

JS: Arrays

НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

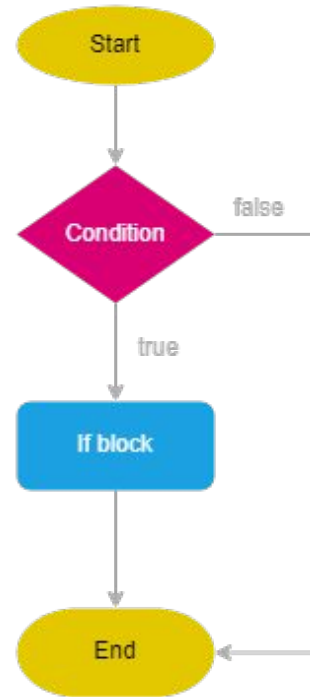
ЦЕЛЬ

Изучить особенности и синтаксис работы с массивами в js. Изучить методы push, pull, shift, unshift.

ПЛАН ЗАНЯТИЯ

- Логические операторы
- Шаблонные строки
- Массивы: доступ к элементам, длина, добавление, удаление

Условные операторы



Условный оператор „?“ (тернарный)

Так называемый «**условный**» оператор «вопросительный знак» позволяет нам сделать это более коротким и простым способом.

Оператор представлен знаком вопроса **?**. Его также называют «тернарный», так как этот оператор, единственный в своём роде, имеет три аргумента.

```
1 let result = условие ? значение1 : значение2;
```

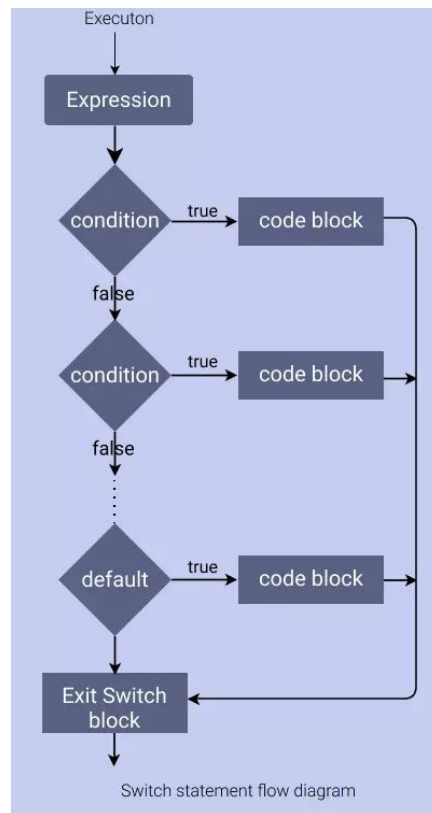


```
1 let accessAllowed = (age > 18) ? true : false;
```

Конструкция "switch"

Конструкция **switch** заменяет собой сразу несколько if.

Она представляет собой более наглядный способ сравнить выражение сразу с несколькими вариантами.



Конструкция **switch** имеет один или более блок **case** и необязательный блок **default**.

Выглядит она так:

```
1  switch(x) {  
2      case 'value1': // if (x === 'value1')  
3          ...  
4          [break]  
5  
6      case 'value2': // if (x === 'value2')  
7          ...  
8          [break]  
9  
10     default:  
11         ...  
12         [break]  
13 }
```

Переменная **x** проверяется на строгое равенство первому значению **value1**, затем второму **value2** и так далее.

Если соответствие установлено – **switch** начинает выполняться от соответствующей директивы **case** и далее, до ближайшего **break** (или до конца **switch**).

Если ни один **case** не совпал – выполняется (если есть) вариант **default**.

Пример использования switch (сработавший код выделен):

```
1  let a = 2 + 2;  
2  
3  switch (a) {  
4      case 3:  
5          alert( 'Маловато' );  
6          break;  
7      case 4:  
8          alert( 'В точку!' );  
9          break;  
10     case 5:  
11         alert( 'Перебор' );  
12         break;  
13     default:  
14         alert( "Нет таких значений" );  
15 }
```

Логические операторы



Папа, живет по принципу
"Если хотя бы кто-нибудь"

|| (ИЛИ)

Оператор «ИЛИ» выглядит как двойной символ вертикальной черты:

```
1 result = a || b;
```

