# 概要



---

# テーブルの実装

## ヘッダ固定のテーブルを実装する

`pages/item.vue` に対してElementUIのサンプルからコードをコピー&ペーストして、ヘッダ固定型のテーブルを作成します。

https://element.eleme.cn/#/en-US/component/table#table-with-fixed-header

上記のソースをコピーしてテーブルを作成します。

```
        :
      <el-col :span="20">
        <div>
          <h2>Items</h2>  ... この部分と置き換えます
        </div>
      </el-col>
        :
```

`<h2>~</h2>` の部分を下記の `<el-table>~</el-table>` と置き換えます。

```
<el-table :data="tableData" height="250" style="width: 100%">
  <el-table-column prop="date" label="Date" width="180"></el-table-column>
  <el-table-column prop="name" label="Name" width="180"></el-table-column>
  <el-table-column prop="address" label="Address"></el-table-column>
</el-table>
```

script部分もコピーします。

`<templete> ~ </template>` 部分の下に、scriptのブロックをそのまま追加します。

```
<script>
export default {
  data() {
```

```
      return {
        tableData: [{
            date: '2016-05-03',
            name: 'Tom',
            address: 'No. 189, Grove St, Los Angeles'
        }, {
            date: '2016-05-02',
            name: 'Tom',
            address: 'No. 189, Grove St, Los Angeles'
        }, {
            date: '2016-05-04',
            name: 'Tom',
            address: 'No. 189, Grove St, Los Angeles'
        }, {
            date: '2016-05-01',
            name: 'Tom',
            address: 'No. 189, Grove St, Los Angeles'
        }, {
            date: '2016-05-08',
            name: 'Tom',
            address: 'No. 189, Grove St, Los Angeles'
        }, {
            date: '2016-05-06',
            name: 'Tom',
            address: 'No. 189, Grove St, Los Angeles'
        }, {
            date: '2016-05-07',
            name: 'Tom',
            address: 'No. 189, Grove St, Los Angeles'
        }]
      }
    }
  }
  </script>
```

動かしてみます。

```
CMD> yarn dev
```

prettierのエラーが出たら... 一旦`Ctrl+C`で停止した後、自動整形処理を実行します。

```
CMD> yarn lint --fix
```

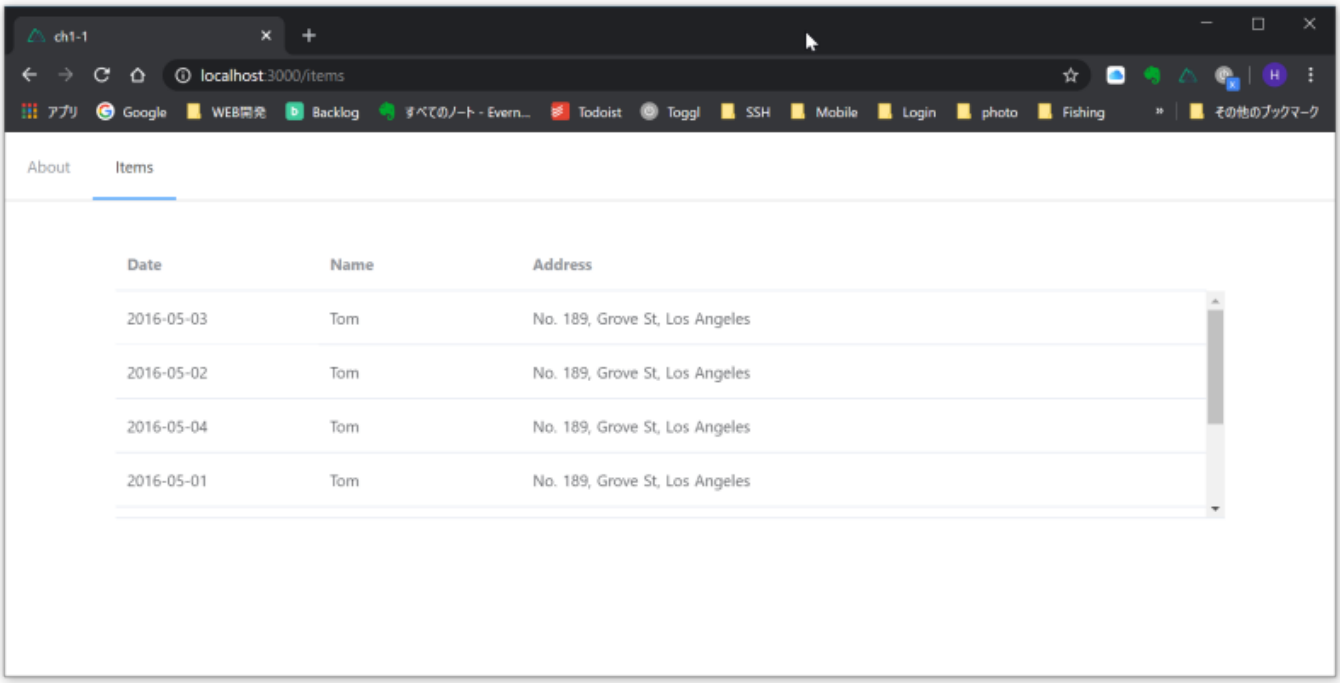
で整形してもらいます。

現在の整形ルールは`.prettierrc`で以下のようになっています。
`printWidth`が入っていない方は、追加しておいてください。そうしないと、幅が80カラムを越えるとエラーが出ます。

```
{
  "semi": false,
  "arrowParens": "always",
  "singleQuote": true,
  "printWidth": 256
}
```

プロジェクトでコーディング規約がある場合、prettierを導入することで、規約違反はエラーとして扱うので、コーディングスタイルの統一に貢献します。(コーディングスタイルを統一することでソースを読みやすくし、開発・メンテナンスの生産性向上に寄与します)

再度起動し、http://localhost:3000/items にアクセスします。




テーブルが実装できました。

## カラムとデータを今回の実習用に変更します

<実習用データ>

| id | title | content | status |
|---|---|---|---|
| 1 | WEB Application作成 | NUXT.js+ElementUIでフロントエンドのWEBアプリを作成する | DONE |
| 2 | RESTful API作成 | Spring Bootを用いてRESTful APIを構築する | DONE |
| 3 | フロントエンドとバックエンドを結合 | NUXTのアプリからaxios経由でREST APIをコールしフロントとバックエンドを繋げる | PROGRESS |
| 4 | MySQLのDockerイメージを作成する | DockerでRESTful APIでアクセスするMySQLのイメージを作成する | PROGRESS |

| id | title | content | status |
|---|---|---|---|
| 5 | バックエンドのDockerイメージを作成する | Javaのイメージにバックエンドのjarをレイヤー化したイメージを作成する | TODO |
| 6 | フロントエンドのDockerイメージを作成する | Node.jsからNUXTをインストールしWEBアプリをコピー後にbuildするイメージを作成する | TODO |
| 7 | Docker-composeでパッケージ化する | フロントエンド・バックエンド・DBの3層をひとつのパッケージにまとめる | TODO |

テーブルのカラムを変更する

```
<el-table ref="itemTable" :data="tableData" height="300" style="width: 100%">
  <el-table-column prop="id" label="ID" min-width="40" header-align="center"
align="right"></el-table-column>
  <el-table-column prop="title" label="タイトル" min-width="200" header-
align="center" show-overflow-tooltip></el-table-column>
  <el-table-column prop="content" label="内容" min-width="400" header-
align="center" show-overflow-tooltip></el-table-column>
  <el-table-column prop="status" label="状態" min-width="100" align="center">
    <template slot-scope="scope">
      <el-tag :type="scope.row.status === 'DONE' ? 'success' : (scope.row.status
=== 'PROGRESS' ? 'primary' : 'warning')" disable-transitions>
        {{ scope.row.status }}
      </el-tag>
    </template>
  </el-table-column>
</el-table>
```

`<el-table>`:

- ref : script内でどのような名前でアクセスするか
- :data : dataというバインド変数にitemTable配列を割り当てる
- height : 表の高さ
- style : スタイル指定

`<el-table-column>`:

- prop : itemTable配列のデータの項目名を設定する
- label : 表のヘッダに表示されるラベルを指定する
- min-width : 最小幅を指定する
- header-align : ヘッダ部分の文字揃えを指定する
- align : データ部分の文字揃えを指定する
- show-overflow-tooltip : データがオーバーフローした場合にツールチップとして表示する場合指定する

`<template slot-scope="scope"> ~ </template>`:
テーブルのデータ部分に、templateで囲われたHTMLを表示します。
これにより、例えば、商品の写真のようなものもテーブル内に表示することが出来るようになります。

`<el-tag>`: 下記URLを参照
https://element.eleme.cn/#/en-US/component/tag

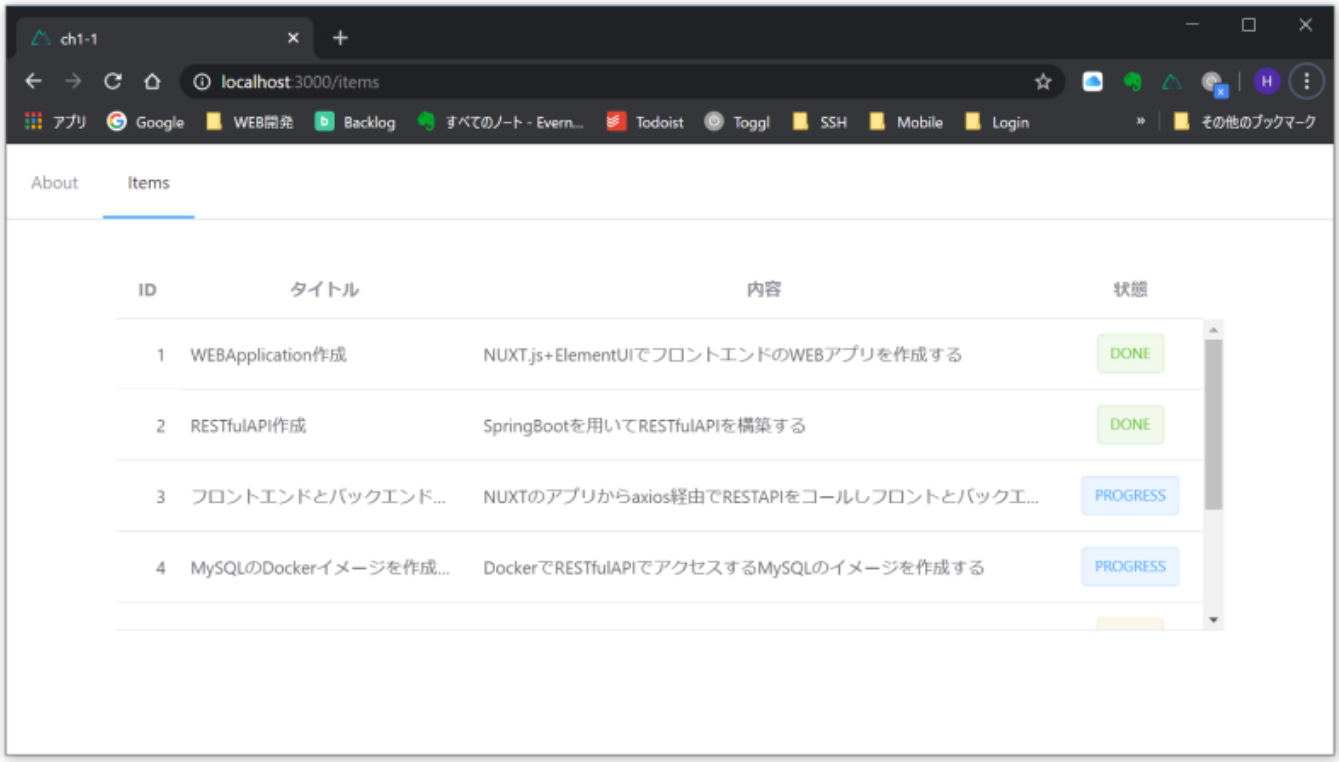
statusの値により、`:type`を`DONE`なら`success`, `PROGRESS`なら`primary`,`TODO`なら`warning`の色に設定しています。

