# Блок 5. Асинхронное программирование. Работа с сервером

## Занятие 1. Изучаем асинхронное программирование

### Часть 1. Используем промисы

На этом занятии мы будем использовать промисы, позволяющие возвращать результат асинхронной операции.

1. Добавьте метод bonus() в Employee, который должен возвращать Promise, содержащий случайное значение бонуса в диапазоне 0…1000; бонус должен быть вычислен после задержки в 1000ms (это имитирует задержку при обращении к серверу)

bonus() {  
 **return new** Promise(resolve=>  
 *setTimeout*(  
 ()=>resolve(***Math***.round(***Math***.random()\*1000)),  
 1000))  
}

1. Добавьте метод total() в Employee который должен вычислить сумму бонуса и зарплаты и вернуть новый Promise:

total() {  
 **return new** Promise(resolve=>  
 **this**.bonus().then(bonus=>  
 resolve(bonus+**this**.salary)))  
}

Обратите внимание, что здесь мы можем обратиться к **this**.**salary** благодаря lexical scope **this** в стрелочной функции.

Для нестрелочной функции нам бы понадобилось использовать метод bind().

1. В main.js переместите код, который модифицирует HTML в отдельную функцию render():

**function** *render*() {  
 **document**.getElementById(**"employees"**).**innerHTML** = html;  
}

И вызовите render() в конце main.js. Добавьте placeholder employees на страницу.

1. Напечатайте общий доход каждого сотрудника в main.js с использованием промисов:

**let** employees = *jsonToEmployees*(DATA.employees);

**for** (**let** e **of** employees) {  
 e.total().then(total=>{  
 html += **`**${e.**name**} **total:** ${total} **<br>`**;  
 *render*();  
 });  
}

Таким образом, значения доходов будут появляться постепенно, по мере “загрузки”.

### Часть 2. Добавляем обработку исключений

1. Измените метод bonus() в классе Employee так, чтобы он отклонял Promise в случае, если бонус больше 700:

bonus() {  
 **var** bonus = **Math**.round(**Math**.random()\*1000);  
 **return new** Promise((resolve,reject)=>  
 setTimeout(()=>bonus<700?resolve(bonus):reject(bonus),1000))  
}

1. Добавьте метод total() в Employee, который обрабатывает исключения в bonus(). Данный метод должен дожидаться получения бонуса (bonus().then), а затем увеличивать зарплату на размер бонуса и возвращать ее как Promise. В случае исключения в методе bonus() метод total() должен также выбрасывать исключение.
2. Обновите выводимый на печать список сотрудников в main.js (версия с Promise), добавив блок перехвата, который будет выводить предупреждение «Недопустимый размер бонуса» для сотрудника в случае исключения:

**for** (**let** e **of** employees) {  
 e.total()  
 .then(total=>  
 html += **`**${e.**name**} **total:** ${total} **<br>`**)  
 .catch(bonus=>  
 html += **`**${e.**name**} **bonus is too big**

**(**${bonus}**!) <br>`**)  
 .then(*render*)  
}

### Часть 3. Использование синтаксиса async/await

Используйте синтаксис async/await в main.js для вывода на печать списка сотрудников и бонусов.

Добавьте в main.js функцию с синтаксисом async/await:

**async function** *printBonus*() {  
 html += **"<br>Async/await version:<br>"**;  
 **for** (**let** e **of** employees) {  
 **let** bonus = **await** e.bonus();  
 html += **`**${e.**name**} **bonus:** ${bonus}  
 **total:** ${e.**salary**+bonus}**<br>`**;  
 *render*();  
 }  
}

Запустите ее: *printBonus*();

### Самостоятельное задание

Перепишите метод total() с использованием синтаксиса async/await.