

Area of a Triangle

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

Examples

`triArea(3, 2)` → 3

`triArea(7, 4)` → 14

`triArea(10, 10)` → 50

Notes

- The area of a triangle is: $(\text{base} * \text{height}) / 2$
- Don't forget to `return` the result.

```
<script>
  const area = (base,height)=>{
    return (base+ height)/2;
  }
  console.log("Area of Triangle is",area(10,520))
  console.log("Area of Triangle is",area(20,30))
  console.log("Area of Triangle is",area(33,44))
</script>
```

```
Area of Triangle is 265
Area of Triangle is 25
Area of Triangle is 38.5
Live reload enabled.
```

Return Something to Me!

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

Examples

```
giveMeSomething("is better than nothing") → "something is better than  
nothing" giveMeSomething("Bob Jane") → "something Bob Jane"  
giveMeSomething("something") → "something something"
```

```
<script>  
  function something(str){  
    return "Something" + str;  
  }  
  console.log (something("is better than nothing."))  
  console.log (something("alan shah"))  
  console.log (something("is wrong."))  
</script>
```

Somethingis better than nothing.	index4.html:13
Somethingalan shah	index4.html:14
Somethingis wrong.	index4.html:15
Live reload enabled.	index4.html:45

Basketball Points

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

Examples

points(1, 1) → 5

points(7, 5) → 29

points(38, 8) → 100

```
<script> function basketball (twopointer,three_pointer){  
  | return two_pointer*2+three_pointer*3  
  }  
  console.log("The total score is",basketball(20,40))  
  console.log("The total score is",basketball(10,8))  
  console.log("The total score is",basketball(19,3))  
</script>
```

The total score is 160 [index4.html:13](#)

The total score is 44 [index4.html:14](#)

The total score is 74 [index4.html:15](#)

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`

`// 83 + 34 = 117`

`lessThan100(3, 77) → true`

```
<script>
function sum(a,b){
  if (a+b<100 ){
    return true}
    else{
      return false
    }
  }
  console.log(sum(92,9) )
  console.log(sum( 83,99 ))
  console.log(sum(54,9))
  console.log(sum(76,8) )
</script>
```

false	index4.html:17
false	index4.html:18
true	index4.html:19
true	index4.html:20

Add up the Numbers 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 $1 + 2 + 3 + 4 = 10$.

Examples

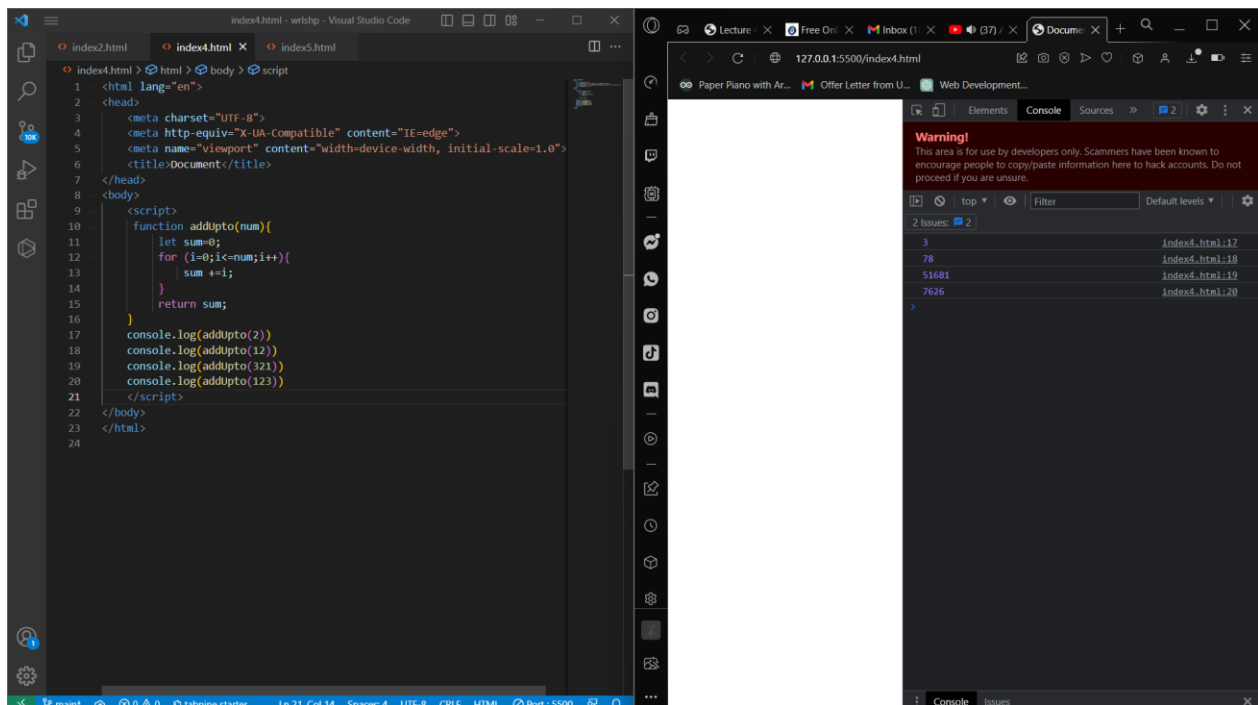
`addUp(4)` → 10

`addUp(13)` → 91

`addUp(600)` → 180300

Notes

Expect any positive number between 1 and 1000.



