

Hands – On Lab

Workshop 3.

AREA OF TRIANGLE

Write a function that takes the base and height of a triangle and **return** its area.

Example:

Areaoftriangle (3, 4) \longrightarrow 6

Areaoftriangle (7, 8) \longrightarrow 28

Notes

- Area of triangle is $(\text{base} * \text{height})/2$
- Don't forget to return the result



The screenshot shows a code editor with a file named 'index.js'. The code defines a function 'areaTriangle' that prompts the user for the base and height, calculates the area, and logs it to the console. The function is then called. The console output shows the prompts and the calculated area of 72.

```
1 // area
2 let areaTriangle = () => {
3   let b = parseInt(prompt("Enter the base"))
4   let h = parseInt(prompt("Enter the height"))
5   console.log((b*h)/2)
6 }
7 areaTriangle()
```

Console output:

```
Enter the base> 12
Enter the height> 12
72
Hint: hit control+c anytime to enter REPL.
```

RETURN SOMETHING TO ME!

Write a function that returns the string "something" joined with a space " " and the given argument.

Examples

giveMeSomething("is better than nothing") \rightarrow "something is better than nothing"

giveMeSomething("Bob Jane") \rightarrow "something Bob Jane"

giveMeSomething("something") \rightarrow "something something"

The screenshot shows a code editor with a file named `index.js`. The code defines a function `takeInput` that prompts the user for words, logs the concatenated result, and calls itself. The console shows the output: `Enter some words to concatenate> prashan` and `Hello prashan`.

```
1 let takeInput = () => {
2   let somewords = prompt("Enter some words to concatenate")
3   console.log(soncatinat(somewords))
4 }
5 let soncatinat = (somewords) => {
6   return `Hello ${somewords}`
7 }
8 takeInput()
```

Console output:
Enter some words to concatenate> prashan
Hello prashan
Hint: hit control+c anytime to enter REPL.

BASKETBALL POINTS

You are counting points for a basketball game, given the amount of 2 – pointer scored and 3 – pointer scored, find the final points for the team and return the value.

Example:

points (3,5) $\longrightarrow 3*2 + 5*3 = 21$

points (1,1) $\longrightarrow 5$

The screenshot shows a code editor with a file named `index.js`. The code defines a function `takeInput` that prompts the user for the number of 2-point and 3-point shots, calculates the total score, and logs the result. The console shows the output: `Enter number of 2 scored> 12`, `Enter number of 3 scored> 23`, and `93`.

```
1 let takeInput = () => {
2   let numberOfTwoScored = parseInt(prompt("Enter number of 2 scored"))
3   let numberOfThreeScored = parseInt(prompt("Enter number of 3 scored"))
4   console.log(totalScore(numberOfTwoScored, numberOfThreeScored))
5 }
6 let totalScore = (a, b) => {
7   return `The total score is `, 2*a + 3*b
8 }
9 takeInput()
```

Console output:
Enter number of 2 scored> 12
Enter number of 3 scored> 23
93
Hint: hit control+c anytime to enter REPL.

LESS THAN 100?

Given two numbers, return true if the sum of both numbers is less than 100.

