# Блок 2. Работа с JavaScript в браузере.

## Занятие 1. Работа с DOM через JavaScript

### 1) Написать функцию, которая выводит список всех сотрудников в список <ul>.

Функция **showEmployees(employees)** должна использовать DOM для создания элементов <li> списка <ul>. В каждом элементе должны выводиться данные об одном сотруднике. Должно выводиться имя и фамилия сотрудника. Результат должен помещаться в PLACEHOLDER, заранее добавленный на страницу:

<**div id="employeesPlaceholder"**></**div**>

Перед повторным заполнением плейсхолдер должен очищаться:

**function** *clearEmployeesPlaceholder*() {

***document***.getElementById(***PLACEHOLDER***).innerHTML = '';

}

Также объявите константу PLACEHOLDER, равную **employeesPlaceholder.**

Функцию нужно поместить в файл ui.js и подключить файл к HTML-файлу.

**function** *showEmployees*(employees) {  
 *clearEmployeesPlaceholder*();  
 **const** ul = ***document***.createElement(**"ul"**);  
  
 **for** (**let** employee **of** employees) {  
 **const** li = ***document***.createElement(**"li"**);  
 ul.appendChild(li);  
  
 li.innerHTML = employee.**name**+**" "**+employee.**surname**;  
  
 }  
 ***document***.getElementById(***PLACEHOLDER***).appendChild(ul);  
}

### 2) Запускать функцию показа списка сотрудников при открытии страницы.

Для этого нужно сделать функцию runUI, в которой будут помещены все действия, выполняемые при загрузке страницы.

Данную функцию можно вызывать после подключения скриптов в HTML:

<**script src="employees-json.js"**></**script**>  
<**script src="service.js"**></**script**>  
<**script src="ui.js"**></**script**>

В конце документа добавьте вызов runUI:

<**script**>  
 *runUI*();  
</**script**>

Сама функция пока что выглядит тривиально:

**function** *runUI*() {  
 *showEmployees*(***DATA***.**employees**);  
}

### 3) Создать форму с возможностью добавить нового сотрудника и полями name и surname.

Сама форма должна располагаться в index.html:

<**div**>  
 Имя:  
 <**input id="name" placeholder="Имя"**>  
  
 Фамилия:  
 <**input id="surname" placeholder="Фамилия"**>  
  
 <**button onclick="***addEmployeeUI*()**"**>Добавить сотрудника</**button**>  
</**div**>

При нажатии на кнопку должна вызываться функция *addEmployeeUI():*

**function** *addEmployeeUI*() {  
 **const** name = ***document***.getElementById(**"name"**).**value**;  
 **const** surname = ***document***.getElementById(**"surname"**).**value**;  
 **const** id = *addEmployee*(name, surname);  
 *showEmployees*(***DATA***.**employees**);  
}

### 4) Нужно предусмотреть валидацию данных формы. Если name или surname не заполнены, должно показываться сообщение об ошибке.

Для этого после формы следует предусмотреть соответствующий плейсхолдер:

<**div id="addEmployeeFormErrorMessage"**></**div**>

А в самой функции нужно его заполнять в случае наличия ошибок:

**function** *addEmployeeUI*() {  
 **let** errorHTML = **""**;

**const** name = ***document***.getElementById(**"name"**).**value**;  
 **if** (name==**""**) {  
 **errorHTML** += **"- Имя сотрудника должно быть задано<br>"**;  
 }  
 **const** surname = ***document***.getElementById(**"surname"**).**value**;  
 **if** (surname==**""**) {  
 **errorHTML** += **"- Фамилия сотрудника должна быть задана<br>"**;  
 }  
 ***document***.getElementById(**"addEmployeeFormErrorMessage"**)  
 .innerHTML = errorHTML;  
 **if** (errorHTML.length != 0) **return**;

*addEmployee*(name, surname);  
 *showEmployees*(***DATA***.**employees**);  
}

### 5) После добавления сотрудника данные в форме следует очищать.

Для этого в функции *addEmployeeUI()* нужно очищать поля формы после отправки:

***document***.getElementById(**"name"**).**value = "";**

***document***.getElementById(**"surname"**).**value = "";**

**Дополнительные задания:**

1) Реализовать возможность задать дату рождения сотрудника

2) Реализовать валидацию поля ввода даты рождения

## Занятие 2. Расширение функциональности. Работа с событиями.

### 1) Реализуем возможность удаления сотрудника из списка

В цикле функции showEmployees() добавьте код для добавления кнопки Удалить для каждого сотрудника в списке:

**const** removeButton = ***document***.createElement(**"button"**);  
removeButton.innerHTML = **"Удалить"**;  
removeButton.addEventListener(**'click'**,  
 () => *removeEmployeeUI*(employee.**id**));  
li.appendChild(removeButton);

Также необходимо реализовать метод removeEmployeeUI. Эта функция должна вызывать removeEmployee(id) из service.js. После удаления сотрудника необходимо перерисовать список всех сотрудников вызовом showEmployees().