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"
```

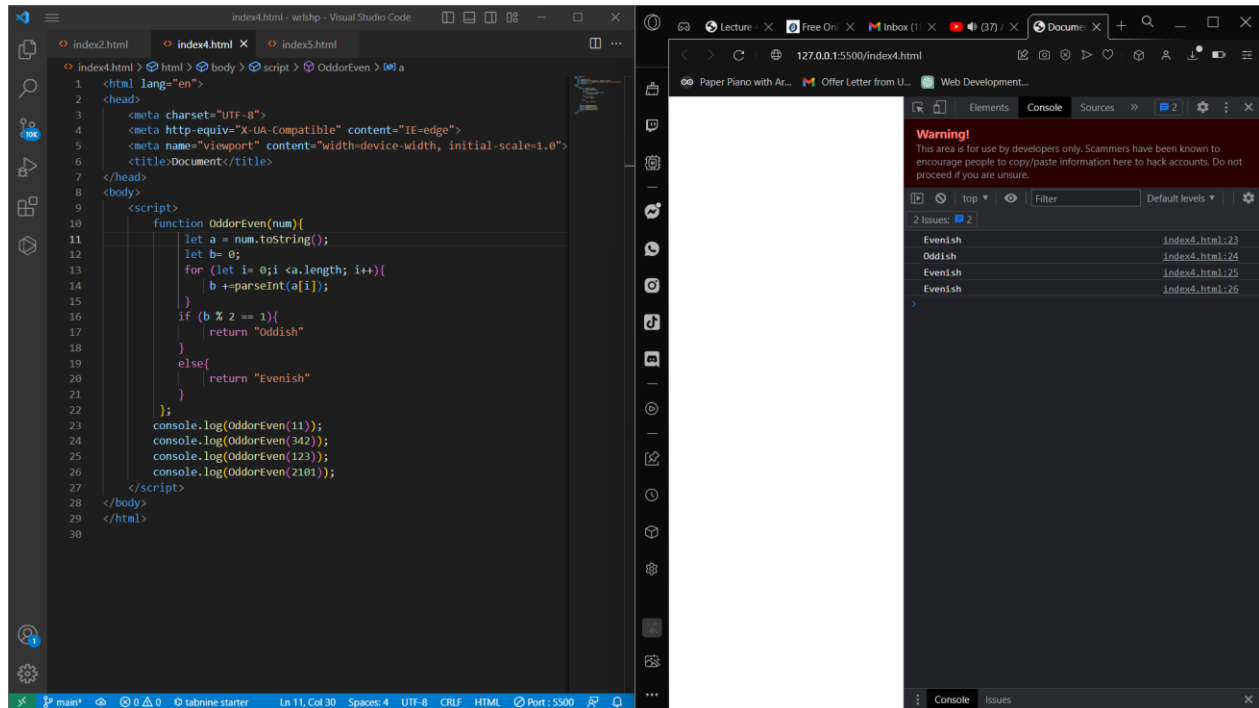
```
// 3 + 7 + 3 = 13
```

```
// 13 % 2 = 1
```

```
oddishOrEvenish(4433) → "Evenish"
```

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

```
// 14 % 2 = 0
```



Any Prime Number in Range

Create a function that returns `true` if there's at least one prime number in the given range

(`n1` to `n2` (inclusive)), `false` otherwise. **Examples**

