Классы ES2015/2019

JavaScript
Courses

www.courses.dp.ua

```
2
3     class Parcel{
4         #code;
5         #width;
6         #length;
7         #height;
8
9         constructor(code, w, l, h){
10            this.#code      = code;
11            this.#width     = w;
12            this.#length    = l;
13            this.#height    = h;
14        }
15
16        getVolume(){
17            return  this.#width * this.#length * this.#height;
18        }
19
20        getReport(){
21            return `Parcel ${this.code}: ${this.getVolume()}`;
22        }
23    }
24
25    let box = new Parcel(100, 20, 45);
26
27    console.log(box.getReport());
28
```

Классы в ECMAScript 2015-2019

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.

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

Классы в ECMAScript 2015-2019

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

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

Классы в ECMAScript 2015-2019

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**.

*В ES2019 была добавлена возможность создавать **приватные** (закрытые) свойства и методы. К этим методам есть возможность обратится только из методов объекта. Из вне они недоступны. Их легко отличить по символу # в начале имени.*

Подробнее: https://learn.javascript.ru/private-protected-properties-methods

```
2
3   class Person{
4       #name;
5       #lastName;
6
7       constructor(name, lastName){
8           this.#name      = name;
9           this.#lastName  = lastName;
10      }
11      getInfo(){
12          return `${this.#name} ${this.#lastName}`;
13      }
14  }
15
16  class Driver extends Person{
17      #driverLicense;
18
19      constructor(name, lastName, driverLicense){
20          super(name, lastName);
21          this.#driverLicense = driverLicense;
22      }
23
24      getInfo(){
25          return super.getInfo() + `, Driver License: ${this.#driverLicense} `;
26      }
27  }
28
29  let driver = new Driver('Jhon', 'Smith', 'AA324356');
30
31  console.log(driver.getInfo());
32
```

Наследование

Класс может расширять функционал (наследовать) другого (родительского) класса, а по сути добавлять в него дополнительные методы и свойства. Для указание этого применяется ключевое слово **extends**. В конструкторе дочернего класса (и в его методах) можно обращаться к конструктору (и методам) класса родителя, для этого применяется ключевое слово **super**.

Подробнее:
https://learn.javascript.ru/extend-natives

Статические свойства и методы

```
2
3     class Demo{
4         static #counter = 0;
5
6         static getNext(){
7             return ++this.#counter;
8         }
9     }
10
11    console.log( Demo.getNext() );
12    console.log( Demo.getNext() );
13    console.log( Demo.getNext() );
14
```

Статические свойства и методы помечаются ключевым словом **static** к ним можно обращаться без создания экземпляра класса (объекта). Статические свойства и методы хороши для данных которые являются общими для всех объектов класса.

