**Лабораторная работа № 9. Встроенные и пользовательские объекты Java Script.**

(трудоемкость 2 часа)

**Цель**: изучить встроенные объекты JavaScript Array,String,Date, их а также научиться способам создания пользовательских объектов, познакомиться с правилами применения специальных операторов: delete, in, instanceof, typeof..

**Порядок выполнения лабораторной работы:**

1. Изучить встроенные объекты JavaScript Array,String,Date, их свойства и методы, научиться их правильно применять в своих скриптах (см. Приложение 1 и презентацию Лекция 12\_javascript-vstroennye-obekty.pptx).

* Создать массив, содержащий элементы любого типа на Ваш выбор. Добавить в начало и в конец массива еще по одному элементу. Вывести исходный массив, полученный массив и их длину (количество элементов).
* Создать объект **String** – строку текста (свои Фамилия Имя Отчество), в которой присутствуют строчные и прописные буквы. Узнать ее длину. Перевести все символы строки в верхний регистр, а затем в нижний. Соединить полученные строки. Заменить свои Фамилия Имя Отчество на ФИО. Вывести исходную и полученные строки на страницу.
* Использовать объект **Date**. Вывести на страницу таблицу с составляющими текущей даты и времени в виде:

|  |  |
| --- | --- |
| Год | 2022 |
| Месяц |  |
| День |  |
| Час |  |
| Минуты |  |
| Секунды |  |

Применить к ячейкам таблицы свойства форматирования.

1. Изучить правила создания и манипулирования пользовательскими объектами (см. Приложение 2).
2. Выполнить ***задание 2:***

Создайте собственный объект, содержащий не менее 5-ти свойств различных типов (в соответствии с Вашей индивидуальной тематикой).

Выведете его свойства (с помощью оператора цикла for..in).

Удалите одно из свойств и проверьте результат удаления.

Напишите функцию, осуществляющую вычисления на основе числовых свойств объекта и проверьте ее действие.

1. Оформить отчет, содержащий[[1]](#footnote-1):

- титульный лист;

- цель работы;

- краткое описание хода выполнения работы с листингом кода обновленных web-страниц и сценариев Java Script для ***заданий №1***и***№2****.*

Приложение 1

## [Обзор объектов](https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Working_with_Objects#%D0%BE%D0%B1%D0%B7%D0%BE%D1%80_%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BE%D0%B2)[[2]](#footnote-2)

Объекты в JavaScript, как и во многих других языках программирования, похожи на объекты реальной жизни. Концепцию объектов JavaScript легче понять, проводя параллели с реально существующими в жизни объектами.

В JavaScript объект — это самостоятельная единица, имеющая свойства и определённый тип. Сравним, например, с чашкой. У чашки есть цвет, форма, вес, материал, из которого она сделана, и т.д. Точно так же, объекты JavaScript имеют свойства, которые определяют их характеристики.

## [Объекты и свойства](https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Working_with_Objects#%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%8B_%D0%B8_%D1%81%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%B0)

В JavaScript объект имеет свойства, ассоциированные с ним. Свойство объекта можно понимать как переменную, закреплённую за объектом. Свойства объекта в сущности являются теми же самыми переменными JavaScript, за тем исключением, что они закреплены за объектом. Свойства объекта определяют его характеристики. Получить доступ к свойству объекта можно с помощью точечной записи:

objectName.propertyName

## Встроенные объекты[[3]](#footnote-3)

**Объект Array (Массив)**

Массив представляет собой упорядоченный набор данных. Его удобно представить себе в виде одностолбцовой таблицы, содержащей некоторое количество строк. В ячейках такой таблицы могут находиться данные любого типа, в том числе и массивы.

Можно создать массивы тремя разными способами.

Первый cпособ:

spec = new Array();

spec[0] = ИСиТ;

spec[1] = "ПОИТ";

spec[2] = "ПОИБМС";

spec[3] = "ДЭВИ";

Второй способ:

var spec = new Array("ИСиТ","ПОИТ","ПОИБМС","ДЭВИ");

Третий способ:

var spec = [("ИСиТ","ПОИТ","ПОИБМС","ДЭВИ")];

Нумерация индексов в массивах начинается не с 1, а с 0. С помощью свойства **length** можно узнать количество элементов в массиве.

Методы Array:

* **concat()** объединяет два и более массива в один;
* **рор()** – удаляет последний элемент массива и возвращает его значение;
* **push(значение|объект)** – добавляет к массиву указанное значение в качестве последнего элемента и возвращает новую длину массива;
* **shift()** – удаляет первый элемент массива и возвращает его значение;
* **sliсe(индекс1 [, индекс2])** – создает массив из элементов исходного массива с индексами указанного диапазона;
* **sort([функция\_сортировки])** – сортирует (упорядочивает) элементы массива с помощью функции сравнения.

