**Основы работы с массивами и объектами в JavaScript**

В программировании очень часто возникает задача **хранения списка похожих значений**, например, всех дней недели или всех месяцев. Можно было бы создавать под каждое значение списка свою переменную, но это очень неудобно и долго - для списка дней недели понадобилось бы 7 переменных, а для месяцев - вообще 12.

А теперь представьте, что вам нужно вывести на экран название третьего месяца. С помощью 12-ти различных переменных это было бы проблематично, так как вам необходимо помнить названия всех переменных, в которые вы записали имена месяцев.

Поэтому для таких вещей был изобретен специальный тип данных. Он называется **массив**.

Массив создается с помощью квадратных скобок **[ ]**:

var arr **=** []; //создаем массив arr

Пока созданный нами массив не содержит никаких данных. **Заполним его** названиями дней недели:

var arr **=** ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];

Каждое значение списка, который мы записали в массив (в нашем случае каждый день недели), называется **элементом** массива.

Элементы разделяются между собой **запятой**. После этой запятой можно ставить пробелы, а можно и не ставить.

Обратите внимание на то, что названия дней недели представляют собой **строки** и поэтому взяты в кавычки. Кроме строк в массиве можно хранить **числа**, и их в кавычки мы не берем:

//В массиве можно хранить как строки, так и числа:

var arr **=** ['пн', 256, 'ср', 34, 38, 'сб', 95];

**Как вывести отдельный элемент массива**

Предположим, мы хотим вывести на экран среду с помощью составленного массива дней недели.

**Делается это так:** нужно после переменной массива (в нашем случае **arr**) написать квадратные скобки **[ ]**, а в них указать **порядковый номер элемента**, который мы хотим вывести: **arr[3]**. Казалось бы, что порядковый номер среды - это 3, но это не так. Потому что в программировании нумерация начинается с нуля. Поэтому 0 - это понедельник, 1 - это вторник, а 2 - это среда.

Посмотрите и повторите пример:

//Выведем слово 'ср':

var arr **=** ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];

alert(arr[2]);

## Ассоциативные массивы (объекты)

В предыдущем примере, чтобы обратиться к нужному элементу массива, мы писали в квадратных скобках его **порядковый номер** (нумерация начинается с нуля, если вы уже забыли). Эти порядковые номера называются **ключами массива**.

То есть мы получали *значение элемента массива по его ключу*.

В нашем случае JavaScript **сам определял ключи** для элементов - это были их **порядковые номера**. Иногда это может оказаться **неудобным** - к примеру, если мы хотим вывести на экран название третьего дня недели (среду), то должны написать в квадратных скобках цифру 2, а не 3.

Поэтому в JavaScript можно указать ключи в **явном** виде – так, как нам нужно. Делается это с помощью **объектов** (в других языках программирования они называются **ассоциативными**массивами). Объекты создаются с помощью фигурных скобок **{ }**, внутри которых пишутся элементы этого объекта в формате **ключ: значение**.

Давайте сделаем так, чтобы **понедельник** имел **ключ 1**, а не ноль, как было раньше (и всем остальным дням прибавим единицу):

//Укажем ключи в явном виде:

var obj **=** {1: 'пн', 2: 'вт', 3: 'ср', 4: 'чт', 5: 'пт', 6: 'сб', 7: 'вс'};

alert(obj[1]); //выведет 'пн'

Синтаксис здесь такой: **ключ**, затем идет двоеточие **:**, а потом **значение**.

Ключи не обязательно должны быть числами, они могут быть и **строками**. Сделаем массив, в котором ключами будут имена работников, а элементами - их зарплаты:

//Массив (объект) работников:

var obj **=** {'Коля': 200, 'Вася': 300, 'Петя': 400};

Узнаем зарплату Васи:

var obj **=** {'Коля': 200, 'Вася': 300, 'Петя': 400};

alert(obj['Вася']); //выведет 300

Кроме того, кавычки вокруг строковых ключей можно и не ставить:

var obj **=** {key1: 200, key2: 300, key2: 400};

