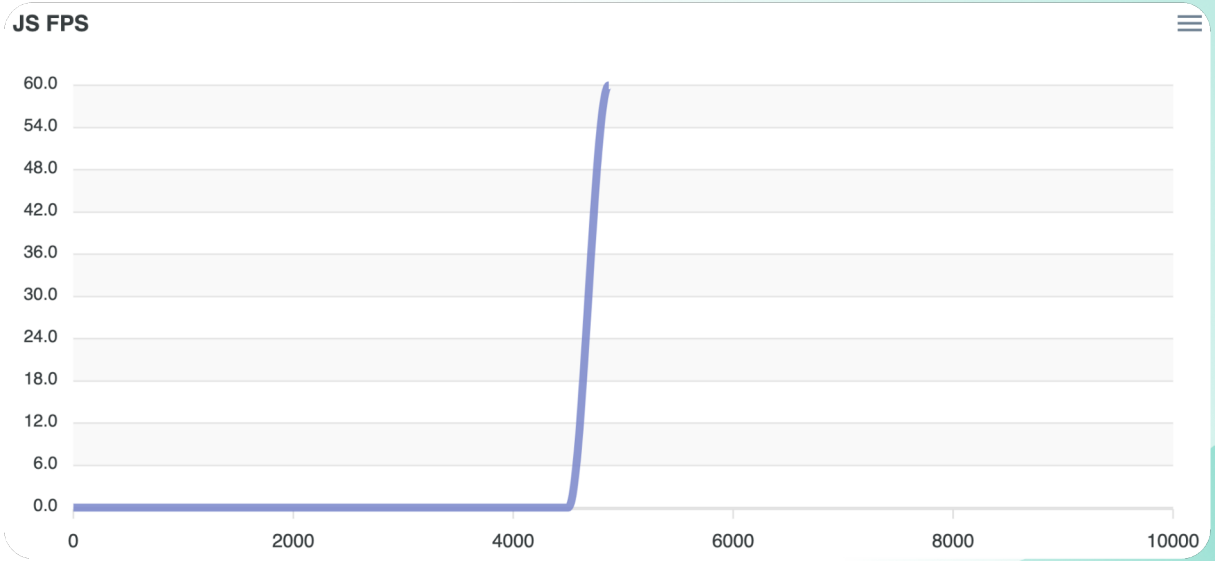
бизнес-логика и React работают асинхронно

