

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Информационные технологии и прикладная
математика»
Кафедра: 806 «Вычислительная математика и программирование»

Реферат
по курсу «Фундаментальная
информатика» I семестр
Тема:
«CRM на JavaScript»

Группа:	М8О-109Б-22
Студент:	Фомин И.Д.
Преподаватель:	Сысоев М.А.
Оценка:	
Дата:	

Москва, 2022

Содержание

1. Сферы применения JavaScript.....	
2. Основные особенности языка.....	
3. CRM. Вёрстка.....	
4. CRM. JS – View.....	
5. CRM. JS – Model+Controller.....	
6. Список литературы.....	

1. Сферы применения JS

1. Динамические веб-страницы

Всплывающие подсказки, движущиеся картинки, падающие снежинки и прочие анимации — за всем этим надо идти к JS. JS-код встроен в веб-страницу, и когда пользователь открывает её, скрипт выполняется прямо в браузере.

2. Веб приложения и игры

Для примера, Google Maps и веб-клиент Gmail используют JavaScript. А если вы хотите написать игру — возьмите JS, HTML5, одну из готовых библиотек (скажем, EaselJS или PixiJS)

3. Расширения для браузера

Раз в основе JavaScript лежит выполнение кода в браузере, это отличный выбор для создания браузерных расширений. Напишите свой чекер почты или, например, счётчик активности, который будет отслеживать, сколько времени вы провели за работой, а сколько за просмотром соцсетей.

4. Серверные приложения

Главное преимущество JavaScript перед PHP, Python или GO — возможность разрабатывать клиентскую и серверную часть на одном и том же языке. Чтобы писать бэкенд на JavaScript, обычно используется движок Node.js — он позволяет выполнять JS-код вне браузера.

5. Мобильные приложения

Чаще всего мобильные приложения разрабатываются на языках, специфичных для операционной системы (Swift для iOS и Java/Kotlin для Android). JavaScript же хорош тем, что позволяет создавать кроссплатформенные приложения — для этого можно использовать фреймворки React Native, Ionic или PhoneGap.

6. Desktopные приложения

GitHub соединил Node.js, движок рендеринга Chromium и разработал фреймворк Electron, на котором можно писать кроссплатформенные десктопные проекты. Среди примеров — Discord, GitHub Desktop, Visual Studio Code, Skype, WordPress Desktop. У Electron есть аналог — NW.js, который в основном используется для создания настольных версий сайтов и игр.

1. Основные особенности языка

7. Необязательная точка с запятой:

```
1 alert('Привет'); alert('Мир');
```

Как правило, перевод строки также интерпретируется как разделитель, так тоже будет работать:

```
1 alert('Привет')
2 alert('Мир')
```

Это так называемая «автоматическая вставка точки с запятой». Впрочем, она не всегда срабатывает, например:

```
1 alert("После этого сообщения ждите ошибку")
2
3 [1, 2].forEach(alert)
```

8. По умолчанию в качестве рабочей среды используется браузер (если писать на чистом JS, а не на NODE.js):

Вот, например, как можно взаимодействовать с пользователем

```
prompt(question, [default])
```

Задаёт вопрос `question` и возвращает то, что ввёл посетитель, либо `null`, если посетитель нажал на кнопку «Отмена».

```
confirm(question)
```

Задаёт вопрос `question` и предлагает выбрать «ОК» или «Отмена». Выбор возвращается в формате `true/false`.

```
alert(message)
```

Выводит сообщение `message`.


9. До того, как объявление переменных через `var` устарело, в js происходило так называемое всплытие переменных: объявление переменных автоматически поднималось в начало области видимости (тогда области видимости выделялись с помощью функций, то есть, если создать переменную, например, в цикле, то к ней можно будет обращаться и вне этого цикла):

```
console.log(shape); // OUTPUT : undefined

var shape = "square";

console.log(shape); // OUTPUT : "square"
```

10. Ну и, конечно же, легендарное неявное приведение типов:

≥ typeof NaN	≥ true==1
◁ "number"	◁ true
> 9999999999999999	≥ true===1
◁ 10000000000000000	◁ false
> 0.5+0.1==0.6	> (!+[]+[]+![]).length
◁ true	◁ 9
≥ 0.1+0.2==0.3	> 9+"1"
◁ false	◁ "91"
≥ Math.max()	≥ 91-"1"
◁ -Infinity	◁ 90
> Math.min()	≥ []==0
◁ Infinity	◁ true
> []+[]	
◁ ""	
≥ []+{}	
◁ "[object Object]"	
≥ {}+[]	
◁ 0	
> true+true+true===3	
◁ true	
> true-true	
◁ 0	

2. CRM. Вёрстка

Вообще, стоит хотя бы кратко упомянуть, что такое CRM. CRM – Это Customer Relationship Management, то есть «управление отношениями с клиентами». Вообще, базовые возможности CRM – это создание/изменение/удаление данных клиентов, удобное их представление, чтобы с клиентом легко можно было связаться. И подобных полезных quality of live изменений можно придумать сколько угодно.

Итак, страница состоит из хедера, таблицы с клиентами и кнопки добавления клиента.