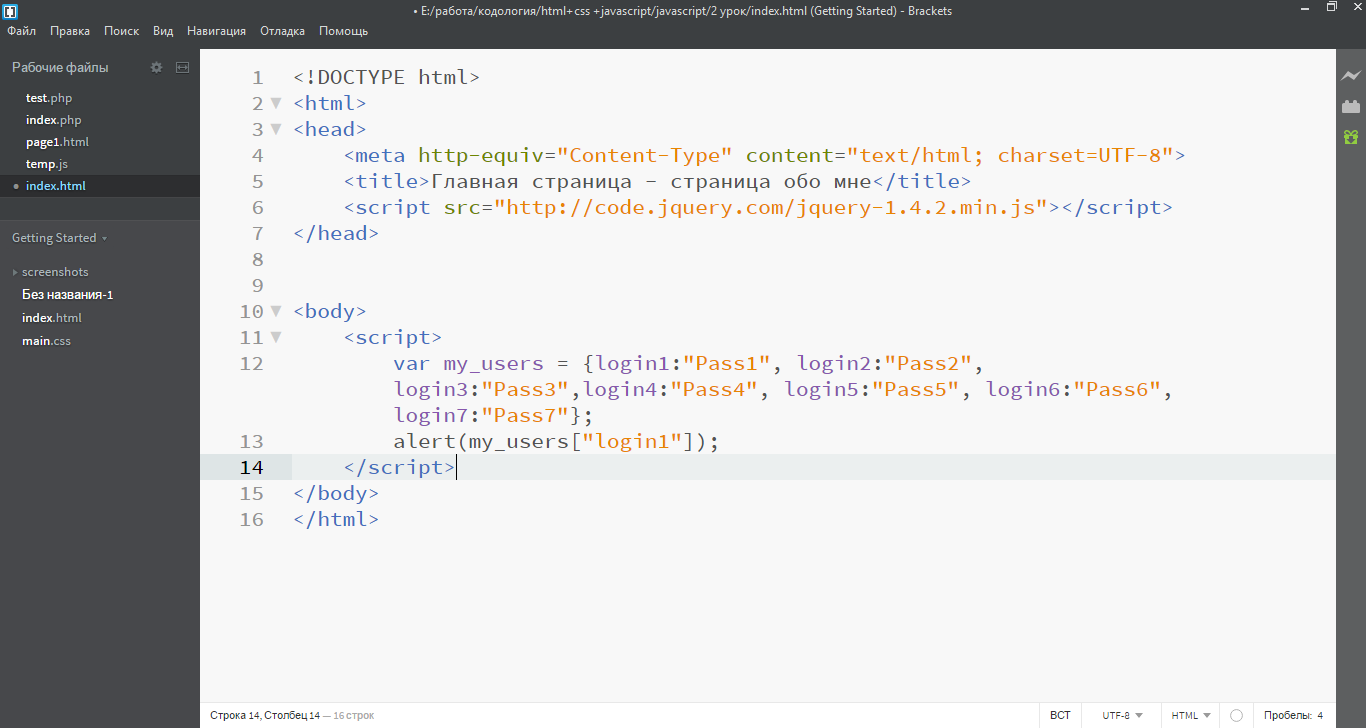
alert(obj['key1']); //выведет 200

Однако, на такие ключи накладываются ограничения: они не должны начинаться с цифры, не должны иметь символ дефиса или символ пробела внутри.

Если у вас есть такой ключ - просто возьмите его в кавычки и все будет ок.

Задачи:

1. Создать массив из 5 всевозможных логинов. Вывести на экран второй.
2. Создать ассоциативный массив с логинами и паролями, вывести на экран любые 3 пароля.



1. Дан массив с 5 числами, вывести на экран их сумму.

## Конструкция if-else

Для того, чтобы напрограммировать что-нибудь **полезное**, одних переменных далеко не достаточно. Нам нужен механизм, который позволит выполнять *определенный код в зависимости от каких-либо условий*.

То есть нам нужно иметь возможность спросить у JavaScript **'Если'**.

**Например так:** *если эта переменная меньше нуля, то вывести 'отрицательно', иначе (то есть если она больше нуля) вывести 'положительно'.*

В JavaScript для таких вопросов предназначена **конструкция if**, которая позволяет выполнять **определенный код при выполнении какого-либо условия**:

if (логическое выражение) {

Этот код выполниться,

если логическое выражение верно (то есть равно true)

} else {

Этот код выполнится,

если логическое выражение неверно (то есть равно false)

}

Обратите внимание на то, что **блок else не обязателен**.

