Описание курса

JavaScript. Уровень 1. Основы JavaScript

Тема	Часы
Модуль 1. Основы программирования *Операторы, переменные, типы данных…*	4
Модуль 2. Управляющие конструкции *if-else, for, while, switch…*	4
Модуль 3. Функции *Синтаксис функции, аргументы, области видимости…*	4
Модуль 4. Объектные типы *Свойства, методы, массивы…*	4
Модуль 5. Объектно-ориентированное программирование *Функция конструктор, прототипы, классы…*	4
Модуль 6. Дополнительная информация *Работа со строками, регулярные выражения, JSON…*	4

Программа курса предусматривает лабораторные работы по каждой теме, а также выполнение домашних заданий и контроль знаний

Бауманский учебный центр
Специалист.ru
Мировые стандарты обучения

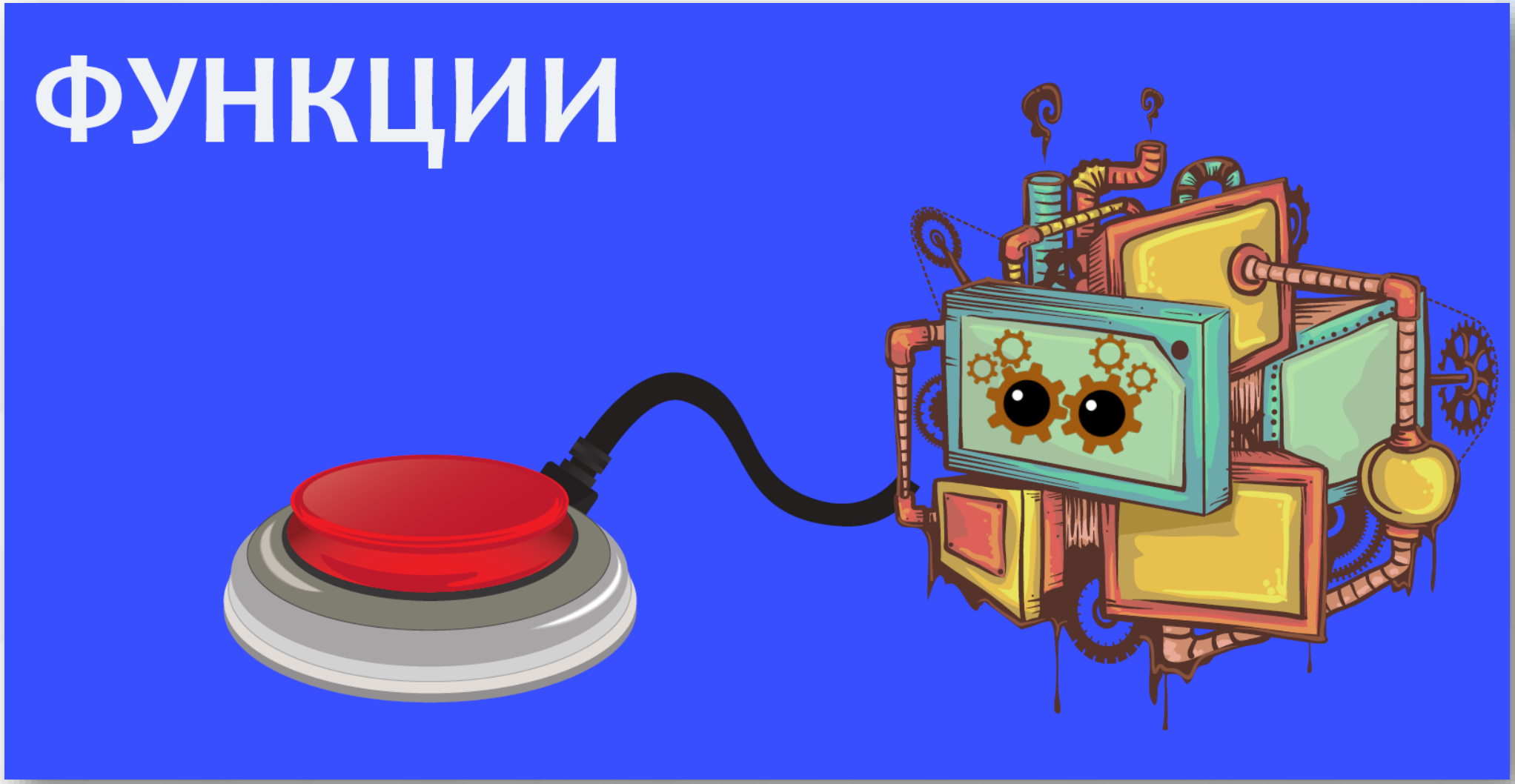
Модуль 3 Функции

Басов Денис

В этом модуле мы рассмотрим:

- Понятие функций
- Объявление функций
- Аргументы и параметры функций
- Возврат значений из функций
- Области видимости
- Функциональное выражение
- Анонимная функция

Общее понимание что такое функции



Функции – переиспользуемые процедуры

- Функции позволяют нам писать код, который мы можем переиспользовать в дальнейшем
- Мы определяем фрагмент кода, который можем выполнить в нужном месте
- Если бы функций не существовало, пришлось бы копировать и вставлять фрагмент кода каждый раз, когда он понадобится
- Функции нужны, чтобы не переписывать один и тот же код много раз
- Функции нужны для разбиения кода на фрагменты, подпрограммы для удобства работы

Два этапа работы с функциями

Объявление функции

Вызов функции

Объявление функции

Для объявления функции используем ключевое слово **function**

Далее указываем название функции

В круглых скобках могут быть параметры функции

В фигурных скобках пишем тело функции

```
1   // объявление функции
2   function sayHello(){
3       console.log('Hello, user');
4   }
5
```

Правила именования функций

- Название функции должно содержать только буквы, цифры, символы $ и _
- Первый символ не должен быть цифрой
- Нелатинские буквы разрешены, но не рекомендуются
- Имена функций регистрозависимы, sayHello и sayhello две разные функции
- Имя функции должно быть описательным
- Так как функция что-то делает, рекомендуется в названии функции использовать глагол
- При именовании функции рекомендуется использовать CamelCase

Вызов функции

После объявления функция ждет и ничего не делает

Чтобы функция сделала свою работу, ее нужно вызвать

```
1   // объявление функции
2   function sayHello() {
3     console.log("Hello, user");
4   }
5
6   // вызов функции
7   sayHello(); // Hello, user
```

Внутри функции мы можем использовать любые возможности JavaScript

```
1   // объявление функции
2   function printArray() {
3     let animals = ["bison", "camel", "duck", "elephant", "cat"];
4     for (let animal of animals) {
5       if (animal.length > 4) {
6         console.log(animal);
7       }
8     }
9   }
10
11  // вызов функции
12  printArray();
```

