50 CODING CHALLENGES

... IN 50 DAYS OR LESS

codeguppy.com

About these coding challenges

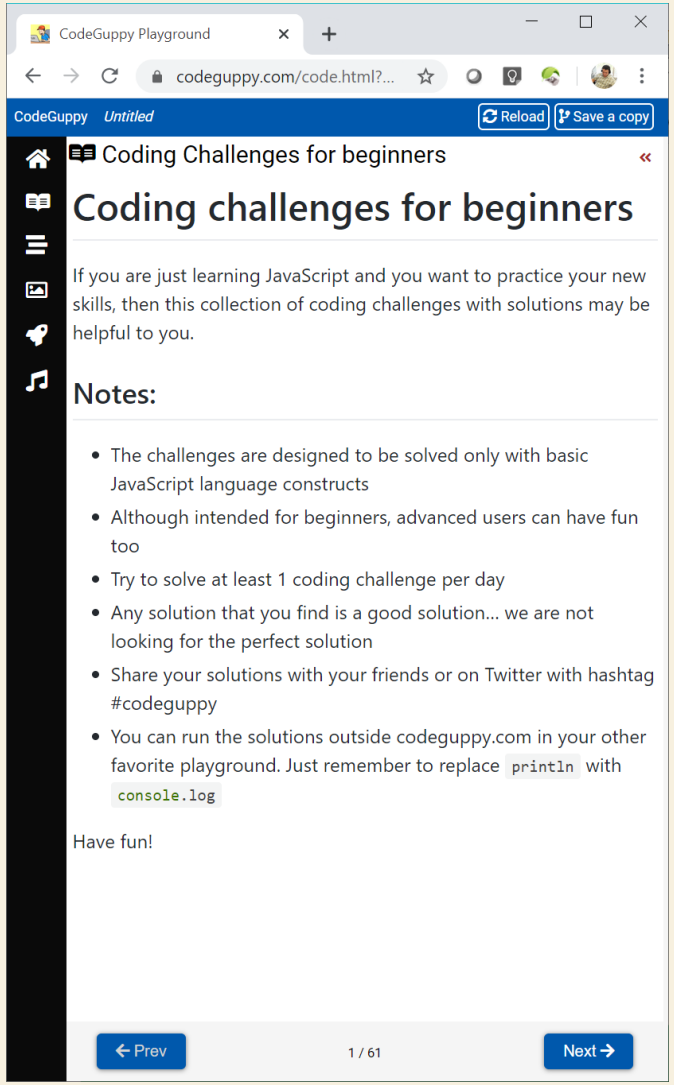
- These coding challenges are intended for code newbies that want to practice JavaScript
- The challenges are designed to be solved only with basic JavaScript language constructs
- Although intended for beginners, advanced users can have fun too
- Try to solve at least 1 coding challenge per day
- Any solution that you find is a good solution… we are not looking for the perfect solution
- For convenience, when you work on these challenges, you can use an online coding playground, such as the one from https://codeguppy.com
- Feel free to share this booklet with your friends and invite them to complete the coding challenges
- This booklet contains also all the solutions to these challenges
- These challenges are great as coding exercises for CS classrooms!
- Have fun!

50 coding challenges – Part 1

1. Print numbers from 1 to 10
2. Print the odd numbers less than 100
3. Print the multiplication table with 7
4. Print all the multiplication tables with numbers from 1 to 10
5. Calculate the sum of numbers from 1 to 10
6. Calculate 10!
7. Calculate the sum of even numbers greater than 10 and less than 30
8. Create a function that will convert from Celsius to Fahrenheit
9. Create a function that will convert from Fahrenheit to Celsius
10. Calculate the sum of numbers in an array of numbers
11. Calculate the average of the numbers in an array of numbers
12. Create a function that receives an array of numbers as argument and returns an array containing only the positive numbers
13. Find the maximum number in an array of numbers
14. Print the first 10 Fibonacci numbers without recursion
15. Create a function that will find the n^{th} Fibonacci number using recursion
16. Create a function that will return a Boolean specifying if a number is prime
17. Calculate the sum of digits of a positive integer number
18. Print the first 100 prime numbers
19. Create a function that will return in an array the first "p" prime numbers greater than "n"
20. Rotate an array to the left 1 position
21. Rotate an array to the right 1 position
22. Reverse an array
23. Reverse a string
24. Create a function that will merge two arrays and return the result as a new array
25. Create a function that will receive two arrays of numbers as arguments and return an array composed of all the numbers that are either in the first array or second array but not in both
26. Create a function that will receive two arrays and will return an array with elements that are in the first array but not in the second