データ部分を書き換える（バックエンドと結合したときはDB内容を表示する）

`tableData: [ ... ]`の配列の中身を以下の内容で書き換えます。

```
{
  id: '1',
  title: 'WEBApplication作成',
  content: 'NUXT.js+ElementUIでフロントエンドのWEBアプリを作成する',
  status: 'DONE'
},
{
  id: '2',
  title: 'RESTfulAPI作成',
  content: 'SpringBootを用いてRESTfulAPIを構築する',
  status: 'DONE'
},
{
  id: '3',
  title: 'フロントエンドとバックエンドを結合',
  content: 'NUXTのアプリからaxios経由でRESTAPIをコールしフロントとバックエンドを繋げる',
  status: 'PROGRESS'
},
{
  id: '4',
  title: 'MySQLのDockerイメージを作成する',
  content: 'DockerでRESTfulAPIでアクセスするMySQLのイメージを作成する',
  status: 'PROGRESS'
},
{
  id: '5',
  title: 'バックエンドのDockerイメージを作成する',
  content: 'Javaのイメージにバックエンドのjarをレイヤー化したイメージを作成する',
  status: 'TODO'
},
{
  id: '6',
  title: 'フロントエンドのDockerイメージを作成する',
  content: 'Node.jsからNUXTをインストールしWEBアプリをコピー後にbuildするイメージを作成する',
  status: 'TODO'
},
{
  id: '7',
  title: 'Docker-composeでパッケージ化する',
  content: 'フロントエンド・バックエンド・DBの3層をひとつのパッケージにまとめる',
  status: 'TODO'
}
```

下記画面のようになりましたでしょうか？




## Edit/Deleteボタン追加

下記サンプルを参考に行単位に編集・削除するための操作ボタンを追加します。

https://element.eleme.cn/#/en-US/component/table#custom-column-template

せっかくなので文字ではなくアイコンボタンにしてみます。
編集は鉛筆アイコン `<i class="el-icon-edit"></i>` に、削除はゴミ箱アイコン`<i class="el-icon-delete"></i>` にします。

https://element.eleme.cn/#/en-US/component/icon

```
<el-table-column label="操作" min-width="120" header-align="center">
  <template slot-scope="scope">
    <el-button size="small" circle @click="handleEdit(scope.$index, scope.row)"><i
class="el-icon-edit"></i> </el-button>
    <el-button size="small" type="danger" circle
@click="handleDelete(scope.$index, scope.row)"><i class="el-icon-delete"></i>
</el-button>
  </template>
</el-table-column>
```

`<el-button>`:

- size: ボタンサイズ
- type: 色

- round: 形状を円にします
- @click: ボタンが押されたときにコールする処理

ボタンを表示するカラムを追加したので、contentのカラム幅を300に減らします。

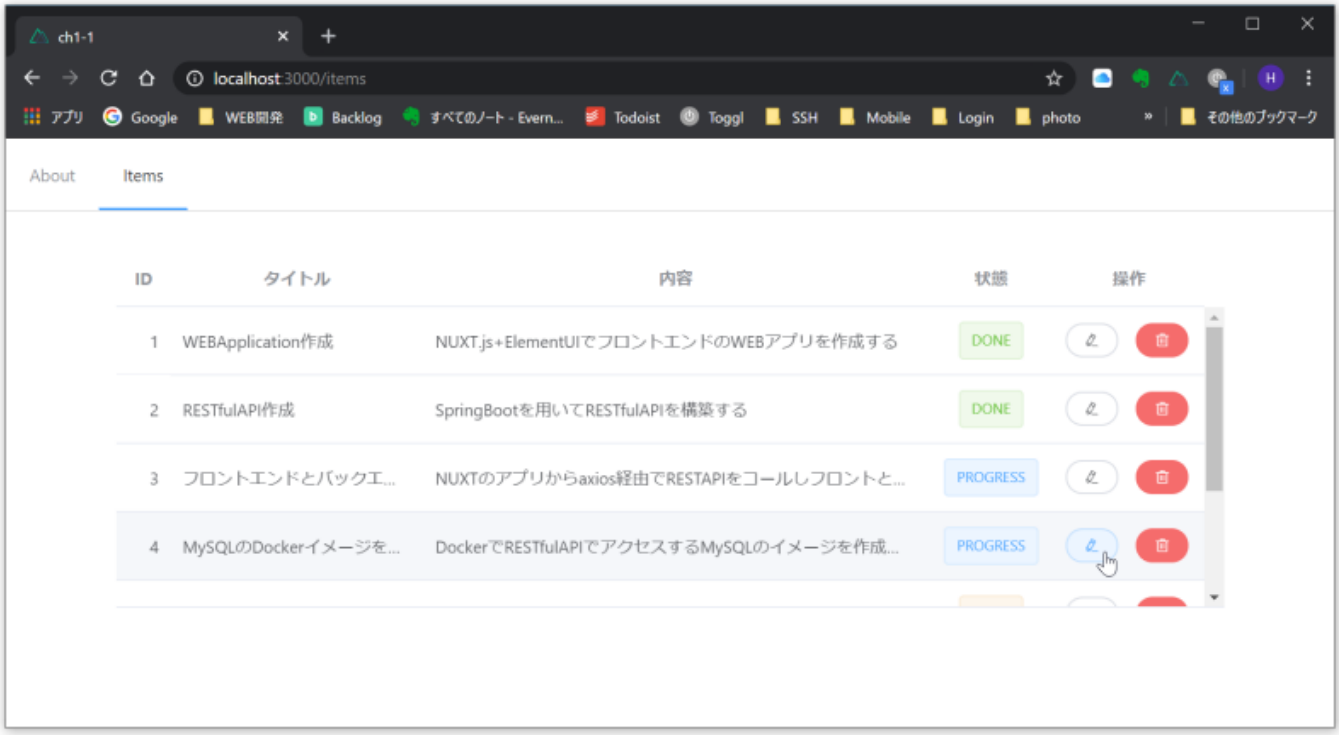
```
<el-table-column prop="content" label="内容" min-width="300" header-align="center"
show-overflow-tooltip> </el-table-column>
```

script追加部分

```
export default {
  data() {
    return {
      tableData: [
        :
      ]
    }
  },
  methods: {
    handleEdit(index, row) {
      console.log(index, row)
    },
    handleDelete(index, row) {
      console.log(index, row)
    }
  }
}
```

