React18 Update

with React Server Components

2022/05/27

はじめに

- Reactはおかげさまで2022/03/29にv18.0が正式リリースされました。
  （実は2022/05/22現時点ではさらにバージョンアップがなされ、2022/04/26に18.1.0がリリースされています。）

- というわけで今回はこの喜ぶべきReact v18.0について新機能を見ていこうと思います。

- ベースはreact conf 2021としています。
  正直ここでのスライドがめちゃくちゃわかりやすかったので、もはやまとめ直す必要すらないのですが、さらに噛み砕いてみます。
  （驚くべきことに字幕の日本語がきちんと設定されているので読みやすいです。）

はじめに

- ただし、React18のUpdateを正しく理解するためには、React Server Componentsも理解してないとあまり意味がないので、その部分についても軽く解説します。

目次

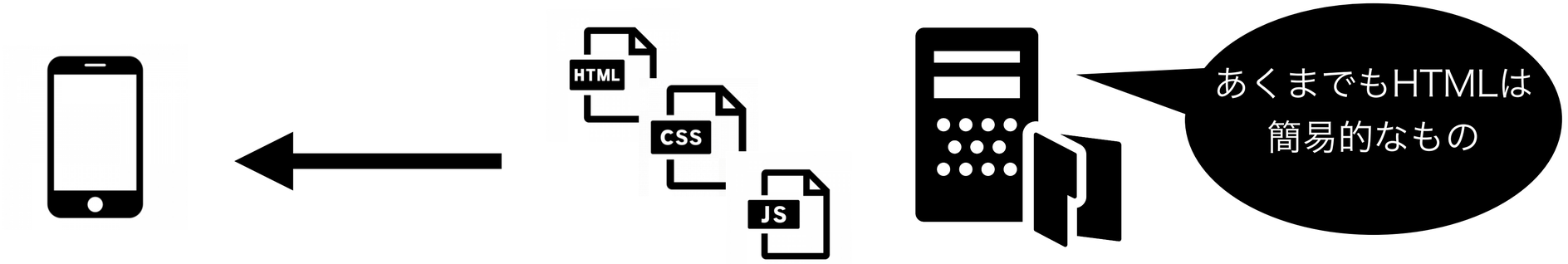
- React Server Components
- React18
  - Suspense
  - 自動バッチ
  - Upgrade方法

React Server Components

歴史

- Reactの最初の出発地点はクライアントにてレンダリングを行うものでありました。（従来はサーバ側でコンテンツを作り、クライアント側は表示するだけ）

- この動作に注目することによって”SPA”と言えるものでありました。要はサーバからはSinglePageApplicationとして一塊のHTML/CSS/Javascriptを渡しクライアント側でそれを使ってアプリケーションのような振る舞いをするものです。

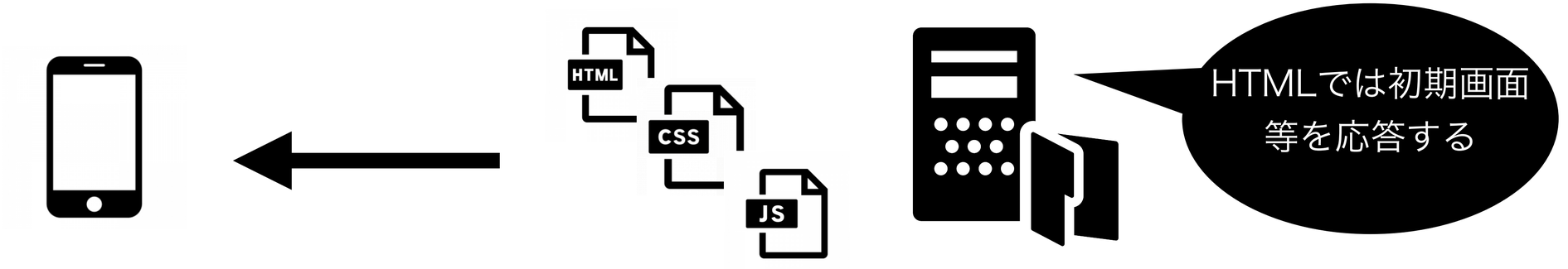


React Server Components

歴史

- SPAにおいて、いくつかの問題点が指摘されていました。例えば、クライアント側でのレンダリングは非常にコストが高く、処理が遅くなったり、データの取得をAPI等で行うため通信回数が多い等です。

