

ООП и JavaScript, часть 2

JS
COURSE
ORT DNIPRO

ORT**DNIPRO**.ORG/**JS**

1. Классы

```

class Parcel{
    #code;
    #width;
    #length;
    #height;

    constructor(code, w, l, h){
        this.#code = code;
        this.#width = w;
        this.#length = l;
        this.#height = h;
    }

    getVolume(){
        return this.#width * this.#length * this.#height;
    }

    getReport(){
        return `Parcel ${this.code}: ${this.getVolume()}`;
    }
}

let box = new Parcel(100, 20, 45);

console.log(box.getReport());

```

Классы в JavaScript

Классы пришли в JavaScript из других (типизированных) языков программирования. В которых классы применяли для описание структуры объектов которые на основе класса создаются. **Класс** выступают своего рода «чертежом» по которому будут создаваться объекты.

Классы в JavaScript/ECMAScript

Классы в JavaScript'е являются лишь надстройкой («маскировкой», «синтаксическим сахаром») над **прототипной** моделью построения объектов. И не являются её заменой.

Классы в JavaScript/ECMAScript

По сути описывая **класс** мы создаём функцию **конструктор** в которой идёт перечисление свойств и методом будущего объекта. А далее эта функция вызывается через оператор **new**.

2. Глобальный объект `globalThis` (`window`)

Глобальный объект **window** (**globalThis**)

Браузер добавляет в JavaScript всего один объект – **window (globalThis)**. Но этот объект содержит все необходимые инструменты для манипуляции HTML-документом.

Глобальный объект **window**

Объект **window** можно использовать неявно, т.е. опускать его имя при написании кода.

Свойства и методы window

```
.setInterval() ;  
.setTimeout() ;  
.alert() ;  
.prompt() ;  
.confirm() ;  
...
```


3. Множество (Set)

Множество / Set

```
2
3  let set = new Set();
4
5  set.add("Jhon");
6  set.add("Helen");
7  set.add("Jhon");
8  set.add("Maria");
9  set.add("Jane");
10 set.add("Bill");
11
12 console.log(set); // {"Jhon", "Helen", "Maria", "Jane", "Bill"}
13
14 console.log( set.has('Maria') ); //true;
15 console.log( set.has('Samuel') ); //false
16
17 set.delete("Jane");
18 console.log(set); //{"Jhon", "Helen", "Maria", "Bill"}
19
```

Set – коллекция без ключей (создаётся при помощи ключевого слова **new**), позволяет хранить любые типы данных. Элемент множества встречаться в нём не более чем один раз. Есть возможность узнать есть ли элемент во множестве (метод **.has(...)**), а также узнать размер множества (свойство **.size**).

Избавление от дубликатов при помощи Set

```
2
3 let arr = ['Jane', 'Jhon', 'Maria', 'Alice', 'Jane', 'Peter', 'Alice', 'Donald'];
4
5 console.log(arr); //["Jane", "Jhon", "Maria", "Alice", "Jane", "Peter", "Alice", "Donald"]
6 console.log("Array length:", arr.length); //8
7
8 let set = new Set(arr);
9
10 console.log(set); // {"Jane", "Jhon", "Maria", "Alice", "Peter", "Donald"}
11 console.log(set.size); //6
12
13 let new_arr = Array.from(set);
14
15 console.log(new_arr); //["Jane", "Jhon", "Maria", "Alice", "Peter", "Donald"]
16 console.log("New array length:", new_arr.length); //6
17
```

4. Объект Date

Дата/Время в JavaScript

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

В JavaScript есть (относительно) удобная возможность работы с датой и временем – объект **Date**. Дату можно преобразовать к **UTC**-виду или **timestamp**'у, 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 работе 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);
- 5) JavaScript даёт определённые возможности по форматирование вывода даты/времени, при помощи методов **.toLocaleString()**, **.toLocaleDateString()**, **.toLocaleTimeString()**. Но эти возможности крайне ограничены.

Подробнее: <https://habr.com/ru/companies/vk/articles/438286/>

5. Принципы модульного тестирования (Unit Testing)

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.

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>

6. Немного практики

Замеры времени выполнения кода

`performance.now()`

```
2
3   let t1 = performance.now();
4
5   for(let i = 0; i < 1000000; i++){
6       //...do something HARD
7   }
8
9   let t2 = performance.now();
10
11  console.log('Time for HARD work (ms):', t2 - t1);
12
```

Метод **performance.now()** возвращает в миллисекундах временную метку. При сравнении двух и более временных меток можно определить время прошедшее между их получением.

Таблица умножения

Разработать скрипт, проверяющий знания таблицы умножения двузначных чисел. Скрипт должен задать пользователю 12 задач на умножение двузначных чисел....

Дополнительное условие: если на ответ пользователю понадобилось более 20 секунд – считать его не верным. Но в консоли указывать для такого ответа, что ошибка именно из-за времени.

На следующем занятии

На следующем занятии

Работа с DOM

(работа с разметкой документа)


Домашнее задание
/сделать


Домашнее задание #F.1

«Проверка ИНН»

КАРТКА
фізичної особи - платника податків

повідомляє, що Григорук Володимир Григорович
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

 (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: **корректен ли код, узнать дату рождения, определить пол и сколько полных лет человеку.**

Демо-пример: <https://decode.org.ua/inn>