

JavaScript

Intro



Возможности JavaScript сильно зависят от окружения, в котором он работает.

FRONT END



может

- взаимодействовать с **DOM** (добавлять, изменять, удалять HTML-код страницы).
- изменять стили элементов
- отправлять и обрабатывать сетевые запросы
- запоминать данные на стороне клиента («local storage»)

не может

- обращаться к данным других страниц и вкладок (Same Origin Policy)
- читать/записывать произвольные файлы на жёстком диске

BACK END



может

- читать/записывать произвольные файлы
- взаимодействовать с БД
- отправлять и обрабатывать сетевые запросы

не может

- взаимодействовать с **DOM** локальной страницы

let, const, var

LET

VAR

CONST

Для создания переменной в JavaScript используйте ключевое слово `let`:

```
let myValue = "The String"; // объявление переменной и присвоение значения  
myValue = 43;               // новое значение переменной
```

`var` является устаревшим и не рекомендованным способом создания переменных

Ключевое слово `const` создает *переменную-константу*, т.е. такую переменную, которой нельзя присвоить новое значение.

```
const greetings = "Hello"; // объявление константы и присвоение значения  
  
greetings = "Привет"; //!!! Error: Assignment to constant variable.
```

Ключевое слово `const` не создает “неизменяемую” переменную. `const` — это константная ссылка на значение, но не всегда константное содержимое.

Переменные в JavaScript имеют динамическую типизацию, т.е. тип переменной определяется типом ее значения

В JavaScript есть 8 основных типов данных. Семь “примитивных”:

- **number** - целые числа или числа с плавающей точкой. Пример: `12.2`, `136`, `0.2`, *Infinity*, *-Infinity* и *NaN* - специальные числовые значения.
- **bigint** - целых числа произвольной длины. Пример: `987243987234987234987324237234n`
- **string** - строки. Пример: `“строка”`, `‘тоже строка’`, ``строка-шаблон ${varName}``
- **boolean** - логическое значение `true/false`.
- **null** - неизвестное значение, – отдельный тип, имеющий одно значение `null`.
- **undefined** - для неприсвоенных значений – отдельный тип, имеющий одно значение `undefined`.
- **symbol** - для уникальных идентификаторов. Пример `Symbol(“id”)`

и “не примитивный” **object** для более сложных структур данных.

Оператор **typeof** отображает строку с названием типа данных

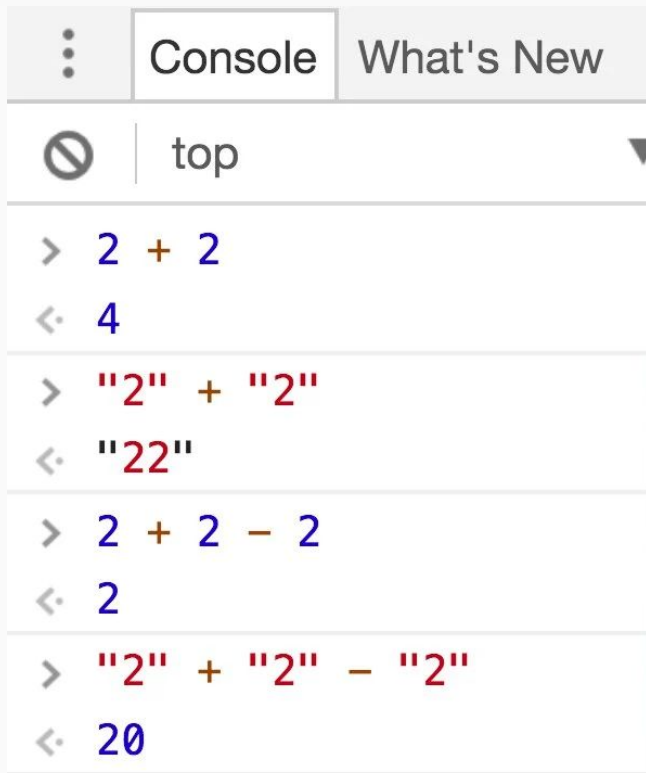
Примитивные типы данных

В JavaScript есть семь примитивных типов данных:

number	числа (целые или числа с плавающей точкой) <i>Пример: 12.2, 136, 0.2</i> Специальные значения: <i>Infinity, -Infinity и NaN</i>
bigint	целых числа произвольной длины <i>Пример: 987243987234987234987324237234n</i>
string	строки <i>Пример: "строка", 'тоже строка', `строка-шаблон \${varName}`</i>
boolean	логическое значение true/false
null	неизвестное значение, — отдельный тип, имеющий одно значение null.
undefined	для неприсвоенных значений — отдельный тип, имеющий одно значение undefined
symbol	для уникальных идентификаторов. <i>Пример Symbol("id")</i>

Оператор **typeof** отображает строку с названием типа данных.

