HTML - Урок 12

Массивы и Циклы в JavaScript

Геннадий Караченцев

www.weblaboratory.in.ua

Что такое массивы?

Массив — тип или структура данных в виде набора компонентов *(элементов массива)*, расположенных в памяти непосредственно друг за другом.

Доступ к отдельным элементам массива осуществляется с помощью индексации, то есть ссылки на массив с указанием номера *(индекса)* нужного элемента.

Зачем нужны массивы?

Чаще всего массивы используются как хранение однотипных или родственных данных внутри одной переменной в виде списка данных.

Пример:

Яблоки, Груши, Бананы, Апельсины – все это фрукты

```
var fruits = ["Яблоки", "Груши", "Бананы", "Апельсины"]
```

Простым языком

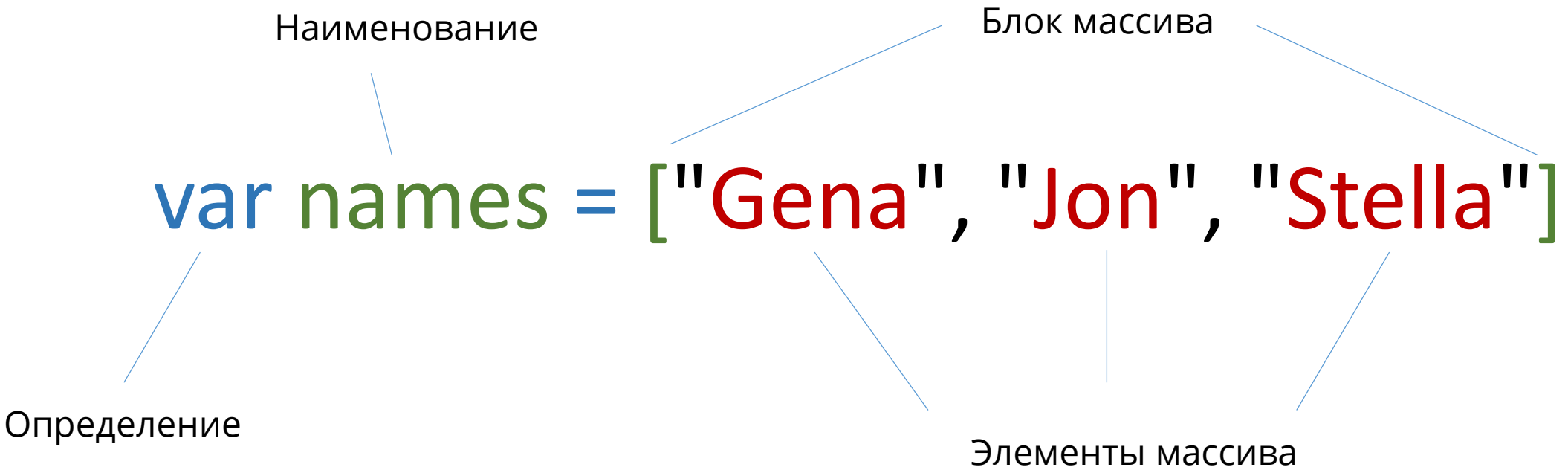
Переменная

var name = "Gena"
Gena

Массив

var names = ["Gena", "Jon", "Stella", "Andrea", "Kevin"]
names[0] -> Gena names[1] -> Jon names[2] -> Stella names[3] -> Andrea

Строение массива



Что можно делать с массивом?

- Получать данные из массива
- Добавлять данные в массив
- Удалять данные из массива
- Объединять массивы в один
- Преобразовывать строку в массив
- Копировать участок массива
- Сортировать массива
- Менять порядок элементов в массиве

Организация данных

0 1 2

Строки - `var names = ["Gena", "Jon", "Stella"]`

Числа - `var names = [1, 2, 3]`

Микс - `var names = [1, "Два", {number: "3"}]`

Работа с данными массива

```
var names = ["Gena", "Jon", "Stella"]
```

Получить данные:

```
console.log( names[0] ); // Gena
```

Заменить данные:

```
names[1]="Karl"; // ["Gena", "Karl", "Stella"]
```

Добавить элемент:

```
names[3]="Same"; // ["Gena", "Karl", "Stella"", “Same"]
```

Узнать длину массива:

```
console.log( names.length ); // 3
```

Если обратиться к ключу, у которого нет значения, то он будет равен undefine

Методы работы с массивом

- pop - Удаляет *последний* элемент из массива и возвращает его
- push - Добавляет элемент *в конец* массива
- shift - Удаляет из массива *первый* элемент и возвращает его
- unshift - Добавляет элемент *в начало* массива

Методы push и unshift могут добавлять сразу по несколько элементов

pop

```
var names = ["Gena", "Jon", "Stella"]

console.log( names.pop() );  // Удалит Stella
console.log( names );  // Gena, Jon
```

push

```
var names = ["Gena", "Jon"]

names.push("Stella");  // Добавит в конец Stella
console.log( names );  // Gena, Jon, Stella
```

shift

```
var names = ["Gena", "Jon", "Stella"]

console.log(  names. shift()  );  // Удалит Gena
console.log( names );  // Jon, Stella
```

unshift

```
var names = ["Jon", "Stella"]

names.unshift( "Gena" );  // Добавит в начало Gena
console.log( names );  // Gena, Jon, Stella
```

push и unshift

Могут добавлять сразу несколько элементов

```
var names = ["Gena"]

names.push("Jon", "Stella");
names.unshift("Marta", "Igor");

// Marta, Igor, Gena, Jon, Stella
console.log( names );
```

Перебор массива

Задача:

Необходимо перебрать массив и вывести значение каждого элемента на экран. Массив может быть любой длинны и мы не можем знать заранее какой.

