

Wartości, zmienne i kontrola sterowania

| | |
|--|---|
| <code>\n</code> | Znak nowego wiersza |
| <code>typeof</code> | tworzy wartość łańcuchową reprezentującą typ podanej mu wartości. |
| <code>3 > 2</code> | jeden ze sposobów na uzyskanie wartości <code>true</code> |
| <code>"Z" < "a"</code> | Wielkie litery są zawsze „mniejsze” od małych. Porównywanie znaków odbywa się na podstawie standardu Unicode |
| <code>5e2</code> | 500 |
| <code>Math.max(2, 4)</code> | przyjmuje dowolną liczbę wartości liczbowych i zwraca największą z nich |
| <code>prompt(text, defaultText)</code> | Pierwszy argument zawiera pytanie, a drugi tekst, który zostanie wstępnie wyświetlony w polu tekstowym na odpowiedź. Gdy użytkownik wpisze jakiś tekst w oknie, funkcja zwróci go jako łańcuch. |
| <code>null == undefined</code> | <code>true</code> |
| <code>false == 0</code> | <code>true</code> |
| <code>"" == 0</code> | <code>true</code> |
| <code>"5" == 5</code> | <code>true</code> |
| <code>null === undefined</code> | <code>false</code> |
| <code>"5" === 5</code> | <code>false</code> |
| <code>NaN == NaN</code> | <code>false</code> |

Wartości, zmienne i kontrola sterowania (cont)

| | |
|--|---|
| <code>//komentarz calego wiersza</code> | |
| <code>/* stanowi początek komentarza, który kończy się ciągiem */</code> | |
| <code>isNaN(value)</code> | Sprawdza czy wartość nie jest liczbą |
| <code>Number(object)</code> | konwertuje różne wartości na liczby |
| <code>x y</code> | Jest to łatwy sposób na zdefiniowanie wartości „awaryjnej”. najpierw sprawdza wartość znajdującą się po jego lewej stronie. Jeśli w wyniku konwersji tej wartości na typ logiczny otrzyma <code>true</code> , zwraca tę wartość znajdującą się po jego lewej stronie. W przeciwnym przypadku zwraca wartość znajdującą się po prawej. |
| <code>x && y</code> | Gdy po jego lewej stronie znajduje się wartość dająca <code>false</code> po konwersji na typ logiczny, zwraca tę wartość. W przeciwnym przypadku zwraca wartość znajdującą się po prawej. |
| <code>x == 0 ? a : b</code> | Jeśli warunek <code>x==0</code> daje wartość <code>true</code> wykonywane jest <code>a</code> . W przeciwnym wypadku <code>b</code> |

Funkcje - deklaracja

```
function add(a, b) {
    return a + b;
}
console.log( add(2, 2));
```

lub

```
var add = function(a, b) {
    return a + b;
};
console.log( add(2, 2));
```

arguments.length - zwraca liczbę argumentów funkcji.
arguments[i] - odnosi się do `i` argumentu.

Ta funkcja przyjmuje zmienną `add`. Jej argumenty nazywają się `a` i `b`. Instrukcja **return a + b;** stanowi treść właściwą tej funkcji. Gdy zostaje wykonana instrukcja **return**, sterowanie jest przekazywane na zewnątrz funkcji do miejsca, w którym ta funkcja została wywołana i wartość zwrótka zostaje przekazana do kodu, który to wywołanie wykonał. Jeśli za instrukcją `return` nie ma żadnego wyrażenia, funkcja zwraca wartość **undefined**.

Funkcje - deklaracja zmiennych

o tym, które zmienne są widoczne w funkcji decyduje położenie tej funkcji w tekście programu. W funkcji widoczne są wszystkie zmienne, które zostały zdefiniowane „nad” jej definicją, czyli zarówno zdefiniowane w funkcjach ją zawierających jak i w głównym środowisku programu. Ta zasada określania dostępności zmiennych nazywa się leksykalnym określeniem **zakresu**.



Struktury danych: obiekty i tablice

`text.length` Każdy łańcuch ma własność o nazwie `length`, która odnosi się do liczby oznaczającej, z ilu znaków ten łańcuch się składa.

`text["length"]` Pierwszy z przedstawionych rodzajów zapisu jest skrótem pierwszego i można go stosować tylko wtedy, gdy nazwa własności mogłaby być poprawną nazwą zmiennej – nie zawiera spacji ani znaków specjalnych oraz nie zaczyna się od cyfry.

Obiekty - deklaracja i pętle

```
var car = {type:"Fiat",
model:"500", color:"white"};
var person = {
  firstName:"John",
  lastName: "Doe ",
  age:50,
  eye color: "blue"
};
// pętla wyświetlająca
wszystkie własności obiektu
person
for (var x in person) console.
log(x);
```

Obiekty - operacje

`delete` Słowo kluczowe `delete` usuwa własności. Próba odczytu nieistniejącej własności powoduje zwrócenie wartości `undefined`.