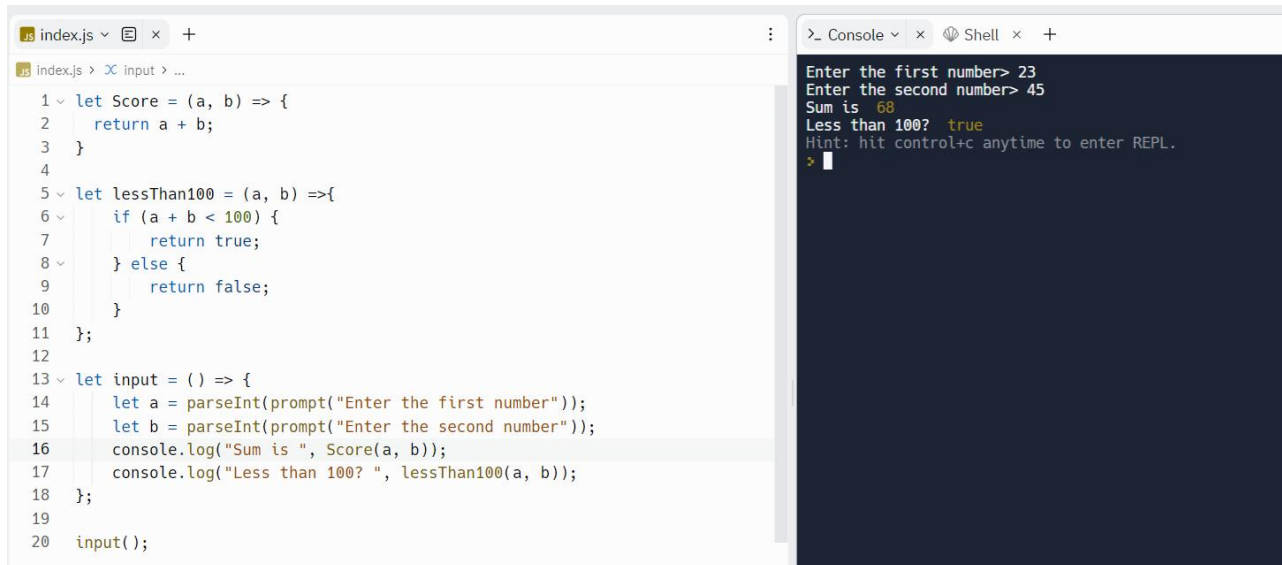
Otherwise return false.

Examples

`lessThan100(22, 15) → true`

`// 22 + 15 = 37`

`lessThan100(83, 34) → false`



The image shows a code editor window with a file named `index.js` and a terminal window. The code in `index.js` defines a function `Score` that returns the sum of two numbers, a function `lessThan100` that checks if the sum is less than 100, and an `input` function that prompts the user for two numbers and logs the results. The terminal shows the execution of the `input` function with inputs 23 and 45, resulting in a sum of 68 and a `lessThan100` result of `true`.

```
1 let Score = (a, b) => {
2   return a + b;
3 }
4
5 let lessThan100 = (a, b) =>{
6   if (a + b < 100) {
7     return true;
8   } else {
9     return false;
10  }
11 };
12
13 let input = () => {
14   let a = parseInt(prompt("Enter the first number"));
15   let b = parseInt(prompt("Enter the second number"));
16   console.log("Sum is ", Score(a, b));
17   console.log("Less than 100? ", lessThan100(a, b));
18 };
19
20 input();
```

Terminal output:

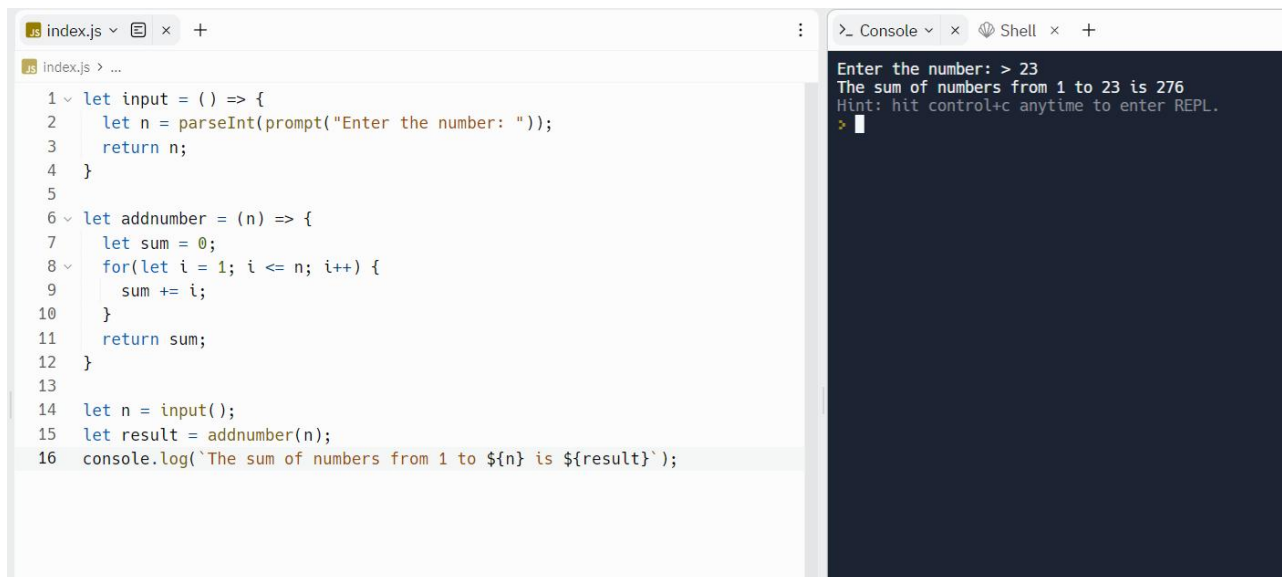
```
>_ Console x Shell x +
Enter the first number> 23
Enter the second number> 45
Sum is 68
Less than 100? true
Hint: hit control+c anytime to enter REPL.
>
```

```
// 83 + 34 = 117
```

```
lessThan100(3, 77) → true
```