Это задача для циклов. Когда происходит выполнение одного и того же действия несколько раз подряд, с заданным кол-во итераций (выполнением).

Задача для циклов

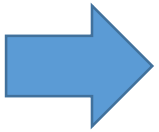
Задача:

Необходимо перебрать массив и вывести значение каждого элемента на экран. Массив может быть любой длинны и мы не можем знать заранее какой.

Это задача для циклов. Когда происходит выполнение одного и того же действия несколько раз подряд, с заданным кол-во итераций (выполнений).

Циклы на примере

Массив
- Яблоки
- Груши
- Апельсины
- Бананы
- Киви
- …
- Мандарины



Выполнить это действие по кругу,
столько раз, сколько его длинна

Вывод элемента ‘n’ на экран

На каждом шаге выполнения
Подставлять вместо ‘n’ номер шага.

Номер шага совпадает с номером элемента
массива, так мы можем обратится
по номеру к элементу и узнать его содержимое

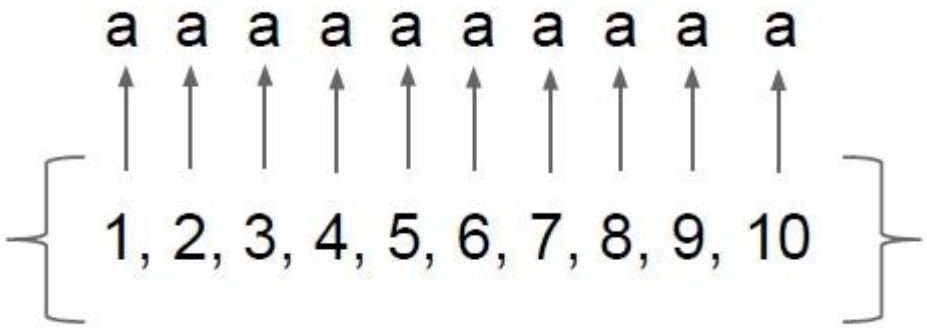
Циклы

Очень часто, при написание программ и скриптов, необходимо провести одно и тоже действие несколько раз.

К примеру цикл очень подойдет, для того, чтобы перебрать список товаров, по порядку. Или просто перебрать интервал чисел, например от 1 до 10 и произвести какие либо действия с каждым.

Для неоднократного повторения участка кода – применяются циклы.

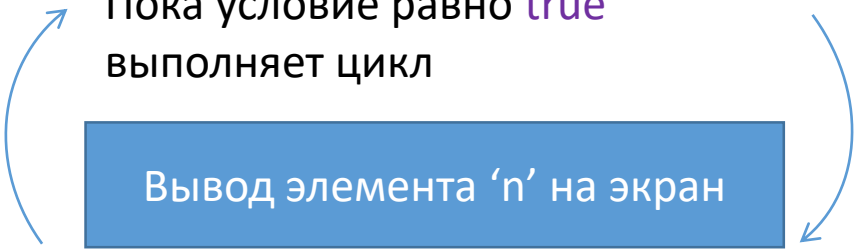
Например у нас есть alert() который нужно вывести 10 раз.



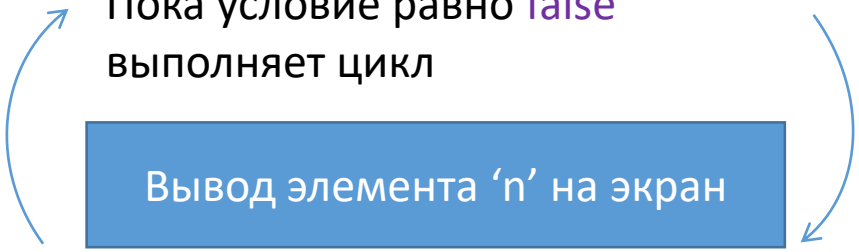
Виды циклов

- Цикл while выполняет действие пока условие верно (true)
- Цикл for выполняет действие пока условие не верно (false)

Цикл while



Цикл for



Цикл for

Самый, часто используемы цикл, это цикл for

Синтаксис

```
for (начало шага; условие прекращения; номер шага){
    // выполняем код
}
```

```
for (var i = 0; i < 3; i++){
    // выполняем код
}
```

- **Начало:** i=0
- **Условие:** i<3.
- **Шаг:** i++
- **Тело:** выполнение кода