**function** *removeEmployeeUI*(id) {  
 *removeEmployee*(id);  
 *showEmployees*(***DATA***.**employees**);  
}

Код для удаления сотрудника может выглядеть так:

**function** *removeEmployee*(id) {  
 **let** index = 0;  
 **for** (**let** e **of *DATA***.**employees**) {  
 **if** (e.**id**===id) **break**;  
 index++;  
 }  
 ***DATA***.**employees**.splice(index, 1);  
}

Также вы можете использовать findIndex для поиска номера элемента с определенным id.

### 2) Создаем возможность добавлять руководителя.

Для сотрудника можно будет задавать руководителя. Это будет делаться путем выбора из выпадающего списка всех сотрудников.

Для этого в форму добавления сотрудника добавим выпадающий список:

Руководитель:  
<**select id="managerSelect"**></**select**>

Этот список надо заполнить списков всех сотрудников. Для этого реализуйте стандартную функцию **fillSelect(select, values, selectedValue)**.

Данная функция будет принимать ссылку на select в первом параметре, массив объектов с текстом и значением во втором, и выбранное значение - в третьем.

Например, следующий код:

fillSelect(document.getElementById("managerSelect"), [

{ text: " Иван Иванов", value: 133 },

{ text: " Петр Петров", value: 134 }], 133);

Заполнит выпадающий список двумя значениями: Иван Иванов и Петр Петров, при этом Иван Иванов будет выбран изначально.

Вот реализация такого метода:

**function** *fillSelect*(select, values, selectedValue) {  
 **for** (**let** val **of** values) {  
 **const** option = ***document***.createElement(**"option"**);  
 option.text = val.**text**;  
 option.value = val.**value**;  
 **if** (selectedValue==option.value) option.selected=**true**;  
 select.appendChild(option);  
 }  
}

Теперь нам надо создать массив values для заполнения дроп-дауна списком всех сотрудников. Это можно сделать так:

**function** *getEmployeesOptions*() {  
 **let** options = [];  
 **for** (**let** e **of *DATA***.**employees**) {  
 options.push({**text**:e.**name**+**' '**+e.**surname**, **value**:e.**id**});  
 }  
 **return** options;  
}

Остается заполнить дроп-даун списком сотрудников. Для этого добавим в runUI() такой вызов:

*fillSelect*(***document***.getElementById(**"managerSelect"**),  
 *getEmployeesOptions*());

Теперь необходимо при добавлении сотрудника сохранять ссылку на менеджера. Ссылка будет называться managerRef, и будет содержать id сотрудника, являющегося руководителем. Для этого в service.js реализуем функцию setEmployeeManager(id, managerId). Также в функции addEmployeeUI() будем получать и сохранять id менеджера:

**const** id = *addEmployee*(name, surname);  
**const** managerId = ***document***.getElementById(**"managerSelect"**).**value**;  
*setEmployeeManager*(id, managerId);

Теперь нам необходимо выводить руководителя в функции showEmployees(). Это можно сделать следующим образом:

**let** managerHTML = **""**;  
**if** (employee.**managerRef**) {  
 **let** manager = *findById*(employee.**managerRef**);  
 managerHTML = **" <b>Руководитель:</b> "**+manager.**name**

+**" "**+manager.**surname**;  
}  
li.innerHTML = employee.**name**+**" "**+employee.**surname**+managerHTML;

### 3) Добавляем возможность изменить руководителя в списке.

Реализуем возможность показывать руководителя. Для этого в функции showEmployees() в цикле по сотрудникам добавим возможность отображать не просто имя руководителя, а выпадающий список с выбранным руководителем.

Также добавим обработчик события onchange. При изменении руководителя ссылка managerRef должна обновляться.