**Объект String (Строка)**

Объект String (строковый объект) используется для хранения и обработки текстовой информации.

Примеры создания:

mystring = new String ("Привет!")– первый способ;

mystring = "Привет!" – второй способ.

С помощью свойства **length**можно узнать длину строки:

document.write(mystring.length)

С помощью метода **toUpperCase()**можно перевести все символы текста в верхний регистр, а с помощью**toLowerCase()**– в нижний.

Метод **concat()**позволяет объединить две и более строки и вывести результат на страницу:

str='Кафедра';

document.write(str.concat('ИСиТ'));

Метод **replace()**позволяет заменить одно произвольное слово в строке на другое:

document.write(str.replace('ИСиТ','ПОИТ')).

**Объект Date (Дата)**

Объект **Date**позволяет производить различные операции с датой и временем. Некоторые методы объекта представлены в таблице.

Таблица 7.

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| getDate() | Возвращает день месяца (может принимать значения от 1-31) заданной даты. |
| getDay() | Возвращает день недели (может принимать значения от 0-6, причем 0-Воскресенье, а 6-Суббота) заданной даты. |
| getFullYear() | Возвращает год (4 числа, например 2017) заданной даты. |
| getHours() | Возвращает час (может принимать значения от 0-23) заданной даты. |
| getMiliseconds() | Возвращает миллисекунду (может принимать значения от 0-999) заданной даты. |
| getMinutes() | Возвращает минуту (может принимать значения от 0-59) заданной даты. |
| getMonth() | Возвращает месяц (может принимать значения от 0-11) заданной даты. |
| getSeconds() | Возвращает секунду (может принимать значения от 0-59) заданной даты. |
| toDateString() | Преобразует часть объект, содержащую дату, в строку. |
| toString() | Преобразует объект в строку. |
| toTimeString() | Преобразует часть объекта, содержащую время, в строку. |

После того, как объект создан, можно с помощью доступных методов производить над ним различные операции.

Пример.

//Создадим объект Date

x=new Date();

//Извлечем день месяца и выведем результат на страницу

document.write(x.getDate());

//Извлечем год из объекта x и выведем результат на страницу

document.write(x.getFullYear());

Приложение 2[[4]](#footnote-4)

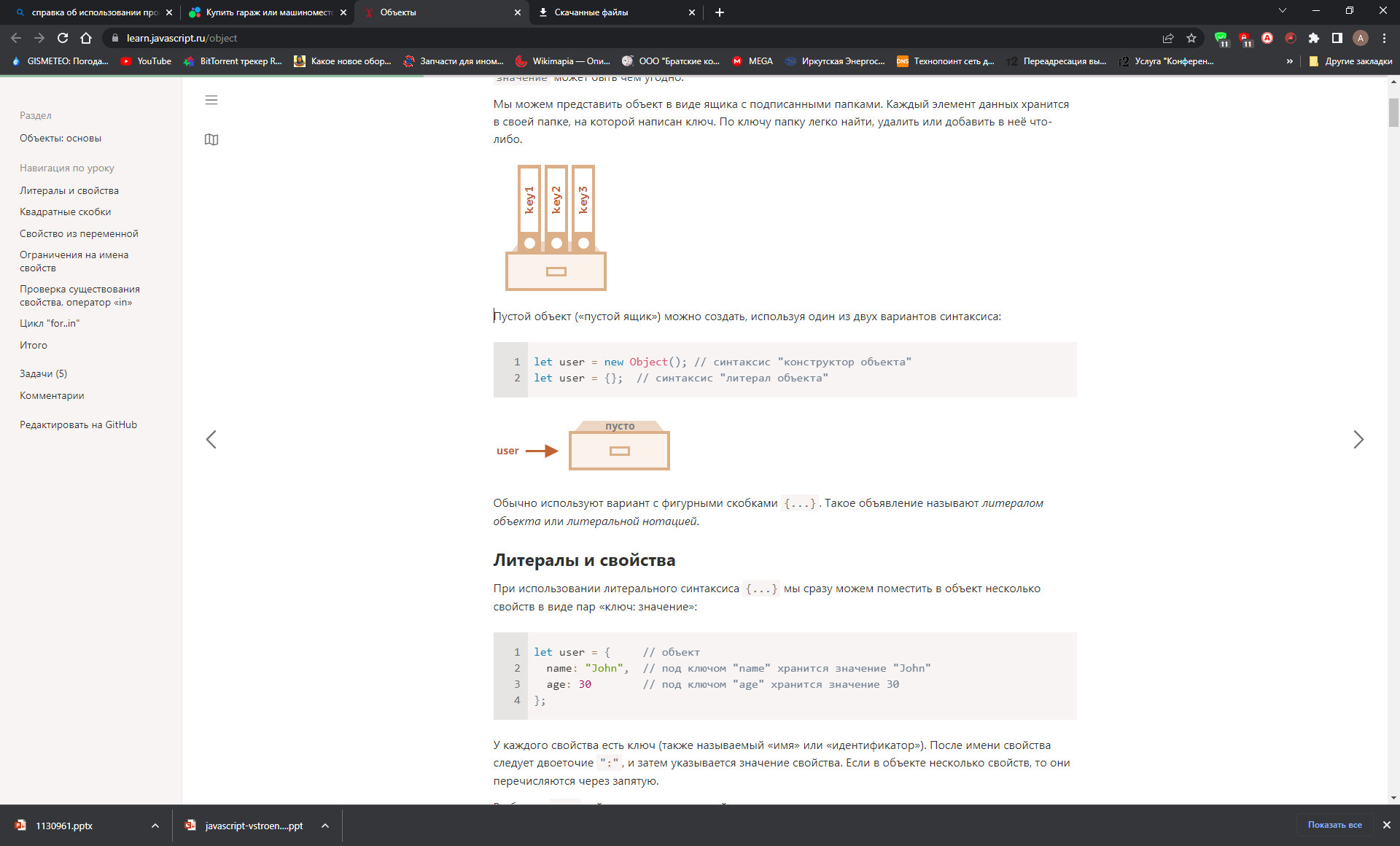
## Сущность понятия «Пользовательские или собственные объекты»