50 coding challenges – Part II

27. Create a function that will receive an array of numbers as argument and will return a new array with distinct elements
28. Calculate the sum of first 100 prime numbers and return them in an array
29. Print the distance between the first 100 prime numbers
30. Create a function that will add two positive numbers of indefinite size. The numbers are received as strings and the result should be also provided as string.
31. Create a function that will return the number of words in a text
32. Create a function that will capitalize the first letter of each word in a text
33. Calculate the sum of numbers received in a comma delimited string
34. Create a function that returns an array with words inside a text.
35. Create a function to convert a CSV text to a "bi-dimensional" array
36. Create a function that converts a string to an array of characters
37. Create a function that will convert a string in an array containing the ASCII codes of each character
38. Create a function that will convert an array containing ASCII codes in a string
39. Implement the Caesar cypher
40. Implement the bubble sort algorithm for an array of numbers
41. Create a function to calculate the distance between two points defined by their x, y coordinates
42. Create a function that will return a Boolean value indicating if two circles defined by center coordinates and radius are intersecting
43. Create a function that will receive a bi-dimensional array as argument and a number and will extract as a unidimensional array the column specified by the number
44. Create a function that will convert a string containing a binary number into a number
45. Create a function to calculate the sum of all the numbers in a jagged array (contains numbers or other arrays of numbers on an unlimited number of levels)
46. Find the maximum number in a jagged array of numbers or array of numbers
47. Deep copy a jagged array with numbers or other arrays in a new array
48. Create a function to return the longest word in a string
49. Shuffle an array of strings
50. Create a function that will receive n as argument and return an array of n random numbers from 1 to n. The numbers should be unique inside the array.
51. Find the frequency of letters inside a string. Return the result as an array of arrays. Each subarray has 2 elements: letter and number of occurrences.
52. Calculate Fibonacci(500) with high precision (all digits)
53. Calculate 70! with high precision (all digits)