ADD UPTO THE NUMBER FROM A SINGLE NUMBER

Create a function that takes a number as an argument. Add up all the numbers from 1 to the number you passed to the function. For example, if the input is 4 then your function should return 10 because $1+2+3+4 = 10$



```
index.js > ...
1 let input = () => {
2   let n = parseInt(prompt("Enter the number: "));
3   return n;
4 }
5
6 let addnumber = (n) => {
7   let sum = 0;
8   for(let i = 1; i <= n; i++) {
9     sum += i;
10  }
11  return sum;
12 }
13
14 let n = input();
15 let result = addnumber(n);
16 console.log(`The sum of numbers from 1 to ${n} is ${result}`);
```

```
>_ Console x Shell x +
Enter the number: > 23
The sum of numbers from 1 to 23 is 276
Hint: hit control+c anytime to enter REPL.
>
```

ANY PRIME NUMBER IN RANGE

Create a function that return true if there is at least one prime number in the given range(n_1 to n_2) inclusive, false otherwise.


Example:

```
primeInRange(10,15) → true
```

```
// prime number is range : 11, 13
```

```
primeInRange(3,1) → true
```

```
// prime number is range : 3, 5
```



The screenshot shows a code editor with a JavaScript file named `index.js`. The code defines a function `isPrime` that checks if a number is prime. It uses a loop to test divisibility from 2 to $n-1$. If a divisor is found, it returns `false`; otherwise, it returns `true`. Below the function, there is a call to `primeInRange()` which logs the result. The console on the right shows the execution: the first number entered is 34, the second is 33, and the output is `False`. A hint suggests using `control+c` to enter REPL.

```
3 let n2 = parseInt(prompt("Enter second number:"))
4
5 for (let i = n1; i <= n2; i++) {
6   let isPrime = true;
7   if (i < 2) {
8     isPrime = false;
9   } else {
10    for (let j = 2; j < i; j++) {
11      if (i % j == 0) {
12        isPrime = false;
13        break;
14      }
15    }
16  }
17  if (isPrime) {
18    return true;
19  }
20 }
21 return false;
22 }
23
24 if (primeInRange()) {
25   console.log("True");
26 } else {
27   console.log("False");
28 }
```

Console output:
Enter first number:> 34
Enter second number:> 33
False
Hint: hit control+c anytime to enter REPL.

ODDISH VS. EVENISH

Create a function that determines whether a number is Oddish or Evenish. A number is Oddish if the sum of all of its digits is odd, and a number is Evenish if the sum of all of its digits is even. If a number is Oddish, return "Oddish". Otherwise, return "Evenish".

For example, `oddishOrEvenish(121)` should return "Evenish", since $1 + 2 + 1 =$

4. `oddishOrEvenish(41)` should return "Oddish", since $4 + 1 = 5$.