Как мы знаем из главы [Типы данных](https://learn.javascript.ru/types), в JavaScript существует 8 типов данных. Семь из них называются «примитивными», так как содержат только одно значение (будь то строка, число или что-то другое).

Объекты же используются для хранения коллекций различных значений и более сложных сущностей. В JavaScript объекты используются очень часто, это одна из основ языка. Поэтому мы должны понять их, прежде чем углубляться куда-либо ещё.

Объект может быть создан с помощью фигурных скобок {…} с необязательным списком свойств. Свойство – это пара «ключ: значение», где ключ – это строка (также называемая «именем свойства»), а значение может быть чем угодно.

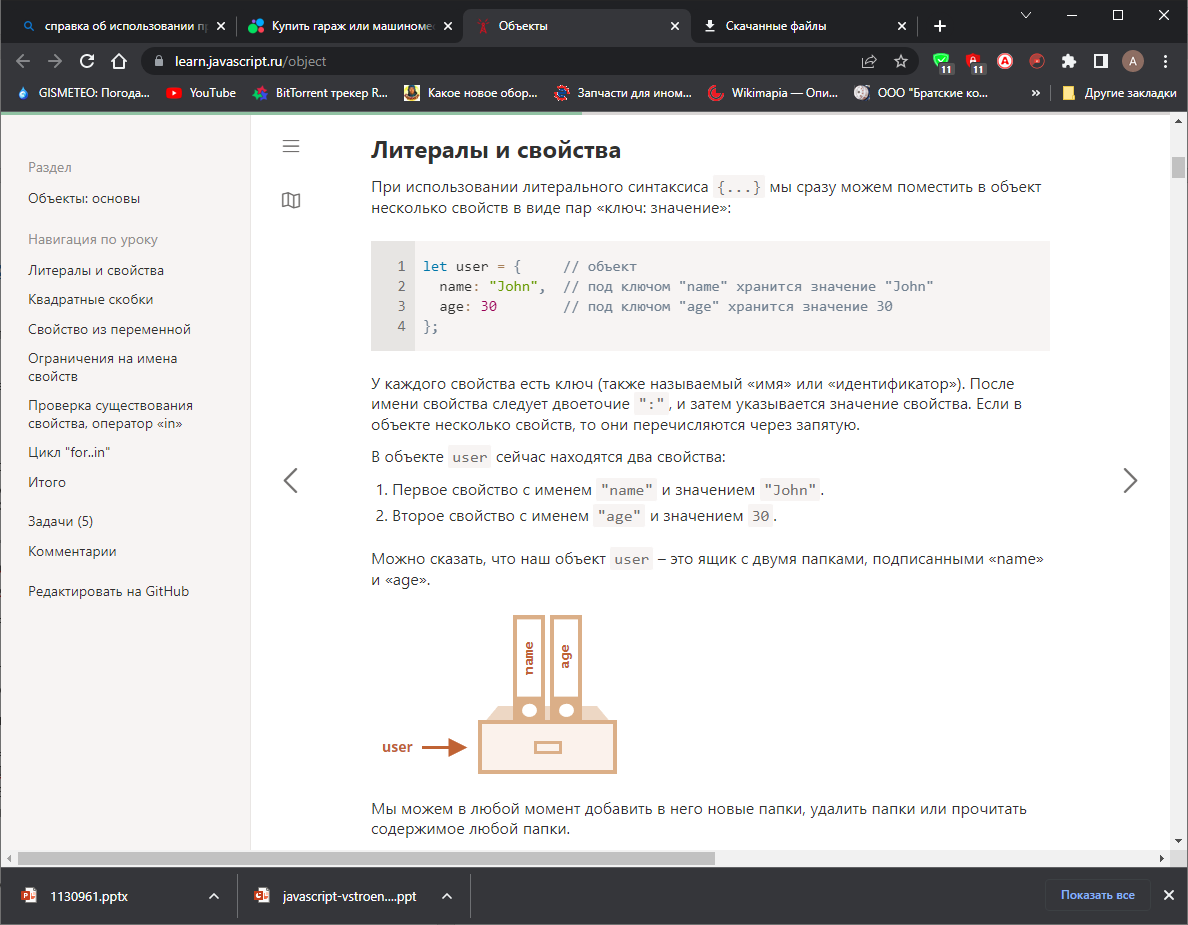
Мы можем представить объект в виде ящика с подписанными папками. Каждый элемент данных хранится в своей папке, на которой написан ключ. По ключу папку легко найти, удалить или добавить в неё что-либо.



