Батчинг в React’е

Чтобы лучше понять, что такое батчинг, давайте рассмотрим пример

```
const [country, setCountry] = useState('');
const [language, setLanguage] = useState('');

const onClick = (newCountry, newLanguage) => {
  setCountry(newCountry);
  setLanguage(newLanguage);
}
```

```
const [country, setCountry] = useState('');
const [language, setLanguage] = useState('');

const onClick = (newCountry, newLanguage) => {
  setCountry(newCountry);
  setLanguage(newLanguage);
}
```

1

- 2 вызова setState
- 1 рендер

```
const onClick = async (formData) => {
  setLoading(true);

  await createDeal(formData)

  setLoading(false);
}
```

```
const onClick = async (formData) => {
  setLoading(true);

  await createDeal(formData)

  setLoading(false);
}
```

2

В чем проблема ?

```
const onClick = async (formData) => {
  setLoading(true);

  await createDeal(formData)

  setLoading(false);
}
```

Памятка

- setTimeout
- setInterval
- Promise
- XMLHttpRequest

- 2 вызова setState
- 2 рендера
- await меняет логику

```
const onClick = async (formData) => {
  setLoading(true);
  setSubmitButtonDisabled(true);

  const deal = await createDeal(formData)

  setDeal(deal);
  setSubmitButtonDisabled(false);
  setLoading(false);
}
```

v17: 4

v18: 2

- 5 вызовов setState
- в разных версиях разное количество рендеров
- await меняет логику

Как это группируется ?

V17.0.2

```
const onClick = async (formData) => {

  setLoading(true);
  setSubmitButtonDisabled(true);

  const deal = await createDeal(formData);

  setDeal(deal);

  setSubmitButtonDisabled(false);

  setLoading(false);

}
```

4

V18.2.0

```
const onClick = async (formData) => {

  setLoading(true);
  setSubmitButtonDisabled(true);

  const deal = await createDeal(formData);

  setDeal(deal);
  setSubmitButtonDisabled(false);
  setLoading(false);

};
```

2

Давайте немного подебажим

```
const onClick = async (formData) => {

    sleep(3000);
    setLoading(true);

    sleep(3000);
    setSubmitButtonDisabled(false);

    const deal = await createDeal(formData);

    sleep(3000);
    setSubmitButtonDisabled(true);

    sleep(3000);
    setDeal(deal);

    sleep(3000);
    setLoading(false);

};
```

```javascript
const sleep = (duration) => {
    const start = new Date().getTime();
    let end = start;

    while (end < start + duration) {
        end = new Date().getTime();
    }
};
```

```
const onClick = async (formData) => {

    sleep(3000);
    setLoading(true);

    sleep(3000);
    setSubmitButtonDisabled(false);

    const deal = await createDeal(formData);

    sleep(3000);
    setSubmitButtonDisabled(true);

    sleep(3000);
    setDeal(deal);

    sleep(3000);
    setLoading(false);

};
```

```
const onClick = async (formData) => {

    sleep(3000);
    setLoading(true);

    sleep(3000);
    setSubmitButtonDisabled(false);

    const deal = await createDeal(formData);

    sleep(3000);
    setSubmitButtonDisabled(true);

    sleep(3000);
    setDeal(deal);

    sleep(3000);
    setLoading(false);

};
```

Вот это и есть батчинг

Что поменялось в 18 версии ?

Automatic batching for fewer renders in React 18 #21

gaearon on 28 May 2021 · 11 comments · 35 replies

```
      // React 17 and earlier does NOT batch these because
      // they run *after* the event in a callback, not *during* it
      setCount(c => c + 1); // Causes a re-render
      setFlag(f => !f); // Causes a re-render
    });
  }

  return (
    <div>
      <button onClick={handleClick}>Next</button>
      <h1 style={{ color: flag ? "blue" : "black" }}>{count}</h1>
    </div>
  );
}
```

- 🟡 Demo: React 17 does NOT batch outside event handlers. (Notice two renders per click in the console.)

Until React 18, we only batched updates during the React event handlers. Updates inside of promises, setTimeout, native event handlers, or any other event were not batched in React by default.

What is automatic batching?

Starting in React 18 with `createRoot`, all updates will be automatically batched, no matter where they originate from.

This means that updates inside of timeouts, promises, native event handlers or any other event will batch the same way as updates inside of React events. We expect this to result in less work rendering, and therefore better performance in your applications:

```
// Before
import { render } from 'react-dom';
const container = document.getElementById('app');
render(<App tab="home" />, container);

// After
import { createRoot } from 'react-dom/client';
const container = document.getElementById('app');
const root = createRoot(container); //
createRoot(container!) if you use TypeScript
root.render(<App tab="home" />);
```

Определение

Батичнгом в React’е называют процесс группировки нескольких вызовов обновления состояния в один этап рендера. Другими словами, пакетная обработка.