```
primeInRange(10, 15) →true //
```

Prime numbers in range: 11, 13

```
primeInRange(62, 66) →false
```

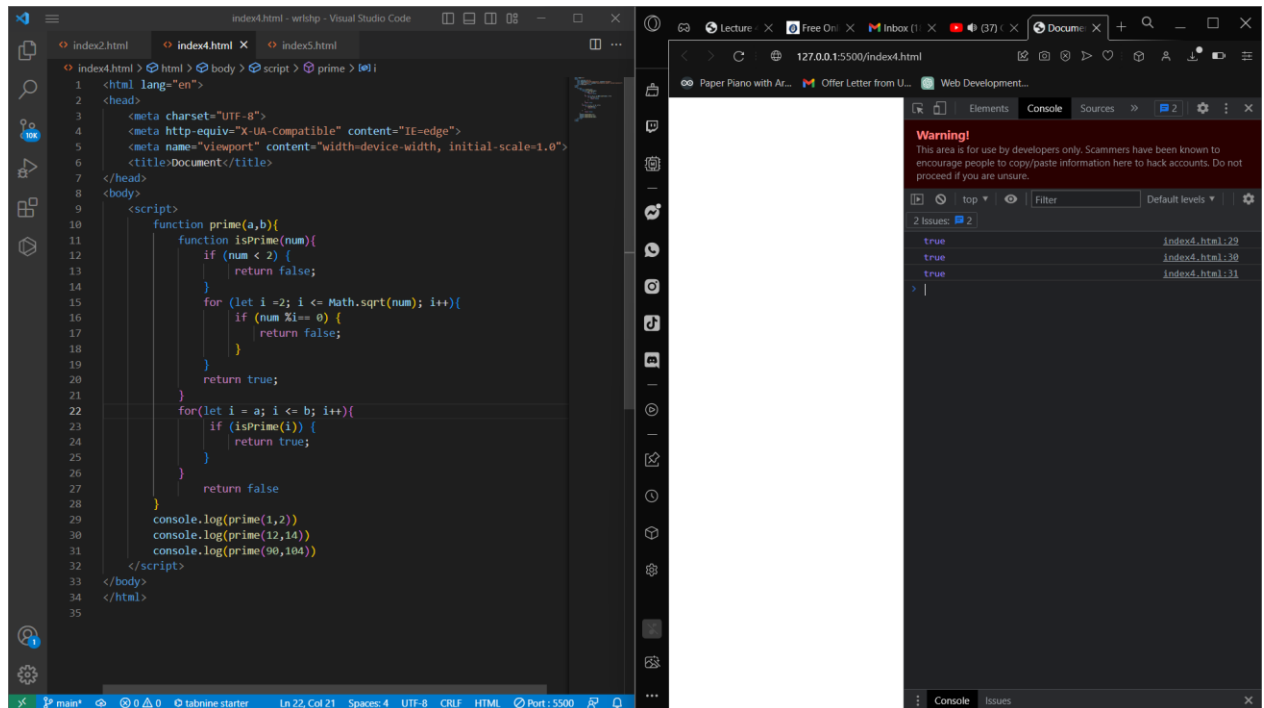
```
// No prime numbers in range.
```

```
primeInRange(3, 5) →true
```

```
// Prime numbers in range: 3, 5
```

Notes

- n_2 is always greater than n_1 .
- n_1 and n_2 are always positive.
- 0 and 1 aren't prime numbers.



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`

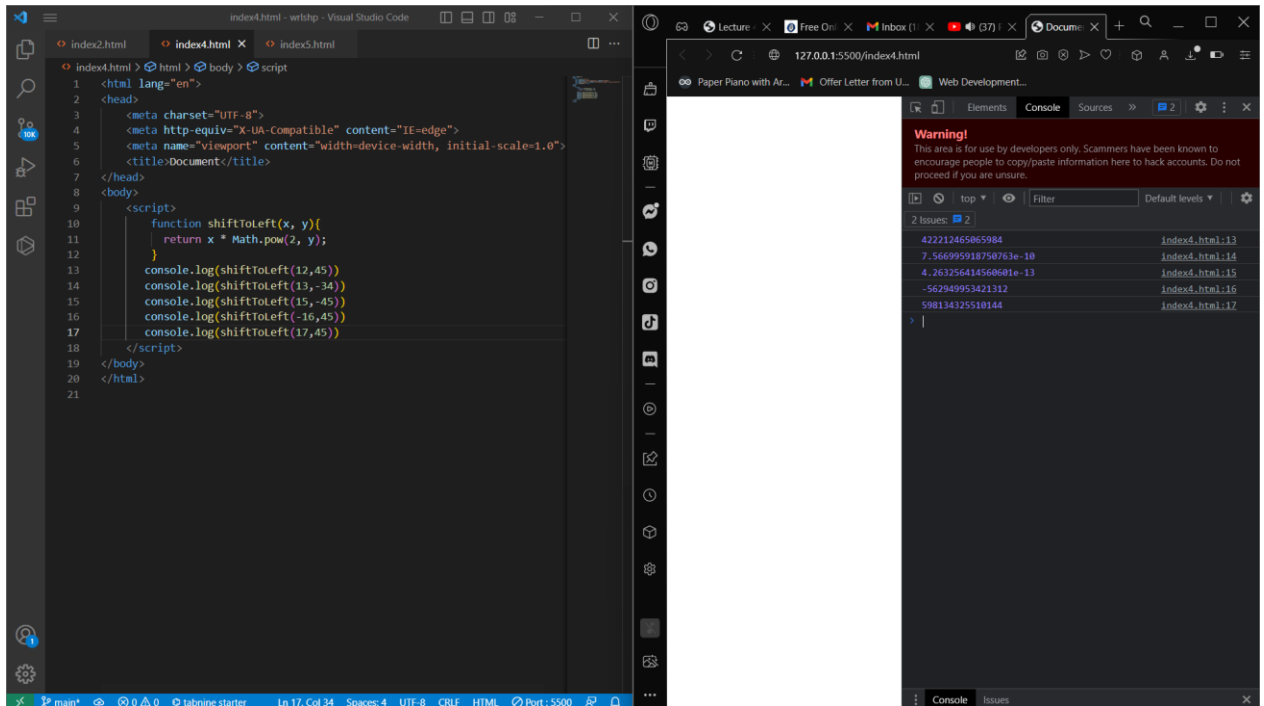
`shiftToLeft(-6, 5) → -192`

`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.



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 x2. 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".

