<http://subnets.ru/blog/?p=5521>

<https://www.youtube.com/playlist?list=PLCjIBdZ4kPo0PVkDsOtwZn6O3IkT7_dqR>

<https://habr.com/ru/company/sberbank/blog/533612/>

[https://developers.sber.ru/docs/ru/va/reference/assistant-client/overview#](https://developers.sber.ru/docs/ru/va/reference/assistant-client/overview)

<https://webdevblog.ru/15-poleznyh-javascript-primerov-map-reduce-i-filter/>

\_\_\_\_\_\_\_\_\_\_\_\_actions.js

function addNote(note, context) {

addAction({

type: "add\_note",

note: note

}, context);

}

function doneNote(id, context){

addAction({

type: "done\_note",

id: id

}, context);

}

function chooseAnswer(id, context){

addAction({

type: "choose\_answer",

id: id

}, context);

}

chooseAnswer.sc

theme: /

state: ВыборОтвета

q!: (~ответ|выбери)

$AnyText::anyText

script:

var item\_id = get\_id\_by\_selected\_item(get\_request($context));

chooseAnswer(item\_id,$context);

a: Выбираю

addNote.sc

theme: /

state: ДобавлениеЭлемента

q!: (~добавить|~установить|запиши|поставь|закинь|~напомнить|~творить)

[~напоминание|~заметка|~задание|~задача]

$AnyText::anyText

random:

a: Добавлено!

a: Сделано!

script:

addNote($parseTree.\_anyText, $context);

addSuggestions(["Добавь задачу купить машину"], $context);

require: slotfilling/slotFilling.sc

module = sys.zb-common

# Подключение javascript обработчиков

require: js/getters.js

require: js/reply.js

require: js/actions.js

# Подключение сценарных файлов

require: sc/addNote.sc

require: sc/doNote.sc

require: sc/chooseAnswer.sc

require: sc/noteDone.sc

patterns:

$AnyText = $nonEmptyGarbage

theme: /

state: Start

# При запуске приложения с кнопки прилетит сообщение /start.

q!: $regex</start>

# При запуске приложения с голоса прилетит сказанная фраза.

q!: (запусти | открой | вруби) Салют Демо Апп

a: Начнём.

state: Fallback

event!: noMatch

a: Я не понимаю

Вспомогательные классы:

TaskItem.jsx

import React from "react";

import "../App.css";

export const TaskItem = (props) => {

  const { item, index, onDone } = props;

  return (

    <li

      className = "task-item"

      //key       = {item.id}

    >

      <span>

        <span

          style = {{ fontWeight: "bold" }}

        >{index + 1}. </span>

        <span

          style = {{ textDecorationLine: item.completed ? "line-through" : "none", }}

        >

          {item.title}

        </span>

      </span>

      <input

        className = "done-item"

        type      = "checkbox"

        checked   = {item.completed}

        onChange  = {(event) => onDone(item) }

      />

    </li>

  )

}

TaskItem.jsx

import React from "react";

import {TaskItem} from './TaskItem';

import "../App.css";

export const TaskItemList = (props) => {

  const { items, onDone } = props

  return (

    <ul className="notes">

      {

        items.map((item, index) => (

          <TaskItem

            item   = { item }

            key    = { index }

            index  = { index }

            onDone = { () => onDone(item) }

          />

        ))

      }

    </ul>

  )

}

AddTask.jsx

import React from "react";

import "../App.css";

export class AddTask extends React.Component {

  constructor (props) {

    super(props)

    this.state = {

      note: '',

    }

  }

  render () {

    const { onAdd } = this.props;

    return (

      <form

        onSubmit={(event) => {

          event.preventDefault();

          onAdd(this.state.note);

          this.setState({

            note: '',

          })

        }}

      >

        <input

          className   = "add-task"

          type        = "text"

          placeholder = "Add Note"

          value       = { this.state.note }

          onChange    = {({ target: { value } }) => this.setState({

            note: value,

          })}

          required

          autoFocus

        />

      </form>

    )

  }

}

TaskList.jsx

import React from 'react';

import {AddTask} from '../components/AddTask';

import {TaskItemList} from '../components/TaskItemList';

export const TaskList = (props) => {

  const { items, onAdd, onDone } = props;

  return (

    <main className="container">

      <AddTask

        onAdd = { onAdd }

      />

      <TaskItemList

        items  = { items }

        onDone = { onDone }

      />

    </main>

  )

}

import React from "react";

import {

createSmartappDebugger,

createAssistant,

} from "@sberdevices/assistant-client";

import "./App.css";

import { TaskList } from './pages/TaskList';

const initializeAssistant = (getState/\*: any\*/) => {

if (process.env.NODE\_ENV === "development") {

return createSmartappDebugger({

token: process.env.REACT\_APP\_TOKEN ?? "",

initPhrase: `Запусти ${process.env.REACT\_APP\_SMARTAPP}`,

getState,

});

}

return createAssistant({ getState });

};

export class App extends React.Component {

constructor(props) {

super(props);

console.log('constructor');

this.state = {

notes: [],

}

this.assistant = initializeAssistant(() => this.getStateForAssistant() );

this.assistant.on("data", (event/\*: any\*/) => {

console.log(`assistant.on(data)`, event);

const { action } = event

this.dispatchAssistantAction(action);

});

this.assistant.on("start", (event) => {

console.log(`assistant.on(start)`, event);

});

}

++++++

Index. tsx

+++++

componentDidMount() {

console.log('componentDidMount');

}

getStateForAssistant () {

console.log('getStateForAssistant: this.state:', this.state)

const state = {

item\_selector: {

items: this.state.notes.map(

({ id, title }, index) => ({

number: index + 1,

id,

title,

})

),

},

};

console.log('getStateForAssistant: state:', state)

return state;

}

dispatchAssistantAction (action) {

console.log('dispatchAssistantAction', action);

if (action) {

switch (action.type) {

case 'add\_note':

return this.add\_note(action);

case 'done\_note':

return this.done\_note(action);

case 'delete\_note':

return this.delete\_note(action);

default:

throw new Error();

}

}

}

add\_note (action) {

console.log('add\_note', action);

this.setState({

notes: [

...this.state.notes,

{

id: Math.random().toString(36).substring(7),

title: action.note,

completed: false,

},

],

})

}

done\_note (action) {

console.log('done\_note', action);

this.setState({

notes: this.state.notes.map((note) =>

(note.id === action.id)

? { ...note, completed: !note.completed }

: note

),

})

}

delete\_note (action) {

console.log('delete\_note', action);

this.setState({

notes: this.state.notes.filter(({ id }) => id !== action.id),

})

}

render() {

console.log('render');

return (

<TaskList

items = {this.state.notes}

onAdd = {(note) => { this.add\_note({ type: "add\_note", note }); }}

onDone = {(note) => { this.done\_note({ type: "done\_note", id: note.id }) }}

/>

)

}

}