

Obiekty

Pojęcie obiektu

Zgodnie z jego definicją znajdującą się w Wikipedii:

- Jest to wyodrębniony element rzeczywistości, mający znaczenie w rozpatrywanym modelu. Obiekt – element – może być materialny lub abstrakcyjny. Termin ten ma różne znaczenia w różnych dziedzinach, z uwagi na wielość użytych kontekstów.
-
- W informatyce jest to abstrakcja programistyczna posiadająca tożsamość, stan i zdefiniowany sposób zachowania.

Pytania

Pytanie 1: Jak rozumiesz pojęcie

- tożsamość,
- stan,
- zdefiniowany sposób zachowania

Pytanie 2: W jakich dziedzinach nauki posługujemy się pojęciem obiekt

Pytanie 3: Gdzie w życiu posługujemy się pojęciem obiekt

A serene landscape photograph of a wooden dock extending into a calm lake. The dock is made of weathered wooden planks and has a metal ring at its end. The water is still, reflecting the surrounding trees and the bright, hazy sky. The trees on the left and right banks are covered in autumn foliage, with shades of orange, yellow, and brown. The overall atmosphere is peaceful and quiet.

Obiekty wbudowane języka JavaScript

Obiekty wbudowane

JavaScript posiada kilka specjalnych obiektów, które są szczególnie przydatne w różnych sytuacjach. Mowa tu o obiektach wbudowanych, (ang. built-in objects) czyli obiektach predefiniowanych w języku JavaScript, które są dostępne od razu po załadowaniu skryptu. Są to obiekty, które służą do różnych celów, takich jak praca z tablicami, ciągami tekstowymi, datami, funkcjami itp.

Przykładowe obiekty wbudowane

MATH - służy do wykonywania różnych obliczeń matematycznych

DATE - służy do pracy z datami i godzinami

ARRAY - służy do tworzenia i wykonywania operacji na tablicach

STRING - służy do pracy z ciągami tekstowymi

Obiekt *String* - deklaracja

Jest to dość często spotykany obiekt, służący do pracy z ciągami tekstu.

Deklaracja i inicjalizacja zmiennej łańcuchowej:

- I. let tekst1;
 tekst1 = „Ala ma kota”;
- II. let tekst2 = „Ala ma kota”;

length – właściwość obiektu *String*

String posiada jedną właściwość: *length*, która zwraca długość łańcucha (ciągu znaków).

```
<script>  
    let tekst="Ala ma kota";  
    document.write(tekst.length);  
</script>
```

Poniższy skrypt wypisze na ekranie liczbę 11, ponieważ do ciągu znaków reprezentowanych przez zmienną tekst należy 9 liter i 2 spacje.

charAt() – metoda obiektu *String*

Zwraca ona znak występujący w łańcuchu pod danym indeksem.

Jako argument podajemy indeks danego elementu w ciągu, poczynając od 0.

Przykład: Poniższy skrypt wypisze na ekranie Aaaa

```
<script>
    let tekst="Ala ma kota";
    document.write(tekst.charAt(0));
    document.write(tekst.charAt(2));
    document.write(tekst.charAt(5));
    document.write(tekst.charAt(10));
</script>
```

Metody dające dostęp do podciągów obiektu String

```
graph TD; A[Metody dające dostęp do podciągów obiektu String] --> B[substring(start, end)]; A --> C[slice(start, end)]; A --> D[substr(start, length)];
```

`substring(start, end)`

`slice(start, end)`

`substr(start, length)`

Gdzie:

start – indeks pierwszego elementu podciągu (dla slice i substr również pozycja od końca podana ze znakiem „-”)

end - indeks ostatniego elementu +1

length – długość podciągu

Metody dające dostęp do podciągów obiektu *String*

Przykład użycia metod obiektu *String* do wyciągnięcia z podanego łańcucha „tekst” podciągu „kot”. W przypadku metody *substring* z liczbami ujemnymi jako argumentami zwraca pusty łańcuch:

```
<script>
  let tekst="Ala ma kota";
  document.write(tekst.slice(7,10)+"<br>");
  document.write(tekst.slice(-4,-1)+"<br>");
  document.write(tekst.substring(7,10)+"<br>");
  document.write(tekst.substring(-4,-1)+"<br>");
  document.write(tekst.substr(7,3)+"<br>");
  document.write(tekst.substr(-4,3)+"<br>");
</script>
```

kot
kot
kot

kot
kot

Metoda *repalce()*

Służy ona do zamiany pewnego podciągu znajdującego się w zmiennej na inny przy czym:

- wielkość liter w podciągu ma znaczenie.
- zamieniane jest pierwsze wystąpienie podciągu

Posiada dwa argumenty. Pierwszym argumentem jest dotychczasowy podciąg, a drugi to podciąg na który chcemy zmienić.