Подробнее: https://learn.javascript.ru/static-properties-methods

глобальный объект window

Глобальный объект window

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

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

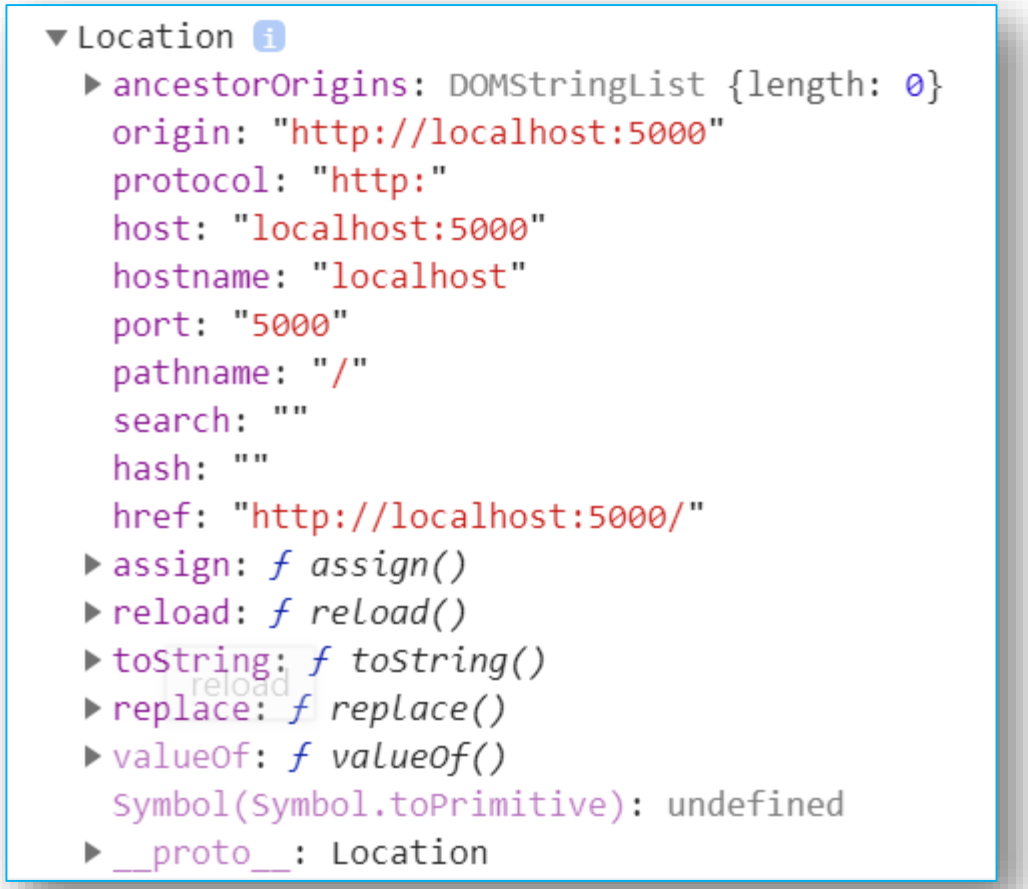
Глобальный объект window

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

Свойства и методы window

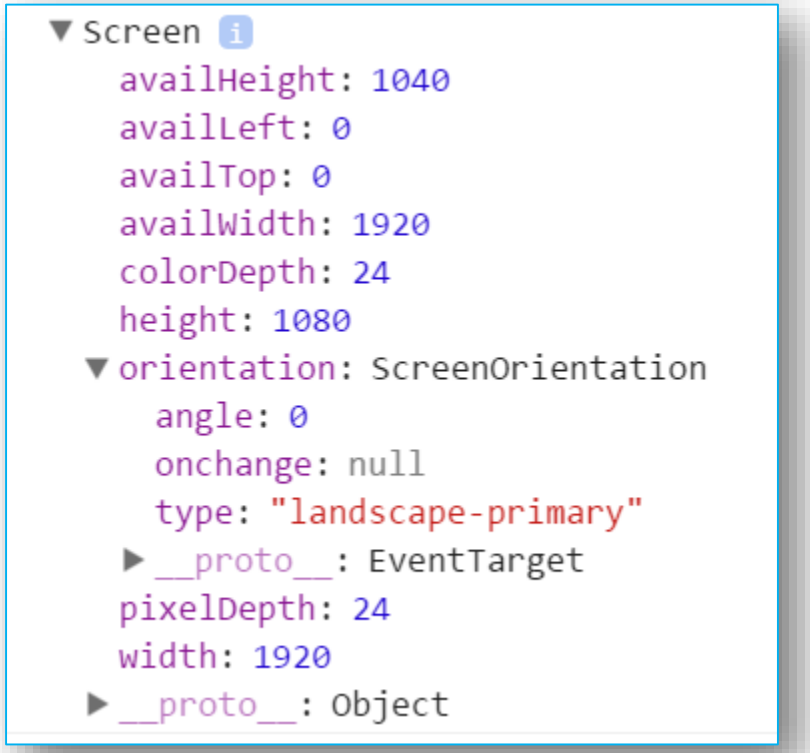
```
.setInterval();
.setTimeOut();
.alert();
.prompt();
.confirm();
…
```