Examples

`oddishOrEvenish(43) → "Oddish"`

`// 4 + 3 = 7`

`// 7 % 2 = 1`

`oddishOrEvenish(373) → "Oddish"`

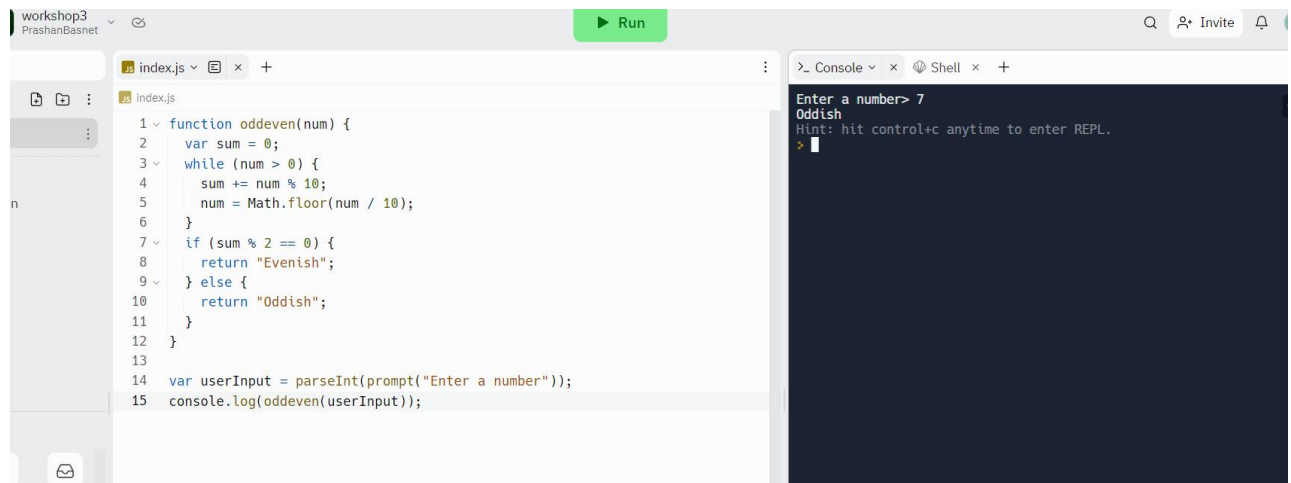
`// 3 + 7 + 3 = 13`

`// 13 % 2 = 1`

`oddishOrEvenish(4433) → "Evenish"`

`// 4 + 4 + 3 + 3 = 14`

`// 14 % 2 = 0`



LEFT SHIFT BY POWERS OF TWO

The left shift operation is similar to multiplication by powers of two.

Sample calculation using the left shift operator (\ll):

$$10 \ll 3 = 10 * 2^3 = 10 * 8 = 80$$

$$-32 \ll 2 = -32 * 2^2 = -32 * 4 = -128$$

$$5 \ll 2 = 5 * 2^2 = 5 * 4 = 20$$

Write a function that mimics (without the use of \ll) the left shift operator and returns the result from the two given integers.