Обычно используют вариант с фигурными скобками {...}. Такое объявление называют литералом объекта или литеральной нотацией.

## [Литералы и свойства](https://learn.javascript.ru/object" \l "literaly-i-svoystva) (способы создания и объявления новых объектов)

При использовании литерального синтаксиса {...} мы сразу можем поместить в объект несколько свойств в виде пар «ключ: значение»:

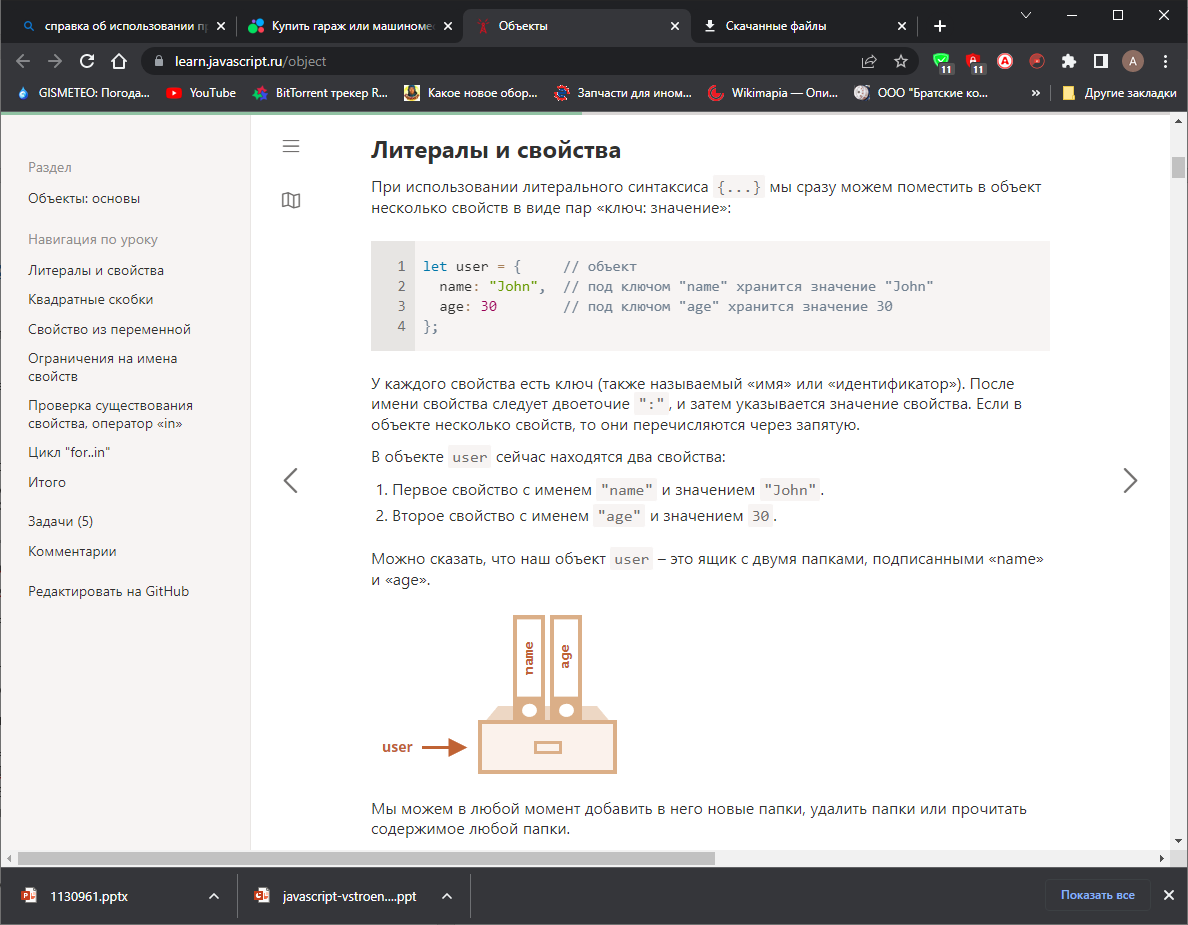
У каждого свойства есть ключ (также называемый «имя» или «идентификатор»). После имени свойства следует двоеточие ":", и затем указывается значение свойства. Если в объекте несколько свойств, то они перечисляются через запятую.