Wspomniane wcześniej ograniczenia można zmienić przy użyciu flag:

- g- zamienia wszystkie podciągi
- i – ignoruje wielkość liter w szukanym podciągu

Metoda replace() - przykłady

```
let tekst="Ala ma kota";  
let tekst1=tekst.replace("kota", "psa");  
document.write(tekst1+"<br>");  
let tekst2=tekst.replace("A", "O");  
document.write(tekst2+"<br>");  
let tekst3=tekst.replace("a", "o");  
document.write(tekst3+"<br>");  
let tekst4=tekst.replace(/a/g, "o");  
document.write(tekst4+"<br>");  
let tekst5=tekst.replace(/a/i, "o");  
document.write(tekst5+"<br>");  
let tekst6=tekst.replace(/a/ig, "o");  
document.write(tekst6+"<br>");
```

Małgorzata Tokarek

Ala ma psa
Ola ma kota
Alo ma kota
Alo mo koto
ola ma kota
olo mo koto

Konwersje małych i wielkich liter

Konwersji łańcucha na wielkie litery dokonuje metoda *toUpperCase()*, a na małe metoda *toLowerCase()*. Nie potrzebują one argumentów, bo jako metody są już wywoływane z pewnym *Stringiem*.

```
let tekst="Ala ma kota";  
document.write(tekst.toUpperCase()+"<br>");  
document.write(tekst+"<br>");  
document.write(tekst.toLowerCase()+"<br>");  
document.write(tekst+"<br>");
```

ALA MA KOTA
Ala ma kota
ala ma kota
Ala ma kota

Metoda *trim()*

Służy do usuwania białych znaków z początku i końca łańcucha.

Przykład

```
let napis = "   Szkoła   ";  
document.write(napis.length+"<br>");  
let napis1=napis.trim()  
document.write(napis1.length+"<br>");  
let napis2=napis.trimStart()  
document.write(napis2.length+"<br>");  
let napis3=napis.trimEnd()  
document.write(napis3.length+"<br>");
```

22

6

19

9

Pytania

Pytanie 1: Wymień właściwości obiektu *String*

Pytanie 2: Wymień metody obiektu *String*

Pytanie 3: Wymień metody do wyciągania podciągu z łańcucha. Podaj ich argumenty.

Pytanie 4: Podaj metodę służącą do wyciągania łańcucha o zadanej długości.

Pytanie 5: Jakimi sposobami można zmieniać wielkości liter (wymień różne sposoby, a nie tylko dedykowane temu metody)

Pytanie 6: W jaki sposób uzyskać dostęp do n-tego elementu łańcucha.

Zadania

Zadanie 1: Napisz skrypt, który dla zmiennej napis (let napis=„Dobranoc”) wyciągnie i wypisze na ekranie słowo noc. Pobranie podciągu należy zrobić na 4 różne sposoby, tj. trzema różnymi metodami obiektu *String* i litera po literze.

Zadanie 2: Napisz skrypt, który w napisie wprowadzonym przez użytkownika pozmienia wszystkie litery na nieparzystej pozycji na wielkie litery, a na parzystej na małe i wypisze wynik na ekranie. Przykładowe rozwiązania: Dzleń DoBrY, MiłeGo dNiA.

Zadanie 3: Napisz skrypt, który w napisie „Tego fajnego dnia pogoda była naprawdę fajna.” pozmienia wszystkie słowa fajny, fajna itd. na wspaniały, wspaniała itd. Spróbuj to zrobić jedną instrukcją. Wskazówka: pozmieniał wszystkie „fajn” na „wspaniał”.

Zadania cd.

Zadanie 4: Napisz skrypt, którego zadaniem będzie wypisanie wspak (w odwrotnej kolejności) słowa wprowadzonego przez użytkownika.