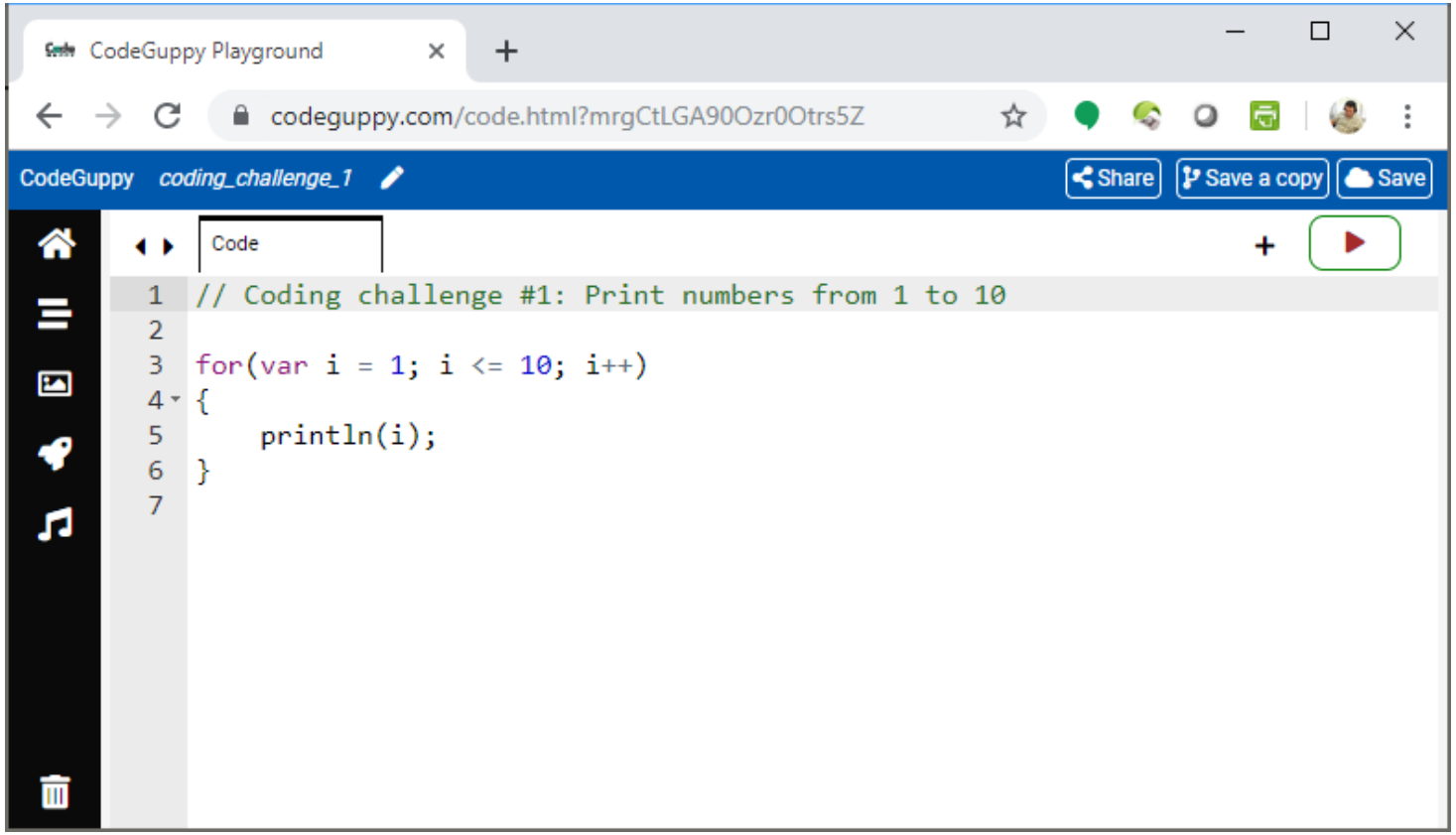


The source code of the solutions presented in this booklet is available online at:

https://codeguppy.com/code.html?t=coding_challenges

