第6章 TODOの投稿処理をオフライン対応

Todo表示処理をオフライン化しましたが、Todo投稿や削除処理はオフラインの状態では失敗してしまいます。これには二つの問題があります。

- オフライン時には投稿内容をキューに保存する
- オンラインになったタイミングでキューを実行する

オフライン時には投稿内容をキューに保存する

Webブラウザのオンライン、オフライン判定を行うのは `navigator.onLine` になります。trueの場合はオンライン、falseの場合はオフラインになります。キューをグローバルな変数として用意しておき、オフライン時にはその変数に保存しましょう。ただし変数に入れたままだとWebブラウザを再読込した時に消えてしまいます。そこでlocalStorageを使ってデータを恒久的に残しておきます。 `todo.js` で `// オフライン時の処理（第6章で追加）` を探して追記してください。

```
// オフライン時の処理（第6章で追加）
if (!navigator.onLine) {
  // オフライン時の処理
  var task = {
    _id: '_local_' + Math.random().toString(32).substring(2),
    todo: todo
  };
  queues.add.push(task);
  localStorage.setItem('addQueue', JSON.stringify(queues.add));
  // 表示を更新
  return res(task);
}
```

逆にWebブラウザを開いた時にはキューを復元します。これは todo.js の一番上に記載されています。

```
// すでに記載済み
// オフライン時に追加/削除するTodo入れておくキュー
var queues = {add: [], delete: []};
```

```
  for (var name of ['add', 'delete']) {
    var value = localStorage.getItem(name + 'Queue');
    if (value) {
      queues[name] = JSON.parse(value);
    }
  }
```

これでキューに保存、そして復元する処理が完了です。

削除処理も同様に

Todoを削除した際にも同様の処理が必要です（こちらはすでに記載済みです）。

```
  // オフライン時
  // すでに記載済み
  if (!navigator.onLine) {
    queues.delete.push(todo);
    localStorage.setItem('deleteQueue', JSON.stringify(queues.delete))
    return res({todo});
  }
```

これで削除するTodoもキューに追加されました。

オンラインになった時にキューを処理する

ではWebブラウザがオンラインに戻った時にキューを処理しましょう。これは `document.addEventListener` で実行できます。hifiveの場合は `'{window} online'` と'定義します。 `todo.js` で オンライン復帰時の処理（第6章用） を探して追記してください。

```
  // オンライン復帰時の処理（第6章用）
  '{window} online': function(context) {
    // 以下を追加
    this.executeQueue(queues);
  }
```

処理は `addTodo` と `deleteTodo` を呼ぶだけです。すでに記述済みです。

```
  // 以下はすでに追加済みです。
  // Todo追加、削除の実行
  executeTask: function(action, todo) {
    switch (action) {
    case 'add':
      this.addTodo(todo);
      break;
    case 'delete':
      this.deleteTodo(todo);
      break;
    }
  },
  // キューを処理する
  executeQueue: function(queues) {
    var me = this;
    for (var action of ['add', 'delete']) {
      for (var todo of queues[action]) {
        this.executeTask(action, todo);
      }
      queues[action] = [];
      localStorage.setItem(action + 'Queue', []);
    }
  },
```

Todoの削除も同様です。今回はTodo名しか使っていないので、同じラベルで複数作ってしまうとまとめて削除されてしまいますので注意してください。

これでオフライン対応も完了です。

ここまででPWAとしての基本であるオフライン対応が完了します。リソースの取得はService Workerがうまく行ってくれますが、データの追加/更新/削除については自分たちでうまく実装しなければなりません。また、Web APIの場合はデータが動的に変わるのでキャッシュを最適なタイミングで更新しないと古いデータが表示されてしまう可能性があるので注意してください。

では次回はこのTodoアプリにWebリモートプッシュ通知を実装してみましょう。