Финальный раунд: Сколько будет рендеров ?

```
const onClick = async (formData) => {
    setLoading(true);
    setSubmitButtonDisabled(true);

    const deal = await createDeal(formData);

    setDeal(deal);
    setDealCounter((prevDeal) => prevDeal + 1);

    const deal2 = await createDeal2(formData);

    setDeal2(deal2);
    setDealCounter((prevDeal) => prevDeal + 1);
    setSubmitButtonDisabled(false);
    setLoading(false);
}
```

v17: 7

v18: 3

V18.2.0

```
const onClick = async (formData) => {
    setLoading(true);
    setSubmitButtonDisabled(true);

    const deal = await createDeal(formData);

    setDeal(deal);
    setDealCounter((prevDeal) => prevDeal + 1);

    const deal2 = await createDeal2(formData);

    setDeal2(deal2);
    setDealCounter((prevDeal) => prevDeal + 1);
    setSubmitButtonDisabled(false);
    setLoading(false);
}
```

3

V17.0.2

```
const onClick = async (formData) => {
    setLoading(true);
    setSubmitButtonDisabled(true);

    const deal = await createDeal(formData);

    setDeal(deal);
    setDealCounter((prevDeal) => prevDeal + 1);

    const deal2 = await createDeal2(formData);

    setDeal2(deal2);
    setDealCounter((prevDeal) => prevDeal + 1);
    setSubmitButtonDisabled(false);
    setLoading(false);
}
```

7

```
import { unstable_batchedUpdates } from "react-dom";
```

V17.0.2

```
const onClick = async (formData) => {
  setLoading(true);
  setSubmitButtonDisabled(false);

  const deal = await createDeal(formData);

  unstable_batchedUpdates(() => {
    setDeal(deal);
    setDealCounterCounter((prevDeal) => prevDeal + 1);
  });

  const deal2 = await createDeal(formData);

  unstable_batchedUpdates(() => {
    setDeal2(deal2);
    setDealCounterCounter((prevDeal) => prevDeal + 1);
    setSubmitButtonDisabled(true);
    setLoading(false);
  });
};
```

