# **Number**

Skaičiai

## **Sukurti Number**

Javascript turi tik vieno tipo skaičius - number. Nesvarbu ar jie naudojami per kablelį ar ne.

| let x = 3.14; *// Skaičius su dešimtosiomis dalimis*  let y = 3; *// Sveikas skaičius* |
| --- |

Laba dideli skaičiai gali būti rašomi su moksline notacija. (scientific (exponent) notation)

| let x = 123e5; *// 12300000* let y = 123e-5; *// 0.00123* |
| --- |

Numerius galima kurti per konstruktorių.

| const x = Number('123'); |
| --- |

## **NaN - Not a Number**

NaN yra bazinis žodis kuris pasako, kad reikšmė nėra teisingas Javascript numeris. Bandant daryti aritmetines operacijas su ne numeric reikšmėmis išmes NaN rezultatą.

| let x = 100 / "Apple"; *// NaN* |
| --- |

Norint patikrinti, ar gauta reikšmė yra Nan tam galime pasitelkti globalią funkcija isNaN();

NaN yra numerio tipas. typeof NaN gražins number.

| let x = 100 / "Apple"; isNaN(x); *// true* |
| --- |

| let x = NaN; let y = "5"; let z = x + y; |
| --- |

## 

## **Infinity**

Bazinis Infinity arba -Infinity tai yra reikšmė, kurią Javascript išmes jei pasieksite maximalų numerį.

| let myNumber = 2; *// Ciklas suksis iki Infinity* while (myNumber != Infinity) {  myNumber = myNumber \* myNumber; } |
| --- |

Dalyba iš 0 irgi sugeneruos Infinity.

Infinity yra numerio tipas. typeof Infinity gražins number.

## **Hexadecimal**

Javascript interpretuoja skaičių konstantas kaip hexadecimal jei yra nurodytos su “0x”

| let x = 0xFF; // 255 |
| --- |

## **Number properties**

| MAX\_VALUE | Gražina didžiausia teigiama skaičių |
| --- | --- |
| MIN\_VALUE | Gražina mažiausia neiteigiama skaičių |
| POSITIVE\_INFINITY | Gražina teigiama Infinity |
| NEGATIVE\_INFINITY | Gražina neigiama Infinity |

## **Number metodai**

**Metodas 1: toString()** gražina skaičių kaip tekstą.

| let x = 123; x.toString(); (123).toString(); (100 + 23).toString(); |
| --- |

**Metodas 2: toExponential()** suapvalina skaičių ir gražina kaip string.

| let x = 9.656; x.toExponential() *// 9.656* x.toExponential(2); *// 9.7* x.toExponential(4); *// 9.656* x.toExponential(6); *// 9.65600* |
| --- |

**Metodas 3: toFixed()** apvilina skaičių su nurodytom šimtosiomis dalimis ir gražina kaip string.

| let x = 9.656; x.toFixed(0); *// 10* x.toFixed(2); *// 9.66* x.toFixed(4); *// 9.6560* x.toFixed(6); *// 9.656000* |
| --- |

## **Konvertavimas į number reikšmę**

Yra trys būdai konvertuoti reikšmę į skaičius.

* The Number() method
* The parseInt() method
* The parseFloat() method

**Number()**

| Number(true); *// 1* Number(false); *// 0* Number("10"); *// 10* Number(" 10"); *// 10* Number("10 "); *// 10* Number(" 10 "); *// 10* Number("10.33"); *// 10.33* Number("10,33"); *// NaN* Number("10 33"); *// NaN* Number("John"); *// NaN* |
| --- |

**parseInt()**

| parseInt("-10"); *// -10* parseInt("-10.33"); *// -10*  parseInt("10"); *// 10* parseInt("10.33"); *// 10* parseInt("10 20 30"); *// 10* parseInt("10 years"); *// 10* parseInt("years 10"); *// NaN* |
| --- |

**parseFloat()**

| **parseFloat("10"); *// 10* parseFloat("10.33"); *// 10.33* parseFloat("10 20 30"); *// 10* parseFloat("10 years"); *// 10* parseFloat("years 10"); *// NaN*** |
| --- |