Функции без параметров и аргументов

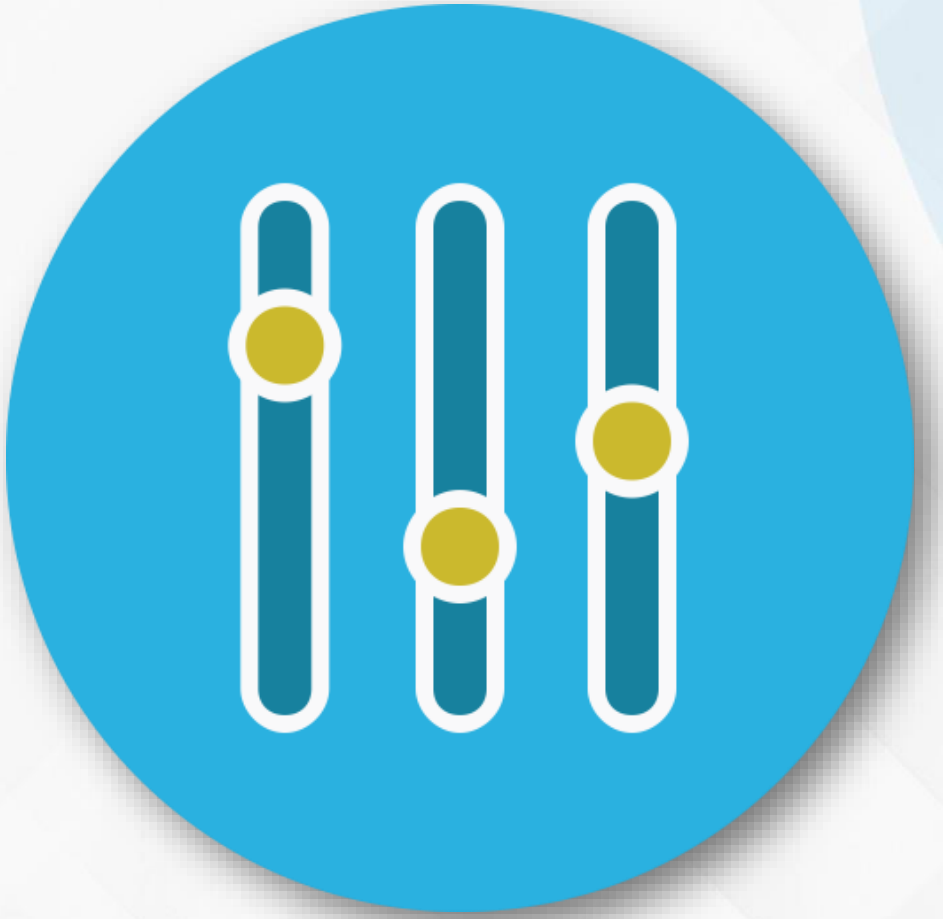
Пока наши функции не очень полезны. При каждом вызове результат одинаковый

```
1   // объявление функции
2   function sayHello() {
3     console.log("Hello, user");
4   }
5
6   // вызов функции
7   sayHello(); // Hello, user
8   sayHello(); // Hello, user
9   sayHello(); // Hello, user
10  sayHello(); // Hello, user
```

Параметры функции

Параметры – это внутренние переменные функции, которым присваиваются значения переданных при вызове функций аргументов

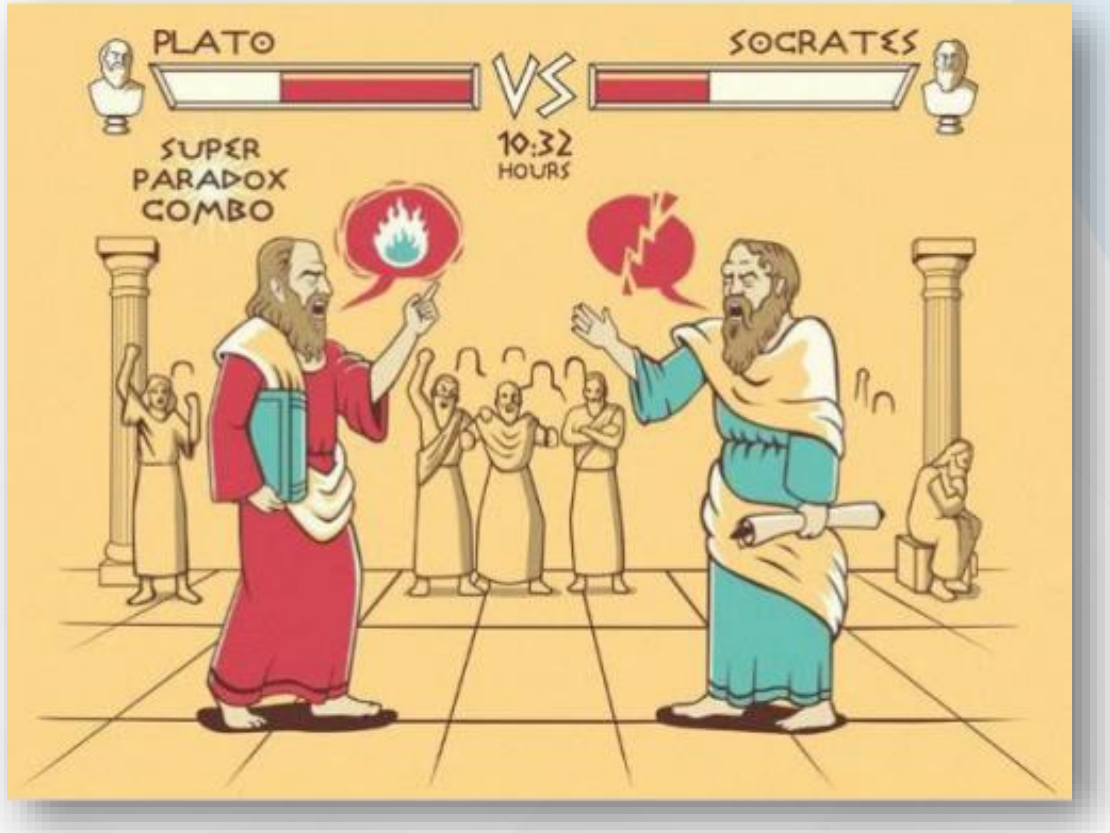
Параметры указываются в круглых скобках при объявлении функции