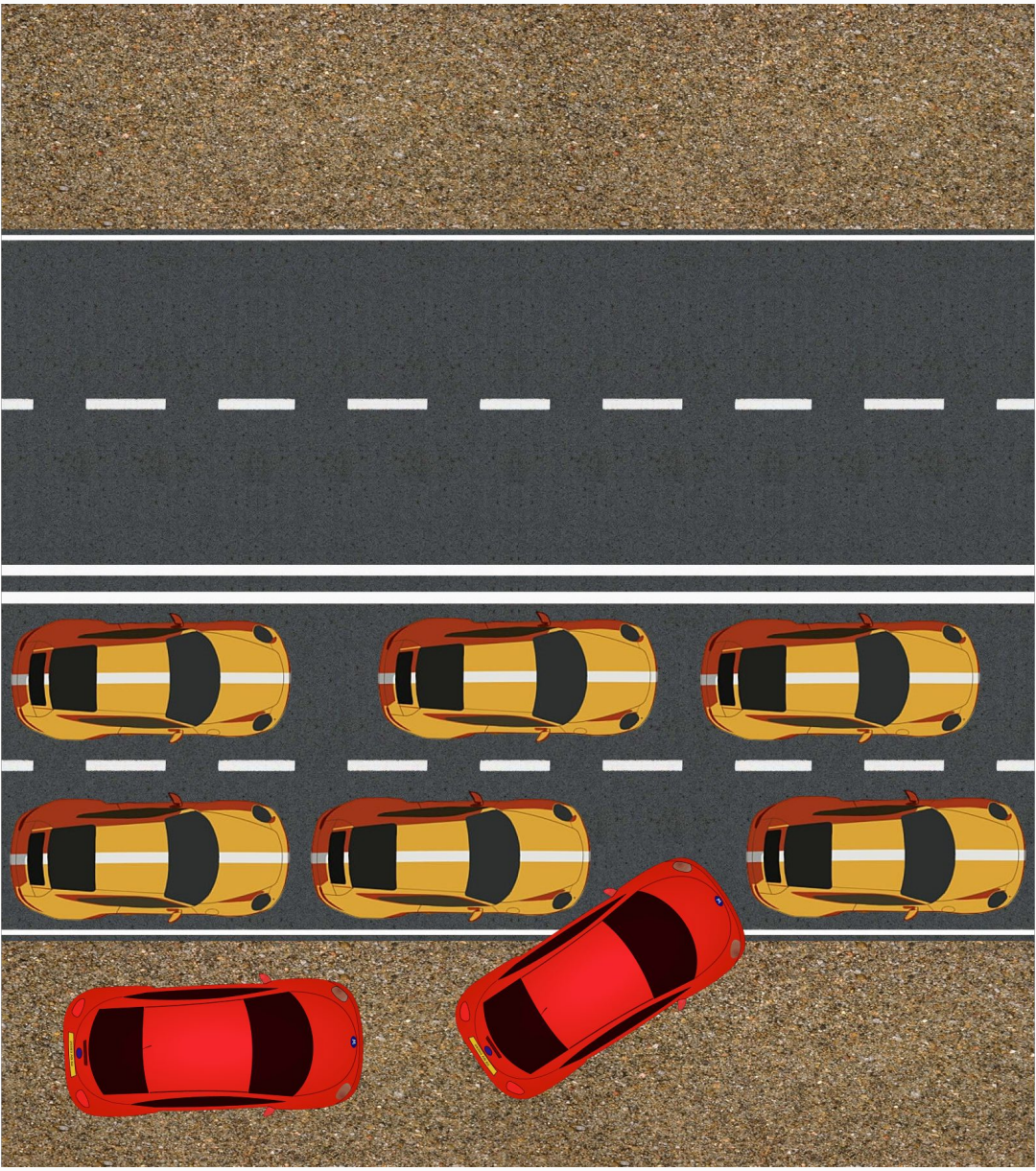
3

Как работает unstable_batchUpdates ?

main react / packages / react-dom / src / client / ReactDOM.js / Jump to

```
192  export {
193    createPortal,
194    batchedUpdates as unstable_batchedUpdates,
195    flushSync,
```

```
1569  export function batchedUpdates<A, R>(fn: A => R, a: A): R {
1570    const prevExecutionContext = executionContext;
1571    executionContext |= BatchedContext;
1572    try {
1573      return fn(a);
1574    } finally {
1575      executionContext = prevExecutionContext;
1576      // If there were legacy sync updates, flush them at the end of the outer
1577      // most batchedUpdates-like method.
1578      if (
1579        executionContext === NoContext &&
1580        // Treat `act` as if it's inside `batchedUpdates`, even in legacy mode.
1581        !(__DEV__ && ReactCurrentActQueue.isBatchingLegacy)
1582      ) {
1583        resetRenderTimer();
1584        flushSyncCallbacksOnlyInLegacyMode();
1585      }
1586    }
1587  }
```



Это конечно хорошо, а можно реальный пример

Да, можно !

17 Версия

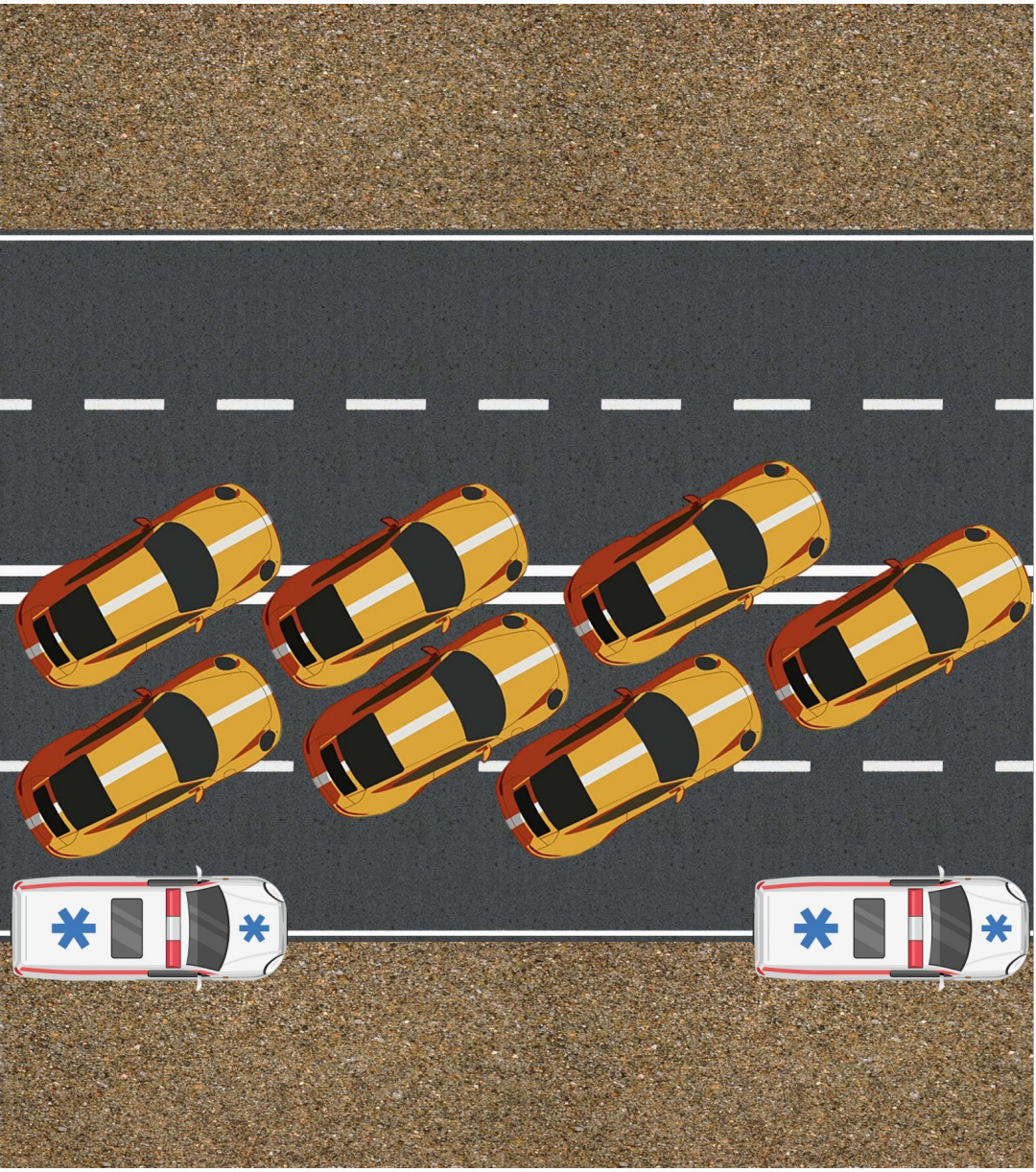
- onSelectChange не так прост, как хотелось бы
- 14 рендеров

