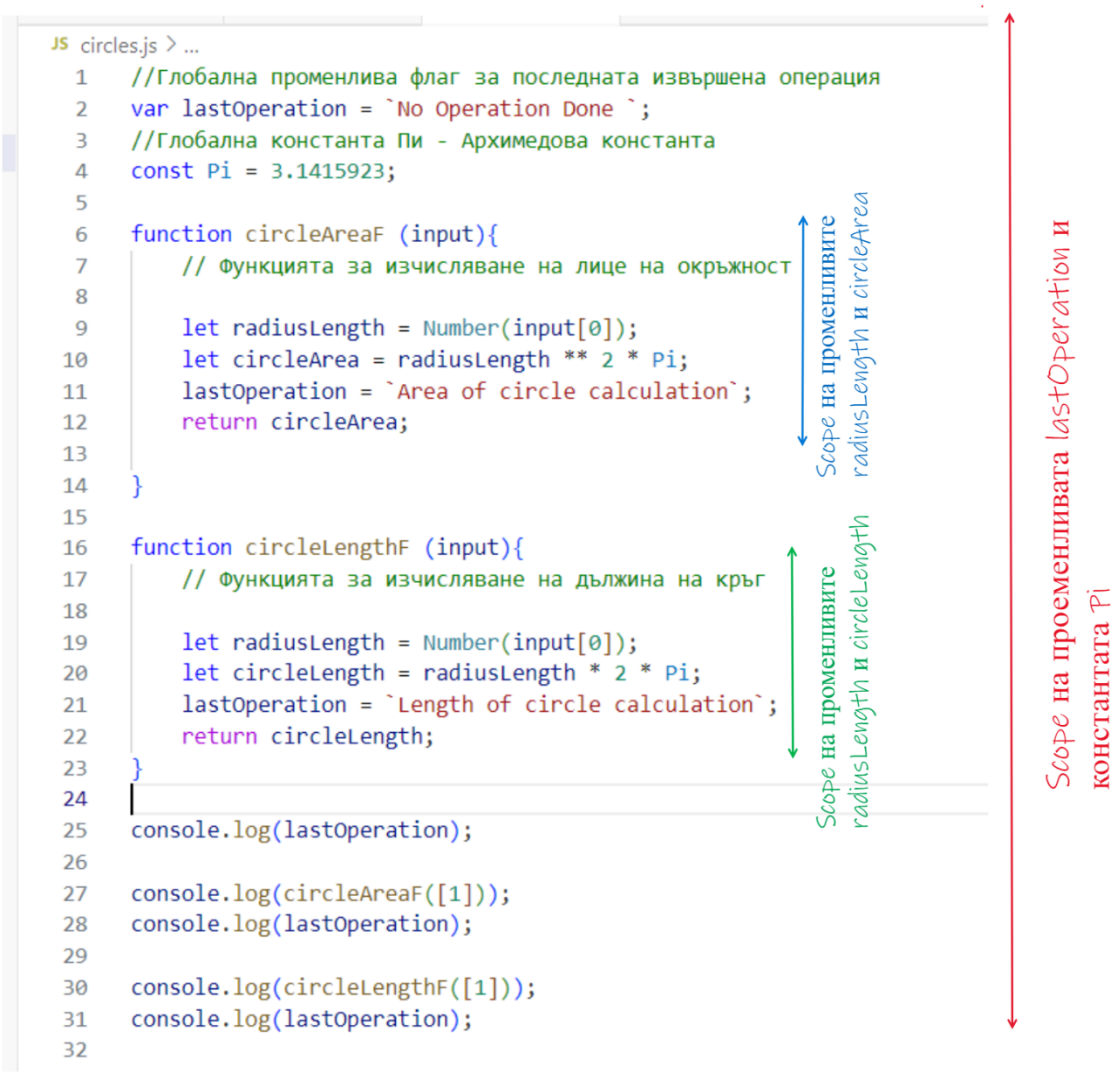
Var, let, const и scope на променливи в Java Script



Тъй като в групата се обсъждат теми относно някой подробности в Java Script и по-точно в сегашната му имплементация, след въвеждането на деклариране на променливи със запазената дума let, може би е добра идея да си изясним експериментално какъв е по-точно обхвата на променливите в кода и как точно работят запазените думи *var, let* и *const*.

В курса на това ниво се използват основно два подхода за интерпретиране на Java Script код – в конзолата на браузър и от Node JS, като за целта на втория подход разписваме кода във Visual Studio Code.

Първо трябва да кажем, че Node се различава принципно от браузърите – докато при тях Java Script се използва за създаване на функционалност във front-end частта на web приложение, то Node е нещо като рамка за back-end частта на такива приложения. От друга страна, различните браузъри използват различни JS енджини, а този енджин, който се използва от Google Chrome се използва и от Node JS.

Трябва да кажем също така, че Java Script е очевидно нетипизиран програмен език, което означава, че при дефинирането на променлива не казвате задължително на интерпретатора какъв тип данни ще съдържа тя и можете доста свободно да я използвате в програмния код след това. В типизираните езици ще трябва да подходите по обратния начин.