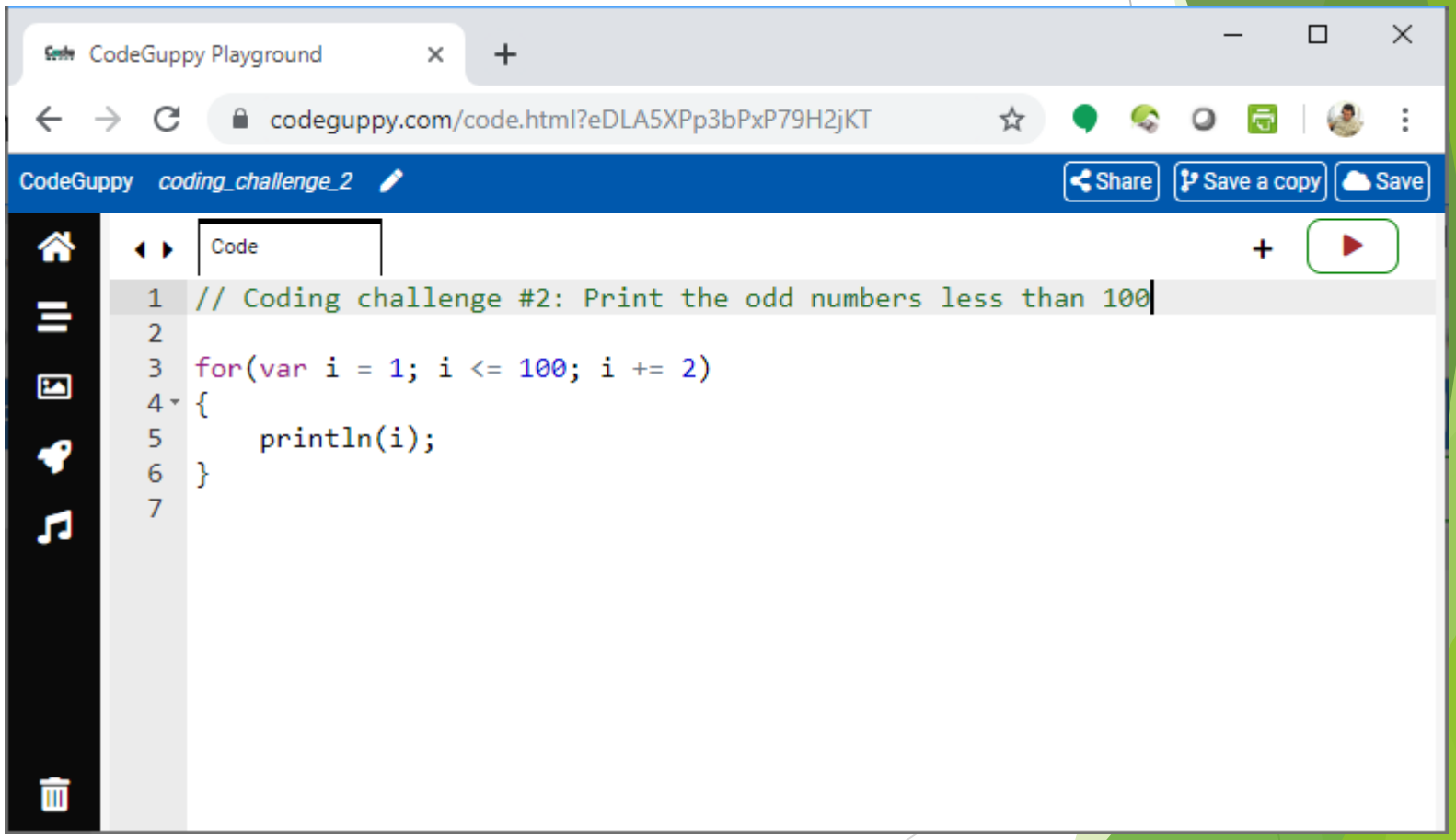
You can type-in the full link above, or just go to codeguppy.com and browse to locate the "Coding Challenges" project.