Examples

`shiftToLeft(5, 2) → 20`

`shiftToLeft(10, 3) → 80`

`shiftToLeft(-32, 2) → -128`

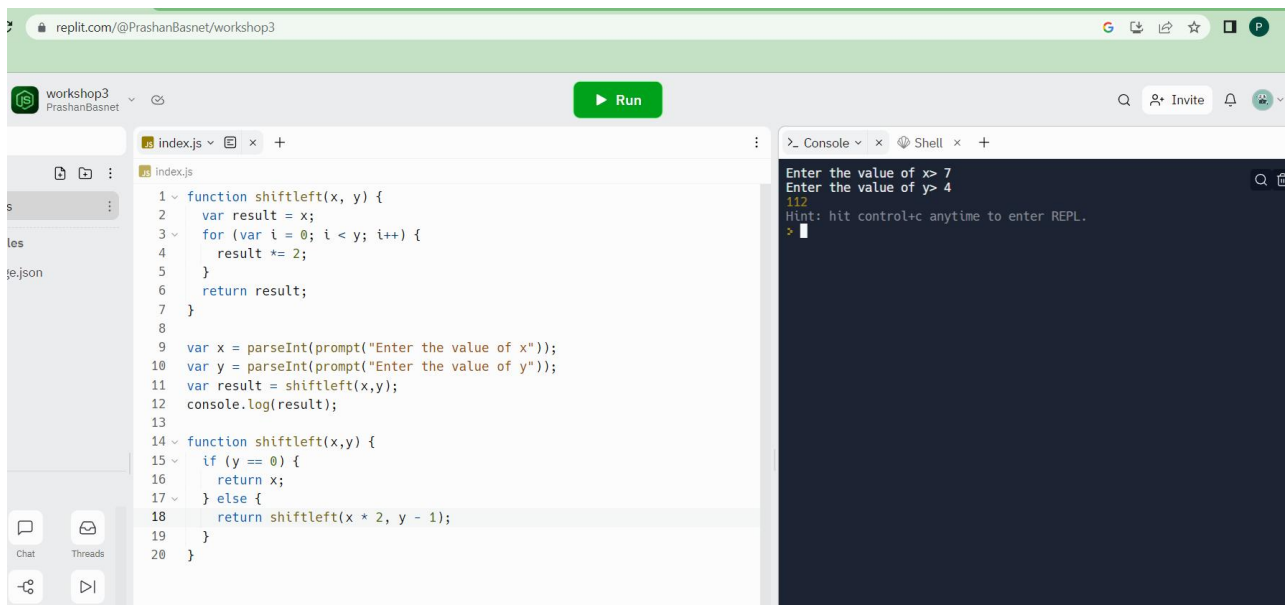
`shiftToLeft(-6, 5) → -192`

`shiftToLeft(12, 4) → 192`

`shiftToLeft(46, 6) → 2944`

Notes

- There will be no negative values for the second parameter y .
- This challenge is more like recreating the left shift operation, thus, the use of the operator directly is prohibited.
- Alternatively, you can solve this challenge via recursion.



The screenshot shows a Replit workspace with a file named `index.js` and a terminal window. The code in `index.js` defines a `shiftLeft(x, y)` function that uses a loop to multiply x by 2 for y times. It also includes a recursive version of the function. The terminal shows the user entering `x = 7` and `y = 4`, resulting in the output `112`.

```
1 function shiftLeft(x, y) {
2   var result = x;
3   for (var i = 0; i < y; i++) {
4     result *= 2;
5   }
6   return result;
7 }
8
9 var x = parseInt(prompt("Enter the value of x"));
10 var y = parseInt(prompt("Enter the value of y"));
11 var result = shiftLeft(x, y);
12 console.log(result);
13
14 function shiftLeft(x, y) {
15   if (y == 0) {
16     return x;
17   } else {
18     return shiftLeft(x * 2, y - 1);
19   }
20 }
```

Console output:

```
Enter the value of x > 7
Enter the value of y > 4
112
Hint: hit control+c anytime to enter REPL.
```

CONVERT A NUMBER TO BASE-2

Create a function that returns a base-2 (binary) representation of a base-10 (decimal) string number. To convert is simple: ((2) means base-2 and (10) means base-10)

$$010101001(2) = 1 + 8 + 32 + 128.$$

Going from right to left, the value of the most right bit is 1, now from that every bit to the left will be $\times 2$. The values of an 8 bit binary number are (256, 128, 64, 32, 16, 8, 4, 2, 1).

Examples

`binary(1) → "1"`


```
// 1*1 = 1 binary(5)
```

```
→ "101"
```

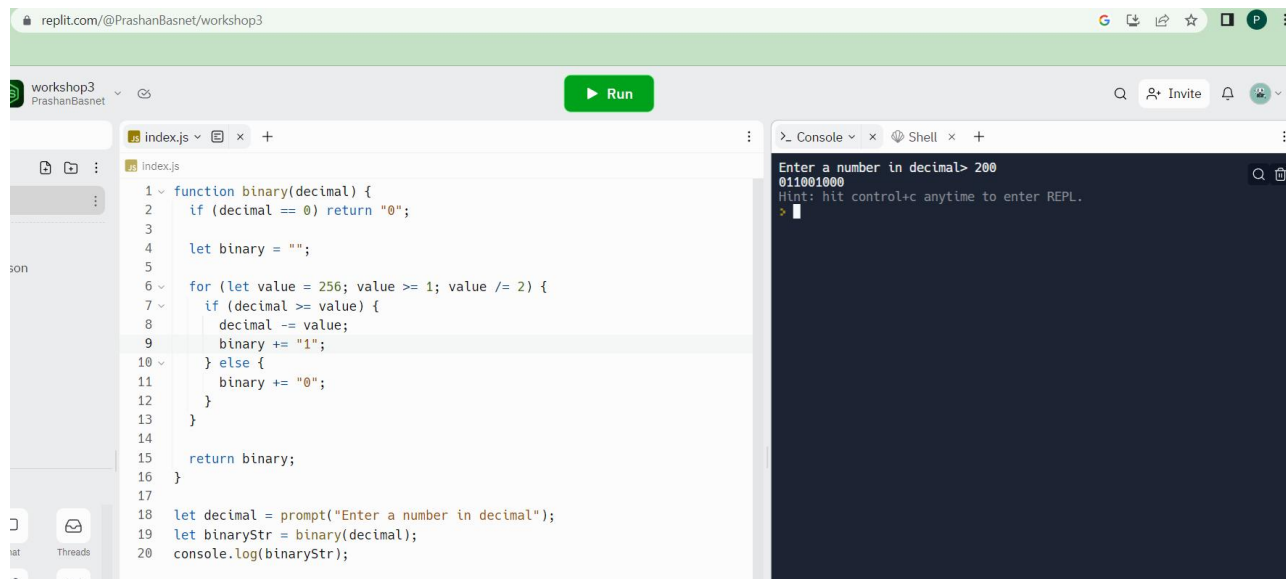
```
// 1*1 + 1*4 = 5
```

```
binary(10) → "1010"
```

```
// 1*2 + 1*8 = 10
```

