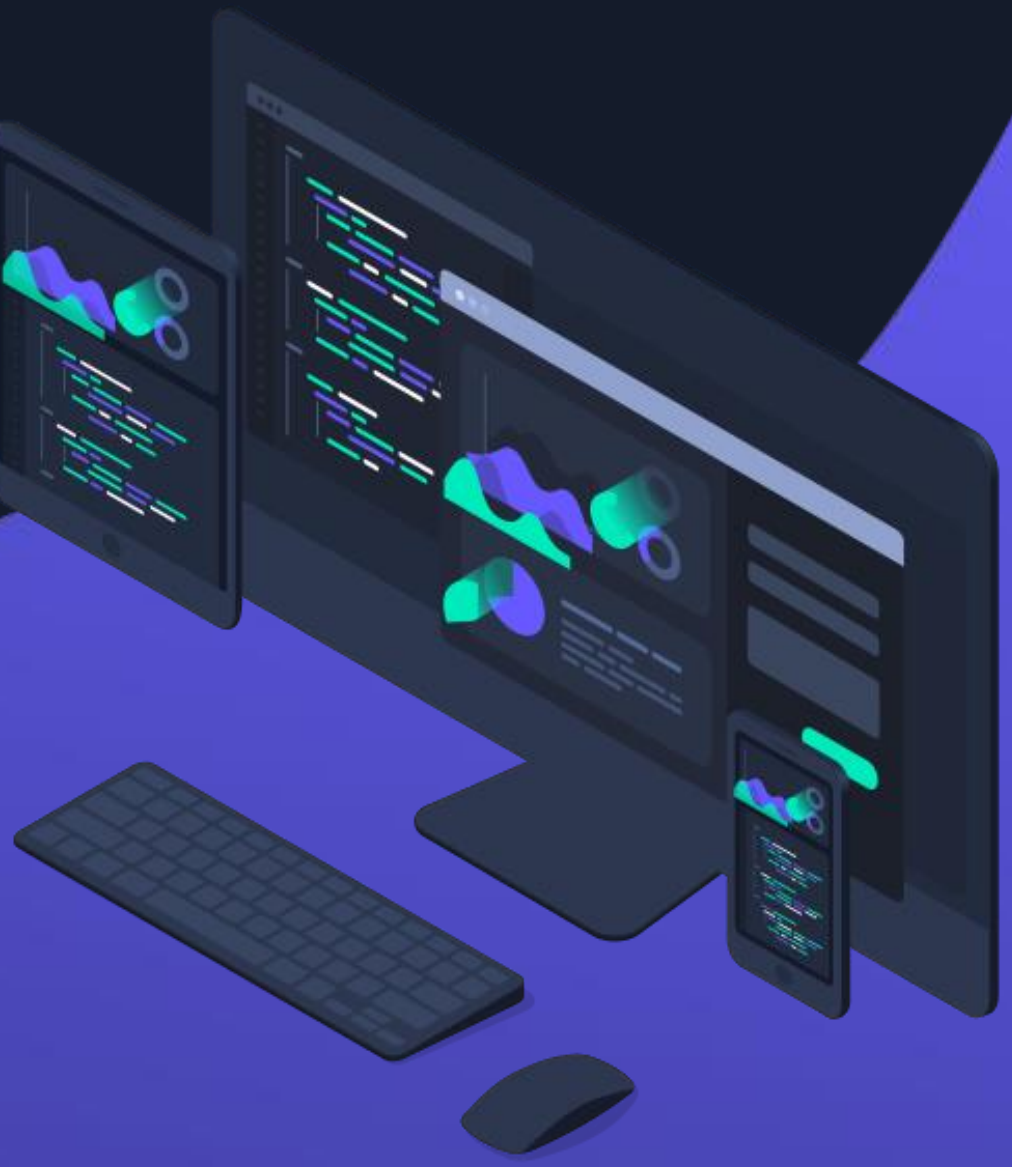
ITEA

ONLINE



ITEA ONLINE

На сегодняшнем занятии:

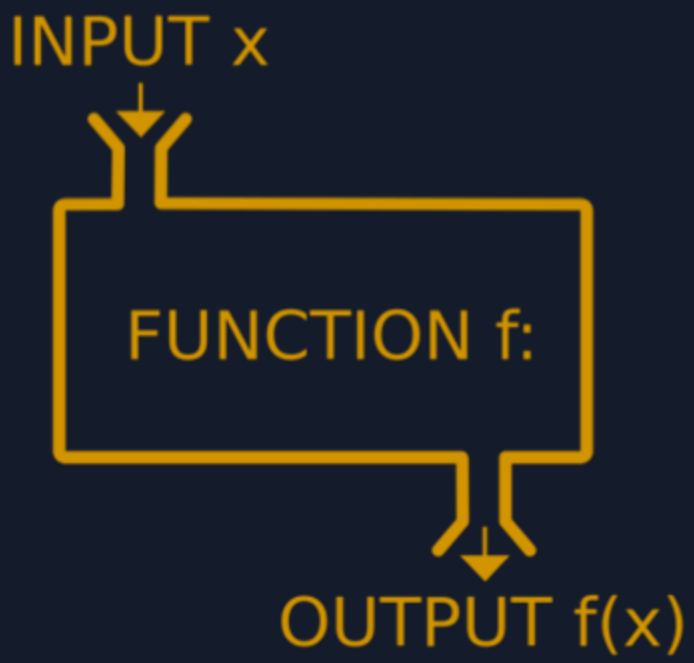
1. Функции
2. Объявление функции (Function Declaration)
3. Локальные и глобальные переменные
4. Параметры
5. Параметры по умолчанию
6. Возврат значения
7. Выбор имени функции
8. Функциональное выражение (Function Expression)
9. Function Expression vs Function Declaration
10. Стрелочные функции

ITEA ONLINE

Функции

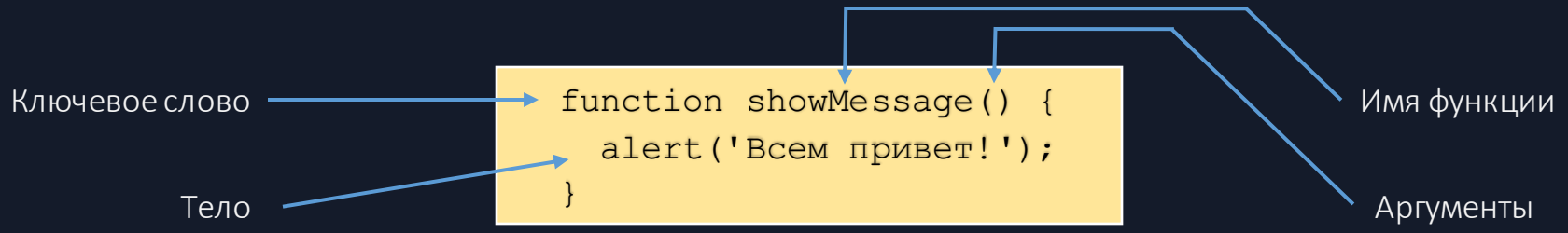
Зачастую нам надо повторять одно и то же действие во многих частях программы.
Например, необходимо красиво вывести сообщение при приветствии посетителя, при выходе посетителя с сайта, ещё где-нибудь.
Чтобы не повторять один и тот же код во многих местах, придуманы функции.

Функции являются основными «строительными блоками» программы.



Объявление функции *(Function Declaration)*

Для создания функций мы можем использовать объявление функции.
Пример объявления функции:



Вначале идёт ключевое слово function, после него имя функции, затем список параметров в круглых скобках через запятую (в вышеприведённом примере он пустой) и, наконец, код функции, также называемый «телом функции», внутри фигурных скобок.

```
showMessage(); //Вызов функции
```

ITEA ONLINE

Локальные и глобальные переменн-ые

Локальные – переменные, объявленные внутри функции, видны только внутри этой функции.
Глобальные – переменные, которые объявлены вне функций и могут изменяться внутри функции.
Глобальная переменная используется, только если внутри функции нет такой локальной.

Локальная

```
function showMessage() {
  let msg = "Всем привет!";
  alert(msg);
}
```

Глобальная

```
let msg = "Всем привет!";

function showMessage() {
  msg = msg + "Как дела?";
  alert(msg);
}
```

ITEA ONLINE

Параметры

Мы можем передать внутрь функции любую информацию, используя *параметры* (также называемые аргументами функции).

В нижеприведённом примере функции передаются два параметра: from и text.

```
// аргументы: from, text
function showMessage(from, text) {
  alert(from + ': ' + text);
}
showMessage('Аня', 'Привет!'); // Аня: Привет!
showMessage('Аня', "Как дела?"); // Аня: Как дела?
```

Все переданные значения функции хранятся в «псевдо-массиве» arguments.