- これを克服するために、サーバサイドレンダリング（SSR）という仕組みも考えられ、ある程度のコンテンツはサーバ側で作成して渡すという手法も考えられました。



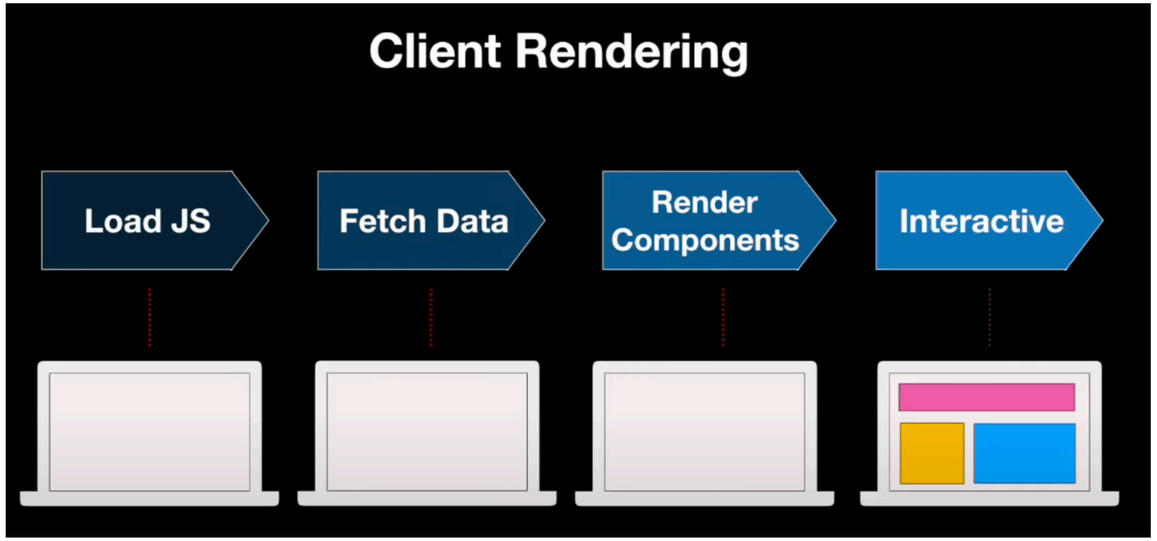
React Server Components

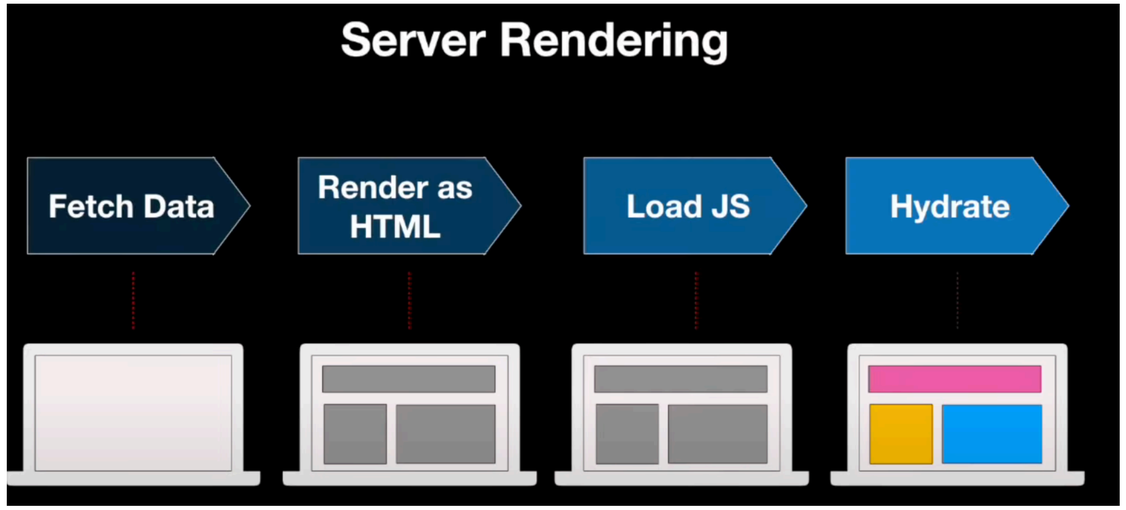
仮想DOMとSSR

- SSRも万能ではなく、SSR/CSRそれぞれに以下の特色があり、その両方の影響を受けることになります。
  - SSR：ただのHTMLを返すのみであり操作とかはできません。また、データ全体を読み込むまで、画面は真っ白になってしまいます。
  - CSR：何かを表示する前に、全てロードする必要があります。
