

Функции в JavaScript



JavaScript
Courses

www.courses.dp.ua

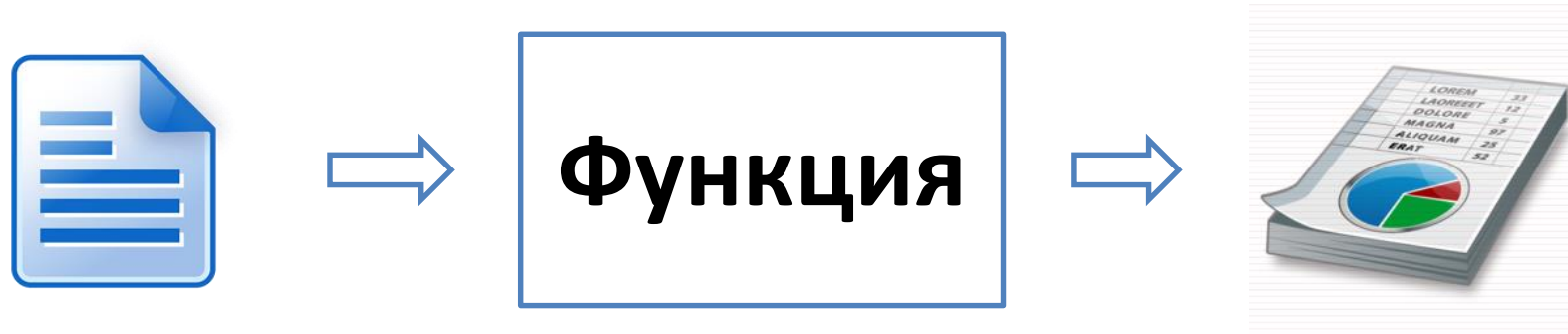
Функции

Функция – фрагмент кода, у которого есть имя, который можно вызывать из любого места в программе. **Функции** уменьшают количество кода в программе, код функции пишется один раз, используется многократно.



Идея функций заключается в следующем: **зачем писать многократно одно и то же, лучше сказать программе: я уже такое писал, возьми и повтори это здесь, там, и еще вот там.**

Функции в JavaScript



Функция также называют «подпрограммами» (программа в программе). Как и у программы в целом задача функции получить данные на входе и дать результат их обработки на выходе (хотя получение данных и/или выдача результатов не является обязательным).

Какая польза от функций?

1. Уменьшаем дублирование (повторение) кода;
2. Проще вносить изменения;
3. Абстрагирование от деталей;

Функции в JavaScript

```
2
3  function action(a, b, c){
4      |   let sum = a + b + c;
5      |   return sum;
6      | }
7
8  let process = function(a, b, c){
9      |   let sum = a + b + c;
10     |   return sum;
11     | }
12
13  let calculate = (a, b, c) => a + b + c;
14
15  typeof action; //function
16  typeof process; //function
17  typeof calculate; //function
18
```

Функции в JavaScript – блоки кода которые возможно вызывать (выполнять) многократно. Синтаксисом JS предусмотрено несколько способов определения функций: Объявление функции (***Function Declaration***) (3), Функциональное выражение (***Function Expression***, она же «анонимная» функция) (8), и стрелочные-функции (***arrow-function***, они же лямбда-функции) (13). Функции в JavaScript – тип данных, функцию мы можем размещать в переменных, как и другие типы данных. Отличие в том, что функции мы можем вызывать.

Подробнее: <https://learn.javascript.ru/function-basics>

Подробнее: <https://learn.javascript.ru/arrow-functions-basics>

rest-оператор и функции

```
2
3   let process = function(a, b, c, ...others){
4       console.log(others);
5       let sum = a + b + c;
6       return sum;
7   }
8
9   process(1,2,3,4,5,6,7); // return 6;
10  // in console: [4,5,6,7];
11
```

Функция может принимать параметры и возвращать результат своей работы для дальнейшего использования (оператор *return*).

Но при помощи оператора `...` (в данном случае его называют *rest-оператором*) мы можем принять любое количество параметров и работать с ними как с массивом (**ES2015**).