18 Версия

- 4 рендера

Как работает flushSync ?

```
1207 export function flushSync<A, R>(fn: A => R, a: A): R {
1208   const prevExecutionContext = executionContext;
1209   if ((prevExecutionContext & (RenderContext | CommitContext)) !== NoContext) {
1210     if (__DEV__) {
1211       console.error(
1212         'flushSync was called from inside a lifecycle method. React cannot ' +
1213           'flush when React is already rendering. Consider moving this call to ' +
1214           'a scheduler task or micro task.',
1215       );
1216     }
1217     return fn(a);
1218   }
1219   executionContext |= BatchedContext;
1220
1221   if (decoupleUpdatePriorityFromScheduler) {
1222     const previousLanePriority = getCurrentUpdateLanePriority();
1223     try {
1224       setCurrentUpdateLanePriority(SyncLanePriority);
1225       if (fn) {
1226         return runWithPriority(ImmediateSchedulerPriority, fn.bind(null, a));
1227       } else {
1228         return (undefined: $FlowFixMe);
1229       }
1230     } finally {
1231       setCurrentUpdateLanePriority(previousLanePriority);
1232       executionContext = prevExecutionContext;
1233       // Flush the immediate callbacks that were scheduled during this batch.
1234       // Note that this will happen even if batchedUpdates is higher up
1235       // the stack.
1236       flushSyncCallbackQueue();
1237     }
```



“это достаточно наиграно и можно вынести в редакс”

```
const asyncAction = () => {
  dispatch(setDirties(dirties));
  dispatch(setValues(values));
  dispatch(setErrors(errors));
  dispatch(setTouches(touches));
  dispatch(setDisables(disables));
}
```

batch()

```
const asyncAction = () ⇒ {
  batch(() ⇒ {
    dispatch(setDirties(dirties));
    dispatch(setValues(values));
    dispatch(setErrors(errors));
    dispatch(setTouches(touches));
    dispatch(setDisables(disables));
  });
}
```

- без batch() - 9 рендеров
- с batch() - 5 рендеров

unstable_batchUpdates история

20 ноября 2016 добавили в unstable_batchUpdates

история батчинга

График изменений

0.4.0
впервые добавили батчинг

Jul 17, 2013

0.12.0
addons.batchedUpdates добавили в API

Oct 28, 2014

0.13.0
setState в методах жизненного цикла теперь асинхронный

Mar 10, 2015

0.14.0
Появился unstable_batchedUpdates

Oct 7, 2015

15.0.0
задиприкейтили addons.batchedUpdates

Apr 7, 2016

17.0.0
Поменяли приоритеты батчинга

Oct 20, 2020

18.0.0
Добавили автоматический батчинг для асинхронщины

Mar 29, 2022

React 0.4.0 использование функции batchedUpdates

Updates to your component are batched now, which may result in a significantly faster re-render of components.
this.setState now takes an optional callback as it's second parameter.
If you were using onClick={this.setState.bind(this, state)} previously,
you'll want to make sure you add a third parameter so that the event is not treated as the callback.

Отдельная благодарность

АйТи Синяк

Антоха Горелов

Спасибо за внимание !