Wartości, zmienne i kontrola sterowania

\n	Znak nowego wiersza
typeof	tworzy wartość łańcuchową reprezentującą typ podanej mu wartości.
3 > 2	jeden ze sposobów na uzyskanie wartości true
"Z" < "a"	Wielkie litery są zawsze „mniejsze" od małych. Porównywanie znaków odbywa się na podstawie standardu Unicode
5e2	500
Math.max(2, 4)	przyjmuje dowolną liczbę wartości liczbowych i zwraca największą z nich
prompt(text, defaultText)	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.
null == undefined	true
false == 0	true
"" == 0	true
"5" == 5	true
null === undefined	false
"5" === 5	false
NaN == NaN	false

Wartości, zmienne i kontrola sterowania (cont)

//komentarz calego wiersza

/* stanowi początek komentarza, który kończy się ciągiem */

isNaN(value)	Sprawdza czy wartość nie jest liczbą
Number(object)	konwertuje różne wartości na liczby
x || y	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 true, zwraca tę wartość znajdującą się po jego lewej stronie. W przeciwnym przypadku zwraca wartość znajdującą się po prawej.
x && y	Gdy po jego lewej stronie znajduje się wartość dająca false po konwersji na typ logiczny, zwraca tę wartość. W przeciwnym przypadku zwraca wartość znajdującą się po prawej.
x == 0 ? a : b	Jeśli warunek x==0 daje wartość **true** wykonywane jest **a**. W przeciwnym wypadku **b**

Funkcje - deklaracja

```
function add(a, b) {
    return a + b;
}
consol e.l og( add(2, 2));
lub
var add = functi on(a, b) {
    return a + b;
};
consol e.l og( add(2, 2));
argume nts.length- zwraca liczbę argumentów funkcji.
argume nts[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 zwrotna 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 określaniem **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 = {
        fir stN ame :"Jo hn",
        las tNa me: " Doe ",
        age:50,
        eye Col or: " blu e"
};
// pętla wyświe tlająca
wszystkie własności obiektu
person
for (var x in person) consol -
e.l og(x);
```

Obiekty - operacje

delete person.age;	Słowo kluczowe delete usuwa własności. Próba odczytu nieistniejącej własności powoduje zwrócenie wartości undefined.

Obiekty - operacje (cont)

person.nowaWlasnosc = "nowaWartosc"	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 wartosci **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 wywoływania 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 (tez 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 przypisac 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)	Zaokragla liczbę **n**.
Math.floor(n)	Zaokrągla liczbę **n** w dół.
Math.ceil(n)	Zaokrągla liczbę **n** w górę.