Глобальный объект window



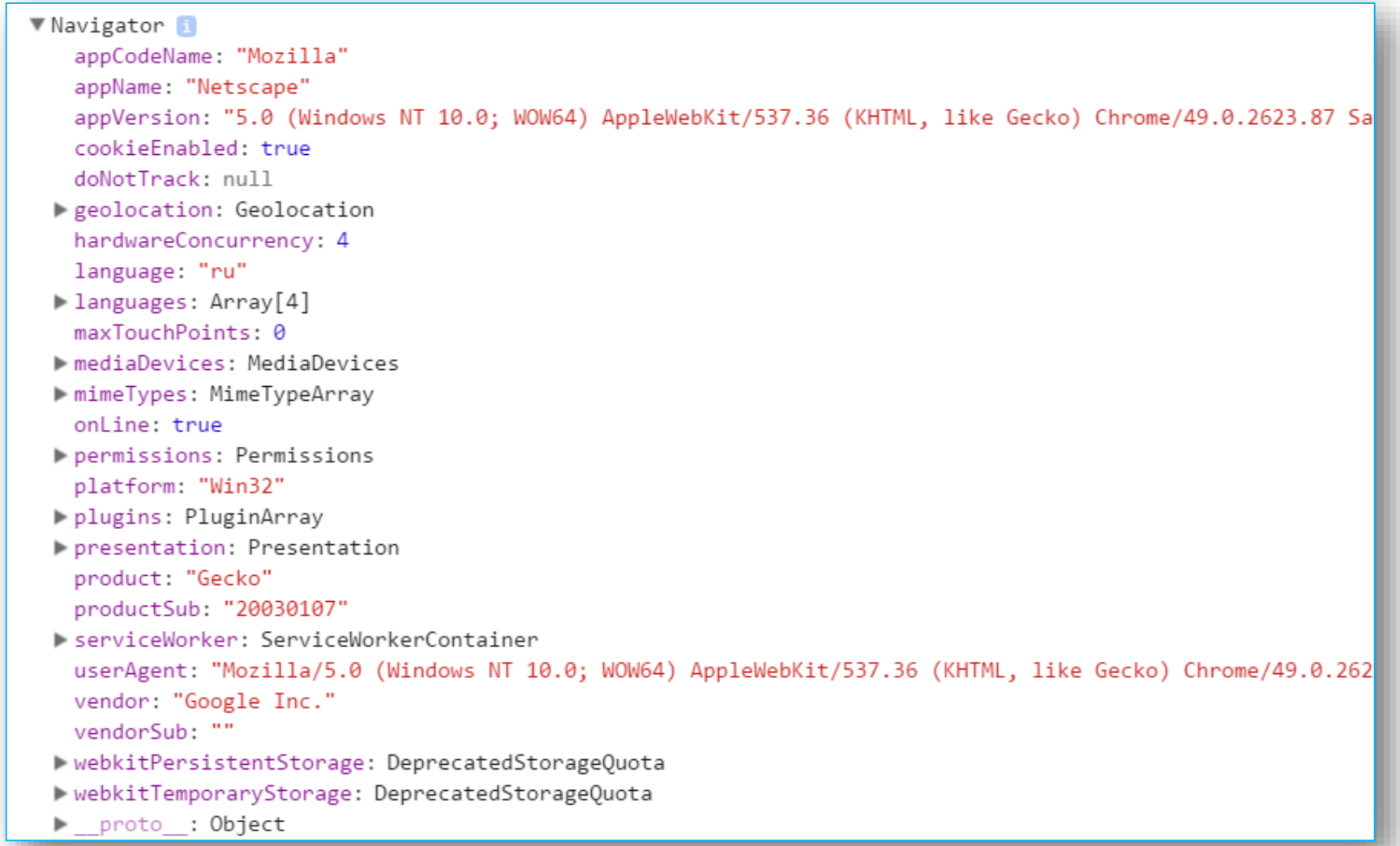
window.location – свойство определяющее какую страницу содержит окно браузера.

Глобальный объект window



window.screen – информация об экране, размерах, ориентации и т.д.

Глобальный объект window



window.navigator – информация о браузере.

window.document

(корень DOM-дерева)