Notes

- Numbers will always be below 1024 (not including 1024).
- The && operator could be useful.
- The strings will always go to the length at which the most left bit's value gets bigger than the number in decimal.
- If a binary conversion for 0 is attempted, return "0".



The screenshot shows a Replit workspace with a file named `index.js` and a console window. The code in `index.js` defines a `binary` function that converts a decimal number to a binary string. It uses a loop to divide the decimal by 2 and build the binary string from the remainders. The console shows the prompt "Enter a number in decimal" and the input "200", resulting in the output "011001000".

```
1 function binary(decimal) {  
2   if (decimal == 0) return "0";  
3  
4   let binary = "";  
5  
6   for (let value = 256; value >= 1; value /= 2) {  
7     if (decimal >= value) {  
8       decimal -= value;  
9       binary += "1";  
10    } else {  
11      binary += "0";  
12    }  
13  }  
14  
15  return binary;  
16 }  
17  
18 let decimal = prompt("Enter a number in decimal");  
19 let binaryStr = binary(decimal);  
20 console.log(binaryStr);
```

Console output:
Enter a number in decimal> 200
011001000
Hint: hit control+c anytime to enter REPL.

GUESSING GAME

Generate a random number (do research) and store it in a variable. Write a program to take input from the user and tell them whether their guessed number is correct, greater or lesser than the original number. (100 – number of guesses) is the score of user. The program is expected to terminate once the number is guessed. Number should be between 1 – 100.

Example:

Random number generated by computer: 54

User input: 34

// lesser than original number

User input: 67

// greater than original number

User input: 54

// congratulations!!! The number you guessed matched the original number. Your score is 97!

HIGHER ORDER ARRAY METHODS

Const age = [23,34,12,54,23,54,11,9,29,17,15,19,20,21,13,7]

- a. Filter the array of age who can apply for citizenships
- b. Find the average age of a given array