**if** (employee.**managerRef**) {  
 **let** manager = *findById*(employee.**managerRef**);  
 **const** managerSpan = ***document***.createElement(**"span"**);  
 **const** managerSelect = ***document***.createElement(**"select"**);  
 *fillSelect*(managerSelect, *getEmployeesOptions*(), employee.**managerRef**);  
 managerSelect.addEventListener(**'change'**,  
 () => employee.**managerRef**=managerSelect.value);  
 managerSpan.innerHTML = **" <b>Руководитель:</b> "**;  
 li.appendChild(managerSpan);  
 li.appendChild(managerSelect);  
}

**Дополнительные задания**

1) Реализовать возможность изменить имя, фамилию и дату рождения сотрудника

## Занятие 3. Поиск по сотрудникам. Динамическое изменение стилей.

### 1) Поиск по сотрудникам.

Реализуем возможность искать по сотрудникам.

Искать будем по имени, фамилии, названию департамента и руководителю.

Для этого добавим форму поиска в index.html.

<**form**>  
 Поиск сотрудника:  
  
 <**input id="nameSearch" placeholder="Имя" size="30"**>  
 <**input id="surnameSearch" placeholder="Фамилия" size="30"**>  
  
 Поиск по менеджеру:  
 <**select id="managerSearch"**></**select**>  
  
 <**button type=button onclick="***searchEmployeeUI*()**"**>Найти сотрудников</**button**>  
 <**input type="reset" value="Сбросить форму"**>  
</**form**>

Обратите внимание: здесь реализована кнопка для поиска сотрудников, а также кнопка для сброса данных формы. <input type="reset"> предназначен как раз для сброса данных форм.

Теперь реализуем функцию **searchEmployeeUI()** в файле ui.js:

Данная функция должна получать данные формы, осуществлять поиск и затем выводить результаты, перерисовывая список сотрудников.

Также в файл service.js добавим функцию *searchEmployees*(name, surname, managerRef). Данная функция должна возвращать сотрудников по заданным параметрам. Если какой-то параметр не задан (null или undefined или пустая строка), параметр должен игнорироваться. Таким образом, вызов **searchEmployees("Иван")** должен находить всех Иванов, вне зависимости от фамилии или того, кто его менеджер.

### 2) Отображение результатов валидации путем изменения вида input-полей

Мы будем подсвечивать input-поля розовым в случае некорректности параметров ввода. Для этого мы будем использовать возможность менять стиль у произвольного элемента( также можно это сделать через классы).

Изменить валидацию формы в функции addEmployeeUI(), подсвечивая фон.

### 3) Реализуем интерфейс с закладками

Мы будем использовать работу с CSS для переключения между закладками.

У нас на текущий момент будет 2 закладки: поиск сотрудника и добавление сотрудника.

Сначала реализуем сами закладки. Поместим их в index.html.

Далее опишем необходимые стили в **style.css**.

Теперь поместим форму поиска сотрудника в <**div id="searchPane" class="tabcontent"**>

Также поместим форму добавления сотрудников в <**div id="addPane" class="tabcontent"**>

Теперь нам нужно реализовать функцию **openTab(event, tabId)** в **ui.js**:

*/\*\*  
 \* Активирует выбранный таб  
 \** ***@param*** *evt событие, вызывающее активацию  
 \** ***@param*** *id идентификатор таба  
 \*/***function** *openTab*(evt, id)

Теперь табы должны работать. Остается активировать первый таб при загрузке страницы.

### 4) Реализуем возможность отправки данных по нажатию Ввод в input-полях

Реализуем функцию, которая будет находить все input-поля и навешивать на нее слушатель события keyup. Если нажатая кнопка - кнопка ввода (она имеет код 13), то мы отправляем данные путем нажатия на первую кнопку

**function** *assignSendOnEnter*(paneId, buttonId)

Остается добавить навешивание события при старте в функцию runUI()

Теперь можно проверить, что форма отправляется при нажатии Enter на любом поле ввода.

*На этом реализация возможностей программы "вчерную" завершается, и мы начинаем перерабатывать наш код, вводя современные концепции, которые позволят делать наш код легко читаемым и расширяемым, и позволят минимизировать вероятность ошибок в большом проекте.*

**Дополнительные задания**

1) Реализовать поиск при любом изменении полей поисковой формы (без нажатия на кнопку поиска)

2) Реализовать возможность поиска по дате рождения