async / await | Unit testing

JavaScript Courses

www.courses.dp.ua

async/await

(ECMAScript-2017)

async/await – упрощение кода Promise’ов

```
2
3   let url = 'https://bank.gov.ua/NBUStatService/
    v1/statdirectory/exchange?json';
4
5   (async function(){
6
7       let result  = await fetch(url);
8       result      = await result.json();
9
10      console.log(result);
11
12  })();
13
```

async/await – надстройка над **Promise** позволяющая писать код в полностью привычном синхронном стиле, при этом откладывая ожидания завершения операций до тех пор пока её результат действительно понадобиться;

async – отмечает функцию как асинхронную (результат такой функции оборачивается в **Promise**);

await – при вызову асинхронных функций указывает, что не нужно ждать результата сейчас

Подробнее: https://learn.javascript.ru/async-await

Цикл for-await-of (ES-2018)

```
23
24      let promises = [new Promise(), new Promise(), new Promise(), ];
25
26      for await(p of promises){
27          console.log( p.someResultData );
28      }
29
```

Цикл **for-await-of** позволяет перебрать итерируемую (перебираемую, массив или псевдомассив) состоящий из объектов типа **Promise**. Цикл будет ожидать когда разрешится каждый из **Promis'ов** и только тогда начинать выполнение каждого шага цикла.

Подробнее: https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Statements/for-await...of

Модульное тестирование (Unit Testing) и функции

Unit testing – модульное тестирование

```
2
3   function calc_sum(a, b){
4       let result = a + b;
5       return result;
6   }
7
8   (function(){
9       let control = calc_sum(2, 3);
10
11      if(control === 5){
12          console.log("calc_sum() - OK");
13      }else{
14          console.log("calc_sum() - FAIL");
15      }
16  })();
17
```

Идея **модульного тестирования** (**Unit testing**) в том, чтобы писать код который будет проверять работу основного кода. Функция, как пример модуля, может быть протестирована другой, написанной нами функцией. Основная польза модульного тестирования в том, что при изменении кода функции мы может оперативно определить не поломался ли её функционал.

Подробнее: https://habr.com/ru/post/169381/

Unit testing – модульное тестирование

```
2
3     function calc_sum(a, b){
4         let result = a + b+1;
5         return result;
6     }
7
8     (function(){
9         let control = calc_sum(2, 3);
10
11        console.assert(control === 5, "TEST: calc_sum(2,3)");
12
13    })();
14
```

Метод **console.assert()** – удобный способ добавить вывод информации об ошибках в консоль разработчика.

Подробнее: https://developer.mozilla.org/ru/docs/Web/API/Console/assert

Полезные методы .toString() / .valueOf() у объектов

Методы .toString()/.valueOf() у объектов

```
2
3     let user = {
4         name: 'Ivan',
5         lastName: 'Ivanov',
6         age: 28,
7         toString: function(){
8             return `USER: ${this.name} ${this.lastName}`;
9         },
10        valueOf: function(){
11            return this.age;
12        }
13    }
14
15    alert(user);
16    console.log('Number:',  +user );
17
```

Метод **.toString()**, если он определен у объекта – позволяет браузеру корректно преобразовать объект **к строке**. Также есть метод **.valueOf()** для преобразования **к числу**.

Подробнее: https://learn.javascript.ru/object-conversion

Объект **Date**
или
о работе с датой/временем

Дата/Время в JavaScript

```
2
3     let currentDateTime = new Date();
4     console.log(currentDateTime);
5     console.log(currentDateTime.toUTCString());
6
7     let dateA = new Date(2019, 10, 18, 17, 23, 56);
8
9     console.log(dateA, +dateA);
10
```

В JavaScript есть (*относительно*) удобные возможность работы с датой и временем – объект **Date**. Дату можно преобразовать к **UTC-**виду, и получить отдельные её компоненты (год, месяц, … минуты, секунды).

Подробнее: https://learn.javascript.ru/datetime

Дата/Время в JavaScript

```
2
3     let newYear2020 = new Date(2020, 0, 1, 0,0,0);
4     let now         = new Date();
5
6     let diff = newYear2020 - now;
7
8     diff = Math.floor(diff / (1000 * 60 * 60 * 24));
9
10    console.log(`New Year 2020 after ${diff} days`);
11
```

Две даты можно вычитать одну из другой, в результате мы можем получить разницу в миллисекундах между этими датами. Это возможно за счёт преобразования даты к числу (**Timestamp'y**) которое показывает кол-во миллисекунд прошедшее от начала Unix-эпохи.

Подробнее: https://learn.javascript.ru/datetime

Дата/Время в JavaScript

Важные моменты при работу с датой/временем:

1) Не забывать про разницу между местным и UTC-временем;
2) Не забывать про смещение (метод: **.getTimezoneOffset()**);
3) Помнить о возможности преобразования даты времени в Timestamp и обратно;
4) Помнить о возможности выполнять вычитание дат (и тем самым находить продолжительность какого-либо процесса).

Подробнее: https://habr.com/ru/company/mailru/blog/438286/

API Нової Пошти

API компании «Нова Пошта»

Для работы с сервисом потребуется ключ

44e91d173783612570185e7b41eea14c

https://devcenter.novaposhta.ua/

Пример запроса к API «Нова Пошта»

```
2
3   let key = '44e91d1737836125701 85e7b41eea14c';
4   let url = 'https://api.novaposhta.ua/v2.0/json/';
5
6   let params = {
7       "modelName": "Address",
8       "calledMethod": "getCities",
9       "apiKey": key
10  }
11
12  fetch(url, {
13      method: 'POST',
14      body: JSON.stringify(params)
15  }).then(answer => answer.json()).then(answer => console.log(answer));
16
```

Параметры запроса передаются не в URL а в теле запроса,
также запрос выполняется методом **POST**.