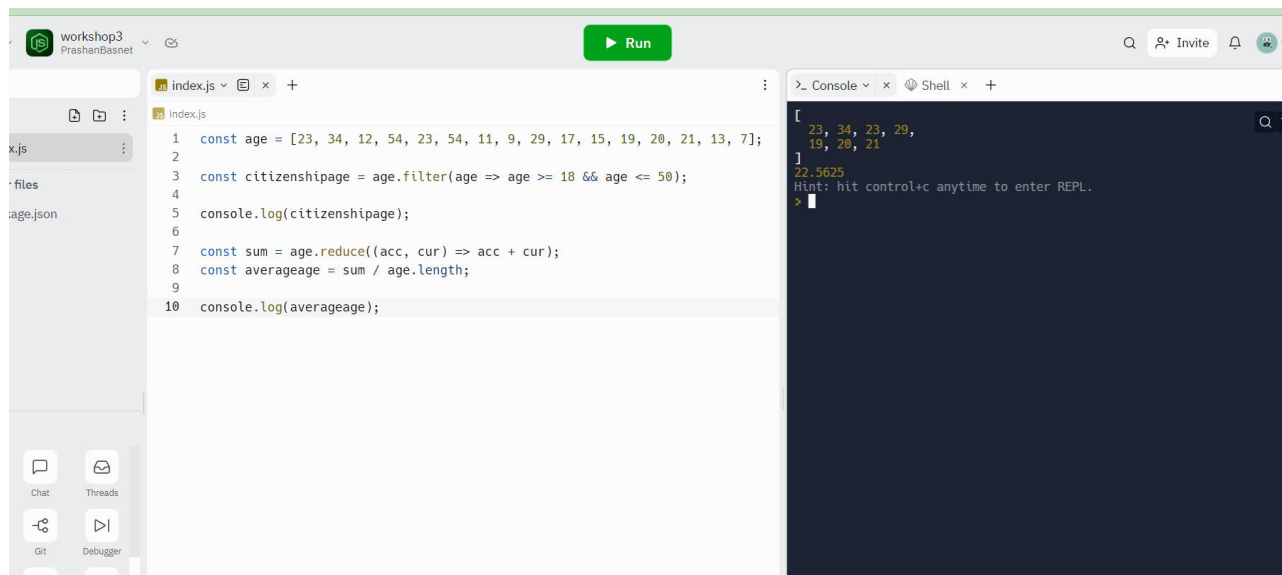
Const companies = [

{ name: "ABC", category: "Finance", start: 1981, end: 2004 },

{ name: "XYZ", category: "Retail", start: 1991, end: 20012 },

```
{ name: "DGF", category: "Finance", start: 1976, end: 2008 },  
{ name: "LFT", category: "Retail", start: 1971, end: 1979 },  
{ name: "MND", category: "Retail", start: 1995, end: 2010 },  
{ name: "HCK", category: "Technology", start: 1987, end: 2011 },  
{ name: "BMC", category: "Technology", start: 1989, end: 2009 },  
{ name: "TIC", category: "Retail", start: 1993, end: 2005 },  
{ name: "NAC", category: "Technology", start: 1991, end: 2010 },  
{ name: "ITC", category: "Finance", start: 1998, end: 2016 }  
];
```

- Filter the retail companies
- Get the 80s companies from the array
- Get the companies that lasted for 10 or more years



The screenshot shows a VS Code editor window with a file named `index.js`. The code in the file is as follows:

```
1 const age = [23, 34, 12, 54, 23, 54, 11, 9, 29, 17, 15, 19, 20, 21, 13, 7];  
2  
3 const citizenshipage = age.filter(age => age >= 18 && age <= 50);  
4  
5 console.log(citizenshipage);  
6  
7 const sum = age.reduce((acc, cur) => acc + cur);  
8 const averageage = sum / age.length;  
9  
10 console.log(averageage);
```

The terminal window on the right shows the output of the code:

```
[  
  23, 34, 23, 29,  
  19, 20, 21  
,  
]  
22.5625  
Hint: hit control+c anytime to enter REPL.  
>
```

index.js

index.js

```
1 const companies= [  
2   { name: "ABC", category:" Finance", start:1981, end: 2004},  
3   { name: "XYZ", category:" Retail", start:1991, end: 2012},  
4   { name: "DGF", category:" Finance", start:1976, end: 2008},  
5   { name: "LFT", category:" Retail", start:1971, end: 1979},  
6   { name: "MND", category:" Retail", start:1995, end: 2010},  
7   { name: "MCK", category:" Technology", start:1987, end: 2011},  
8   { name: "BMC", category:" Technology", start:1989, end: 2009},  
9   { name: "TIC", category:" Retail", start:1993, end: 2005},  
10  { name: "NAC", category:" Technology", start:1991, end: 2010},  
11  { name: "ITC", category:" Finance", start:1998, end: 2016},  
12 ];  
13  
14 const retailCompanies = companies.filter(company => company.category ===  
   "Retail");  
15  
16 const eightlesscompanies = companies.filter(company => company.start  
   >=1900 && company.start < 1990);  
17  
18 const tenYearcompanies = companies.filter(company => (company.end -  
   company.start) >= 10);  
19  
20 console.log(retailCompanies);  
21 console.log(eightlesscompanies);  
22 console.log(tenYearcompanies);
```



Chat



Threads



Git



Debugger



Console



Secrets

RAM

Storage

twriter