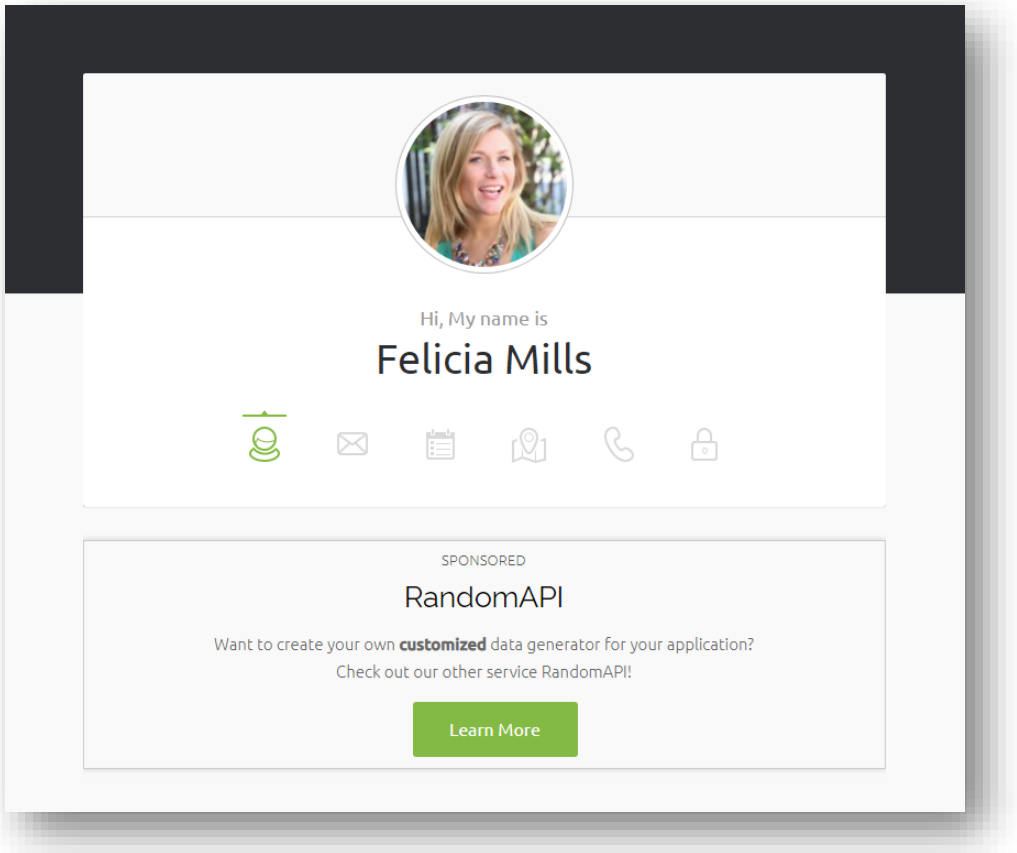
хранилище HTML-документа

Подробнее: https://learn.javascript.ru/dom-nodes

Воспользуйтесь шаблоном /sourse/document.html из репозитория этого занятия

Полезный API-сервис
Генерация данных пользователей

RANDOM USER GENERATOR

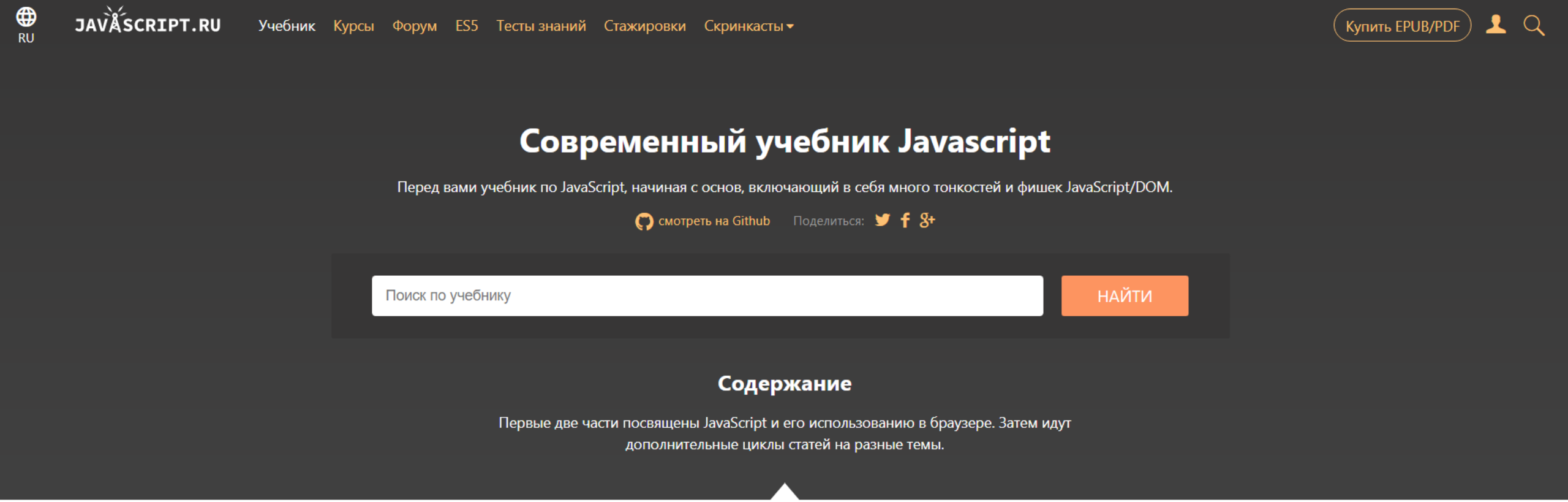


Сервис генерирует (фейковые) данные пользователей. Что может быть крайне полезным для наполнения и тестирования страниц наших приложений.

https://randomuser.me/

https://randomuser.me/documentation#multiple

Домашнее задание
/узнать



Предварительные знания – лучший помощник в обучении, поэтому к следующему занятию жду, что **пройдёте разделы 2-й части 1.1-1.11**

http://learn.javascript.ru/