- 要は、SSRで描画を早くできる、のですが、CSRの制約で全てロードするまで操作等（インタラクティブ）はできません。

React Server Components

仮想DOMとSSR

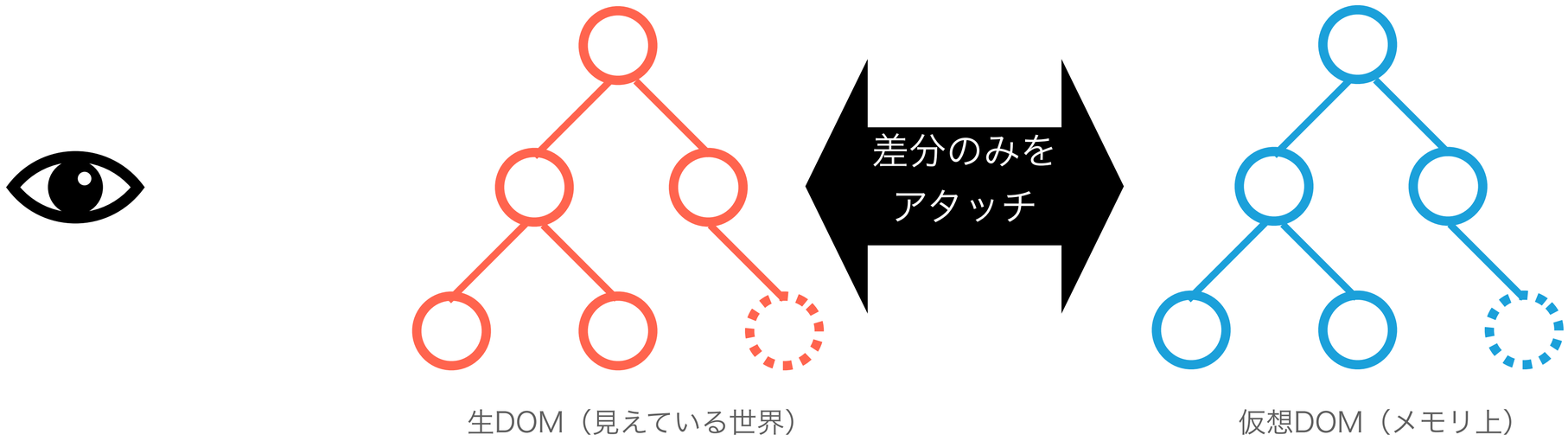




React Server Components

仮想DOMとSSR

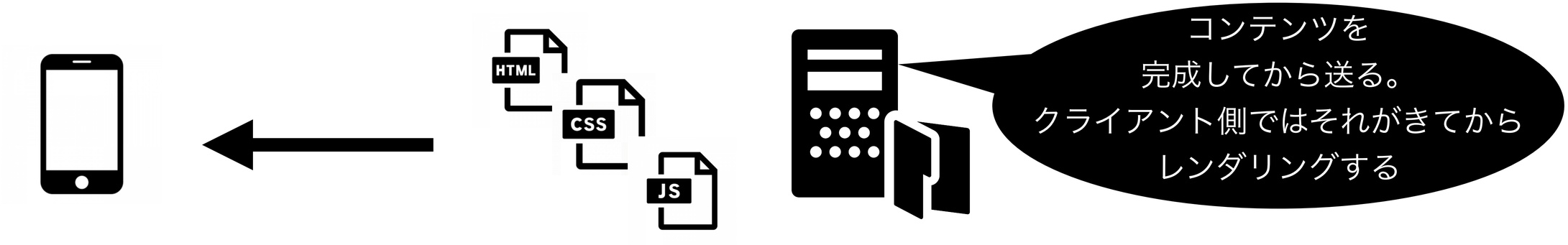
- ここでReactのキモである仮想DOMについても少し触れておきますと、「なぜ仮想DOMという概念が俺達の魂を震えさせるのか」でも話題になっていた通り、実際に見えているDOMと、それに対応する仮想DOMというという仕組みによって技術的革新が得られたのです。