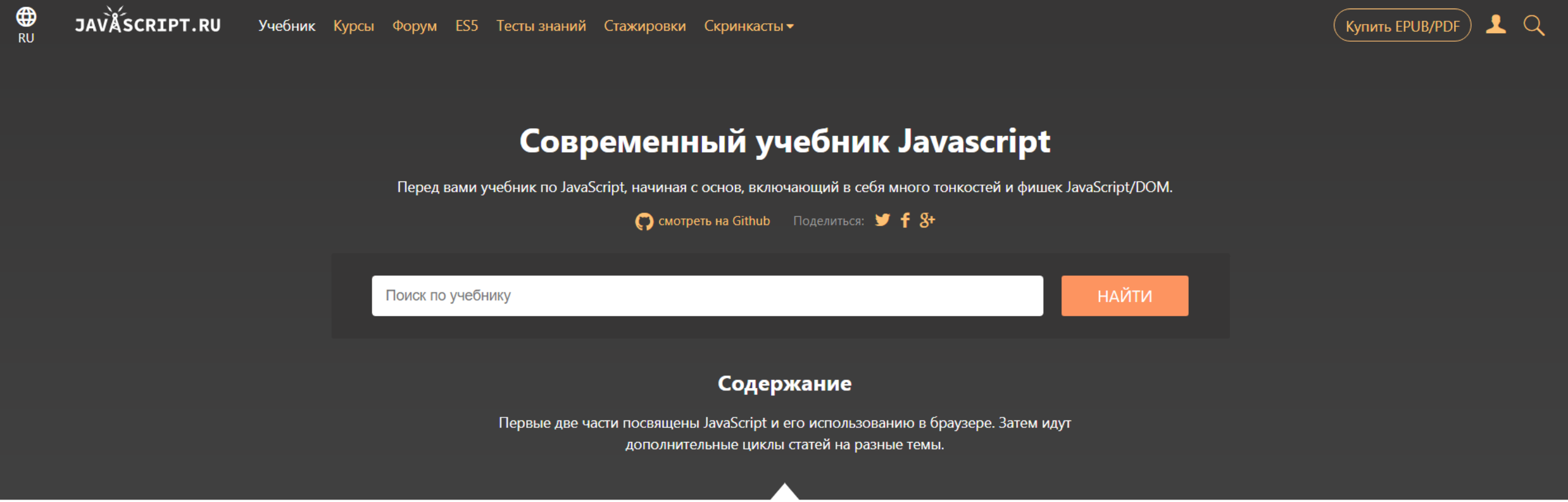
О HTTP-запросах

```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh;… )… Firefox/51.0
Accept: text/html,application/xhtml+xml,…,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-12656974
Content-Length: 345

-12656974
(more data)
```

HTTP-запрос состоит из заголовков и тела запроса, при работе с API «Нова Пошта» параметры передаются в теле запроса, которое в запросе идёт после заголовков.

Домашнее задание
/узнать



Предварительные знания – лучший помощник в обучении, поэтому к следующему занятию жду, что **пройдёте разделы 1-й части 11.8, также из 2-й части 1.1 и из 3-й части 3.1**

http://learn.javascript.ru/

Домашнее задание
/сделать

Домашнее задание #F.1 | «Проверка ИНН»

КАРТКА

фізичної особи - платника податків

повідомляє, що

одержав(ла) ідентифікаційний номер
наданий Державною податковою адміністрацією України
згідно з даними, заповненими ним (нею) в обліковій картці.
Дата занесення до Державного реєстру фізичних осіб - 06/02/1998
(картка видана для пред'явлення до органів державної реєстрації,
установ банків та інших).

М.П. Для довідок

(підпис) (прізвище та ініціали посадової особи органу Державної податкової служби)

(дата видачі картки)

Для проверки:
***3463463460** – пол женский, д.р. 28.10.1994;*
***2063463479** – пол мужской, д.р. 29.06.1956.*

*Пользователь вводит ИНН (физ. лица Украины), Необходимо определить: **нет ли ошибки в коде**, узнать **дату рождения**, определить **пол** и сколько **полных лет** человеку.*

Скрипт должен содержать **функцию**, которая принимает **ИНН** в виде строки (строк может содержать проблемы, необходимо отчистить её). По результатам работы функция должна возвращать объект следующей структуры (поля **sex, dateOfBirth** и **fullYears** для некорректного номера не создаются):

```
{
        code: "1234567890",
        isCorrect: true, //or false
        sex: "female", //or "male"
        dateOfBirth: "1988-12-23",
        fullYears: 29
}
```

И не забудьте написать UNIT-test к созданной функции!

Домашнее задание #F.2

Игра в кости

Необходимо написать скрипт который позволяет играть в кости.

Игроку при старте даётся 1000 гривен. У игрока спрашивается какую сумму он ставит (только целые числа), и на какой результат (от 1 до 12 включительно). После этого компьютер «бросает кости» генерирует два числа от 1 до 6 включительно. Если сумма чисел совпала с загаданным числом пользователя он получает удвоенную ставку, если при этом оба выпавшие числа равны между собой то пользователь получает утроенную ставку. О результатах каждого «бросания» необходимо уведомлять пользователя и о сумме его выигрыша или проигрыша.

Игра продолжается до тех пор пока пользователь не накопит 10 000 гривен или пока у него не закончатся деньги. В случае выигрыша необходимо уведомить пользователя о том, сколько раундов у него заняла игра.