Zadanie 5: Napisz skrypt, w którym użytkownik wprowadza jakieś trzywyrazowe zdanie, a na ekranie wyrazy użyte w zdaniu zostaną wypisane w odwrotnej kolejności.

Obiekt Array

Służy on do pracy z tablicami. Podczas deklaracji nie trzeba określać rozmiaru tablicy. Tablica może przechowywać różne dane. Do kolejnych danych dostajemy się poprzez indeksy. Indeks pierwszego elementu to 0, a ostatniego rozmiar-1. Jego deklaracji można dokonać na jeden z poniższych sposobów:

- I. `let tablica = new Array();`
- II. `let tablica = [];`

Deklarowanie tablicy można połączyć z nadaniem wartości, a do elementu w tablicy można się dostać poprzez jego indeks:

```
let tablica1 = new Array(4, 8, 10);  
let tablica2 = ['Ala', 'Ela', 'Ula'];  
document.write(tablica1[1]+" "+tablica2[0]);
```

8 Ala

Jak widać na ekranie pojawi się liczba 8 (drugi element tablicy1 ma indeks 1), a po nim słowo Ala (pierwszy element tablicy2 ma indeks 0)

length – właściwość obiektu *Array*

Rozmiar tablicy, czyli liczba przechowywanych w niej elementów może w JavaScriptcie ulec zmianie. Rozmiar tablicy przydaje się podczas różnych operacji, dlatego potrzebny jest dostęp do jego rozmiaru.

Zapewnia to właściwość *length*

```
let tablica = ['Ala', 'Ela', 'Ula', 'Ola'];  
for (let i=0; i<tablica.length; i++)  
    document.write(tablica[i] + "<br>");
```

Ala
Ela
Ula
Ola

indexOf() – metoda obiektu Array

Metoda zwraca indeks elementu o podanej wartości

```
let tablica = ['Ala', 'Ela', 'Ula', 'Ola'];  
document.write(tablica.indexOf('Ola') + "<br>");  
document.write(tablica.indexOf('Ala') + "<br>");  
document.write(tablica.indexOf('Ula') + "<br>");  
document.write(tablica.indexOf('Ela') + "<br>");
```

3
0
2
1

Dodawanie i usuwanie elementów tablicy

tablica – metody obiektu zmieniające tablicę

dodanie elementu
do tablicy

wyciąganie
elementu z tablicy

`nazwa.push(...)`
dodanie elementów
na koniec

`nazwa.unshift(...)`
dodanie elementów
na początek

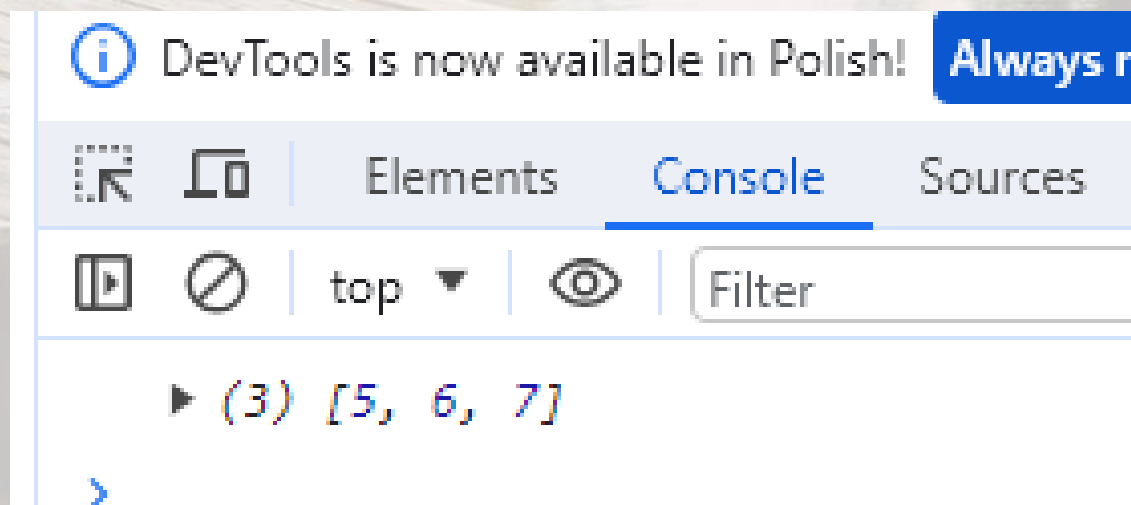
`nazwa.shift()`
pobiera(zabiera)
pierwszy element

`nazwa.pop()`
pobiera(zabiera)
ostatni element

Dodawanie elementu na koniec tablicy – metoda *push()*

Metody tej używamy tak jak innych obiektów czyli
`obiekt.metoda(wartość) : tablica.push(wartość)`

```
let liczby = [];  
//dokładamy do tablicy liczby  
liczby[0]=5;  
liczby.push(6);  
liczby.push(7);  
console.log(liczby);
```



Dodawanie elementu na początek tablicy – metoda *unshift()*

Metody tej używamy tak jak innych obiektów czyli
`obiekt.metoda(wartość) : tablica.unshift(wartość)`

```
console.log(liczby);  
liczby.unshift(4);  
liczby.unshift(3);  
console.log(liczby);  
document.write(liczby);
```

3,4,5,6,7

DevTools is now available in Po

Elements Console

top Filter

▶ (3) [5, 6, 7]

▶ (5) [3, 4, 5, 6, 7]

Usuwanie elementów z początku tablicy - metoda *shift()* i końca - metoda *pop()*

```
console.log(liczby);  
liczby.shift();  
console.log(liczby);  
liczby.pop();  
console.log(liczby);
```

► (5) [3, 4, 5, 6, 7]

► (4) [4, 5, 6, 7]

► (3) [4, 5, 6]

Liczby losowe

To generowania liczb losowych służy metoda *random* obiektu *Math*. Zwraca ona liczby z przedziału $[0,1)$. Zatem jeśli chcemy mieć wylosowane inne liczby to należy wykonać pewne działania matematyczne. Poniższa funkcja zwraca liczby z przedziału od 0 do wartości argumentu.

```
function losowa(a){  
  let b = Math.random()*(a+1);  
  return Math.floor(b);  
}
```


Liczby losowe – testowanie funkcji losowa(a)

<code>console.log(losowa(2));</code>	2
<code>console.log(losowa(2));</code>	1
<code>console.log(losowa(2));</code>	0
<code>console.log(losowa(2));</code>	0
<code>console.log(losowa(2));</code>	2
<code>console.log(losowa(2));</code>	0
<code>console.log(losowa(2));</code>	2
<code>console.log(losowa(2));</code>	1
<code>console.log(losowa(2));</code>	0
<code>console.log(losowa(2));</code>	2
<code>console.log(losowa(2));</code>	1
<code>console.log(losowa(2));</code>	2
<code>console.log(losowa(2));</code>	1
<code>console.log(losowa(2));</code>	2

Wypełnianie tablicy liczbami losowymi

```
let liczby = [];  
let i;  
for (i=0; i<10; i++)  
|   liczby[i] = losowa(50);  
for (i=10; i<20; i++)  
|   liczby.push(losowa(50) +100);  
console.log(liczby);  
for (i=20; i<30; i++)  
|   liczby.unshift(losowa(50) +500);  
console.log(liczby);
```

► (20) [44, 25, 28, 29, 50, 48, 9, 41, 46, 20, 145, 146, 114, 140, 114, 127, 140, 143, 122, 125]

► (30) [502, 543, 540, 502, 523, 500, 500, 501, 545, 541, 44, 25, 28, 29, 50, 48, 9, 41, 46, 20, 145, 146, 114, 140, 114, 127, 140, 143, 122, 125]

Zadania

Utwórz na pulpicie folder o nazwie takiej jak Twoje imię Do zadań napisz funkcję o nazwie *losujLiczbę(a)*, która będzie zwracała całkowitą liczbę losową z przedziału $[0,a]$. Kolejne zadania powinny być wykonywane po wciśnięcie właściwych przycisków:

Zadanie 1

Zadanie 2

Zadanie 3

Zadanie 4

Zadanie 1: Napisz skrypt wywołujący funkcję, która na trzy różne sposoby wpisze wylosowaną liczbę $a \in [0,5]$ do tablicy o nazwie *tablica1*. Wyświetl na ekranie nową zawartość tablicy.

Zadania cd.

Zadanie 2

W istniejącym skrypcie dopisz funkcję o nazwie *parzyste()*, która wypełni tablicę *tablica2* dziesięcioma liczbami parzystymi $a \in [0, 20]$. Wskazówka zrób wewnętrzną pętlę *do while()*, która losuje dopóki wylosowana liczba jest nieparzysta. Efekt wyświetl na ekranie.

Zadanie 3

W istniejącym skrypcie dopisz funkcję o nazwie *gdyParzysta()*, która 10 razy losuje liczbę $a \in [0, 20]$ i dopisuje element do tablicy o nazwie *tablica3i4* tylko wtedy gdy wylosowana liczba jest parzysta. Efekt wyświetl na ekranie.

Zadanie 4

Tak długo jak w tablicy *tablica3i4* jest więcej niż 3 liczby usuwaj pierwszy element tablicy.

Metoda reverse()

Jak nazwa metody wskazuje ... Zmienia oryginał!!!

```
console.log(liczby);  
liczby.reverse();  
console.log(liczby);
```

► (10) [18, 21, 37, 34, 38, 36, 2, 2, 31, 2]

► (10) [2, 31, 2, 2, 36, 38, 34, 37, 21, 18]

Metoda *join()*

Za pomocą metody `join()` można łączyć elementy tablicy w jeden ciąg znaków. W metodzie tej można opcjonalnie podać parametr, który określi znak oddzielający kolejne elementy tablicy. Jeżeli nie zostanie podana wartość tego parametru, domyślnym znakiem będzie przecinek.

```
//tworzenie tablicy
let tablica = new Array();
for (let i=0; i<ile; i++)
    tablica[i]=randomToMax(max);
document.write("Wylosowano liczby:&nbsp;&nbsp; ");
document.write(tablica.join() + "<br><br>");
```

Wylosowano liczby: 58, 22, 76, 66, 93, 31, 33, 85, 84, 46

Sortowanie tablic

Do sortowania elementów tablicy służy metoda `sort()`. Modyfikuje ona oryginalną tablicę – nie tworzy żadnej kopii. Domyślnie tablica jest sortowana leksykograficznie - alfabetycznie. Powoduje to, że liczba 12459 będzie mniejsza od 4567, ponieważ w liczbie 12459 cyfra na pierwszej pozycji jest mniejsza. Aby samemu zdefiniować kryteria sortowania w wielu językach tworzy się samemu funkcję (zwyczajowo nazywaną `comparator`), która w JS jest argumentem funkcji `sort`.

<https://podstawyjs.pl/sortowanie-tablicy-js/>

Comparator - założenia

- jeżeli funkcja $f(a, b)$ zwróci wartość mniejszą od 0, to wartości a zostanie nadany indeks mniejszy od indeksu przyznanego wartości b ,
- jeżeli funkcja $f(a, b)$ zwróci wartość równą 0, to wartości indeksów pozostaną bez zmian,
- jeżeli funkcja $f(a, b)$ zwróci wartość większą od 0, to wartości a zostanie nadany indeks większy od indeksu przyznanego wartości b .

Przykłady comparatorów

Poniżej napisano dwa comparatory, wykorzystywane w dalszej części skryptu, jeden do sortowania rosnącego, a drugi malejącego:

```
function comparatorR(a, b) {  
    return a - b;  
}  
function comparatorM(a, b) {  
    return b - a;  
}
```


Metoda *sort* bez comaparatora

```
tablica.sort();  
document.write("Sortowanie domyślne:&nbsp;");  
document.write(tablica.join() + "<br><br>");
```

Sortowanie domyślne: 14,46,54,60,65,68,69,8,86,91

Comparator jako argument metody *sort*

Z komparatorem do sortowania rosnąco

```
document.write("Sortowanie rosnąco:&nbsp;&nbsp;&nbsp;");  
tablica.sort(comparatorR);  
document.write(tablica.join() + "<br><br>");
```

Sortowanie rosnąco: 8,14,46,54,60,65,68,69,86,91

Comparator jako argument metody *sort*

Z komparatorem do sortowania malejąco

```
document.write("Sortowanie malejąco:&nbsp;&nbsp;&nbsp;");  
tablica.sort(comparatorM);  
document.write(tablica.join() + "<br><br>");
```

Sortowanie malejąco: 91,86,69,68,65,60,54,46,14,8

obiekt *Date*

Małgorzata Tokarek



Pojęcie *timestamp*

W informatyce powszechne jest przechowywanie dat jako liczby. Sposób w jaki liczby te odpowiadają konkretnym datom trochę się różni w zależności od programu. W języku JavaScript (i PHP) wartości daty i czasu są przechowywane w formacie *timestamp*, czyli jako liczba milisekund, które upłynęły od północy 1 stycznia 1970 roku.

Data i czas

Dzięki *timestamp* obiekt *Date* przechowuje wartości daty i czasu oraz pozwala wykonywać na nich różne operacje, można np.:

- odczytać wartość daty i czasu,
- pobrać oddzielnie rok, miesiąc, dzień, ...
- pobrane części niezależnie od siebie modyfikować

Tworzenie obiektów Date.

Podczas tworzenia dat, podobnie jak i innych obiektów używamy słowa kluczowego *new*. Jeśli nie podamy żadnych argumentów to podawana jest aktualna data (i czas).

```
let dzisiajData = new Date();  
document.write(dzisiajData + "<br>");
```

Thu Nov 02 2023 16:43:59 GMT+0100 (czas środkowoeuropejski standardowy)

Tworzenie obiektów *Date*.

Można utworzyć obiekt z określoną liczbą parametrów. Może ich być od dwóch do siedmiu (rok, miesiąc, dzień, godzina, minuty, sekundy, milisekundy). Taki obiekt będzie zawierał ściśle określoną wartość daty i godziny.

```
let dzisiajData = new Date();  
let wybranaData = new Date(2022, 2, 2);  
document.write(dzisiajData + "<br>");  
document.write(wybranaData + "<br>");
```

Thu Nov 02 2023 16:43:59 GMT+0100 (czas środkowoeuropejski standardowy)
Wed Mar 02 2022 00:00:00 GMT+0100 (czas środkowoeuropejski standardowy)

Pytanie

Co się nie zgadza na poprzednim slajdzie ?

Miesiące liczone są od zera czyli

0 to styczeń

1 to luty

2 to marzec

itd

Zastosowanie numeracji od 0

Wygodnym zastosowaniem numeracji miesięcy i dni tygodnia od zera może być utworzenie tablic jak poniżej i stosowanie ich w skrypcie:

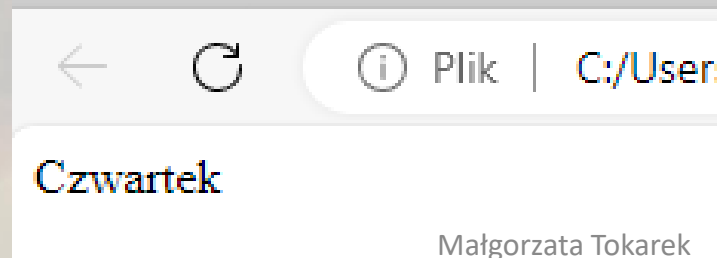
```
//definicja tablic
let miesiace = ["stycznia", "lutego", "marca", "kwietnia", "maja",
    "czerwca", "lipca", "sierpnia", "września", "października", "listopada",
    "grudnia"];
let dni = ["Niedziela", "Poniedziałek", "Wtorek", "Środa", "Czwartek",
    "Piątek", "Sobota"];
```

Zastosowanie tablicy w dacie

Efektem kodu:

```
//dzisiejsza data  
let data = new Date();  
let dzienTygodnia = data.getDay();  
document.write(dni[dzienTygodnia]);
```

jest wyświetlenie na ekranie bieżącego dnia tygodnia w języku polskim



Podstawowe metody obiektu Date

Metoda	Opis
<i>getDate()</i>	dzień miesiąca (wartość z przedziału 1 - 31)
<i>getDay()</i>	dzień tygodnia (0 dla niedzieli, 1 dla poniedziałku, 2 dla wtorku itd.)
<i>getYear()</i>	liczba reprezentująca rok (dla zakresu dat 1900 - 1999 jest to 2-cyfrowa liczba, a dla późniejszych jest to liczba 4-cyfrowa)
<i>getFullYear()</i>	pełna liczba reprezentująca rok
<i>getHours()</i>	zwraca aktualną godzinę (z przedziału 0 - 23)
<i>getMilliseconds()</i>	zwraca milisekundy (z przedziału 0 - 999)
<i>getMinutes()</i>	zwraca minuty (z przedziału 0 - 59)
<i>getMonth()</i>	zwraca aktualny miesiąc (0 - styczeń, 1 - luty itd.)
<i>getSeconds()</i>	zwraca aktualną liczbę sekund (z przedziału 0 - 59)
<i>getTime()</i>	zwraca aktualny czas jako liczbę reprezentującą liczbę milisekund która upłynęła od godziny 00:00 1 stycznia 1970 roku

Wyświetlenie daty według własnego sposobu

```
//dzisiejsza data
let data = new Date();

//składowe dzisiejszej daty
let rok = data.getFullYear();
let mies = data.getMonth();
let nr_dzien = data.getDate();
let dzien = data.getDay();
let godz = data.getHours();
let min = data.getMinutes();
let sek = data.getSeconds();
```

```
//uzupełnianie zapisu 0 z przodu
if (min < 10) {
    min = "0" + min;
}
if (sek < 10) {
    sek = "0" + sek;
}
```

```
let p_data_czas = dni[dzien] + ", " + nr_dzien +
    " " + miesiace[mies] + " " + rok +
    " roku<br>" + "Godzina: " + godz +
    ":" + min + ":" + sek;

document.write(p_data_czas);
```

Czwartek, 30 listopada 2023 roku
Godzina: 15:42:32

Tworzenie obiektów Date

Do tworzenia obiektu wykorzystamy datę wprowadzoną w formularzu. Sama data daje nam czas o północy.

```
<form>
  <input type="date" id="data" value="">
  <input type="button" value="Wczytaj" onclick="dates()">
</form>
<p id="dzisiaj"></p>
<p id="wybrana"></p>
<p id="formularzowa"></p>
```

Tworzenie obiektów *Date*

I skrypcie pobieramy datę oraz podajemy jej wartość do obiektu

```
function dates(){  
    let zformularza    = document.getElementById("data").value;  
    let dzisiajData    = new Date();  
    let wybranaData    = new Date(2022, 2, 2);  
    let formularzData  = new Date(zformularza)  ;  
    document.getElementById("dzisiaj").innerHTML = dzisiajData;  
    document.getElementById("wybrana").innerHTML = wybranaData;  
    document.getElementById("formularzowa").innerHTML = formularzData;  
}
```


Tworzenie obiektów *Date*

Efekt kodu z poprzednich stron

31.10.2023



Wczytaj

Thu Nov 02 2023 17:19:52 GMT+0100 (czas środkowoeuropejski standardowy)

Wed Mar 02 2022 00:00:00 GMT+0100 (czas środkowoeuropejski standardowy)

Tue Oct 31 2023 01:00:00 GMT+0100 (czas środkowoeuropejski standardowy)

Zadanie 1

Utwórz stronę, która po wprowadzeniu daty:

Formularze do wprowadzania dat i obiekt Date w obliczeniach

Podaj datę urodzenia

dd.mm.rrrr



Podaj

Osoba

Przeżyła dni.

Utwórz stronę, która po wprowadzeniu daty

Formularze do wprowadzania dat i obiekt Date w obliczeniach

Podaj datę urodzenia

01.09.2007






Podaj

Osoba *nieletnia.*

Przeżyła 5933 dni.

Gradient liniowy

```
body{  
    background-image:  
        linear-gradient(to bottom,  #0099ff,  white);  
    color:  navy;  
    font-family: 'Charm', sans-serif;  
    font-size: 20pt;  
    height: 100vh;  
}
```

Należy dobrać rozmiar. Wysokość 100vh oznacza 100% obszaru roboczego

Argumenty funkcji linear-gradient

Określają sposób przebiegu oraz użyte kolory i ich rozmiar

```
body{  
    background-image:  
        linear-gradient(45deg, ■ #0099ff, □ white, ■ lightblue);  
    color: ■ navy;  
    font-family: 'Charm', sans-serif;  
    font-size: 20pt;  
    height: 100vh;  
}
```

Zadanie 2

Utwórz stronę, w której po najechaniu myszką na ramkę z tekstem jak poniżej wyświetla się informacja o aktualnej dacie i liczbie dni do wakacji. Wystylizuj stronę za pomocą gradientów liniowych.

Dzisiaj jest , .

Do wakacji zostało dni.

Dzisiaj jest czwartek, 14 grudnia 2023.

Do wakacji zostało 189 dni.