Cheatography

Nestor Javascript Cheat Sheet

by Nestor (Nestor) via cheatography.com/22757/cs/4752/

Typy wartości - proste

Liczby	Dowolna wartość numeryczna
String	
Boolean	Wartość logiczna, dwie wartość **TRUE** i **FALSE**
Function	wartość zwracana przez funkcję
Null	posiada tylko jedną wartość: **NULL**. Oznacza ona *brak wartości* lub *brak obiektu*. Przypisanie tej wartości do zmiennej powoduje wyczyszczenie jej wartości, ale nie powoduje usunięcia zmiennej.
Undefined	posiada tylko jedną wartość: undefined

Operator `typeof()` zwraca string z nazwą typu jaki ma przekazany parametr (np. zmienna). Może on zwrócić jedną z podanych wartości: *number*, *string*, *boolean*, *object*, *function* lub *undefined*.

Typy obiektowe

Object	obiekt, typ złożony, może przechowywać inne typy danych oraz funkcje
Function	wartość zwracana przez funkcję
Array	tablica, grupowanie danych w strukturę, gdzie każdemu elementowi przypisany jest określony indeks
Symbol	symbol
[]	obiekt
{}	obiekt

Operatory arytmetyczne

`x + y`	Sumuje argumenty `x` i `y`
`x - y`	Różnica `y` od `x`
`x * y`	Mnożenie `x` * `y`
`x / y`	Dzielenie `x` przez `y`
`x % y`	modulo - reszta z dzielenia `x` przez `y`
`x++`, `++x`	inkrementacja wartości o 1 *(przed, po)*
`x++`, `++x`	dekrementacja wartości o 1 *(przed, po)*
`-x`	zmiana znaku wartości x

W przypadku *SUMY* gdy jedna wartość ma typ **NUMBER**, a druga **STRING** wynik jest typu **STRING**

Porównania

`x == y`	**TRUE** jeśli *x* = *y*, niezależnie od typu
`x === y`	**TRUE** jeśli *x* = *y* łącznie z typami
`x != y`	**TRUE** jeśli *x* jest różne od *y*
`x !== y`	**TRUE** jeśli *x* i *y* nie są identyczne
`x > y`	**TRUE** jeśli *x* jest większe od *y*
`x >= y`	**TRUE** jeśli *x* jest większe lub równe *y*
`x && y`	**TRUE** jeśli obie wartości są **TRUE**
`x || y`	**TRUE** jeśli co najmniej jedna z wartości jest **TRUE**
`x ^ y`	**TRUE** jeśli jedna z wartości jest **TRUE**
`!x`	**TRUE** jeśli *x* jest **FALSE**

Przypisania

`x = y`	Ustawia *x* wartość *y*
`x += y`	skrócona postać `x = x + y`
`x -= y`	skrócona postać `x = x - y`
`x *= y`	skrócona postać `x = x * y`
`x /= y`	skrócona postać `x = x / y`
`x %= y`	skrócona postać `x = x % y`

Obsługa tablic

`push()`	Dokłada element na koniec tablicy
`pop()`	Usuwa ostatni element tablicy
unshift()	Dokłada element na początku tablicy
`shift()`	Usuwa pierwszy element tablicy
delete	usuwa element, np. po indeksie
`concat()`	Służy do łączenia tabel
`join()`	zwraca *STRING* połączonych elementów, parametr jako delimiter
`length()`	zwraca długość tablicy
splice()	Umożliwia zmianę lub usunięcie kilku elementów
`sort()`	Sortuje elementy typu *STRING*

By **Nestor** (Nestor)
cheatography.com/nestor/

Published 5th August, 2015.
Last updated 12th May, 2016.

Obsługa tablic (cont)

`reverse()`	Odwraca kolejność tablicy

```
var owoce = ["Li mon ka", " Man -
go"];
owoce.p us h("K iwi "); //Dokłada "-
Kiwi" na końcu
owoce.p op(); //Usuwa element "Kiwi" z
owoce
owoce.u ns hif t("K iwi ");
//wkłada "Kiwi" do owoce
owoce.s hi ft(); //Usuwa element "-
Kiwi" z owoce
```

Sortowanie numeryczne można wykonać:

```
var points = [40, 100, 1, 5, 25,
 10];
points.so rt( fun cti on(a, b){r
eturn a-b});
```

lub w odwrotnej kolejności - malejąco:

```
points.so rt( fun cti on(a, b){r
eturn b-a});
```

Instrukcja warunkowa if .. else

```
if (warunek) {
   rób coś;
} else if (warunek 2) {
   rób coś innego;
} else {
   rób coś jeszcze innego;
}
```

Instrukcja warunkowa switch .. case

```
switch(zmienna)
case 0:
    ale rt( " zmienna ma wartość
zero");
  break;
case 1:
    ale rt( " zmienna ma wartość
jeden");
  break;
default:
    ale rt( " zmienna posiada
inną wartoś ć");
  break;
}
```