**«Работа с таймерами в JavaScript»**

**Справочная информация**

**Метод setInterval**

Метод **setInterval** работает следующим образом: первым параметром она принимает **имя функции** (без кавычек и круглых скобок), а вторым параметром - через какой **промежуток** запускать эту функцию. Второй параметр задается в **миллисекундах** (1000 миллисекунд = 1 секунда).

Пример: *window.setInterval(timer, 1000)* - этот код будет запускать функцию **timer** раз в секунду. И каждую секунду эта функция **timer** будет выполнять какие-то полезные операции, например, увеличивать счетчик на странице.

Давайте рассмотрим нужный для этого код. Пусть у нас дан инпут. Будем каждую секунду увеличивать значение его атрибута **value** на единицу:

<input type="text" value="1" id="test">

<input type="submit" onclick="start()">

function start() {

//По нажатию на кнопку наш таймер начнет работать:

window.setInterval(timer, 1000);

}

//Эта функция будет запускаться каждую секунду

function timer() {

var elem **=** document.getElementById('test');

elem.value **=** parseInt(elem.value)**+**1; //parseInt преобразует строку в число

}

Обратите внимание на функцию parseInt - она преобразует строку из атрибута **value** в число.

Это нужно, так как атрибут всегда отдает строку, даже если там хранится число, как у нас, то есть *elem.value* вернет **'1'**, а не **1** (в самом начале таймера, когда в атрибуте еще **1**). И получится, что *elem.value + 1* это **'1'+1**, что дает **'11'**, а не **2**).

Поэтому, если не применять **parseInt** - скрипт начнет складывать наши числа **как строки,** и мы увидим, что в инпуте сначала будет **1**, потом **11**, потом **111** и так далее.

Если же написать **parseInt**, то в инпуте будет сначала **1**, потом **2**, потом **3** и так далее.

## Остановка таймера

Вы уже знаете, как запустить таймер, давайте теперь научимся его останавливать. Для этого используется метод **clearInterval**, который принимает *уникальный номер* того таймера (созданного через **setInterval**), который нужно остановить.

Давайте посмотрим откуда берется этот номер:

<input type="submit" onclick="test()">

function test() {

timerId **=** window.setInterval(timer, 1000);

alert(timerId); //выведет номер таймера

}

//Ничего не делает, просто заглушка

function timer() {

}

То есть при создании таймера через **setInterval** мы можем узнать его номер, и потом передать его методу **clearInterval**, чтобы таймер **остановился**.

В следующем примере демонстрируется работа с **clearInterval**. Обратите внимание на то, что запуск и остановка таймера - это разные функции. Чтобы передать номер таймера из одной функции в другую я использую **глобальные переменные** (свойства объекта window), которые мы с вами уже разбирали раньше:

<input type="text" value="1" id="test">

<input type="submit" value="start" onclick="start()">

<input type="submit" value="stop" onclick="stop()">

//Эта функция запускает таймер

function start() {

window.timerId **=** window.setInterval(timer, 1000);

}

//Эта функция останавливает таймер

function stop() {

window.clearInterval(window.timerId);

}

//Эта функция меняет value для инпута

function timer() {

var elem **=** document.getElementById('test');

elem.value **=** parseInt(elem.value)**+**1;

}

Запустите этот код и понажимайте на кнопки. Кнопка *start* будет **запускать** таймер, а кнопка *stop* - **останавливать**. Причем таймер можно запускать и останавливать **много раз** - счетчик в инпуте продолжит с того места, где он остановился. Только в **window.timerId** при каждом новом запуске будет лежать **новое** значение.

Если еще один нюанс - если много раз понажимать на кнопку *start*, не нажимая *stop*, то вы запустите **много таймеров**, каждый из которых будет менять содержимое инпута. И получится так, что значения атрибута **value** будут меняться не раз в секунду, а гораздо **чаще** (и неравномерно). При этом кнопка *stop* сможет остановить **только последний** таймер (так как каждый новый таймер будет затирать глобальную переменную **window.timerId** и там всегда будет лежать номер последнего запущенного таймера).

## Метод setTimeout

Следующий метод, который нам нужен, называется **setTimeout**. Он позволяет сделать **задержку** перед запуском кода (эта задержка случится только **один раз** и код выполнится только один раз, в отличии от setInterval).

Вторым параметром **setTimeout** принимает **задержку** (в миллисекундах) перед запуском функции, которая задана первым параметром, например, так: *window.setTimeout(func, 3000)* - по сути эта строчка вызывает функции **func**, однако эта функция сработает не сразу, а через **3** секунды.

Давайте сделаем так, чтобы **alert** сработал через **3** секунды после нажатия на кнопку:

<input type="submit" onclick="start()">

function start() {

window.setTimeout(func, 3000);

}

function func() {

alert('!');

}

Что происходит в этом примере - по нажатию на кнопку сработает функция **start**, которая вызывает функцию **func** с задержкой с **3** секунды (**3000** миллисекунд = **3** секунды). Ну, а функция **func** выводит **alert** в тот момент, когда вызвана.

**Задания на тему «Работа с таймерами»**

**Задача 7.1** Создать страницу с отсчетом от 0 до бесконечности и кнопкой, при нажатии на которую и начнется этот отсчет.

**Задача 7.2** Создать страницу с отсчетом от 0 до бесконечности и двумя кнопками, при нажатии на одну из которых отсчет начнется, при нажатии на вторую - остановится.

**Задача 7.3** Создать страницу с тикающими цифровыми часами.

**Задача 7.4** Создать страницу с простым слайдером из 4 изображений (на экране изначально одно изображение, и оно поочередно сменяется другими).

**Задача 7.5** Создать страницу с каруселью из 5 изображений и двумя кнопками: первая – запускает карусель, вторая – останавливает.

**Задача 7.6** Создать страницу с тикающими цифровыми часами, показывающими отсчет оставшегося времени до полуночи.