

# JavaScript: функции

---

**WEB  
COURSE  
ORT DNIPRO**

---

**ORTDNIPRO.ORG/WEB**

# 1. Функции

# Функции в JavaScript

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



# Функции в JavaScript

С функциями мы уже сталкивались: **alert()**, **prompt()**, **parseInt()** и т.д. Для нас это просто названия, мы не знаем как они устроены, но мы знаем, что передав ей определенные **параметры** мы получим на выходе ожидаемый **результат**.

# Функции в JavaScript

**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.

```
2
3   function sum(a, b, c){
4
5       let result = a + b + c;
6
7       return result;
8
9   }
10
```

# Функции в JavaScript

Основной отличительный знак **функций** – круглые скобки: например `alert()` Круглые скобки делают сразу два дела: говорят браузеру что мы хотим **выполнить функцию** имя которой стоит перед скобками, и позволяет передать **функции параметры** (если они необходимы для работы функции):

Имя функции  
которую вызываем  
(выполняем).

Скобки, признак того,  
что мы именно  
вызываем функцию.

Параметры функции  
(данные которые  
функция обрабатывает).

`alert("Сообщение на экран");`

# Параметры функции и возвращаемое значение

## Параметры функции (аргументы)

– такие себе «переменные» которые передаются функции при каждом вызове, и могут влиять на результат её работы.

## Возвращаемое значение –

возможность функции вернуть результат своей работы, при необходимости. Возврат делается оператором **return**.

```
2  
3 → function imt(height, mass){  
4  
5     let index = mass / (height ** 2);  
6  
7     return index; ←  
8  
9     }  
10
```

# Подробнее о параметрах функции

**Параметры** внутри **функции** ведут себя как временные переменные, которые живут только пока **функция выполняется**, при старте им присваивается те значения которые **переданы функции** при вызове. Не обязательно передавать **функции** все **параметры** которые предусмотрены её заголовком, НО в таком случае их значение будет **undefined**.

```
2
3   function imt(height, mass){
4
5       let index = mass / (height ** 2);
6
7       return index;
8
9   }
10
```



# Практическая ценность функций

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

```
2  
3     function imt(height, mass){  
4  
5         let index = mass / (height ** 2);  
6  
7         return index;  
8  
9     }  
10
```

## 2. Функции и события

mousedown



mouseup



click



mousedown



mouseup



click



dblclick

# Функции и события

```
10
11   <h1 id='special' onclick="action()">Hello World!</h1>
12
13  <script>
14
15      function action(){
16          special.style.color = 'red';
17      }
18
19  </script>
20
```

**Функции** используются в **событийной модели** управления. Поскольку функция это в первую очередь «заготовка» кода, то их удобно использовать для **подписки на события**. Мы определяем функцию и «сообщаем» браузеру когда она должна быть вызвана (например, при помощи атрибутов тегов: **onclick**, **onmouseenter** и т.д.).

[https://html5css.ru/jsref/dom\\_obj\\_event.php](https://html5css.ru/jsref/dom_obj_event.php)

# 3. Обращение к тегу

# Обращение к тегу и работа с его данными

```
10
11     <h1 id='special' onclick="action()">Hello World!</h1>
12     <script>
13         function action(){
14             special.innerHTML = 'Some New Content...';
15         }
16     </script>
17
18
19
```

Самый простой способ обратиться к тегу – по его **id**. Для тегов с **атрибутом id** браузер создаёт переменную через которую у нас есть доступ к объекту-тегу. Для того, чтобы получить доступ к содержимому тега мы можем воспользоваться его свойством **.innerHTML**. Для элементов ввода (**input**) используется свойство **.value** (для элементов ввода типа **checkbox** используется свойство **.checked**).

## 4. Немного практики

# Депозитный калькулятор с разметкой

**Заданы:** Есть сумма депозита, годовая процентная ставка, и срок размещения вклада в месяцах. Необходимо **рассчитать** сколько денег по окончании получит вкладчик (сумма **вклада + проценты**). **Считать**, что расчёт процентов выполняется ежемесячно (и их капитализация) без влияния количества дней в месяце, также не учитываем налоги.

*Формула капитализации:*

$$Result = Sum * \left(1 + \frac{Rate}{100\%}\right)^{Term}$$

где **Rate** – процентная ставка за период капитализации (в нашем случае за месяц.)

# Депозитный калькулятор, с разметкой

## Депозитный калькулятор

Сумма вклада (грн.)

Срок вклада (мес.)

Ставка (% годовых)

Капитализация %

☐

Рассчитать

По завершению депозита вы получите: – грн.

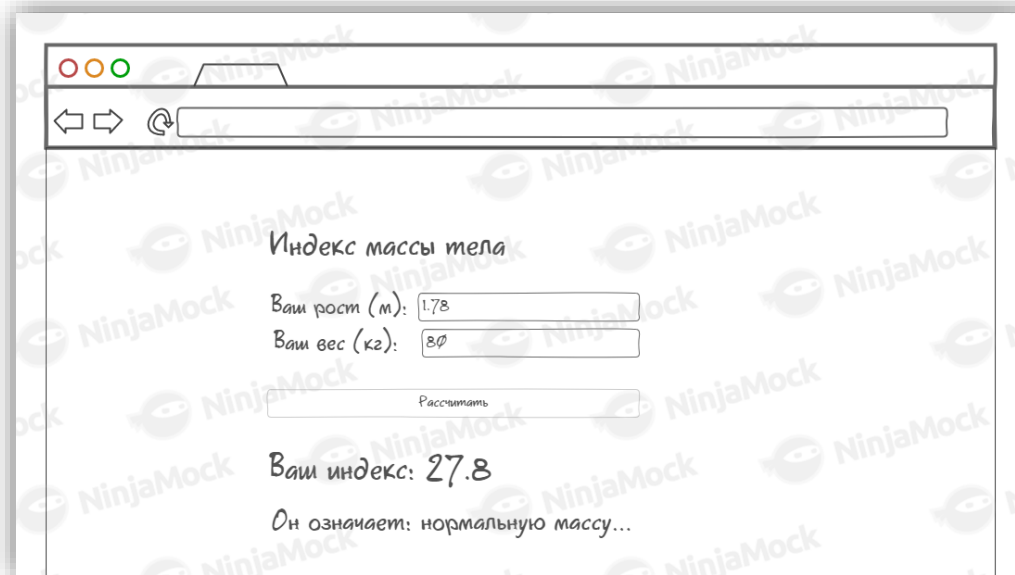
В репозитории занятия: [./src/depo-calc-template](#)



# Домашнее задание

# Домашнее задание #C1

## Индекс массы тела, с разметкой



Индекс массы тела

Ваш рост (м):

Ваш вес (кг):

Ваш индекс: 27.8

Он означает: нормальную массу...

Реализуйте расчёт и вывод **индекса массы тела** (и диагноза) в разметку HTML-документа. Разметку необходимо предварительно подготовить (Bootstrap может помочь). Учтите возможность того, что пользователь по ошибке введёт сантиметры.

**На следующем занятии**

# JS: объекты

```
let person = {  
  firstName: 'John',  
  lastName: 'Doe'  
};
```