JavaScript позволяет нам работать с примитивными типами данных как будто они являются объектами. У них есть и методы.



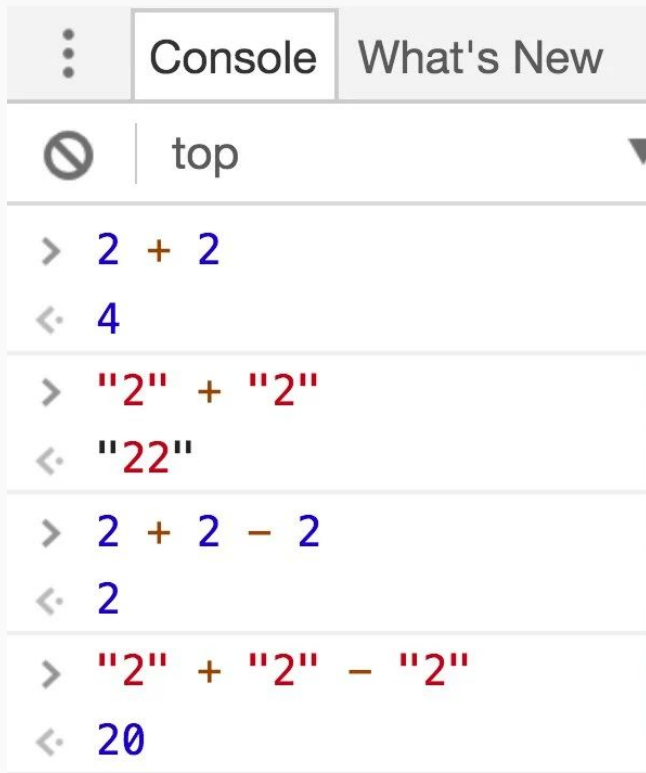
The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays several JavaScript expressions and their results, illustrating type coercion. The expressions and results are as follows:

Expression	Result
<code>2 + 2</code>	<code>4</code>
<code>"2" + "2"</code>	<code>"22"</code>
<code>2 + 2 - 2</code>	<code>2</code>
<code>"2" + "2" - "2"</code>	<code>20</code>

Приоритет преобразований в выражение:

1. строковое
2. численное
3. логическое

Преобразование типов



преобразование в строку:

```
String(true)           // "true"  
String(42)             // "42"
```

Когда хотя бы один из операндов в выражение — строка, `+` превращается в строковую операцию конкатенации, и JavaScript просто склеивает значения как текст.

```
"2" + "2"   // => строка "22"  
"2" + 4     // => строка "24"
```


Преобразование типов

```
⋮ Console What's New
⊘ top
> 2 + 2
< 4
> "2" + "2"
< "22"
> 2 + 2 - 2
< 2
> "2" + "2" - "2"
< 20
```

преобразование в число:

```
Number("123") // 123
Number("abc") // NaN
+"42" // 42 (унарный +)
parseInt("10px") // 10
parseFloat("3.14px") // 3.14
```

Унарный + перед значением, действует аналогично явному численному преобразованию

Если выражение не сводится к конкатенации строк, выполняется преобразование к числам и вычисление математического результата.

```
"3" - 2 // => число 1
```

FALSE

0,
null,
undefined,
NaN,
""

преобразование в boolean:

false:

```
Boolean(0)           // false
Boolean("")           // false
Boolean(null)         // false
Boolean(undefined)    // false
Boolean(NaN)          // false
```

Примеры:

```
Boolean("0")  // true  ← непустая строка
Boolean("false") // true это строка, а не boolean
!!"0"         // true
```

==

нестрогое сравнение (с приведением типа)

2==2 -> true

"2"]==2 -> true

===

строгое сравнение

2===2 -> true

"2"===2 -> false