Хедер свёрстан при помощи grid, он был использован, чтобы поле ввода можно было легче изменять по размеру.

Таблица свёрстана тоже при помощи grid, потому что grid идеален для вёрстки таблиц. Для полей, потенциально занимающих много места, ширина колонок была поставлена fit-content, если известно, что поле не будет слишком большим или auto в иных случаях. Однако для контактов был использован flex, потому что flex лучше всего справляется с выравниванием элементов горизонтально.

Кнопка добавления клиентов свёрстана при помощи flex.

Пройдёмся по основным этапам работы программы:

3. CRM. JS - view

Вся таблица отрисовывается с помощью JS. Процесс такой:

```
function createTable(customers) {  
  createCustomerEntry(elem).forEach(elem => {  
    mainContainer.append(elem);  
  })  
}
```

⇒ для каждого клиента вызывается

⇒

создаётся строка таблицы при помощи

```
function createContactEntry(contactData) {...}  
=  
function createModificationModal({id, contacts, lastName, name, surname}) {
```

И

Которые создают поле контактов и
всплывающее окно при нажатии по кнопкам

* Вся работа с SVG вынесена в отдельный файл svg.js

4. CRM. Model + Controller

Этот код можно назвать и моделью и контроллером, потому что в нём происходит как отрисовка страницы, полученной из view, так и обработка всех действий пользователя и взаимодействие с сервером.

```
export let DELETE_LISTENER = (id) => {  
  fetch( input: `http://localhost:3000/api/clients/${id}`, init: {  
    method: "DELETE",  
  });  
}
```

То, что происходит, когда нажимают на кнопку удалить

```
export let PATCH_LISTENER = (id) => {  
  let surname = document.getElementById("modal-surname").value;  
  let name = document.getElementById("modal-name").value;  
  let lastName = document.getElementById("modal-lastName").value;  
  let contactList = document.getElementById("modal-contact-grid-container").children;  
  let contactData = [];  
  
  let currentType = null;  
  for (const child of contactList) {  
    if (child.tagName.toLowerCase() === "div") {  
      currentType = child.children[0].children[0].textContent; //  
      костыльно, но работает  
    } else if (child.tagName.toLowerCase() === "input" && currentType) {  
      contactData.push({ "type": currentType, "value": child.value });  
      currentType = null;  
    }  
  }  
  console.log(contactData);  
  
  fetch(`http://localhost:3000/api/clients/${id}`, {  
    method: "PATCH",  
    headers: {  
      "Content-Type": "application/json",  
    },  
    body: JSON.stringify({  
      surname,  
      name,  
      lastName,  
      contactData  
    })  
  })  
  .then((msg) => {  
    console.log(msg, id);  
  })  
  .catch(  
    (reason) => {  
      alert(`FAILED TO PATCH: ${reason}`);  
    }  
  )  
  );  
};
```

- то, что происходит при нажатии на кнопку изменить

```

async function generatePage() {
  setEventListeners();
  let formInputFields;

  createTable(customerData);
  createModalNewClient();

  let addClientFromModal = document.getElementById("modal-form-submit");
  addClientFromModal.addEventListener("click", (e) => {
    e.preventDefault();

    formInputFields = saveDataFromModal();

    let data = {};
    Object.entries(formInputFields).forEach((elem) => {
      let [k, v] = elem;
      if (v !== null) data[k] = v;
      else data[k] = null;
    });
    console.log(data);
    if (!addNewClient(data)) alert('FAILED TO ADD NEW CLIENT');
    else location.reload();
  });
}

```

- генерация таблицы и модальных окон + добавление обработчика событий при нажатии на кнопку отправить.

```

function saveDataFromModal() {
  let arrTo = {};

  arrTo.surname = document.getElementById("modal-surname").value;
  arrTo.name = document.getElementById("modal-name").value;
  arrTo.lastName = document.getElementById("modal-lastName").value;
  arrTo.contacts = [];

  let types = document.getElementsByClassName("current-contact-span");
  let data = document.getElementsByClassName("modal__contact-redact-
input__data");

  for (let i = 0; i < types.length; i++) {
    arrTo.contacts.push(
      {"type": types[i].textContent, "value": data[i].value}
    );
  }

  return arrTo;
}

```

- вспомогательная функция, чтобы корректно собрать и отформатировать данные.

```

async function addNewClient({name, surname, lastName, contacts}) {
  // todo validate all fields
  console.log(JSON.stringify({
    name,
    surname,
    lastName,
    contacts,
  }));
}

```

```
let resp = await fetch("http://localhost:3000/api/clients", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    name,
    surname,
    lastName,
    contacts,
  }),
});

return resp.status;
}
```

- то, что вызовется при нажатии на кнопку “Добавить клиента”

Список литературы

- 1) https://github.com/Haliava/CRM_Skillbox - репозиторий с программой на GitHub
- 2) <https://tproger.ru/articles/what-javascript-is-good-for/> - сферы применения JavaScript на сегодняшний день
- 3) https://salesap.ru/crm_sistemy_chno_eto/ - Справки о CRM

Примечание: работа выполнялась в PhpStorm 2021.1.1