Существует всего четыре возможные логические комбинации:

```
1 alert( true || true ); // true
2 alert( false || true ); // true
3 alert( true || false ); // true
4 alert( false || false ); // false
```

|| (или)

Обычно оператор || используется в if для проверки истинности любого из заданных условий.

```
1 let hour = 9;  
2  
3 if (hour < 10 || hour > 18) {  
4   alert( 'Офис закрыт.' );  
5 }
```

Можно передать и больше условий:

```
1 let hour = 12;  
2 let isWeekend = true;  
3  
4 if (hour < 10 || hour > 18 || isWeekend) {  
5   alert( 'Офис закрыт.' ); // это выходной  
6 }
```

&& (И)

Оператор И пишется как два амперсанда **&&** и возвращает true, если оба аргумента истинны, а иначе – false:

```
1 alert( true && true );    // true
2 alert( false && true );   // false
3 alert( true && false );   // false
4 alert( false && false );  // false
```

&& (И)

Пример с if:

```
1 let hour = 12;  
2 let minute = 30;  
3  
4 if (hour == 12 && minute == 30) {  
5     alert( 'Время 12:30' );  
6 }
```

! (НЕ)

Оператор **НЕ** представлен восклицательным знаком !

Оператор принимает один аргумент и выполняет следующие действия:

1. Сначала приводит аргумент к логическому типу true/false.
2. Затем возвращает противоположное значение.

```
1 alert( !true ); // false
2 alert( !0 ); // true
```

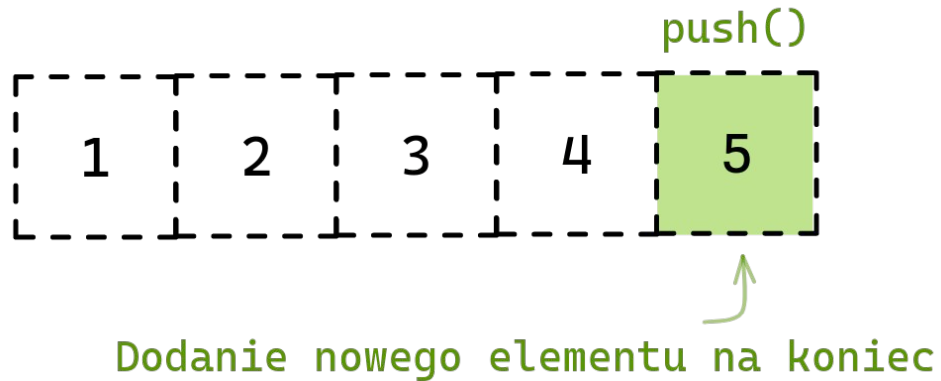
Шаблонные строки (Template Strings)

Шаблонные строки предоставляют удобный способ вставки переменных и выражений в строки.

В шаблонных строках можно использовать выражения, заключенные в `${}`

```
let myName = "Alice";  
let greeting = `Hello, ${myName}!`;   
console.log(greeting); // "Hello,  
Alice!"
```


Массивы



Довольно часто мы понимаем, что нам необходима упорядоченная коллекция данных, в которой присутствуют 1-й, 2-й, 3-й элементы и т.д.

Например, она понадобится нам для хранения списка чего-либо: пользователей, товаров, элементов HTML и т.д.

Для хранения упорядоченных коллекций существует особая структура данных, которая называется массив, Array.

Массив, состоящий из 5 элементов

123	7	50	-9	24
-----	---	----	----	----

0

1

2

3

4

индексы элементов массива

Объявление

Существует два варианта синтаксиса для создания пустого массива:

```
1 let arr = new Array();  
2 let arr = [];
```

Практически всегда используется второй вариант синтаксиса. В скобках мы можем указать начальные значения элементов:

```
1 let fruits = ["Яблоко", "Апельсин", "Слива"];
```

Объявление

Элементы массива нумеруются, начиная с нуля.