Подробнее: <https://learn.javascript.ru/rest-parameters-spread-operator>

Параметры по умолчанию в функциях

```
2
3   let process = function(a = 1, b = 2, c = 3){
4       console.log(a, b, c);
5       let sum = a + b + c;
6       return sum;
7   }
8
9   process(1,2); // return 6;
10  // in console 1, 2, 3
11
```

Передача неполного набора параметров не является ошибкой в JavaScript, но может создать проблемы при работе функции. При помощи синтаксиса параметров по умолчанию мы можем указать значения которые будут использоваться если тот или иной параметр не будет передан (**ES2015**).

Подробнее: <https://learn.javascript.ru/function-basics#parametry-po-umolchaniyu>

Функция в объекте – метод

```
2
3   let arr = ["Jhon", (name) => alert(`Hello ${name}!`) , "Alice"];
4
5   arr[1]('Bill');
6
7   //-----//
8
9   let ob = {
10       name : "Jhon",
11       city : "Dnipro",
12       action: function(name){
13           alert(`Hello ${name}!`);
14       }
15   }
16
17   ob.action("Maria");
18
```

Функции могут размещаться в ячейках массива (коллекций Set и Map) а также в свойствах объекта. При этом для функций в составе объектов есть отдельный термин – **метод**.

Самовывзывающаяся функция

```
2  
3  ✓ (function(){  
4      console.log("...");  
5  })();  
6
```

Самовывзывающаяся функция – удобный механизм выполнить какие-либо действия автоматически, не создавая переменных и внося в код явных вызовов функций. Другими словами не засоряя глобальную область видимости. Активно используется в сторонних библиотеках.

Замыкания

```
2
3   let user_name = "Jhon";
4
5   function test(){
6       |   console.log(`Hello ${user_name}!`);
7   }
8
9   user_name = "Jane";
10
11   test();
12
```

У функций есть доступ к внешним переменным, этот механизм называют **замыканием**, он позволяет обращаться к внешнему контексту и получать оттуда актуальные данные.

Подробнее: <https://learn.javascript.ru/closure>

Таймеры в JavaScript

```
2
3   let f1 = function(){
4       |   console.log("Function for Timeout called");
5       |
6       |
7   let f2 = function(){
8       |   console.log("Function for Interval called");
9       |
10      |
11      let timeout_id = setTimeout(f1, 1000);
12
13      let interval_id = setInterval(f2, 3000);
14
```

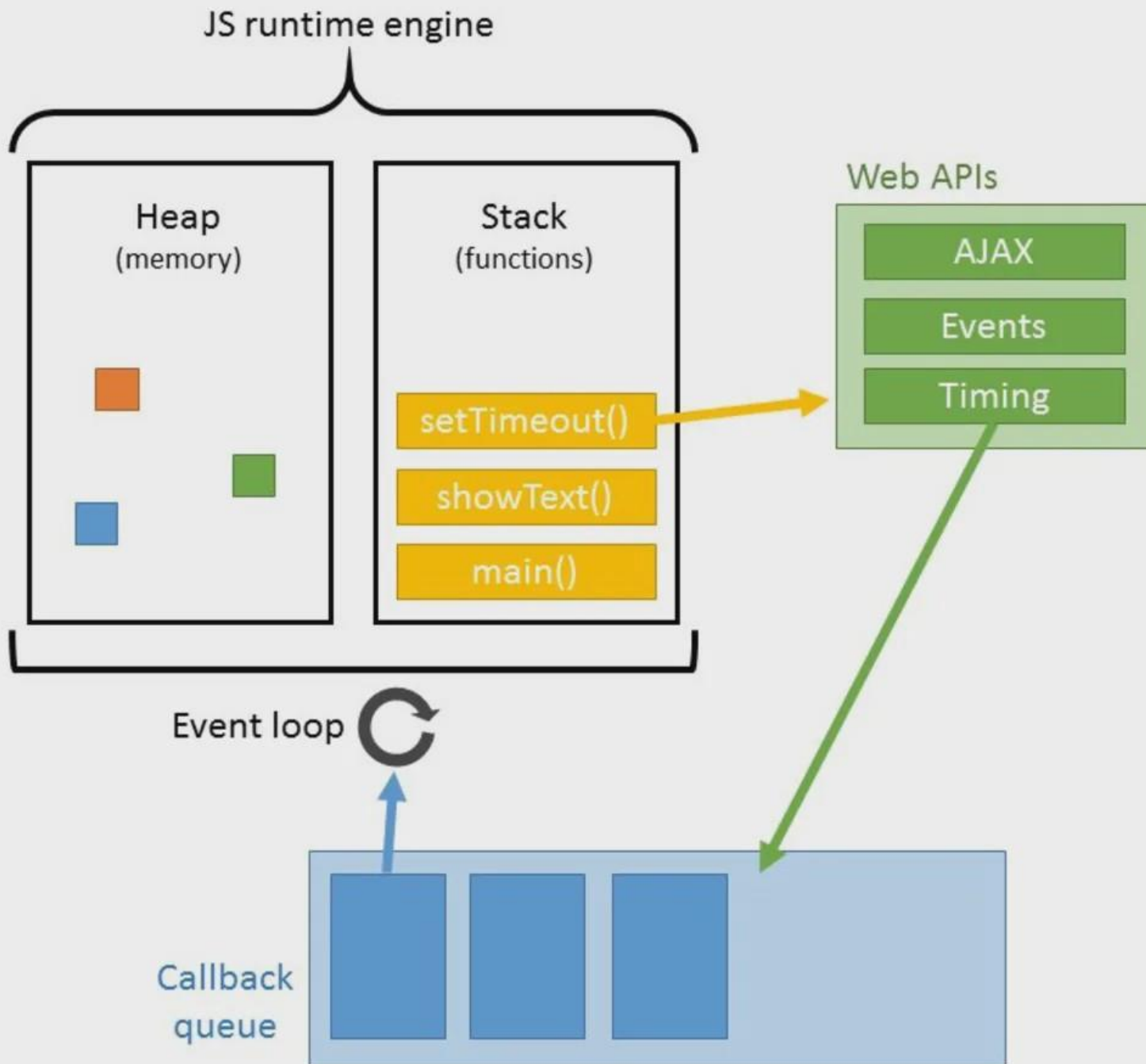
setTimeout(*some_function*, *delay*) – вызовет функцию *some_function* через *delay* миллисекунд. Сделает это один раз.

setInterval(*some_function*, *delay*) – вызовет функцию *some_function* через *delay* миллисекунд. И будет повторять вызов каждые *delay* миллисекунд.

Обе функции возвращают **id** таймера, с помощью которого и функций **clearTimeout** и **clearInterval** уничтожить таймер еще до его вызова.

Обе функции можно отнести к инструментам **асинхронности**.

Подробнее: <https://learn.javascript.ru/settimeout-setinterval>



Event Loop

JavaScript однопоточный язык программирования, но тем не менее нам доступны асинхронные инструменты. Доступны они за счёт функционирования механизма **Event Loop** (или *цикла событий*, но не стоит путать с событиями DOM).

Подробнее:

<https://www.youtube.com/watch?v=8cV4ZvHXQL4>

Методы массивов

Метод .sort() и функция-компаратор

```
2
3 let arr = [23, 4, 67, 117, 34, 0, 55, 78, 5, 9];
4
5 arr.sort(function(a, b){
6     if(a > b){
7         return 1;
8     }else if(a < b){
9         return -1;
10    }else{
11        return 0;
12    }
13 });
14 //arr.sort((a,b) => a - b);
15
16 console.log(arr);
17 //[0, 4, 5, 9, 23, 34, 55, 67, 78, 117]
18
```

Методу **.sort()** массивов можно передать функцию (т.н. функцию-компаратор) которая «подскажет» браузеру как сравнивать два элемента между собой. Функция принимает 2 элемента и должна вернуть 0 если они равны, отрицательное число если второй элемент больше или положительное если первый элемент больше.

Подробнее: <https://learn.javascript.ru/array-methods>

Перебирающий методы массива `.forEach()`

```
2  
3 let arr = [23, 4, 67, 117, 34, 0, 55, 78, 5, 9];  
4  
5 arr.forEach((item, index, array) => console.log(index, item));  
6
```

Функция переданная методу `.forEach()` массива будет применена к каждому элементу. Функция принимает три параметра, которые получают сам элемент (для которого вызывается функция), его индекс в массиве, и ссылка на сам массив. С появлением цикла **for-of** востребованность этого метода уменьшилась.

Подробнее: <https://learn.javascript.ru/array-methods>

Некоторые полезные методы преобразования массивов

.filter();

Метод **.filter()** формирует новый массив занося в него элементы из старого, но только те которые «одобрит» функция переданная методу в качестве параметра.

.map();

Метод **.map()** формирует новый массив занося в него элементы из старого, но предварительно пропуская каждый элемент через функцию переданную методу в качестве параметра. Эта функция может любым образом преобразовать элемент.

.reduce();

Метод **.reduce()** позволяет хранить при переборе элементов какое-либо промежуточное значение, оно передаётся в первом параметре функции (передаваемой методу). При каждом вызове то что возвращает функция становится этим самым «промежуточным» значением для следующего вызова функции. В результате **.reduce()** возвращает самое последнее «промежуточное значение»

Подробнее: <https://learn.javascript.ru/array-methods#preobrazovanie-massiva>

Немного практики #2

Обработка наборов данных

Офіційний курс гривні до іноземних валют та банківських металів

Курс на поточну дату:

<https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange>



Курс на дату (задається у форматі YYYYMMDD, де YYYY - рік, MM - місяць, DD - день):

<https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?date=20191127>



Курс на дату по валюті (код валюті літерний, реєстр значення не має):

<https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?valcode=EUR&date=20191127>



Воспользуемся заготовкой в каталоге репозитория [./source/ex01.html](#)

Данные получены с API-сервисов НБУ

https://bank.gov.ua/control/uk/publish/article?art_id=38441973

try/catch/finally/throw

или

Обработка ошибок

Обработка ошибок (исключений)

```
2
3   let f = function(a, b){
4       |   return a + b;
5   }
6
7   f = 42;
8
9   try{
10      |   let result = f(2, 3);
11  }catch(e){
12      |   console.log("This code if we have error", e);
13  }finally{
14      |   console.log("This code work always");
15  }
16
```

Если в блоке **try** произойдёт ошибка, выполнение блока прекратится и перейдёт к блоку **catch**, в котором могут быть выполнены какие-либо действия направленные на нивелирования влияния ошибки на работу скрипта. Если в блоке **try** ошибка не произошла, то блок **catch** не выполняется. Независимо от того произошла ошибка или нет, после **try-catch** скрипт пойдёт выполняться дальше, как ни в чём не бывало. Блок **finally** выполняется в любом случае. В этом блоке обычно размещается код который должен при любом варианте развития событий завершить те или иные действий (например убрал иконку-лоадер с экрана независимо от того успешна ли была загрузка).

Подробнее: <https://learn.javascript.ru/try-catch>

Генерация ошибки | оператор **throw**

```
2
3     try{
4
5         throw new Error("Info about error!");
6
7     }catch(e){
8         console.log("This code if we have error", e);
9     }finally{
10        console.log("This code work always");
11    }
12
```

При помощи оператора **throw** мы можем «выбросить» свою ошибку, для этого оператору достаточно передать любое значение, но хорошей практикой является использование для этих целей объекта **Error** или производного от него.

Подробнее: <https://developer.mozilla.org/uk/docs/Web/JavaScript/Reference/Statements/throw>

Домашнее задание
/узнать

Современный учебник Javascript

Перед вами учебник по JavaScript, начиная с основ, включающий в себя много тонкостей и фишек JavaScript/DOM.



смотреть на Github

Поделиться:



НАЙТИ

Содержание

Первые две части посвящены JavaScript и его использованию в браузере. Затем идут дополнительные циклы статей на разные темы.

Предварительные знания – лучший помощник в обучении, поэтому к следующему занятию жду, что **пройдёте разделы 4.1, 4.2, 4.4-4.6, 8.1-8.4, 11.1** первой части, а также **раздел 3.8** третьей части

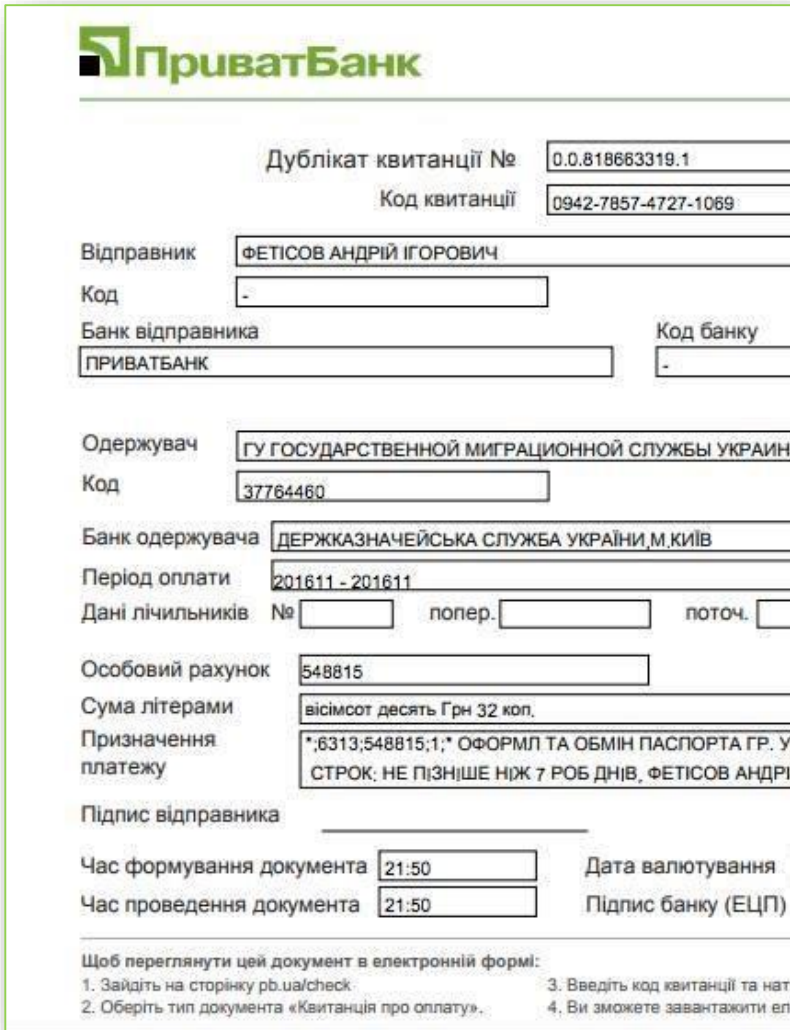
<http://learn.javascript.ru/>

Перебирающие методы

В **JavaScript** есть еще ряд методов массивов, а именно: **.every()**, **.some()**, **.find()**, **.findIndex()** узнайте чем они могут быть полезны.

Домашнее задание
/сделать

Домашнее задание #С.1



ПриватБанк

Дублікат квитанції № 0.0.818663319.1
Код квитанції 0942-7857-4727-1069

Відправник ФЕТІСОВ АНДРІЙ ІГОРОВИЧ
Код -
Банк відправника ПРИВАТБАНК Код банку -

Одержувач ГУ ГОСУДАРСТВЕННОЙ МИГРАЦИОННОЙ СЛУЖБЫ УКРАИНЫ
Код 37764460
Банк одержувача ДЕРЖКАЗНАЧЕЙСЬКА СЛУЖБА УКРАЇНИ, М. КИЇВ

Період оплати 201611 - 201611
Дані лічильників № попер. поточ.

Особовий рахунок 548815
Сума літерами вісімсот десять Грн 32 коп.
Призначення платежу *6313;548815;1;* ОФОРМЛ ТА ОБМІН ПАСПОРТА ГР. У СТРОК: НЕ ПІЗНІШЕ НІЖ 7 РОБ ДНІВ, ФЕТІСОВ АНДРІЙ

Підпис відправника
Час формування документа 21:50 Дата валютування
Час проведення документа 21:50 Підпис банку (ЕЦП)


Щоб переглянути цей документ в електронній формі:
1. Зайдіть на сторінку rb.ua/check. 3. Введіть код квитанції та натисніть «Відслідкувати». 4. Ви зможете завантажити електронний документ.
2. Оберіть тип документа «Квитанція про оплату».

Написать скрипт который будет словами записывать сумму заданную числом которое ввёл пользователь в пределах от 1 до 999 (включительно). Например **643** => «**шестьсот сорок три гривны**» (не забывая добавлять слово гривен, гривна и т.д. в зависимости от необходимого склонения).

Домашнее задание #С.2 | «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) _____

*Пользователь вводит ИНН
(физ. лица Украины),
Необходимо определить:
корректен ли код (нет ли
в нём ошибки) .*

Для проверки: 3463463460; 2063463479.