В объекте user сейчас находятся два свойства:

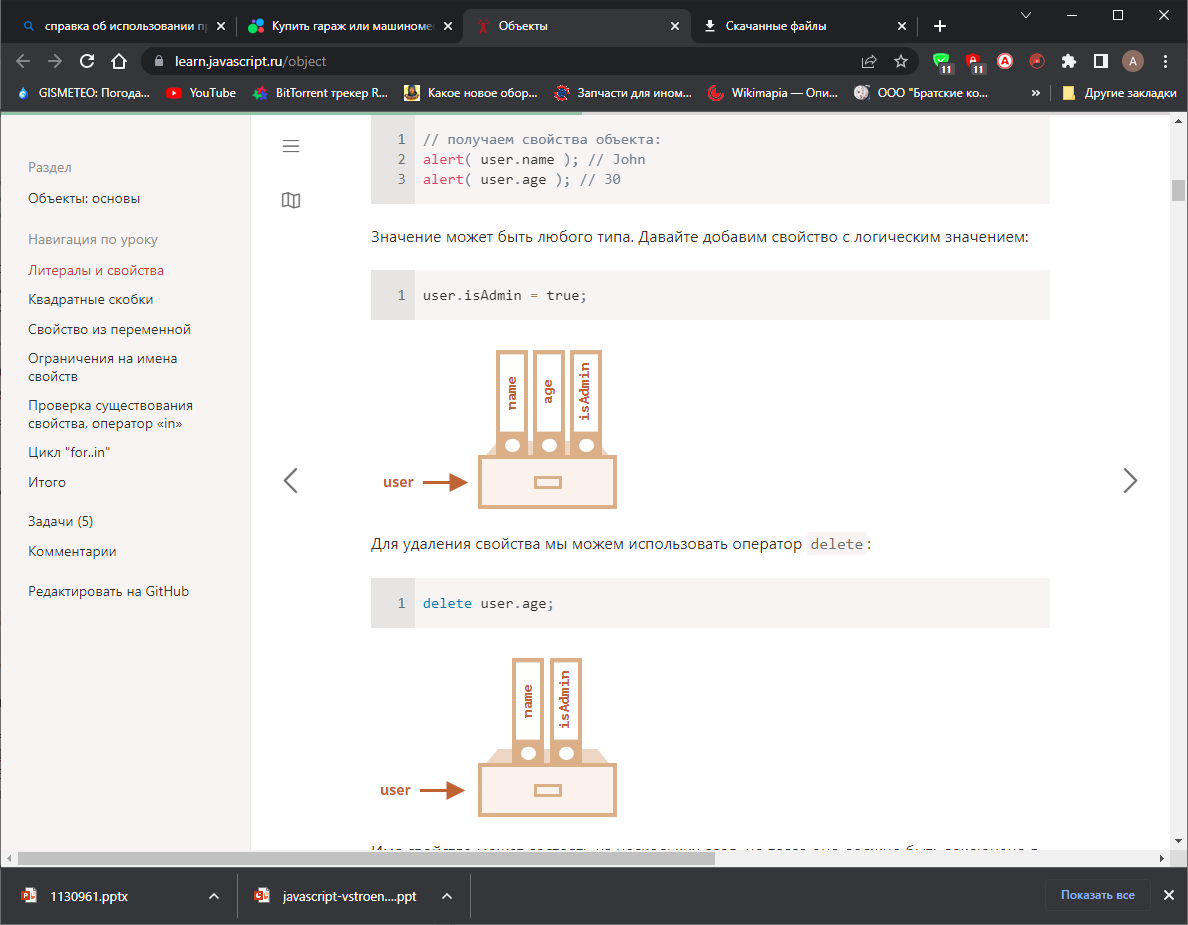
1. Первое свойство с именем "name" и значением "John".
2. Второе свойство с именем "age" и значением 30.

Можно сказать, что наш объект user – это ящик с двумя папками, подписанными «name» и «age».

Мы можем в любой момент добавить в него новые папки, удалить папки или прочитать содержимое любой папки.