React Server Components

仮想DOMとSSR

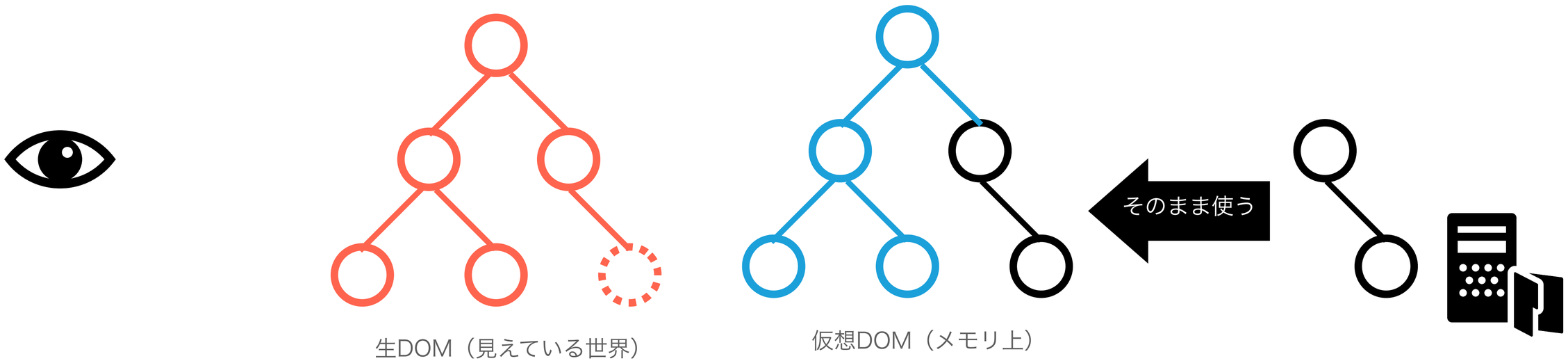
- SSRとは、すなわちページそのものを生成する技術であり、ページ全体の応答という意味では時間がかかります。
- ここで重要なのは、サーバ側でページを作って渡す、という部分で、クライアントの描画との橋渡し、ではありません。あくまで地盤を渡している、というイメージです。



React Server Components

React Server Components

- そこで、SSRとは異なる形で、ReactServerComponents(RSC)という考え方が出てきました。これは、サーバ側でコンポーネントを作り、そのコンポーネントをクライアント側でもそのまま使えるようにする、という技術です。