Мы можем получить элемент, указав его номер в квадратных скобках:

```
1 let fruits = ["Яблоко", "Апельсин", "Слива"];  
2  
3 alert( fruits[0] ); // Яблоко  
4 alert( fruits[1] ); // Апельсин  
5 alert( fruits[2] ); // Слива
```

Мы можем заменить элемент:

```
1 fruits[2] = 'Груша'; // теперь ["Яблоко", "Апельсин", "Груша"]
```

...Или добавить новый к существующему массиву:

```
1 fruits[3] = 'Лимон'; // теперь ["Яблоко", "Апельсин", "Груша", "Лимон"]
```

Длина массива

Общее число элементов массива содержится в его свойстве `length`:

```
1 let fruits = ["Яблоко", "Апельсин", "Слива"];  
2  
3 alert( fruits.length ); // 3
```

Содержание массива

В массиве могут храниться элементы любого типа.

```
let mixedArray = [42, "Hello", true, [1, 2, 3]];
```

Методы pop/push, shift/unshift

Очередь – один из самых распространённых вариантов применения массива. В области компьютерных наук так называется упорядоченная коллекция элементов, поддерживающая два вида операций:

- **push** добавляет элемент в конец.
- **shift** удаляет элемент в начале, сдвигая очередь, так что второй элемент становится первым.



На практике необходимость в этом возникает очень часто. Например, очередь сообщений, которые надо показать на экране.

Методы pop/push, shift/unshift

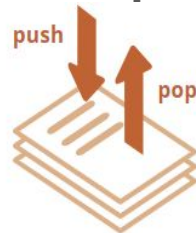
Существует и другой вариант применения для массивов – структура данных, называемая стек.

Она поддерживает два вида операций:

- push добавляет элемент в конец.
- pop удаляет последний элемент.

Таким образом, новые элементы всегда добавляются или удаляются из «конца».

Примером стека обычно служит колода карт: новые карты кладутся наверх и берутся тоже сверху:



Методы, работающие с концом массива:

pop

Удаляет последний элемент из массива и возвращает его:

```
1 let fruits = ["Яблоко", "Апельсин", "Груша"];
2
3 alert( fruits.pop() ); // удаляем "Груша" и выводим его
4
5 alert( fruits ); // Яблоко, Апельсин
```

Методы, работающие с концом массива:

push

Добавляет элемент в конец массива:

```
1  let fruits = ["Яблоко", "Апельсин"];  
2  
3  fruits.push("Груша");  
4  
5  alert( fruits ); // Яблоко, Апельсин, Груша
```

Методы, работающие с началом массива:

shift

Удаляет из массива первый элемент и возвращает его:

```
1 let fruits = ["Яблоко", "Апельсин", "Груша"];
2
3 alert( fruits.shift() ); // удаляем Яблоко и выводим его
4
5 alert( fruits ); // Апельсин, Груша
```

Методы, работающие с началом массива:

unshift

Добавляет элемент в начало массива:

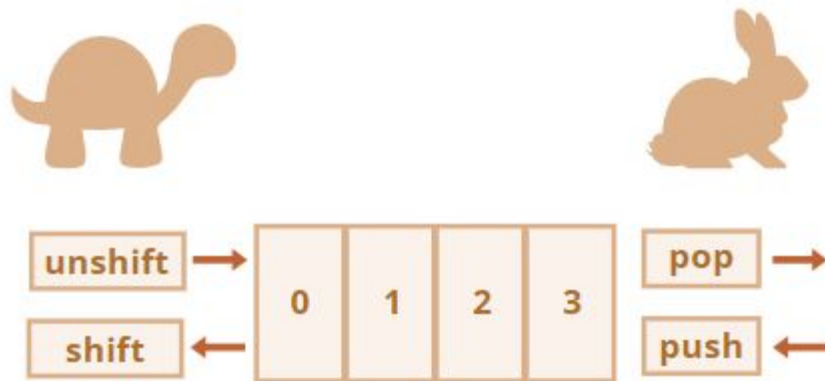
```
1 let fruits = ["Апельсин", "Груша"];
2
3 fruits.unshift('Яблоко');
4
5 alert( fruits ); // Яблоко, Апельсин, Груша
```

Методы `push` и `unshift` могут добавлять сразу несколько элементов:

```
1 let fruits = ["Яблоко"];
2
3 fruits.push("Апельсин", "Груша");
4 fruits.unshift("Ананас", "Лимон");
5
6 // ["Ананас", "Лимон", "Яблоко", "Апельсин", "Груша"]
7 alert( fruits );
```

Эффективность

Методы `push/pop` выполняются быстро, а методы `shift/unshift` – медленно.





Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH