

Operatory zmienne i typy danych

Kurs JS - moduł III - Kamil Cecherz



GIT i Github



git



W poprzednim module wprowadziliśmy do pracy system kontroli wersji GIT, nie będziemy się nim dokładnie zajmować ale konieczne będą podstawy do łatwiejszej nauki i późniejszej pracy nad większymi projektami.

W podstawowej wersji **GIT** działa z wiersza poleceń. Komendy do nauczania:

- **git clone <url>** - kopiuje istniejący projekt z podanej strony.
- **git checkout <branch>** - przełącza się na inną gałąź, miejsce, węzeł projektu.
- **git checkout HEAD^** - cofa się o jeden węzeł do tyłu w projekcie.
- **git add *** - zapisuje wszystkie zmiany w plikach
- **git commit -m "opis"** - tworzy nowy węzeł ze zmianami w projekcie.
- **git push** - przesyła projekt na github.

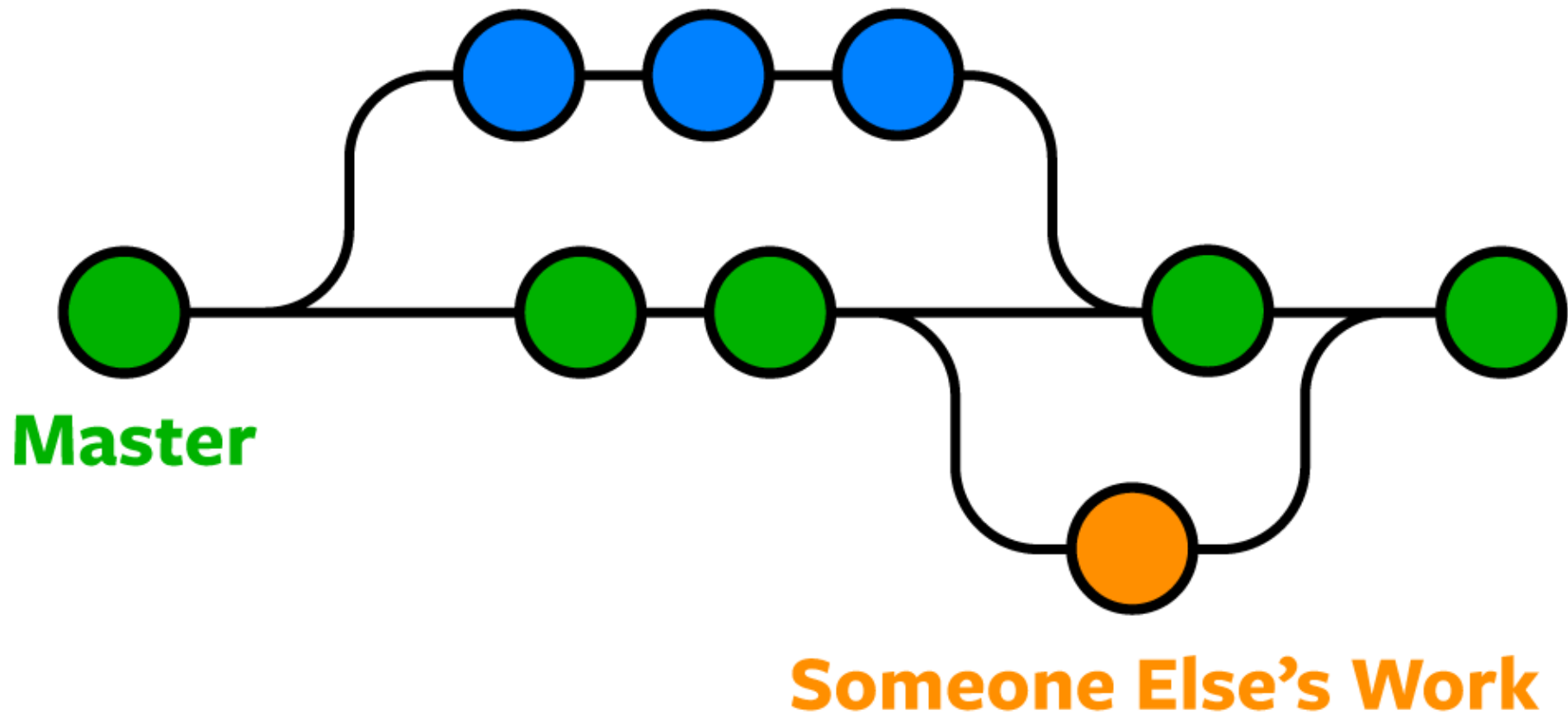
Wstępna konfiguracja

```
git config --global user.name "Anna Pietrzak"
```

```
git config --global user.email annapietrzak765@gmail.com
```

Niezbędna by można było przesyłać zmiany do zdalnego repozytorium. Podstawowym zadaniem systemu GIT-a jest praca w zespole, porównywanie wersji kodu do wspólnie tworzonego projektu. Właściciel repozytorium widzi twoje zmiany i może z nich skorzystać lub nie. Nigdy nic nie zostaje uszkodzone, gdyż można wrócić do poprzednich zapisanych wersji kodu. Przy projektach informatycznych taki system staje się niezbędnym, jego znajomość jest wymagana w każdej firmie.

Your Work



Czym jest Node.js ?

- Node.js to **program**, który potrafi uruchamiać kod JavaScript **poza przeglądarką**.
- Dzięki niemu możesz używać JavaScriptu **na swoim komputerze, serwerze, a nawet w chmurze** – nie tylko na stronie internetowej.
- **Przed Node.js** – JavaScript działał tylko w przeglądarce (np. sterując stroną WWW).
- **Po Node.js** – ten sam język może:
 - ❖ pisać programy uruchamiane w terminalu,
 - ❖ tworzyć serwery internetowe,
 - ❖ czytać i zapisywać pliki,
 - ❖ sterować robotami, dronami czy automatyką.



Instrukcja 1

1. Pobierz **Node.js** ze strony: <https://nodejs.org/en/download>
2. Zainstaluj na komputerze
3. Uruchom w terminalu polecenie **node** i przetestuj kilka instrukcji JavaScriptu wpisując do terminalu cokolwiek.
4. Wyjdź poleceniem **.exit**

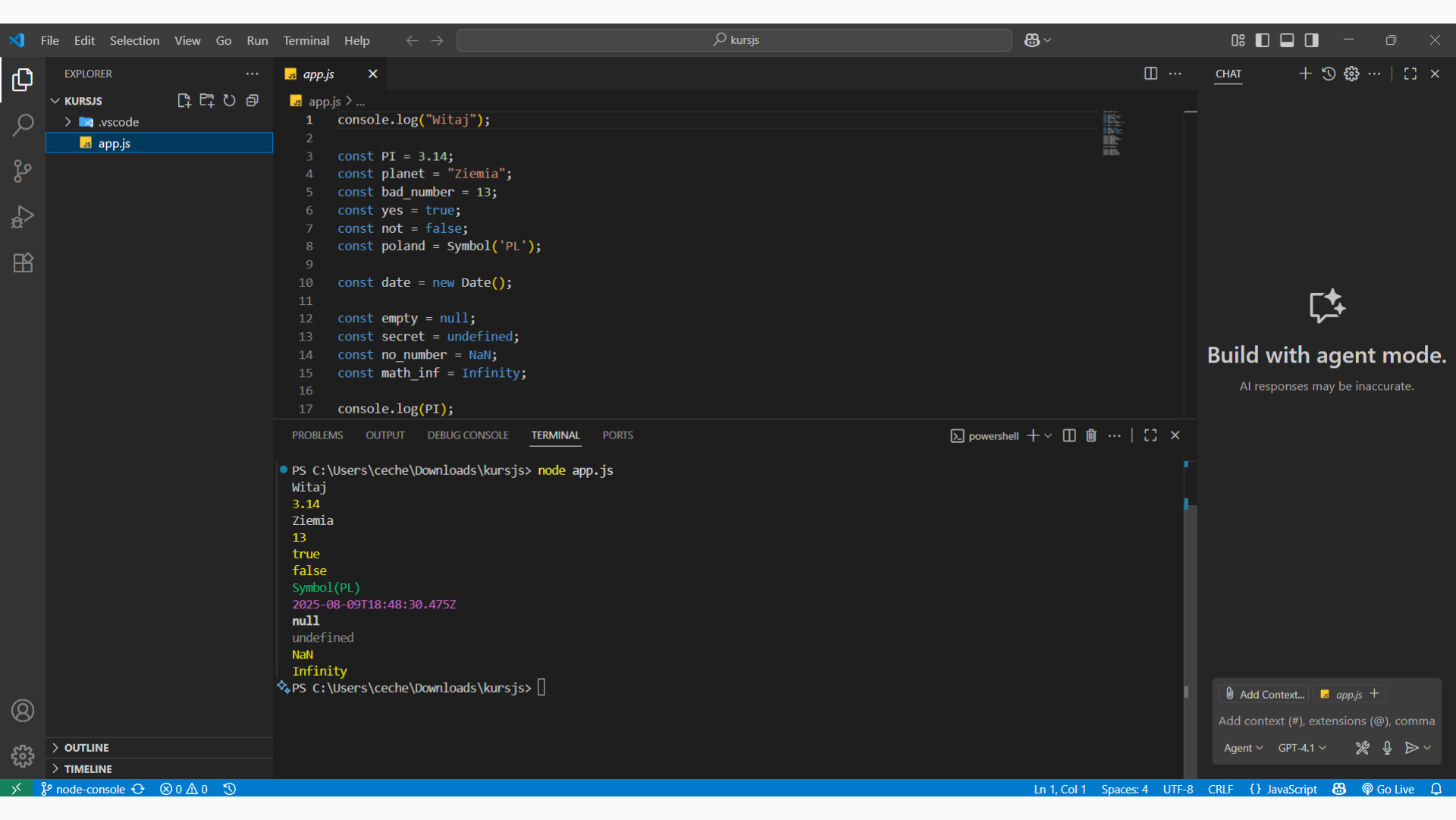
```
Windows PowerShell
PS C:\Users\ceche\Desktop> node
Welcome to Node.js v22.18.0.
Type ".help" for more information.
> console.log("Witaj Kamil");
Witaj Kamil
undefined
> 1*2*3*4*5
120
> "Maciek" + "Kamil"
'MaciekKamil'
> const planeta = "Ziemia";
undefined
> planeta
'Ziemia'
> const PI = 3.14
undefined
> planeta == PI
false
> planeta = PI
Uncaught TypeError: Assignment to constant variable.
> planeta == PI
false
> let tab = [1, 2, 3, 4, 5];
undefined
> tab[2];
3
> |
```

Node.js narzędzie do testów.

- Po uruchomieniu **VS Code** również możesz korzystać z polecenia `node` uruchamiając je we wbudowanym w środowisko terminalu.
- Możesz testować osobno każdy plik **JavaScript** z pominięciem przeglądarki WWW to ułatwia naukę i pracę. Oczywiście **Node.js** ma szerokie bardzo zaawansowane zastosowania ale to wykracza poza zakres kursu podstawowego.

Instrukcja 2

1. Sklonuj repozytorium: **`https://github.com/pangeon/kursjs`**.
2. Otwórz projekt w **VSCode**.
3. Wpisz w **Terminalu** w swoim środowisku: **`git checkout node-console`**.
4. Uruchom plik **`app.js`** poleceniem **`node app.js`**.



Zmienne, stałe i typy danych.

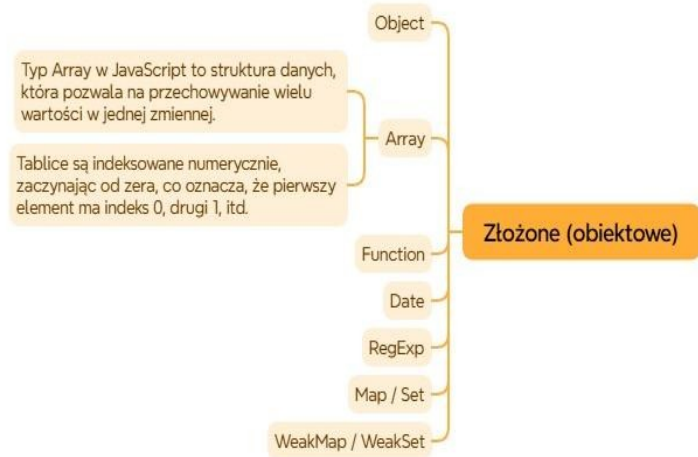
Zmienne i stałe w języku JavaScript

- słowo kluczowe **const** - służy do deklarowania stałych, wartości stałej nie można zmienić podczas działania programu.
- słowo kluczowe **let** oraz już przestarzałe **var** - służy do deklarowania zmiennych, których wartość i typ może się zmieniać podczas działania aplikacji.
- **deklaracja stałej oraz zmiennej** odbywa się poprzez użycie operatora przypisania **=**. Nie wolno mylić operatora przypisania z operatorem równości w matematyce, tej w językach programowania zwykle wygląda inaczej a dokładnie **===**. Co stwarza dodatkową trudność w języku JS mamy dwa operatory porównania: niezbyt intuicyjne luźne porównanie **==** oraz matematycznie ściśle **===**.

Typy zmiennych i stałych

- Gdy tworzymy nowe pole (zmienna lub stała) nigdy z góry nie deklarujemy jego typu. Ta operacja odbywa się “w locie”. Interpreter języka sam domyśla się typu pola. Typy istnieją choć nie musimy ich deklarować z góry.

```
let   zmienna   =   5;  
const stała = “napis”;
```



Typy danych

Proste (prymitywne)



1. Typy proste (Primitive Types)

JavaScript ma 7 typów prymitywnych:

Typ	Opis	Przykład	
string	Tekst w cudzysłowach (<code>'...'</code> , <code>"..."</code> , <code>`...`</code>).	<code>"Diuna"</code>	
number	Liczby całkowite i zmiennoprzecinkowe, także <code>Infinity</code> i <code>NaN</code> .	<code>42</code> , <code>3.14</code>	
bigint	Liczby całkowite o dowolnej długości (kończą się <code>n</code>).	<code>12345678901234567890n</code>	
boolean	Logiczne wartości <code>true</code> lub <code>false</code> .	<code>true</code>	
undefined	Wartość domyślna zmiennej niezainicjalizowanej.	<code>let x;</code> → <code>undefined</code>	
symbol	Unikalny i niezmienny identyfikator.	<code>Symbol('id')</code>	
null	Wartość „brak” lub „pusto” – ustawiona celowo.	<code>let pusta = null;</code>	

2. Typy złożone (Obiektowe)

Wszystko, co nie jest typem prostym, jest obiektem.

Najczęściej spotykane:

Typ	Opis	Przykład
Object	Ogólny typ obiektu.	<pre>{ imie: "Paul", planeta: "Arrakis" }</pre>
Array	Tablica wartości.	<pre>["Gurney", "Duncan", "Thufir"]</pre>
Function	Funkcja jako obiekt pierwszej klasy.	<pre>function walcz() {}</pre>
Date	Data i czas.	<pre>new Date()</pre>
RegExp	Wyrażenia regularne.	<pre>/^Harkonnen\$/</pre>
Map / Set	Kolekcje kluczy/wartości lub unikalnych wartości.	<pre>new Map(), new Set()</pre>
WeakMap / WeakSet	Jak powyżej, ale nie blokują czyszczenia pamięci.	<pre>new WeakMap()</pre>

Operator

Operatory w JavaScript



```
graph LR; A[Operatory w JavaScript] --- B[Operatory warunkowe i specjalne]; A --- C[Operatory bitowe]; A --- D[Operatory logiczne]; A --- E[Operator przypisania]; A --- F[Operatory arytmetyczne]; A --- G[Operatory porównania];
```

A mind map diagram with a central dark blue node labeled 'Operatory w JavaScript'. Six lines radiate from this central node to six peripheral nodes. The nodes on the left are 'Operatory warunkowe i specjalne' (purple), 'Operatory bitowe' (blue-purple), and 'Operatory logiczne' (teal). The nodes on the right are 'Operator przypisania' (orange-red), 'Operatory arytmetyczne' (orange), and 'Operatory porównania' (green). Each node is a rounded rectangle with its text in white.

Operatory warunkowe i specjalne

Operatory bitowe

Operatory logiczne

Operator przypisania

Operatory arytmetyczne

Operatory porównania

1. Operatory przypisania

Służą do nadawania wartości zmiennym.

Operator	Znaczenie	Przykład
=	Przypisanie wartości	<code>x = 5</code>
+=	Dodaj i przypisz	<code>x += 3</code> → <code>x = x + 3</code>
-=	Odejmij i przypisz	<code>x -= 2</code>
*=	Pomnóż i przypisz	<code>x *= 4</code>
/=	Podziel i przypisz	<code>x /= 2</code>
%=	Reszta z dzielenia i przypisz	<code>x %= 2</code>
**=	Potęguj i przypisz	<code>x **= 3</code>

2. Operatory arytmetyczne

Działają na liczbach (w tym `BigInt`).

Operator	Znaczenie	Przykład	Wynik	
<code>+</code>	Dodawanie	<code>5 + 3</code>	<code>8</code>	
<code>-</code>	Odejmowanie	<code>5 - 3</code>	<code>2</code>	
<code>*</code>	Mnożenie	<code>5 * 3</code>	<code>15</code>	
<code>/</code>	Dzielenie	<code>6 / 3</code>	<code>2</code>	
<code>%</code>	Reszta z dzielenia	<code>5 % 2</code>	<code>1</code>	
<code>**</code>	Potęgowanie	<code>2 ** 3</code>	<code>8</code>	
<code>++</code>	Inkrementacja	<code>x++</code> lub <code>++x</code>		
<code>--</code>	Dekrementacja	<code>x--</code> lub <code>--x</code>		

3. Operatory porównania

Służą do oceny relacji między wartościami.

Operator	Znaczenie	Przykład	Wynik
<code>==</code>	Równe (luźne)	<code>"5" == 5</code>	<code>true</code>
<code>===</code>	Równe (ściśle)	<code>"5" === 5</code>	<code>false</code>
<code>!=</code>	Różne (luźne)	<code>"5" != 5</code>	<code>false</code>
<code>!==</code>	Różne (ściśle)	<code>"5" !== 5</code>	<code>true</code>
<code>></code>	Większe niż	<code>7 > 5</code>	<code>true</code>
<code><</code>	Mniejsze niż	<code>3 < 5</code>	<code>true</code>
<code>>=</code>	Większe lub równe	<code>5 >= 5</code>	<code>true</code>
<code><=</code>	Mniejsze lub równe	<code>3 <= 5</code>	<code>true</code>

4. Operatory logiczne

Działają na wartościach logicznych (`true` / `false`).

Operator	Znaczenie	Przykład	Wynik
<code>&&</code>	AND – logiczne „i”	<code>true && false</code>	<code>false</code>
<code> </code>	OR – logiczne „lub”	<code>true false</code>	<code>true</code>
<code>!</code>	NOT – negacja	<code>!true</code>	<code>false</code>
<code>??</code>	Nullish coalescing	<code>null ?? "domyślnie"</code>	<code>"domyślnie"</code>

5. Operatory bitowe

Działają na reprezentacji binarnej liczb całkowitych.

Operator	Znaczenie	Przykład	Wynik	
&	AND bitowy	5 & 1	1	
	OR bitowy	5 1	5	
^	XOR bitowy	5 ^ 1	4	
~	NOT bitowy	~5	-6	
<<	Przesunięcie w lewo	5 << 1	10	
>>	Przesunięcie w prawo z zachowaniem znaku	5 >> 1	2	
>>>	Przesunięcie w prawo bez znaku	5 >>> 1	2	

6. Operatory warunkowe i specjalne

Operator	Znaczenie	Przykład
<code>?:</code>	Operator trójargumentowy (ternary)	<code>wiek >= 18 ? "dorosły" : "nieletni"</code>
<code>typeof</code>	Zwraca typ zmiennej	<code>typeof "Arrakis" → "string"</code>
<code>instanceof</code>	Sprawdza, czy obiekt jest instancją danej klasy	<code>x instanceof Array</code>
<code>delete</code>	Usuwa właściwość z obiektu	<code>delete obiekt.klucz</code>
<code>in</code>	Sprawdza, czy klucz istnieje w obiekcie	<code>"klucz" in obiekt</code>
<code>void</code>	Zwraca <code>undefined</code>	<code>void 0</code>

Testowanie

- Operatorów jest bardzo dużo, warto znać ich typy i zastosowania nie trzeba jednak wszystkiego pamiętać od razu, by sprawdzić co potrafi język i przetestować wszystkie jego funkcje. Najlepiej uruchomić **nodejs** w **Terminalu**.
- Operatory działają inaczej w różnych kontekstach np. **+** służy do sumowania wartości typu **number** (liczby), a w przypadku ciągów znaków, napisów **string** jako operator **konkatenacji**. Skleja dwa napisy w jeden.

Ćwiczenie 1

1. Uruchom plik **data_test.js** poleceniem **node data test.js**.
2. Przypisz do zmiennej dowolną wartość. Co się zmieniło?
3. Sprawdź jak działa skrypt po przypisaniu do zmiennej różnych wartości, przetestuj:
 - a. string
 - b. number
 - c. boolean
 - d. bigint
 - e. null

Ćwiczenie 2

1. W pliku **data_test.js** zadeklaruj dwie dowolne zmienne i przypisz im wartości.
2. Otwórz jedną stałą która połączy wartości z wcześniej zadeklarowanych zmiennych operatorem **+**.
3. Wyświetl wartość zmiennej w terminalu.
4. Użyj operatora **typeof** by sprawdzić typ stałej.

Ćwiczenie 3

1. W pliku **calc.js** dwie zmienne o nazwie **x** oraz **y** w których będziesz przechowywać wartości liczbowe. Przypisz im dowolne wartości.
2. Wyświetl w konsoli sumę tych liczb przy pomocy **console.log()**.
3. Wyświetl w konsoli różnicę tych liczb przy pomocy **console.log()**.
4. Wyświetl w konsoli iloczyn tych liczb przy pomocy **console.log()**.
5. Wyświetl w konsoli iloraz tych liczb przy pomocy **console.log()**.
6. Wyświetl w konsoli dzielenie z resztą tych liczb przy pomocy **console.log()**.
7. W ćwiczeniu oczekuje się 5 instrukcji **console.log()**.

Instrukcja 3

- Mając otwarte **VS Code** wydaj następujące **GIT-a** w **Terminalu**:
 - `git add *`
 - `git commit -m "Ania ćwiczenia"`
 - `git push`
- Jeśli nie pojawiły się żadne błędy znaczy, że udało Ci się zapisać i wysłać zmiany w projekcie do zdalnego repozytorium. Efekty twojej pracy będzie mógł zobaczyć twój trener i ty. Sprawdź adres [www](https://github.com/pangeon/kursjs/tree/node-console):
 - <https://github.com/pangeon/kursjs/tree/node-console>
- Brawo, to co potrafisz pozwoli Ci pracować w zespołach programistycznych. To bardzo cenna umiejętność._

KONIEC