**Логическое выражение** представляет собой тот самый вопрос, который мы хотим задать JavaScript. Например, чтобы спросить *'переменная****a****больше нуля'* мы напишем так: **a > 0**.

Примеры работы:

var a **=** 3;

/\*

Если переменная a больше нуля, то выведи 'верно',

иначе (если меньше или равна нулю) выведи 'неверно'

\*/

if (a **>** 0) {alert('Верно!');} else {alert('Неверно!');} //выведет 'Верно!'

var a **=** **-**3;

/\*

Если переменная a больше или равна нулю, то выведи 'верно',

иначе (если меньше нуля) выведи 'неверно'

\*/

if (a **>=** 0) {alert('Верно!');} else {alert('Неверно!');} //выведет 'Неверно!'

## Равенство по значению и типу

Для того, чтобы сравнить на **равенство** следует использовать оператор двойное равно **==**, а не одиночное **=**, как можно было подумать.

**Почему так?** Потому что **одиночное равно зарезервировано за присваиванием**. Смотрите пример:

var a **=** 0;

/\*

Если переменная a равна нулю, то выведи 'верно',

иначе (если не равна нулю) выведи 'неверно'

\*/

if (a **==** 0) alert('Верно!'); else alert('Неверно!'); //выведет 'Верно!'

**Все операции сравнения**

Возможные операции сравнения, которые можно использовать внутри **if**:

|  |  |
| --- | --- |
| a == b | **a** равно **b** |
| a === b | **a** равно **b** и они одинаковы по типу |
| a != b | **a** не равно **b** |
| a !== b | **a** не равно **b** или **a** равно **b**, но ни разные по типу |
| a < b | **a** меньше **b** |
| a > b | **a** больше **b** |
| a <= b | **a** меньше или равно **b** |
| a >= b | **a** больше или равно **b** |

**Несколько условий сразу**

Иногда может быть нужно составить какое-то **сложное условие**, например, пользователь вводит месяц своего рождения и вам нужно проверить, что введенное число *больше или равно 1 и меньше либо равно 12* (так как в году 12 месяцев).

Для этого существуют операторы **&&** (логическое И) и **||** (логическое ИЛИ).

var a **=** 3;

var b **=** **-**3;

//Если a больше нуля и b одновременно меньше нуля то...

if (a **>** 0 **&&** b **<** 0) alert('Верно!'); else alert('Неверно!'); //выведет 'Верно!'

var a **=** 3;

//Если a больше или равно 1 и меньше или равно 12 то...

if (a **>=** 1 **&&** a **<=** 12) alert('Верно!'); else alert('Неверно!'); //выведет 'Верно!'

var a **=** **-**3;

var b **=** **-**3;

/\*

Если a больше нуля ИЛИ b меньше нуля - хотя бы один из них, то...