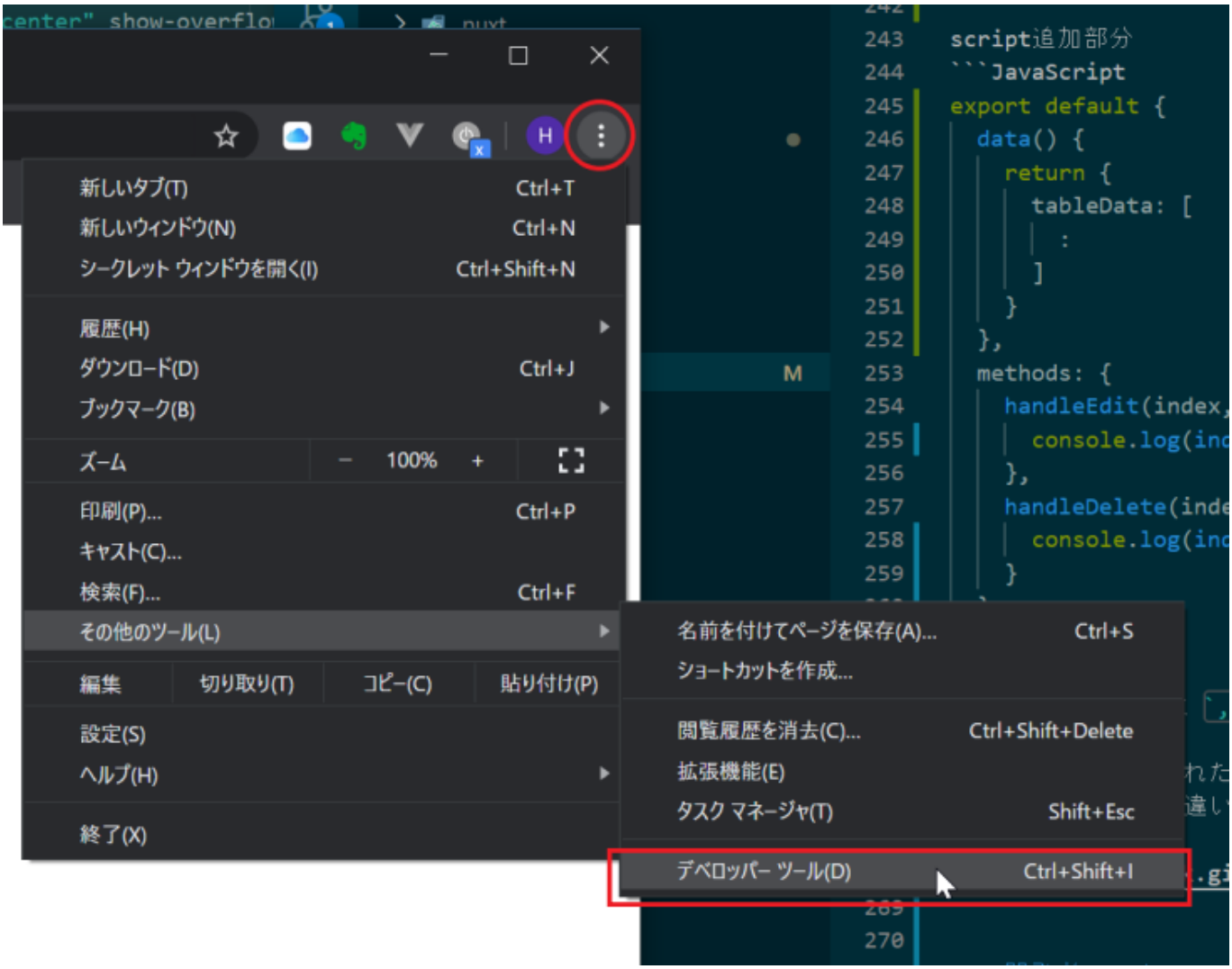
`data() {}` の後に `,` を付けて、`methods: { .. }` 部分を追加します。