По този начин, ако например в **C#** трябва да дефинирате във функция три променливи, първата от които четете от системната конзола, а другите две установявате с първоначална стойност 0, то ще трябва да не само да укажете на компилатора техния тип, но и ако той е числов (в примера по-долу първите две са целочислени, а третата – реално число с плаваща запетия), евентуално ще трябва да парснете текстовия стринг, който получавате от конзолата към целочислен тип:

```
int variableOne = int.Parse(Console.ReadLine());
int variableTwo = 0;
double variableThree = 0;
```

За разлика от това, при **JS** ще трябва просто да обявите променливите, задоволявайки се с доста по-общи указания. В JS не можете да се обърнете директно към конзолата, ще прехвърляте данните през аргумента на функция, нещо от вида

```
function calcOne (input){ … }
```

и съответно ще си извадите тази стойност като първи елемент на масива input:

```
let variableOne = input[0];
let variableTwo = 0;
let variableThree = 0;
```

Не се налага да парсвате текстовия стринг от входа на конзолата към число и така написан кодът ще работи, но нищо не пречи в името на добрия стил да обясните това както на интерпретатора, така и (в много по-голяма степен) на тези, които ще разчитат този код след вас един ден:

```
let variableOne = Number(input[0]);
let variableTwo = 0;
let variableThree = 0;
```

Беше време, в което единственият начин да дефинирате променлива в Java Script беше да използвате ключовата дума *var*. Декларирана по този начин една променлива има за скоуп (или ако предпочитате обхват) на практика целия ви JS документ.

Новият начин за обявяване на променливи със запазената дума *let* ги прави достъпни само в тази част от кода ви, където са обявени. Така например ако в кода по-горе променливата *circleLength* е декларирана във функцията *circlelengthF()*, то тя може да бъде достъпвана единствено вътре в тази функция.

Забележете, че и в двете функции - *circlelengthF()* и *circleAreaF()* са обявени променливи с име *radiusLength*. Тъй като тези променливи са обявени с *let* и вътре във функциите, независимо че са с едно и също име са две съвсем различни променливи, които реферират съответно две различни места в паметта. И двете не могат да бъдат извиквани от частта на документа извън функциите и като ги използваме във всяка от функциите се обръщаме единствено към променливата, дефинирана в съответната функция.

Глобалната променлива *lastOperation*, от друга страна, е достъпна с текущата си стойност от всяко място в кода. В конкретния случай тя е едни вид флаг, в който записваме последната извършена операция. Това е една единствена променлива, която можем да променяме и от двете функции и от всяко място извън тях. В интерес на истината нуждата от нея в случая е доста съмнителна, но за съжаление само такъв пример ми дойде на ум.

Глобалната константа *Pi*, в която в началото на кода сме записали стойността (до седмия знак, защото само до та я помня) на *Архимедовата константа*, за разлика от всички променливи в кода, няма да бъде променяна. В кода сме задали само една единствена стойност на променливата *circleArea*, по същия начин както сме задали една единствена стойност на константата Pi. Каква е тогава разликата между двете? Променливата *circleArea* съдържа текущата изчислена стойност на лицето на кръга. Тя би могла да бъде променяна, ако променим стойността на променливата *radiusLength* и изпълним функцията *circleAreaF(),* то и стойността на променливата *circleArea* ще бъде различна. От друга страна стойността на *Архимедовата константа* винаги ще си остане тази, независимо от конкретната постановка. Тя не се променя не защото програмистът е решил да не я променя, тя не се променя поради своята същност – π винаги ще бъде 3.14…, това е обективно така и не зависи от нашите решения.

При извикването на двете функции за стойност на радиуса на окръжността е избрано 1 за да бъдат по-лесни и очевидни изчисленията на лицето на окръжността и дължината на кръга.

```
JS circles.js > circleAreaF
1   //Глобална променлива флаг за последната извършена операция
2   var lastOperation = `No Operation Done `;
3   //Глобална константа Пи - Архимедова константа
4   const Pi = 3.1415923;
5
6   function circleAreaF (input){
7       // Функцията за изчисляване на лице на окръжност
8       let radiusLength = Number(input[0]);
9       let circleArea = radiusLength ** 2 * Pi;
10      lastOperation = `Area of circle calculation`;
11      return circleArea;
12
13  }
14
15  function circleLengthF (input){
16      // Функцията за изчисляване на дължина на кръг
17      let radiusLength = Number(input[0]);
18      let circleLength = radiusLength * 2 * Pi;
19      lastOperation = `Length of circle calculation`;
20      return circleLength;
21  }
22  console.log(lastOperation);
23  console.log(circleAreaF([1]));
24  console.log(lastOperation);
25  console.log(circleLengthF([1]));
26  console.log(lastOperation);
27

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

C:\Program Files\nodejs\node.exe .\circles.js
No Operation Done
3.1415923
Area of circle calculation
6.2831846
Length of circle calculation
```