Аргументы функции

Аргументы – это данные, которые мы можем передать функции при вызове

Функция обрабатывает предоставленные данные, в результате мы получаем разный результат



Функция с параметрами и аргументами

За счет передачи разных данных при каждом вызове функции, получаем разный результат

```
1   // firstName, lastName - параметры функции
2   function greetUser(firstName, lastName) {
3     console.log(`Привет, ${firstName} ${lastName}!`);
4   }
5
6   // строки, передаваемые функции - аргументы
7   greetUser("Иван", "Иванов"); // Привет, Иван Иванов!
8   greetUser("Анна", "Петрова"); // Привет, Анна Петрова!
9   greetUser("Сергей", "Сидоров"); // Привет, Сергей Сидоров!
10
```

Получение суммарного значения в функции

```
1   // принимаем массив и считаем сумму элементов
2   function getTotalBill(billArray) {
3     let sum = 0;
4     for (let item of billArray) {
5       sum += item;
6     }
7     console.log(sum);
8   }
9
10  // передаем разные счета и получаем сумму по каждому
11  getTotalBill([43, 65, 23, 54]); // 185
12  getTotalBill([56, 87, 32, 56, 32]); // 263
13  getTotalBill([45, 76, 22]); //143
14
```

Значение параметров по умолчанию

Если по каким-то причинам при вызове функции не будет передано значение, мы получим неожиданный результат

```
1   function greetUser(userName) {
2     let greet = `Привет, ${userName}`;
3     console.log(greet);
4   }
5   greetUser(); // Привет, undefined
```

```
1   function calcUserAge(birthYear) {
2     // получаем текущий год
3     let currentYear = new Date().getFullYear();
4     let age = currentYear - birthYear;
5     console.log(age);
6   }
7   calcUserAge(); // NaN
```

Значение параметров по умолчанию

Чтобы сделать нашу функцию более гибкой и избежать возникшей ситуации мы можем задать параметру значение по умолчанию

```
1  function greetUser(userName = "Пользователь") {
2    let greet = `Привет, ${userName}`;
3    console.log(greet);
4  }
5
6  greetUser(); // Привет, Пользователь
7  greetUser("Анна"); // Привет, Анна
8  greetUser("Евгений"); // Привет, Евгений
```

Получение значения из функции

Сейчас мы не можем использовать результат работы функции за ее пределами

После того как функция закончила работать, результат теряется

```
1   // принимаем массив и считаем сумму элементов
2   function getTotalBill(billArray) {
3     let sum = 0;
4     for (let item of billArray) {
5       sum += item;
6     }
7   }
8
9   // пытаемся сохранить рассчитанное значение в переменную
10  let totalBill = getTotalBill([43, 65, 23, 54]);
11  console.log(totalBill); // undefined
```

RETURN

Чтобы получить значение, рассчитанное функцией нужно:

- Вернуть значение из функции с помощью специальной инструкции **return**
- Сохранить возвращенное значение в глобальной переменной

```
1  // в теле функции считаем сумму и возвращаем
2  function getSum(num1, num2) {
3    let result = num1 + num2;
4    return result;
5  }
6
7  // сохраняем в переменную
8  let sum = getSum(4, 8);
9  console.log(sum); // 12
```