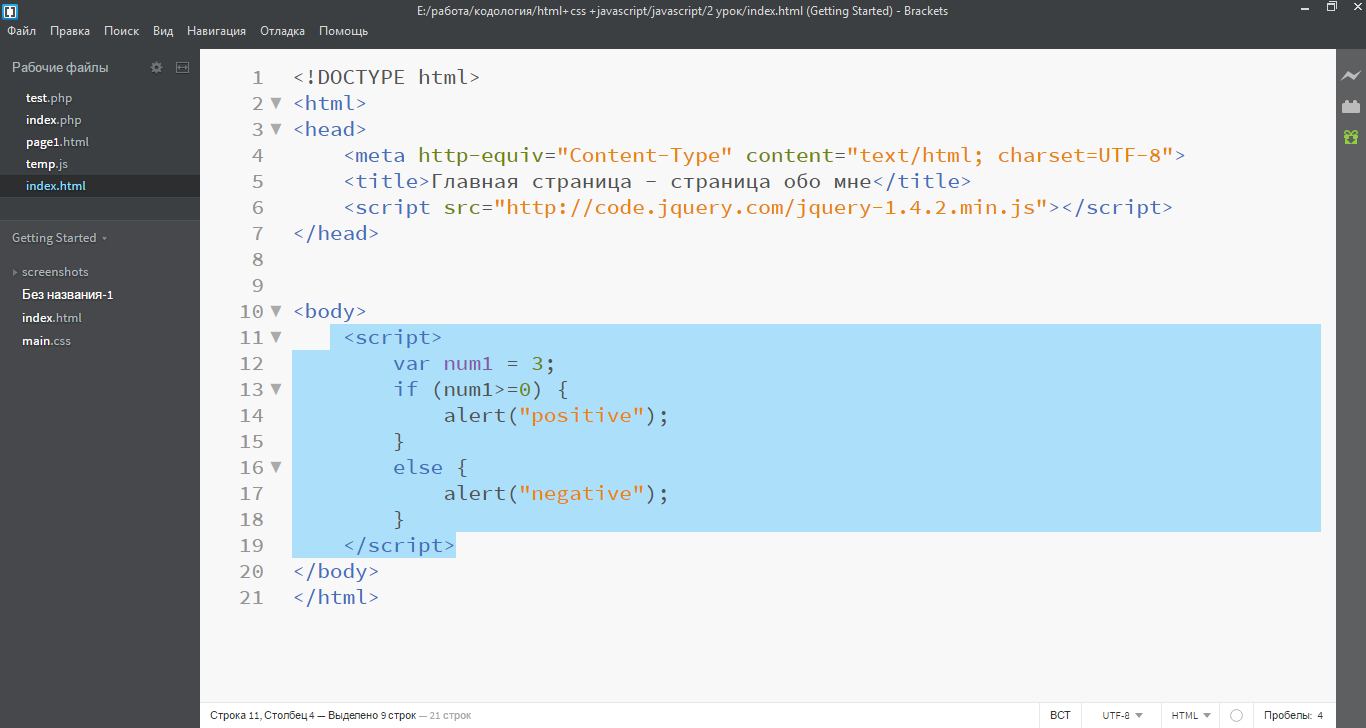
выведет 'Верно!', так как хотя a и не больше нуля,

но одно из условий - b < 0 - выполнится!