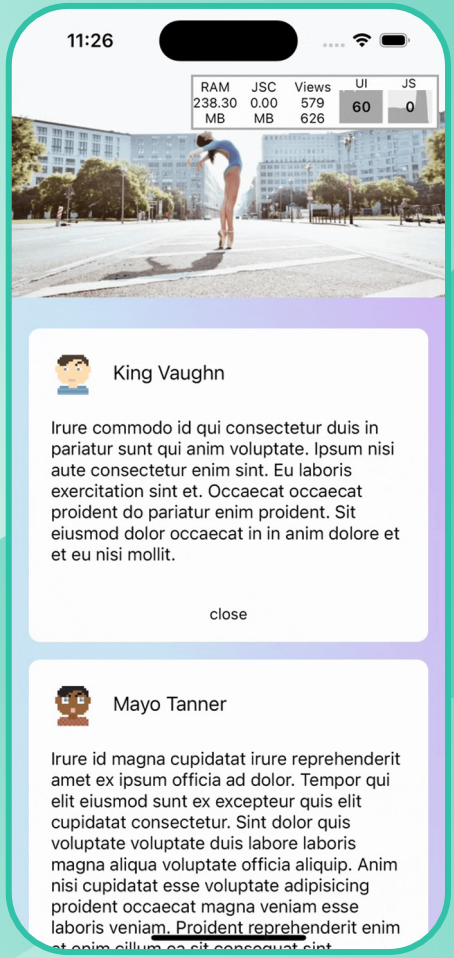


reanimated работает синхронно



замеры


JS FPS
60.0
54.0
48.0
42.0
36.0
30.0
24.0
18.0
12.0
6.0
0.0
0
2000
4000
6000
8000
10000


11:26
RAM
238.30 MB
JSC
0.00 MB
Views
579
626
UI
60
JS
0
King Vaughn
Irure commodo id qui consectetur duis in pariatur sunt qui anim voluptate. Ipsum nisi aute consectetur enim sint. Eu laboris exercitation sint et. Occaecat occaecat proident do pariatur enim proident. Sit eiusmod dolor occaecat in in anim dolore et et eu nisi mollit.
close
Mayo Tanner
Irure id magna cupidatat irure reprehenderit amet ex ipsum officia ad dolor. Tempor qui elit eiusmod sunt ex excepteur quis elit cupidatat consectetur. Sint dolor quis voluptate voluptate duis labore laboris magna aliqua voluptate officia aliquip. Anim nisi cupidatat esse voluptate adipisicing proident occaecat magna veniam esse laboris veniam. Proident reprehenderit enim

tech

наш пример на Reanimated

```
25  const isHide = useSharedValue(false)
26  const toggleText = useSharedValue(isHide.value ? 'open' : 'close')
27  const maxHeight = useSharedValue(isHide.value ? StartSize : MaxSize)
```

наш пример на Reanimated

```
33  const toggleMore = () => {
34    'worklet'
35    const newIsHeight = !isHide.value
36    isHide.value = newIsHeight
37    maxHeight.value = newIsHeight ? StartSize : MaxSize
38    toggleText.value = newIsHeight ? 'open' : 'close'
39  }
```

Reanimated — исследуем границы

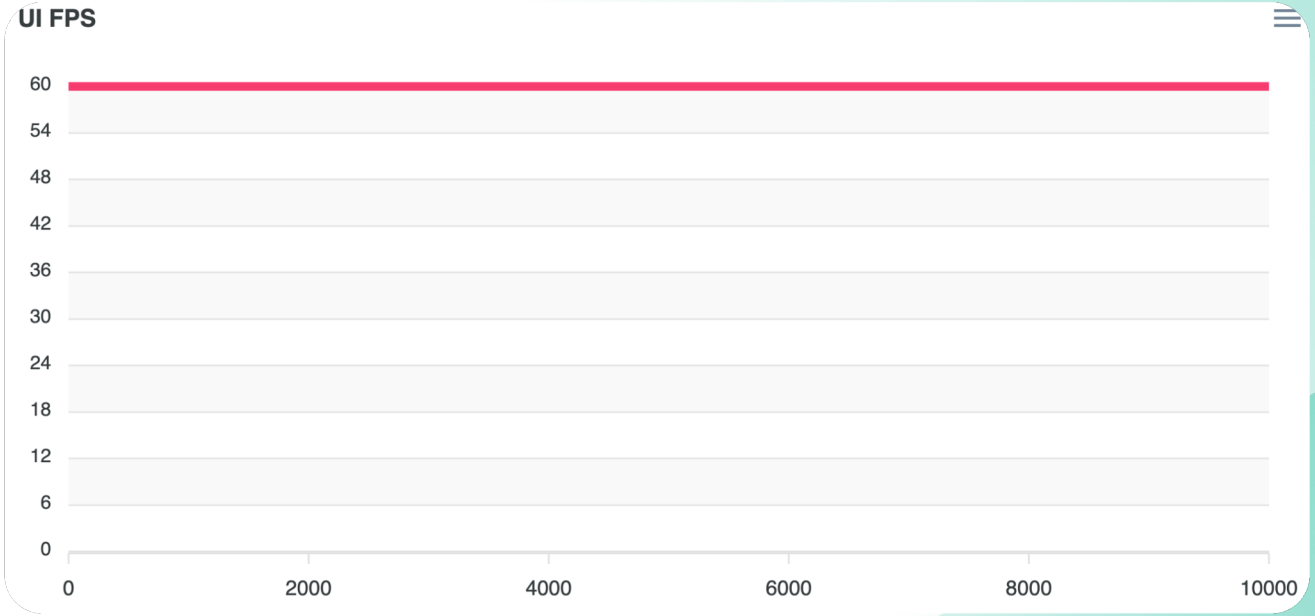
```
28  const endAngle = useSharedValue(generateEndAngle())
29  const size = useSharedValue(40)
30  const translateX = useSharedValue(startValue)
31  const opacity = useSharedValue(0)
32
33  const animStyle = useAnimatedStyle(() => {
34    return {
35      transform: [
36        {
37          rotateZ: `${endAngle.value}deg`,
38        },
39        {
40          translateX: translateX.value,
41        },
42      ],
43      width: size.value,
44      height: size.value,
45      opacity: opacity.value,
46    }
47  })
```

Reanimated — исследуем границы

```
51      size.value = withRepeat(withTiming(20, {duration: 6000}), -1)
52      opacity.value = withDelay(
53        500,
54        withRepeat(
55          withSequence(
56            withTiming(1, {duration: 1500}),
57            withTiming(0, {duration: 4500}),
58          ),
59          -1,
60        ),
61      )
62      translateX.value = withRepeat(
63        withTiming(
64          startValue + Math.abs(Math.sin(endAngle.value) * diffValue),
65          {duration: 6000},
66        ),
67        -1,
68      )
```

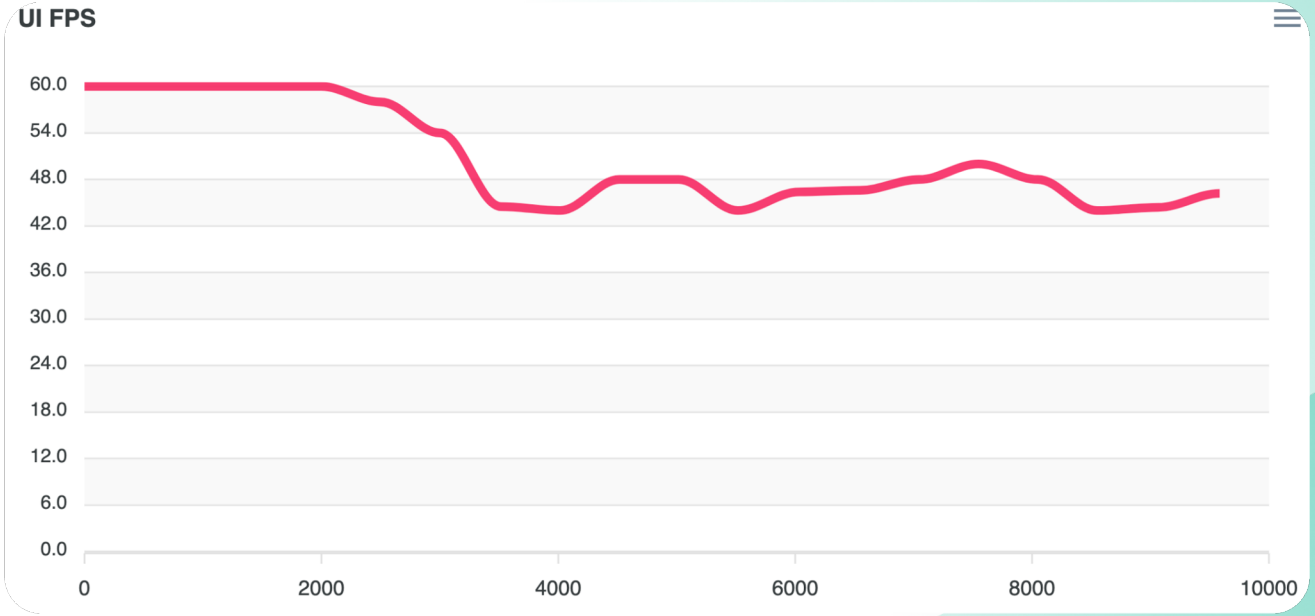
Reanimated — исследуем границы, 10 нод

Reanimated — исследуем границы, 100 нод



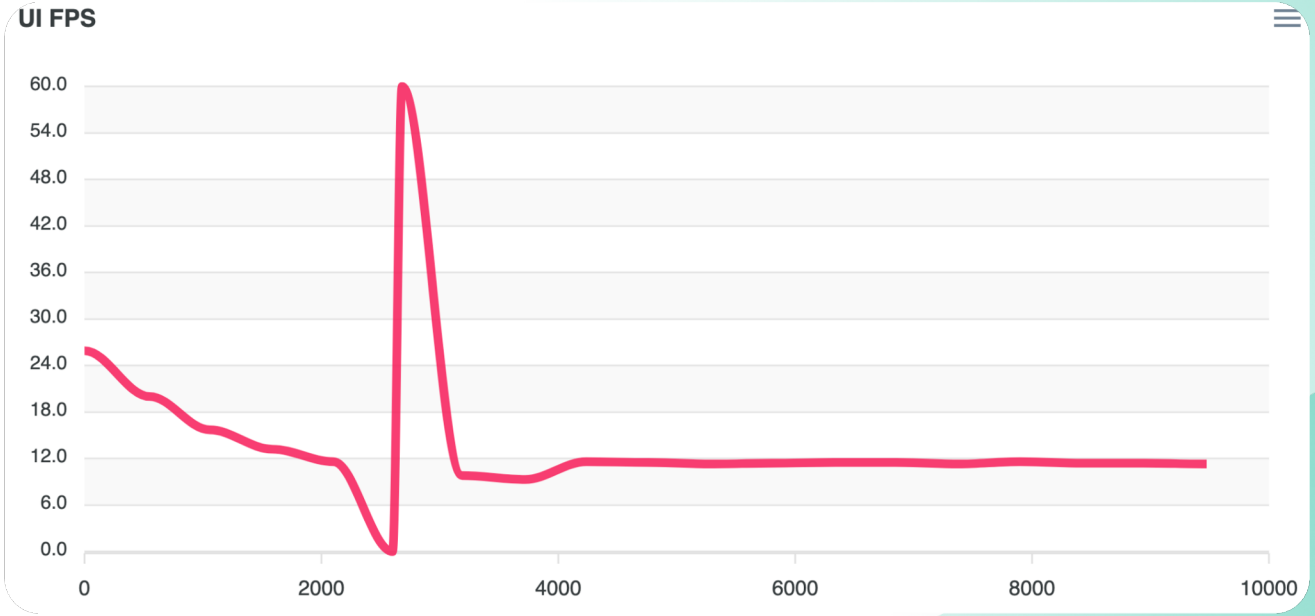
tech

Reanimated — исследуем границы, 300 нод



tech

Reanimated — исследуем границы, 1000 нод



tech

Reanimated — заключение

1. Работает синхронно
2. Управляется из JS, что очень удобно
3. Не загружает основной JS поток лишними вычислениями
4. Решает все проблемы Animated и даже больше, RN хорош)
5. Есть лимит по количеству одновременно анимированных элементов

tech

когда что брать

Animated

1. Для простых анимаций
2. Когда надо сделать анимацию, которая не перерисовывает макет: transform, opacity, borderRadius…
3. Если не хочется разбираться с более трудной библиотекой
4. Всегда надо использовать useNativeDriver: true

Reanimated

1. Если свойства, которые надо анимировать перерисовывают макет
2. Если надо реализовать сложную анимацию, где разные свойства зависят друг от друга
3. Когда используем жесты

что можно сделать сейчас

Animated → **useNativeDrive: true**

Вопросы

tech