下記のように表示されたでしょうか？
ボタンの形状が少し違いますが、画像が少し古いだけですので気にしないでください。

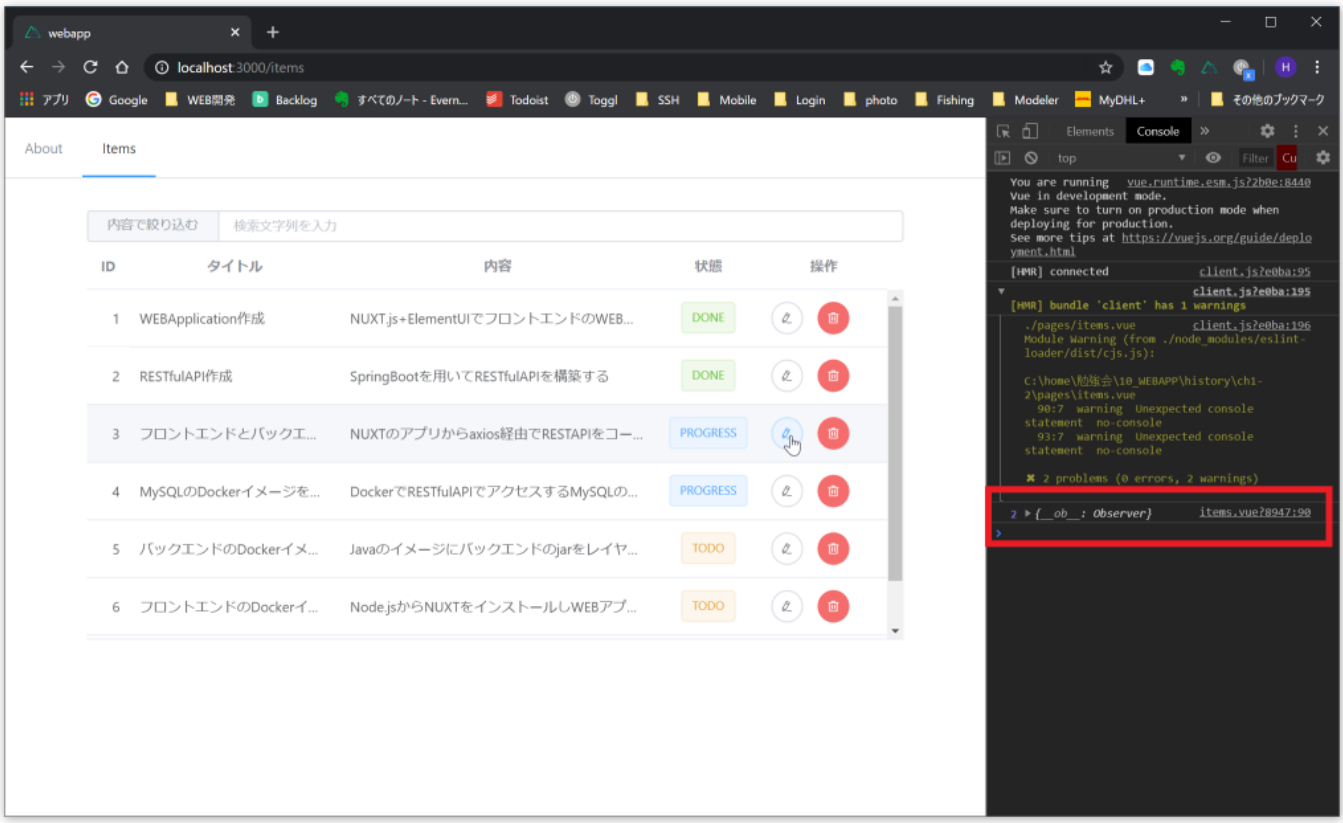



## 開発者ツールでconsole出力内容を確認

google chromeを使って、開発ツールを表示します。



編集ボタンをクリックした行のINDEXとオブジェクト内容が開発ツールウインドウの「console」タブ内に表示されます。




# 絞り込み検索の実装

内容を部分一致でデータを絞り込む機能を実装します。

## 入力欄の追加

下記サンプルを元に入力欄を追加します。

https://element.eleme.cn/#/en-US/component/input#mixed-input

`<el-table>` の上部分に入力欄を追加します。

```
<el-input v-model="search" size="small" placeholder="検索文字列を入力">
  <template slot="prepend">内容で絞り込む</template>
</el-input>
```

- v-model: `search` という変数に入力値をバインドします
- size: 入力欄の大きさ　small にします。(好みで..)
- placeholder: 入力欄に薄く表示される文字を指定します
- `<template slot="prepend">~</templete>`: 入力欄のタイトルを左側に表示します。アイコンなども利用できます。

scriptに検索文字列を保持する変数「search」追加

```
  data() {
    return {
      tableData: [
        :
      ],
      search: ''
    }
  },
```

tableDataにfilterの機能を付けます

https://element.eleme.cn/#/en-US/component/table#table-with-custom-header

```
<el-table ref="itemTable"
  :data="tableData.filter((data) => !search ||
data.content.toLowerCase().includes(search.toLowerCase()))"
  height="400"
  style="width: 100%"
>
```

tableDataの各要素に対して、search が空でなければ、contentを小文字化したものがsearchを小文字化したものを含むものに絞り込んで(filter)返す
という処理をやっています。この書き方が最近流行っているとか...

実行イメージは以下のようになります。
内容に「rest」を含むデータが抽出されています。