todo

`src/App.js`

```
import React from 'react';
// import logo from './logo.svg';
// import './App.css';
import './todo.css'

class App extends React.Component {
  constructor () {
    super()
    this.state = {
      curretTodoName: '',
      currendEditId: '',
      // 全选全不选
      isAllSelect: '',
      filter: '',
      //
      todos: [
        {
          id: 1,
          name: '吃饭饭',
          check: false
        },
        {
          id: 2,
          name: '睡觉觉',
          check: true
        },
        {
          id: 3,
          name: '打豆豆',
          check: false
        },
      ]
    }
  }
  componentDidMount () {
    // 获取路由 监听路由
    this.renderByHash()
  }
  // 渲染任务列表
  // 列表渲染
  renderTodos () {
    return this.state.todos.map( item => {
      if (this.state.filter === '#/active' && item.check) return null
      if (this.state.filter === '#/completed' && !item.check) return null
      return (
    // 动态class
    <li className={ [item.check ? 'completed' : null, this.state.currendEditId === item.id ? 'editing' : null].join(' ') } key={ item.id }>
      <div className="view">
        <input className="toggle" type="checkbox" checked={ item.check }
         onChange={ e => this.changeCheckBox(e, item) } />
        <label onDoubleClick={ e => this.setCurrentEditId(e, item) } >{ item.name }</label>
        <button className="destroy" onClick={ e => this.removeTodo(item.id) }></button>
      </div>
      <input id="input1" ref={ `eidInput_${item.id}` } className="edit" value={ item.name } onBlur={ e=> this.setCurrentEditId(e, item, -1)  } onChange={ e=> this.editDoto(e, item) } autoFocus />
    </li>
   )
  })
 }
 // 14: 根据hash来过滤输出
 renderByHash () {
  // 监视锚点后续改变
  window.addEventListener('hashchange',()=>{
    this.setState({
     filter: window.location.hash
    })
  })
  this.setState({
   filter: window.location.hash
  })
 }
 // 赋值当前任务标题
 setCurrentName (e) {
  this.setState({
   curretTodoName: e.target.value
  })
 }
 // 添加任务
 addTodo (e) {
  e.preventDefault()
  console.log('1')
  if (this.state.curretTodoName.trim() === '') {
   return false
  }
  let minId = -1
  this.state.todos.forEach(item => {
   if (item.id > minId) {
    minId = item.id
   }
  })
  minId++
  this.state.todos.push({
   id: minId,
   name: this.state.curretTodoName,
      check: false
    })
    // 更新视图层
    this.setState({
      curretTodoName: ''
    })
  }
  // 删除任务
  removeTodo (id) {
    let todoIndex = this.state.todos.findIndex( item => {
      return item.id === id
    })
    this.state.todos.splice(todoIndex, 1)
    this.setState({})
  }
  // 改变ckechbox
  changeCheckBox (e, todo) {
    todo.check = e.target.checked
    this.setState({})
  }
  // 设置当前编辑的任务
  setCurrentEditId (e, todo, canel) {
    if (canel === -1) {
      this.setState({
        currendEditId: -1
      })
      return
    }
    // 更新后 才能使用DOM
    // 回调 refs
    this.setState({
      currendEditId: todo.id
    }, () => {
      this.refs[`eidInput_${todo.id}`].focus()
    })
  }
  // 对当前的todo 进行编辑
  // 直接就是双向的绑定 为什么要写这么多代码
  editDoto (e, todo) {
    todo.name = e.target.value
    this.setState({})
  }
  // 全选
  allSelect (e) {
    this.state.todos.forEach( item => {
      item.check = e.target.checked
    })
    this.setState({
      isAllSelect: e.target.checked
    })
  }
  render () {
    return (
      <React.Fragment>
        <section className="todoapp">
          <header className="header">
              <h1>todos</h1>
              <form onSubmit={ e=> this.addTodo(e) }>
                <input className="new-todo" value={ this.state.curretTodoName } onChange={ e=> this.setCurrentName(e) }  placeholder="What needs to be done?" autoFocus />
              </form>
          </header>
          {/* This section should be hidden by default and shown when there are todos */}
          <section className="main">
            <input id="toggle-all" className="toggle-all" type="checkbox" checked={ this.state.isAllSelect } onChange={ e=> this.allSelect(e) } />
            <label htmlFor="toggle-all">Mark all as complete</label>
            <ul className="todo-list">
              { this.renderTodos() }
            </ul>
          </section>
          {/* This footer should hidden by default and shown when there are todos */}
          <footer className="footer">
              {/* This should be `0 items left` by default */}
              <span className="todo-count">
                <strong>{
                  this.state.todos.filter(item => {
                    return !item.check
                  }).length
                }</strong>
                item left
              </span>
              {/* Remove this if you don't implement routing */}
              <ul className="filters">
                  <li>
                      <a className={ this.state.filter === '#/' ? 'selected' : null } href="#/">All</a>
                  </li>
                  <li>
                      <a className={ this.state.filter === '#/active' ? 'selected' : null } href="#/active">Active</a>
                  </li>
                  <li>
                      <a className={ this.state.filter === '#/completed' ? 'selected' : null } href="#/completed">Completed</a>
                  </li>
              </ul>
              {/*Hidden if no completed items are left ↓ */}
              <button className="clear-completed" onClick={ e=> {
                this.setState({
                  todos: this.state.todos.filter(item => {
                    return !item.check
                  })
                })
              } }>Clear completed</button>
          </footer>
        </section>
        <footer className="info">
            <p>Double-click to edit a todo</p>
            {/* Remove the below line ↓ */}
            <p>Template by <a href="http://sindresorhus.com">Sindre Sorhus</a></p>
            {/* Change this out with your name and url ↓ */}
            <p>Created by <a href="http://todomvc.com">you</a></p>
            <p>Part of <a href="http://todomvc.com">TodoMVC</a></p>
        </footer>
      </React.Fragment>
    )
  }

}

export default App;

```