Для обращения к свойствам используется запись «через точку»:



Имя свойства может состоять из нескольких слов, но тогда оно должно быть заключено в кавычки:

**let user = {**

**name: "John",**

**age: 30,**

**"likes birds": true // имя свойства из нескольких слов должно быть в кавычках**

**};**

Последнее свойство объекта может заканчиваться запятой:

**let user = {**

**name: "John",**

**age: 30,**

**}**

Это называется «висячая запятая». Такой подход упрощает добавление, удаление и перемещение свойств, так как все строки объекта становятся одинаковыми.

## [Квадратные скобки](https://learn.javascript.ru/object" \l "kvadratnye-skobki)

Для свойств, имена которых состоят из нескольких слов, доступ к значению «через точку» не работает:

// это вызовет синтаксическую ошибку

**user.likes birds = true**

JavaScript видит, что мы обращаемся к свойству user.likes, а затем идёт непонятное слово birds. В итоге синтаксическая ошибка.

Точка требует, чтобы ключ был именован по правилам именования переменных. То есть не имел пробелов, не начинался с цифры и не содержал специальные символы, кроме $ и \_.

Для таких случаев существует альтернативный способ доступа к свойствам через квадратные скобки. Такой способ сработает с любым именем свойства:

**let user = {};**

// присваивание значения свойству

**user["likes birds"] = true;**

// получение значения свойства

**alert(user["likes birds"]); // true**

// удаление свойства

**delete user["likes birds"];**

Сейчас всё в порядке. Обратите внимание, что строка в квадратных скобках заключена в кавычки (подойдёт любой тип кавычек).

Квадратные скобки также позволяют обратиться к свойству, имя которого может быть результатом выражения. Например, имя свойства может храниться в переменной:

let key = «likes birds»;

// то же самое, что и user[«likes birds»] = true;

user[key] = true;

Здесь переменная key может быть вычислена во время выполнения кода или зависеть от пользовательского ввода. После этого мы используем её для доступа к свойству. Это даёт нам большую гибкость.

Пример:

**let user = {**

**name: “John”,**

**age: 30**

**};**

**let key = prompt(«Что вы хотите узнать о пользователе?», «name»);**

// доступ к свойству через переменную

**alert( user[key] ); // John (если ввели «name»)**

Запись «через точку» такого не позволяет:

**let user = {**

**name: "John",**

**age: 30**

**};**

**let key = "name";**

**alert( user.key ); // undefined**

## [Вычисляемые свойства](https://learn.javascript.ru/object" \l "vychislyaemye-svoystva)

Мы можем использовать квадратные скобки в литеральной нотации для создания вычисляемого свойства.

Пример:

**let fruit = prompt("Какой фрукт купить?", "apple");**

**let bag = {**

**[fruit]: 5, // имя свойства будет взято из переменной fruit**

**};**

**alert( bag.apple ); // 5, если fruit="apple"**

Смысл вычисляемого свойства прост: запись [fruit] означает, что имя свойства необходимо взять из переменной fruit.

И если посетитель введёт слово "apple", то в объекте bag теперь будет лежать свойство {apple: 5}.

По сути, пример выше работает так же, как и следующий пример:

**let fruit = prompt("Какой фрукт купить?", "apple");**

**let bag = {};**

// имя свойства будет взято из переменной fruit

**bag[fruit] = 5;**

…Но первый пример выглядит лаконичнее.

Мы можем использовать и более сложные выражения в квадратных скобках:

**let fruit = 'apple';**

**let bag = {**

**[fruit + 'Computers']: 5 // bag.appleComputers = 5**

**};**

Квадратные скобки дают намного больше возможностей, чем запись через точку. Они позволяют использовать любые имена свойств и переменные, хотя и требуют более громоздких конструкций кода.

Подведём итог: в большинстве случаев, когда имена свойств известны и просты, используется запись через точку. Если же нам нужно что-то более сложное, то мы используем квадратные скобки.

## [Ограничения на имена свойств](https://learn.javascript.ru/object" \l "ogranicheniya-na-imena-svoystv)

Как мы уже знаем, имя переменной не может совпадать с зарезервированными словами, такими как «for», «let», «return» и т.д.

Но для свойств объекта такого ограничения нет:

// эти имена свойств допустимы

**let obj = {**

**for: 1,**

**let: 2,**

**return: 3**

**};**

**alert( obj.for + obj.let + obj.return ); // 6**

Иными словами, нет никаких ограничений к именам свойств. Они могут быть в виде строк или символов (специальный тип для идентификаторов, который будет рассмотрен позже).

Все другие типы данных будут автоматически преобразованы к строке.