Obiekty - operacje (cont)

`person.novaWlasnosc = "novaWartosc"` Jeżeli operator `=` zostanie użyty do ustawienia własności, która jeszcze nie istnieje, to taka własność zostanie utworzona i dodana do obiektu.

`"age" in person` Do sprawdzenia czy obiekt ma określoną własność służy operator `in`. Zwraca on wartość logiczną.

`var object2 = object1` Jeśli zmienimy wartość `object1`, zmienia się również wartość `object2`, w przeciwieństwie do np. liczb czy stringów.

Tablice

`let arr = new Array(x);` tworzy nową tablicę o długości `x`, wypełnioną wartościami `undefined`.

`tablica[i] = x` Dodanie do tablicy w miejsce o indeksie `i` wartości `x`.

`tablica.push(x)` Dodanie wartości `x` na koniec tablicy. Przypisanie do zmiennej `tablica` powoduje błąd.

Tablice (cont)

`tablica.pop()` Usuwa i zwraca ostatnią wartość tablicy. Tablica będzie pozbawiona ostatniego elementu. Komenda będzie miała wartość usuniętego elementu jeśli przypisze się ją do zmiennej.

`tablica.join("")` Tworzy pojedynczy długi łańcuch z tablicy łańcuchów. Parametr jej wywołania jest wstawiany między wartościami tablicy.

`tablica.length = n;` Zmienia ilość elementów w tablicy na `n`.

`tablica.sort()` Sortuje tablicę.

`tablica.reverse()` Odwraca kolejność elementów w tablicy.

`tablica.sort(function(a, b) {return a-b});` Sortowanie rosnąco liczb.

Tablice (cont)

`tablica.map(function (x){return modyfikacja_x})` Metoda tworzy nową tablicę zawierającą wyniki wywołania podanej funkcji dla każdego elementu wywołującej tablicy. Nie modyfikuje tablicy, na której jest wywołany.

Stringi - operacje na stringach

`"String".toUpperCase()` Własność ta zwraca kopię łańcucha, w której wszystkie litery są wielkie.

`"String".toLowerCase()` Własność ta zwraca kopię łańcucha, w której wszystkie litery są małe.

`"String".split(" ")` Metoda ta tnie łańcuch na fragmenty, które zapisuje w elementach tablicy, a jako znaku podziału używa łańcucha przekazanego jej jako argument. **By zapamiętać utworzoną tablicę należy przypisać ją do zmiennej.**

`"String".charAt(i)` Zwraca znak z pozycji o indexie i. Dla nieistniejącego znaku zwraca "".

Stringi - operacje na stringach (cont)

`"String".slice(i,z)` Metoda ta kopiuje fragment łańcucha zaczynając od miejsca określonego liczbowo w pierwszym argumencie i kończąc przed znakiem znajdującym się na pozycji wyznaczonej przez drugi argument (też znak nie jest wliczany). Dla nieistniejącego znaku ignoruje tę część, która nie istnieje. Metoda **slice**, gdy przekaże się jej tylko jeden argument zwraca część łańcucha od określonej w tym argumencie pozycji do końca.

`"String".indexOf(searchvalue, start)` Zwraca index pozycji pierwszego pojawienia się szukanego ciągu znaków. Jeśli nie znajdzie, zwraca -1.

`"String".trim()` Usuwa spacje z obu stron. Trzeba przypisać do zmiennej.

Obiekty - data

`var now = new Date()` Zostanie utworzony obiekt zawierający bieżącą datę i godzinę

`var when = new Date(year, month, day, hours, minutes, seconds, milliseconds)`

`now.getFullYear()` Zwraca rok.

`now.getMonth()` Zwraca miesiąc (0-11).

`new.getDay()` Zwraca dzień tygodnia (0-6).

`new.getDate()` Zwraca dzień miesiąca (1-31).

`new.getHours()`

`new.getMinutes()`

`new.getSeconds()`

`new.getMilliseconds()`

`new.getTime()` Zwraca liczbę milisekund, jaka upłynęła od 1 stycznia 1970.

`new.getTimezoneOffset()` Zwraca różnicę w minutach między GMT (Londyn).

`data1.getTime() == data2.getTime()` porównanie takich samych dat

Argumenty te kolejno oznaczają rok, miesiąc, dzień, godzinę, minutę, sekundę oraz milisekundę. **Cztery** ostatnie argumenty są opcjonalne i jeśli nie zostaną zdefiniowane, nadawana jest im wartość 0. **Miesiące** w tych obiektach są numerowane od 0 do 11, co może powodować pomyłki. Co ciekawe, numeracja **dni** zaczyna się od 1.



Math

Math.round(n) Zaokrągla liczbę **n**.

Math.floor(n) Zaokrągla liczbę **n** w dół.

Math.ceil(n) Zaokrągla liczbę **n** w górę.



By **vizman**

cheatography.com/vizman/

Not published yet.

Last updated 18th December, 2017.

Page 4 of 4.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>