

JS Basics

Falsy ⇒ "", 0, null, undefined, NaN, fals

var - hoisting, let, const - blocks

Temporal Dead Zone начинается в начале области видимости переменной и заканчивается ее объявлением

Primitives: Undefined, Boolean, String, Symbol, BigInt, Null

Non Primitives:

Object, Array, Function, RegExp, Date, Map, Set, Error, Promise, enum

Strings

`text.length`

`text.slice(start, end) => new string [start, end)`

`text.substring(start, end) new string [start, end)`

`text.substr(start, length) new string [start... length number of characters.`

`text.replace(oldValue, newValue)`

`text.replaceAll(oldValue, newValue)`

`text.toLowerCase() text.toLowerCase()`

`concat(string1, string2, ..., stringN)`

`text.trim() trimStart() trimEnd()`

`padStart(targetLength, padString): adds padding to the beginning`

`padEnd(targetLength, padString): adds padding to the end`

`charAt(index) charCodeAt(index)`

`split(separator, limit)`

Timers

Методы `setInterval(func, delay, ...args)` и `setTimeout(func, delay, ...args)` позволяют выполнять func регулярно или только один раз после задержки delay, заданной в мс.

Для отмены выполнения необходимо вызвать `clearInterval/clearTimeout` со значением, которое возвращают методы `setInterval/setTimeout`.

Timers (cont)

Планирование с нулевой задержкой `setTimeout(func, 0)` не гарантирует, что три фазы жизненного цикла будут выполнены сразу же. `setTimeout(func)` используется для вызова функции, цель которой — выполнение (bubbling). Разница между этими фазами заключается в порядке вызова обработчиков событий. В фазе захвата обработчики вызываются сверху вниз, а в фазе всплывания — снизу вверх. В фазе цели вызывается только один обработчик, связанный с элементом цели.

Браузер ограничивает 4-мя мс минимальную задержку между вызовами `setTimeout`, а также более вложенными вызовами `setTimeout`, а также более вложенными вызовами `setTimeout`, начиная с 5-го вызова.

```
let timerId = setTimeout(...);
clearTimeout(timerId);
```

Objects

Creating Objects

Literal/ Constructor Function/ Class

```
{x: "y"} (new function(){} new Object)
same as {} class Banana {} Object.create
```

```
Object.prototype.setPrototypeOf(proto[, proper tie sOb ject])
delete book.user.sReading
```

Обход объекта

```
for (let key in myObject) Object.entries() => [key, value]
```

```
Object.keys() => [key] Object.values() => [value]
```

Методы

```
someObject.hasOwnProperty('key1')
obj.__proto__ === person => Object.getPrototypeOf(obj)
```

```
Object.setPrototypeOf(obj, protoTypeOf(obj))
Object.getPrototypeOf(obj)
```

The `valueOf()` method of [Object] instances converts the this value [to an object]

Object.create()

```
function MyClass() {
  SuperClass.call(this);
  OtherSuperClass.call(this, ...args);
}
MyClass.prototype = Object.create(SuperClass.prototype);
```

позволяет создавать новый объект и устанавливать прототип этого объекта в объект, переданный в качестве параметра

Events

Event Phases — три фазы жизненного цикла события: захват (capturing), цель (target) и всплывание (bubbling).

Разница между этими фазами заключается в порядке вызова обработчиков событий. В фазе захвата обработчики вызываются сверху вниз, а в фазе всплывания — снизу вверх. В фазе цели вызывается только один обработчик, связанный с элементом цели.

Mouse Events: - click - dblclick - mousedown - mouseup - mousemove - mouseover - mouseout - contextmenu

Keyboard Events: - keydown - keyup - keypress

Form: formdata reset submit;

Input: change, input, cut, copy, paste

Array NOT Mutating

```
concat(array1, array2, ..., arrayN)
Array.from(callback(e1, index), thisValue)
```

```
Array.map(callback(element, index, array), thisValue)
```

```
Array.slice(start, end): new Array
```

```
Array.reduce(callback(accumulator, currentValue, index, array), initialValue)
```

```
Array.reduceRight(callback(accumulator, currentValue, index, array), initialValue)
```

Array Mutations

```
push(element1, element2, ..., elementN)
pop()
```

```
shift()
unshift(element1, element2, ..., elementN)
```

```
splice(start, deleteCount, item1, ..., itemN)
```

```
sort(compareFunction)
reverse()
```

```
fill(value, start, end)
```

Iterate Array

```
for (let i = 0; i <
scores.length; i++) {}
for (i in scores) {}
for (score of scores) {}
scores.forEach( (score) =>
{});
every( function (element,
index, array) { / ... / },
thisArg)
array.reduce( functi on( total,
el, i, arr),i nit ial Value )
every( fun ction (element,
index, array) { / ... / },
thisArg)
```

Copy Array

```
const arrNew = [...arrOld];
const newArray = [...oldAr -
ray];
{ const json = JSON.s tri ngi -
fy( old Array);
const newArray = JSON.p ars -
e(j son); }
const arrNew = arrOld.sl ice();
const arrNew = Array.f ro m(a -
rrOld);
const arrNew = [].con cat (ar -
rOld);
```



By itsMe
cheatography.com/itsme/

Published 9th July, 2023.
Last updated 9th July, 2023.
Page 2 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>