Например, если использовать число 0 в качестве ключа, то оно превратится в строку "0":

**let obj = {**

**0: "Тест" // то же самое что и "0": "Тест"**

**};**

// обе функции alert выведут одно и то же свойство (число 0 преобразуется в строку "0")

**alert( obj["0"] ); // Тест**

**alert( obj[0] ); // Тест (то же свойство)**

Есть небольшой подводный камень, связанный со специальным свойством \_\_proto\_\_. Мы не можем установить его в необъектное значение:

**let obj = {};**

**obj.\_\_proto\_\_ = 5; // присвоим число**

**alert(obj.\_\_proto\_\_); // [object Object], значение - это объект, т.е. не то, что мы ожидали**

Как мы видим, присвоение примитивного значения 5 игнорируется.

Мы более подробно исследуем особенности свойства \_\_proto\_\_ в следующих главах [Прототипное наследование](https://learn.javascript.ru/prototype-inheritance), а также предложим [способы исправления](https://learn.javascript.ru/prototype-methods) такого поведения.

## [Проверка существования свойства, оператор «in»](https://learn.javascript.ru/object" \l "proverka-suschestvovaniya-svoystva-operator-in)

В отличие от многих других языков, особенность JavaScript-объектов в том, что можно получить доступ к любому свойству. Даже если свойства не существует – ошибки не будет!

При обращении к свойству, которого нет, возвращается undefined. Это позволяет просто проверить существование свойства:

**let user = {};**

**alert( user.noSuchProperty === undefined ); // true означает "свойства нет"**

Также существует специальный оператор "in" для проверки существования свойства в объекте.

Синтаксис оператора:

"key" in object

Пример:

**let user = { name: "John", age: 30 };**

**alert( "age" in user ); // true, user.age существует**

**alert( "blabla" in user ); // false, user.blabla не существует**

Обратите внимание, что слева от оператора in должно быть имя свойства. Обычно это строка в кавычках.

Если мы опускаем кавычки, это значит, что мы указываем переменную, в которой находится имя свойства. Например:

**let user = { age: 30 };**

**let key = "age";**

**alert( key in user ); // true, имя свойства было взято из переменной key**

Для чего вообще нужен оператор in? Разве недостаточно сравнения с undefined?

В большинстве случаев прекрасно сработает сравнение с undefined. Но есть особый случай, когда оно не подходит, и нужно использовать "in".

Это когда свойство существует, но содержит значение undefined:

**let obj = {**

**test: undefined**

**};**

**alert( obj.test ); // выведет undefined, значит свойство не существует?**

**alert( "test" in obj ); // true, свойство существует!**

В примере выше свойство obj.test технически существует в объекте. Оператор in сработал правильно.

Подобные ситуации случаются очень редко, так как undefined обычно явно не присваивается. Для «неизвестных» или «пустых» свойств мы используем значение null. Таким образом, оператор in является экзотическим гостем в коде.

## [Цикл "for..in"](https://learn.javascript.ru/object" \l "forin)

Для перебора всех свойств объекта используется цикл for..in. Этот цикл отличается от изученного ранее цикла for(;;).

Синтаксис:

**for (key in object) {**

**// тело цикла выполняется для каждого свойства объекта**

**}**

К примеру, давайте выведем все свойства объекта user:

**let user = {**

**name: "John",**

**age: 30,**

**isAdmin: true**

**};**

**for (let key in user) {**

**// ключи**

**alert( key ); // name, age, isAdmin**

**// значения ключей**

**alert( user[key] ); // John, 30, true**

**}**

Обратите внимание, что все конструкции «for» позволяют нам объявлять переменную внутри цикла, как, например, let key здесь.

Кроме того, мы могли бы использовать другое имя переменной. Например, часто используется вариант "for (let prop in obj)".

## Выводы:

Объекты – это ассоциативные массивы с рядом дополнительных возможностей.

Они хранят свойства (пары ключ-значение), где:

* Ключи свойств должны быть строками или символами (обычно строками).
* Значения могут быть любого типа.

Чтобы получить доступ к свойству, мы можем использовать:

* Запись через точку: obj.property.
* Квадратные скобки obj["property"]. Квадратные скобки позволяют взять ключ из переменной, например, obj[varWithKey].

Дополнительные операторы:

* Удаление свойства: delete obj.prop.
* Проверка существования свойства: "key" in obj.
* Перебор свойств объекта: цикл for for (let key in obj).

То, что мы изучали в этой главе, называется «простым объектом» («plain object») или просто Object.

В JavaScript есть много других типов объектов:

* Array для хранения упорядоченных коллекций данных,
* Date для хранения информации о дате и времени,
* Error для хранения информации об ошибке.
* … и так далее.

У них есть свои особенности, которые мы изучим позже. Иногда люди говорят что-то вроде «тип данных Array» или «тип данных Date», но формально они не являются отдельными типами, а относятся к типу данных Object. Они лишь расширяют его различными способами.

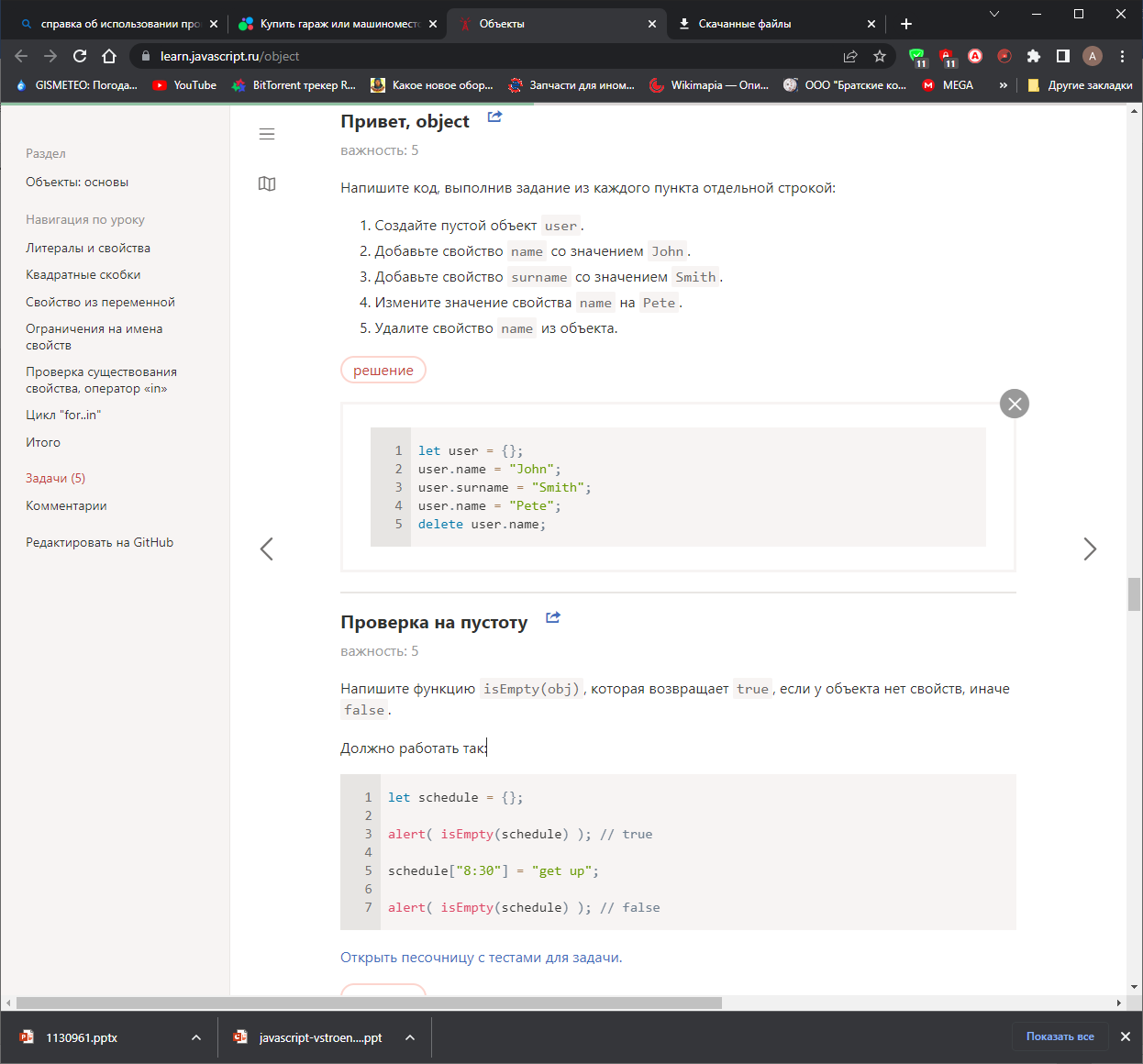
Объекты в JavaScript очень мощные. Здесь мы только немного углубились в действительно огромную тему. Мы будем плотно работать с объектами и узнаем о них больше в следующих частях учебника.

## Задачи и примеры

## Задача 1:

1. Создайте пустой объект user.
2. Добавьте свойство name со значением John.
3. Добавьте свойство surname со значением Smith.
4. Измените значение свойства name на Pete.
5. Удалите свойство name из объекта.

## Решение:



## Задача 2:

## 

## Решение:

## 

## Задача 3:

## 

## Решение:

## 

1. Для студентов, сдающих работу в указанный преподавателем срок, допускается сдача отчета в электронном виде. [↑](#footnote-ref-1)
2. Из источника: https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Working\_with\_Objects [↑](#footnote-ref-2)
3. Обзор большего количества встроенных объектов JavaScript можно посмотреть по следующей ссылке:

   https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global\_Objects [↑](#footnote-ref-3)
4. Источник:

   https://learn.javascript.ru/object [↑](#footnote-ref-4)