React Server Components

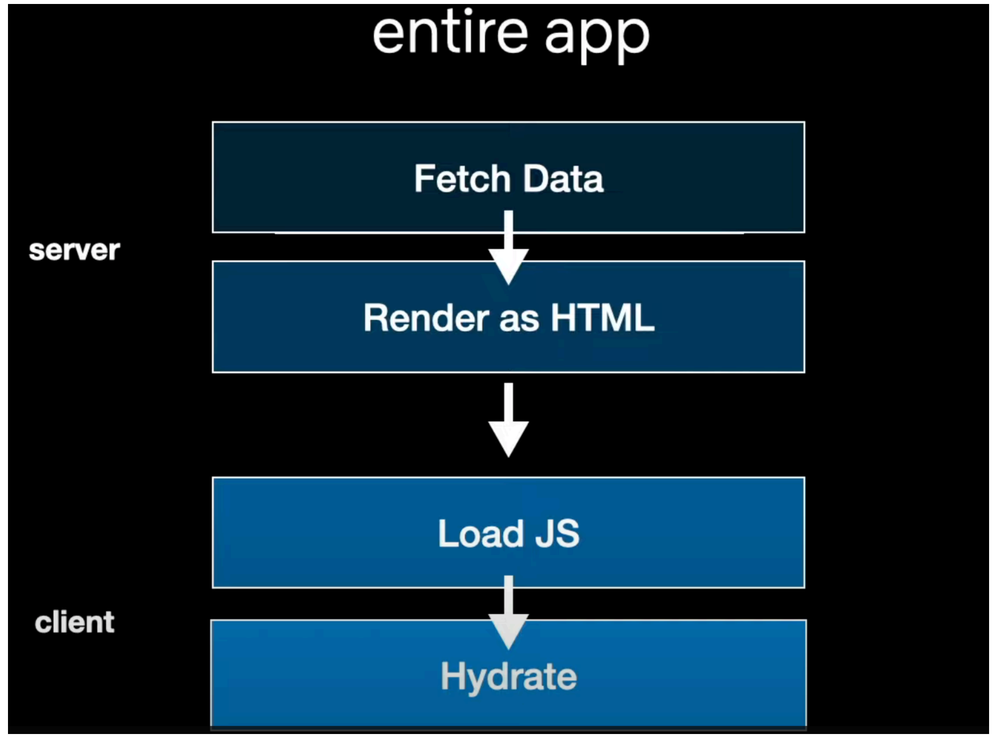
React Server Components

- メリットはいくつかあります。
- バンドルサイズ削減→高速化
  - 処理として時間のかかるものをサーバ側でコンポーネントを作ることによって、クライアント側のコードからは無くすことができます。
- DB等の処理
  - 今まではサーバ側の処理を書くことがなかったため、できなかったような、DBやローカルファイル（サーバ側）へのアクセスが可能になります。

React18

Suspence

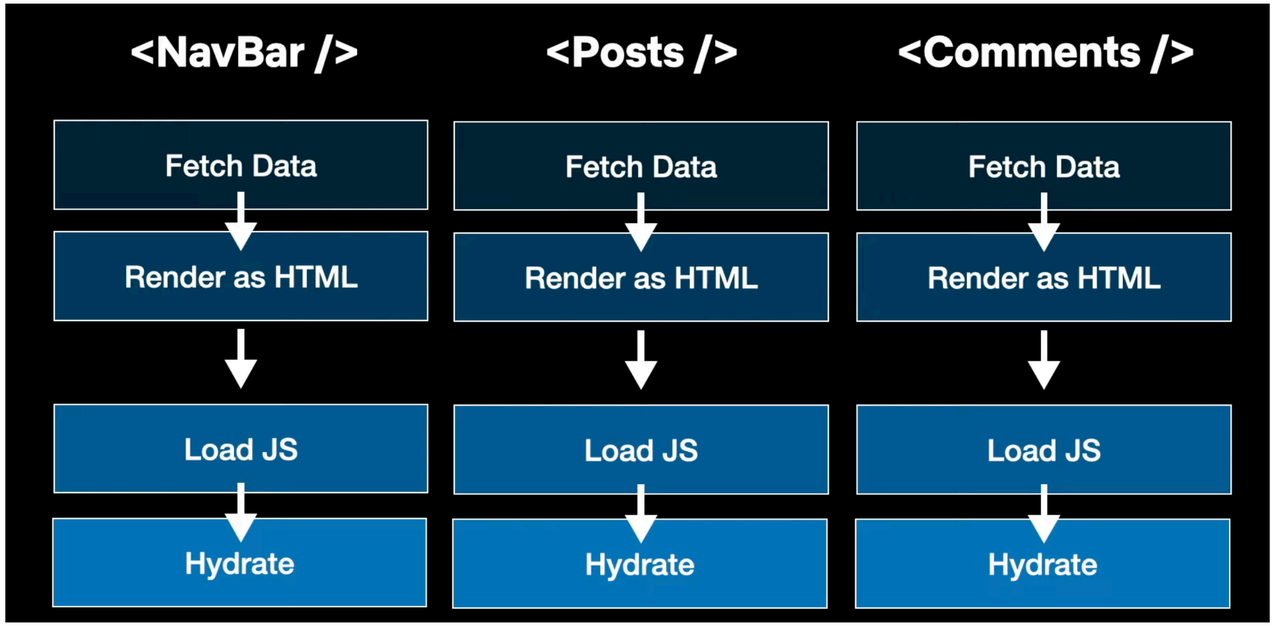
- RSCの登場でサーバ側でもコンポーネントが作れるようになりました。ただし、その場合でも、SSR/CSRの制約は残ります。つまり、SSRで素早くコンポーネントを作ることができても、CSRを行い、それが使える状態になるまでには、どうしても時間が必要でした（全てが出揃わないとインタラクティブとして扱うことができないのです。）



React18

Suspence

- React18からはこれをコンポーネント単位で行うことができるようになりました。
- Suspenseで囲むことによって、それぞれの部分として分離することができるようになったのです。




```
<Suspense fallback={<Spinner />}>
  <AnyComponent />
</Suspense>
```

React18

Suspence

- これによって、以下の問題を解決し、ユーザビリティ向上につながります。
  - fetch everything show anything：必要なものを全て取ってきてから表示するというところから、後からサーバから受け取ることができます。
  - load everything hydrate anything：全てを読み込んでから使えるようにするというところも、使えるところから使えるようにすることができる。
- あとは選択的ハイドレーションという仕組みで、カーソルが当たっている部分から優先的に処理することもできるようになりました。

React18

automatic batching

- Reactは高速化のために、イベントハンドラとかの関数の中では、ステート変更の処理は関数の最後に実施されていました。

- ただ、イベントハンドラ以外の関数の中では再レンダリングが毎回行われていました。これがReact18では、この関数であっても再レンダリングが関数の最後となりました。
※ただし、挙動が変わるので要注意。

```
function handleClick() {
  setIsFetching(false);
  setError(null);
  setFormStatus('success');
}
```

re-rendered 1 times

```
fetch('/something').then(() => {
  setIsFetching(false);
  setError(null);
  setFormStatus('success');
});
```

re-rendered 3 times

```
fetch('/something').then(() => {
  setIsFetching(false);
  setError(null);
  setFormStatus('success');
});
```

re-rendered 1 times

React18
Upgrade

- Upgradeもとても簡単で以下のステップだけで完了です。
  - インストール： npm install react react-dom
  - レンダリングの変更：App.jsの以下を修正
    - 修正前：render(<App tab=“home” />,container);
    - 修正後：root.render(<App tab=“home” />;

React18

おわりに

- 今回のUpdateを見ると、まずはUX改善が軸になっているなと感じます。（コードの書き方、というよりはスピードを始めとしたユーザ体験）
- とはいえ、RSCのようにフロントエンドとしての領域の拡張もあったりとかして、これからも進化は続いていくんだなと強く感じます。
- UpdateについてもReactとしてはとても大切にされており、Upgradeは容易であるものの、下位互換性もきちんと確保されている感じが伝わってきて開発者に寄り添っているなと感じます。（Upgradeしても壊れない、というのが繰り返されていたのが印象的でした。）

React18

おわりに

- 今回紹介できていないだけでもっと多くの機能追加がなされていますので、ぜひぜひ見てみてください（ConcurrentやTransitionやuseldなどの新しいフック）
- 全体的に去年試験導入されたものが正式に導入されたという感じがします。このことからもカンファレンスに出たり、新しい機能をどんどん試していくことの重要性（新規技術へのキャッチアップ）を改めて感じました。
- Reactをはじめフロントエンドは進化の速度が早いですね！頑張ってこれからも追いついていきましょう！

引用

- React 18 for app developers
  - https://www.youtube.com/watch?v=ytudH8je5ko
- Streaming Server Rendering with Suspense
  - https://www.youtube.com/watch?v=pj5N-Khihgc
- UpgradeGuide
  - https://ja.reactjs.org/blog/2022/03/08/react-18-upgrade-guide.html