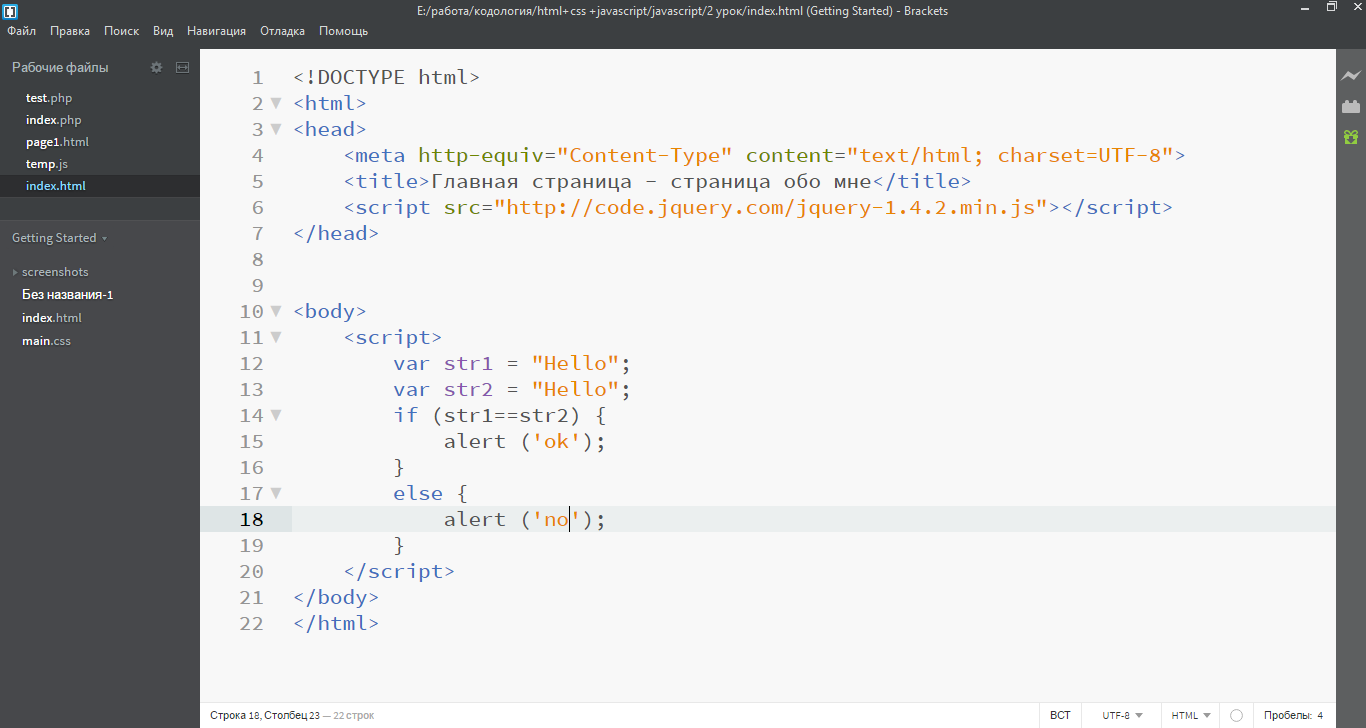
\*/

if (a **>** 0 **||** b **<** 0) alert('Верно!'); else alert('Неверно!');

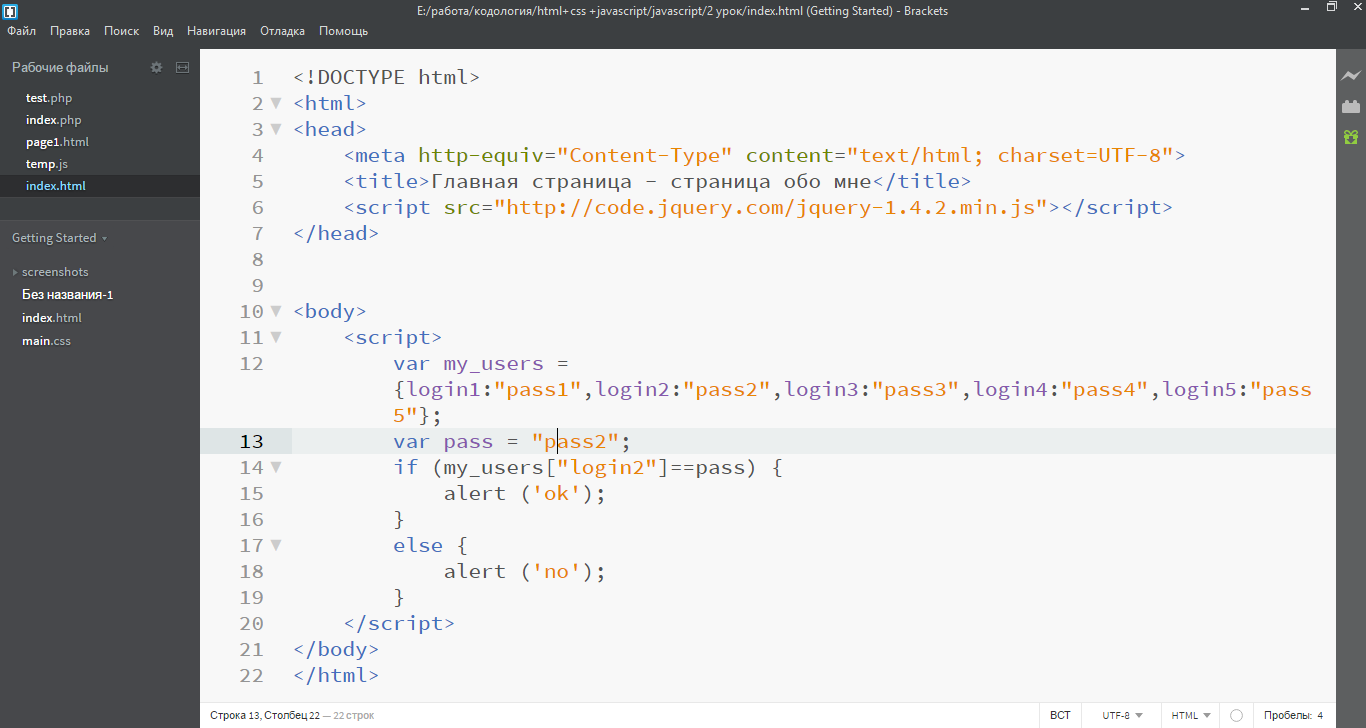
Задачи положительное или отрицательное число



Одинаковые ли строки



Проверить такой ли пароль у 3 пользователя.



Главная задача. Спросить у пользователя логин и пароль. При верном написать правильная авторизация.



Домашнее задание: вывести день. Утро, вечер, ночь в зависимости от времени суток на компьютере