При передаче параметра в функцию – он копируется. Все изменения внутри функции производятся над копией.

ITEA ONLINE

Параметры

Если в коде совпадает название глобальной переменной и аргумента функции – то внутри функции используется аргумент функции. Например:

```
let msg = "Hello";         // Объявляем глобальную переменную

function changeMsg(msg) {
   msg += " world";        // Изменяем аргумент функции, а не глобальную переменную
}

changeMsg(msg);            // Вызываем функцию
console.log(msg + "!");    // Выводим глобальную переменную + «!» // Hello!
```

ITEA ONLINE

Параметры по умолчанию

Если параметр не указан, то его значением становится undefined.
Например, функция showMessage(from, text) может быть вызвана с одним аргументом:

```
showMessage('Аня'); // Аня: undefined
```

Если мы хотим задать параметру text значение по умолчанию, мы должны указать его после =:

```
function showMessage(from, text = "текст не добавлен") {
  alert( from + ": " + text );
}

showMessage("Аня"); // Аня: текст не добавлен
```

Теперь, если параметр text не указан, его значением будет "текст не добавлен"

Возврат значения

Функция может вернуть результат, который будет передан в вызвавший её код.
Простейшим примером может служить функция сложения двух чисел:

```
function sum(a, b) {
  return a + b;
}

let result = sum(1, 2);
alert(result); // 3
```

Переменная (beans)

Функция (doCoffee)

Результат (cupOfCoffee)

Директива return может находиться в любом месте тела функции. Как только выполнение доходит до этого места, функция останавливается, и значение возвращается в вызвавший её код (присваивается переменной result выше).
Вызовов return может быть несколько

I T E A
ONLINE

Выбор имени функции

Функция – это действие. Поэтому имя функции обычно является глаголом. Оно должно быть простым, точным и описывать действие функции, чтобы программист, который будет читать код, получил верное представление о том, что делает функция.

"`show ...`" – возвращают значение,
"`get...`" – возвращают значение,
"`calc...`" – что-то вычисляют,
"`create...`" – что-то создают,
"`check...`" – что-то проверяют и возвращают логическое значение, и т.д.

```
showMessage(..)     // показывает сообщение
getAge(..)          // возвращает возраст (в каком либо значении)
calcSum(..)         // вычисляет сумму (и обычно возвращает её)
createForm(..)      // создаёт форму (и обычно возвращает её)
checkPermission(..) // проверяет доступ, возвращая true/false
```

ITEA ONLINE

Функциональное выражение *(Function Expression)*

Существует ещё один синтаксис создания функций, который называется *Function Expression*. Оно выглядит вот так:

```
let sayHi = function() {
  alert( "Привет" );
};

console.log(sayHi); // выведет код функции
```

Обратите внимание, что последняя строка не вызывает функцию sayHi, после её имени нет круглых скобок. Существуют языки программирования, в которых любое упоминание имени функции совершает её вызов. JavaScript – не один из них.

```
sayHi(); // Вызов функционального выражаения
```

I T E A
ONLINE

Function Expression vs Function Declaration

```
// Function Expression
let sum = function(a, b) {
  return a + b;
};
```

Function Expression: функция, созданная внутри другого выражения или синтаксической конструкции. В данном случае функция создаётся в правой части «выражения присваивания» =:

Function Expression создаётся, когда выполнение доходит до него, и затем уже может использоваться.

```
// Function Declaration
function sum(a, b) {
  return a + b;
}
```

Function Declaration: функция объявляется отдельной конструкцией «function…» в основном потоке кода.

Function Declaration можно использовать во всем скрипте (или блоке кода, если функция объявлена в блоке).

ITEA ONLINE

Стрелочные функции

Существует ещё более простой и краткий синтаксис для создания функций, который часто лучше, чем синтаксис Function Expression. Он называется «функции-стрелки» или «стрелочные функции» (arrow functions), т.к. выглядит следующим образом:

```
let func = (arg1, arg2, ...argN) => expression
```

Такой код создаёт функцию func с аргументами arg1..argN и вычисляет expression с правой стороны с их использованием, возвращая результат.

```
let sum = (a, b) => a + b;

// Более короткая форма для:
let sum = function(a, b) {
  return a + b;
};
```

```
// Если один параметр - скобки можно опустить
let double = n => n * 2; // неявно вызовет return

// Если параметров нет - ставятся пустые скобки
let sayHi = () => alert("Hello!");
```

ITEA ONLINE

Q&A

ITEA

ONLINE