infoShare ACADEMY

JavaScript operacje na typach prostych

infoShare Academy

infoShareAcademy.com

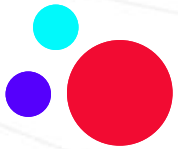
HELLO

Maciej Mikulski

Senior Front End Developer
@JIT.Team @Dialecticanet.com

infoShare ACADEMY

Number



Podstawowe operacje matematyczne

Dodawanie i odejmowanie – opertory + i -

```
let suma = 4 + 5
let roznica = 3 - 10
```

Mnożenie i dzielenie – operatory * i /

```
let iloczyn = 2 * 3
let iloraz = 10 / 2
```

Dzielenie przez 0

```
let pamietajCholero = 4 / 0
//pamietajCholero == Infinity
```

Modulo i potęgowanie

Modulo (czyli reszta z dzielenia, remainder) – operator %

```
let reszta = 5 % 2
```

Często wykorzystywane do sprawdzania czy element jest parzysty, np.

```
let licznik = 3
let parzysty
if (licznik % 2 === 0) { parzysty = true } else { parzysty = false }
```

Potęgowanie (exponentiation) – operator **

```
let potega = 2**4
```

Obiekt Math

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math

Obiekt Math opereuje tylko na typach Number – nie obsługuje typu BigInt.

- Math.max – największy element z tablicy

```
let najwieksza = Math.max([1, 2, 3, 4])
```

- Math.min – najmniejszy element z tablicy

```
let najmniejsza = Math.max([1, 2, 3, 4])
```

- Math.pow – potęga (równoznaczne z **)

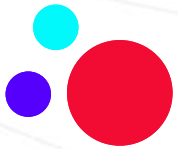
```
let potega = Math.pow(2,5)
```

- Match.sqrt – pierwiastek kwadratowy

```
let pierwiastek = Math.sqrt(64)
```

02. String

infoShareAcademy.com



Konkatenacja – łączenie ciągów znaków

Operator + lub metoda String.concat()

```
let data = ‘5’ + ‘ ’ + ‘marca’ + ‘ ’ + ‘ ‘ + ‘2022’
let grzeczniej = ‘Pani ’.concat(‘Joanna’)
```

Długość ciągu znaków, pozycja znaku

Długość – atrybut length

```
let dlugosc = ‘tekst’.length
```

Pozycja ciągu znaków w innym ciągu znaków – indexOf

```
let index = ‘123 PLN’.indexOf(‘PLN’)
// index = 4
```

Inne metody - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String

- charAt() – zwraca znak z wybranej pozycji
- Includes() – sprawdza czy tekst występuje w ciągu znaków
- trim() – usuwa puste znaki na początku i na końcu

Podział ciągu znaków

Wytnij ciąg znaków - slice()

```
let zdanie = ‘Nazywam się Maciej Mikulski’
let imie = zdanie.slice(12, 6)
```

Przyjmuje argumenty ujemne

```
let cena = ‘123,45 USD’
Let waluta = cena.slice(-3)
```

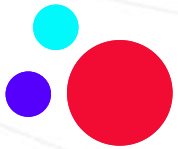
Interpolacja – “szablony”

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals

W ciągu znaków ograniczonym symbolami ` (backtick) możemy umieścić tzw. Tagi, które mogą być dowolnym wyrażeniem, które zostanie obliczone, a następnie zamienione na ciąg znaków I umieszczone w ciągu znaków.

```
let imie = ‘Maciej’
let powitanie = `Witaj ${imie}`

Let wyniki = `dwa plus dwa równa się ${2 + 2}`
```

03. Boolean



Operacje logiczne i porównania

Równa się bez względu na typ - operator ==

Równa się z zachowaniem typu - operator ===

Nie równa się bez względu na typ – operator !=

Nie równa się z zachowaniem typu – operator !==

Większ lub mniejsz – operatory >,<

Większy lub równy, mniejszy lub równy – operatory >=, <=

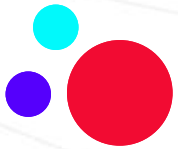
Logiczne AND - operator &&

Logiczne OR – operator ||

Logiczne NOT – operator !

03. Koercja typów

Type coertion



Operacje logiczne na typach nie boolowskich

Operatory && oraz || możemy stosować nie tylko do typu Boolean (true, false) ale także do pozostałych typów, które JavaScript zamieni na `truthy` lub `falsy`.

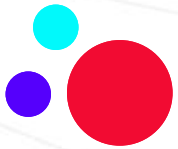
“Fałszywe” wartości:

- false
- null
- NaN
- 0
- pusty string (“” lub ‘’ lub ``);
- undefined

“Prawdziwe” wartości:

- !”Falłszywe”

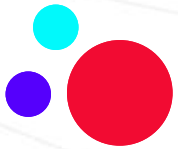
Operatory && oraz || zwracają oryginalną wartość nie zamieniając jej na true lub false.



Konwersje na typ Boolean

Jeśli chcemy wymusić zamianę wartości truthy lub falsy używamy operatora !!

```
let jakbyPrawda = !!0
let jakbyPrawda = !!''
let jakbyPrawda = !!'sdfsd'
```

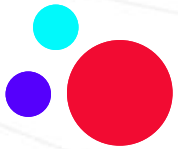


Short circuit evaluation

Operator && (logiczny AND) korzysta z właściwości, która mówi, że wynik będzie zawsze false jeśli lewa strona jest false. W takiej sytuacji wyrażenie po prawej nie zostanie w ogóle wykonane.

Wykorzystujemy to do warunkowego wykonania operacji, np.:

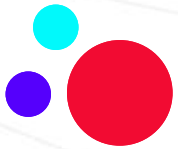
```
let kodRabatowy = undefined

function obliczRabat(produktyWKoszyku) { /*kod obliczający rabat */ }

let rabat = kodRabatowy && obliczRabat(produktyWKoszyku)
```



Koercja (type coercion)

JavaScript automatycznie zmienia typy danych tak aby umożliwić wykonanie operacji.

```
let wynik1 = ‘5’ + 5
let wynik2 = 5 + ‘5’
```



NaN – Not-A-Number

```
Math.sqrt('blabla’)
// NaN
5 + NaN
//Nan

('b' + 'a' + + 'a' + 'a').toLowerCase()
```

Fun Time

https://dev.to/damxipo/javascript-versus-memes-explaining-various-funny-memes-2o8c

https://www.freecodecamp.org/news/js-type-coercion-explained-27ba3d9a2839/

```
> typeof NaN
"number"
> 9999999999999999
10000000000000000
> 0.5+0.1==0.6
true
> 0.1+0.2==0.3
false
> Math.max()
-Infinity
> Math.min()
Infinity
> []+[]
""
> []+{}
"[object Object]"
> {}+[]
0
> true+true+true===3
true
> true-true
0
> true==1
true
> true===1
false
> (!+[]+[]+![]).length
9
> 9+"1"
"91"
> 91-"1"
90
> []==0
true
```

infoShare ACADEMY

Dziękuję za uwagę!

infoShareAcademy.com