RETURN

Считаем сумму элементов массива и сохраняем полученное значение для дальнейшего использования

```
1   // принимаем массив и считаем сумму элементов
2   function getTotalBill(billArray) {
3     let sum = 0;
4     for (let item of billArray) {
5       sum += item;
6     }
7     return sum; // возвращаем полученной значение
8   }
9
10  // сохраняем рассчитанное значение в переменную
11  let totalBill = getTotalBill([43, 65, 23, 54]);
12  console.log(totalBill); // 185
```

RETURN

Сохранив значение в переменной мы дальше можем использовать это значение по своему усмотрению

```
1   function getTotalBill(billArray) {
2     let sum = 0;
3     for (let item of billArray) {
4       sum += item;
5     }
6     return sum;
7   }
8
9   let totalBill = getTotalBill([43, 65, 23, 54]);
10  let totalBillTax = totalBill * 0.2; // считаем налог
11  console.log(totalBillTax); // 37
```

RETURN

Return прекращает выполнение функции. Все что вы напишете после него будет проигнорировано

```
1   function getTotalBill(billArray) {
2     let sum = 0;
3     for (let item of billArray) {
4       sum += item;
5     }
6     return sum;
7     console.log(sum);
8   }
9
10  let totalBill = getTotalBill([43, 65, 23, 54]);
```

Множественный RETURN

В функции может быть и несколько инструкций return, но всегда выполняется только одна из них

```
1   function checkUserAge(userAge) {
2     if (userAge >= 18) {// если возраст 18 и более
3       return true; // возвращаем истину
4     } else {
5       return false; // иначе возвращаем ложь
6     }
7   }
8
9   // сохраняем результат проверки и выводим сообщение
10  let checkUserResult = checkUserAge(23);
11  checkUserResult ?
12    console.log("Добро пожаловать") :
13    console.log("Доступ запрещен");
```

Область видимости переменных

ОБЛАСТЬ

видимости переменных

- Область, где переменная объявлена **определяет то, где мы можем получить к ней доступ**

Область видимости переменных

```
1   let userName = "Иван";
2   let userAge = 22;
3   let userHobby = "Кататься на санках";
4
5   function printUserInfo(name, age, hobby) {
6       let message = `Меня зовут ${name}, мне ${age} лет,
7           я люблю ${hobby.toLowerCase()}`;
8       console.log(message);
9   }
10
11  printUserInfo(userName, userAge, userHobby);
12
```

Глобальная область видимости

Функциональная область видимости

Глобальная область видимости

Область видимости переменных

Может показаться, что мы можем использовать переменные из функции в глобальном коде. Но нет

```
1   function greetUser(userName) {
2     let greet = `Привет, ${userName}`;
3   }
4
5   console.log(greet); // ReferenceError: greet is not defined
6   // говорит, что переменная greet не определена
```

Область видимости переменных

И даже если вызвать функцию, это не работает

```
1   function greetUser(userName) {
2     let greet = `Привет, ${userName}`;
3   }
4
5   greetUser("Светлана");
6   console.log(greet); // ReferenceError: greet is not defined
7   // говорит, что переменная greet не определена
```

Внутренние переменные функции за ее пределами недоступны, не видны. Говорят, что они находятся в локальной (функциональной) области видимости.

Область видимости переменных

Получить значение из функции можно с помощью return

```
1   function greetUser(userName) {
2     let greet = `Привет, ${userName}`;
3     return greet;
4   }
5
6   let greetText1 = greetUser("Светлана");
7   let greetText2 = greetUser("Петр");
8   let greetText3 = greetUser("Ирина");
9   console.log(greetText1); // Привет, Светлана
10  console.log(greetText2); // Привет, Петр
11  console.log(greetText3); // Привет, Ирина
```

Область видимости переменных

Тем не менее, в JS функции могут получить доступ к переменным, объявленным во внешней области видимости

```
1   // в глобальной переменной получаем текущий год
2   let currentYear = new Date().getFullYear();
3
4   // функция для расчета возраста пользователя
5   function calcUserAge(birthYear) {
6     console.log(currentYear - birthYear);
7   }
8
9   calcUserAge(1985); // 39
10  calcUserAge(2001); // 23
11  calcUserAge(1934); // 90
```

Спасибо
за внимание!

Ваши вопросы...