Coding challenge #1: Print numbers from 1 to 10



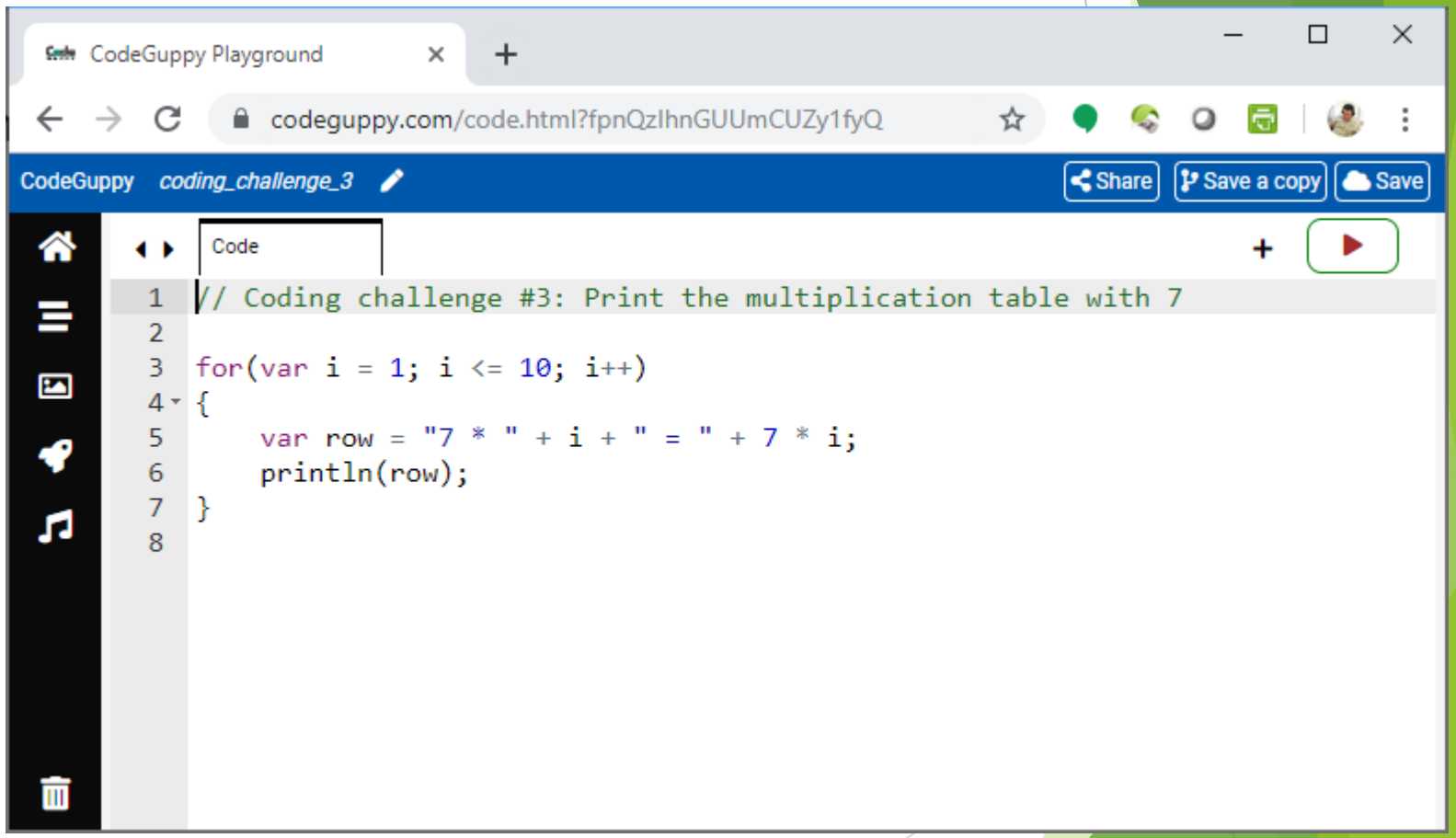
Coding challenge #2

Print the odd numbers less than 100



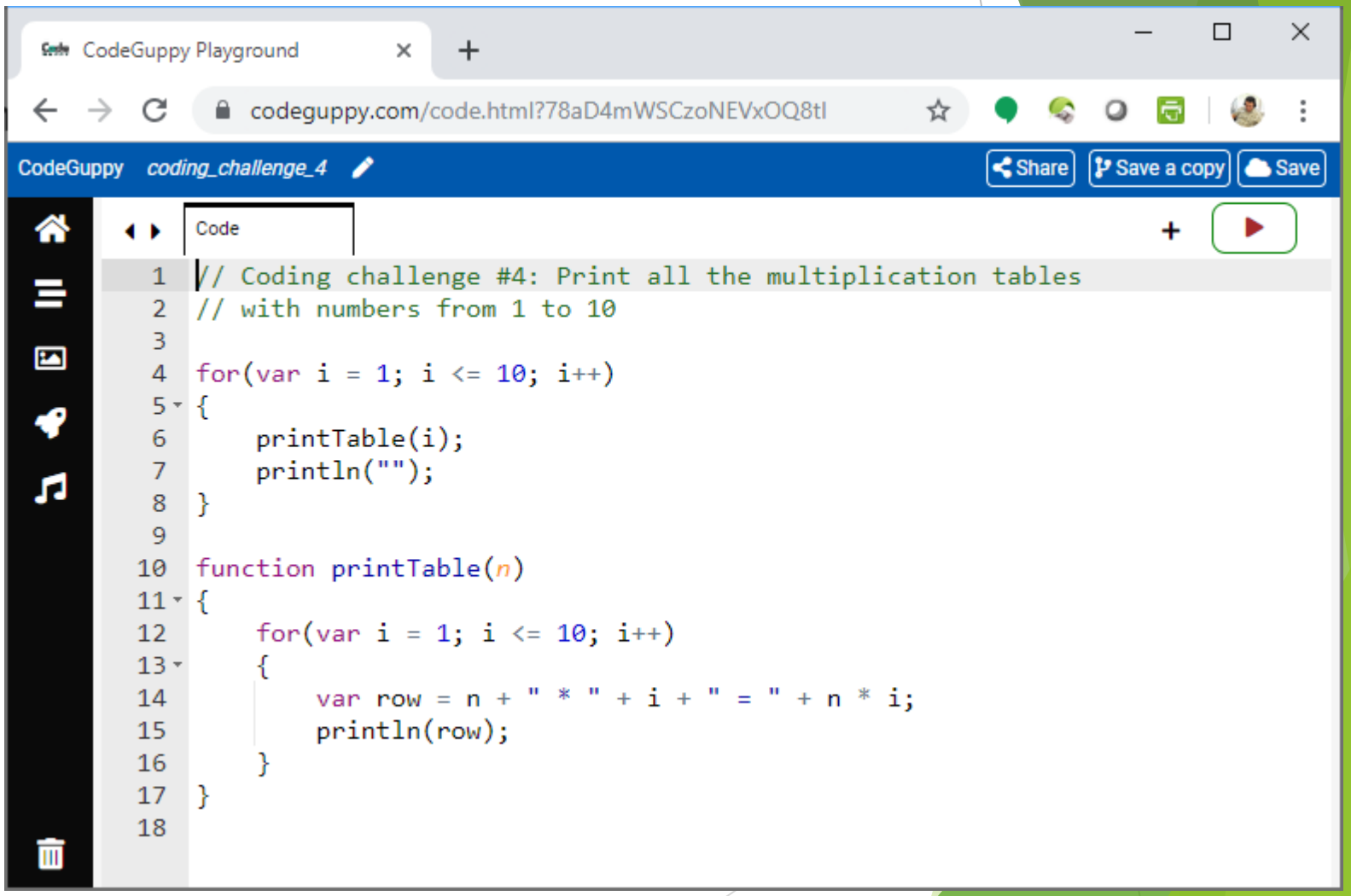
Coding challenge #3

Print the multiplication table with 7



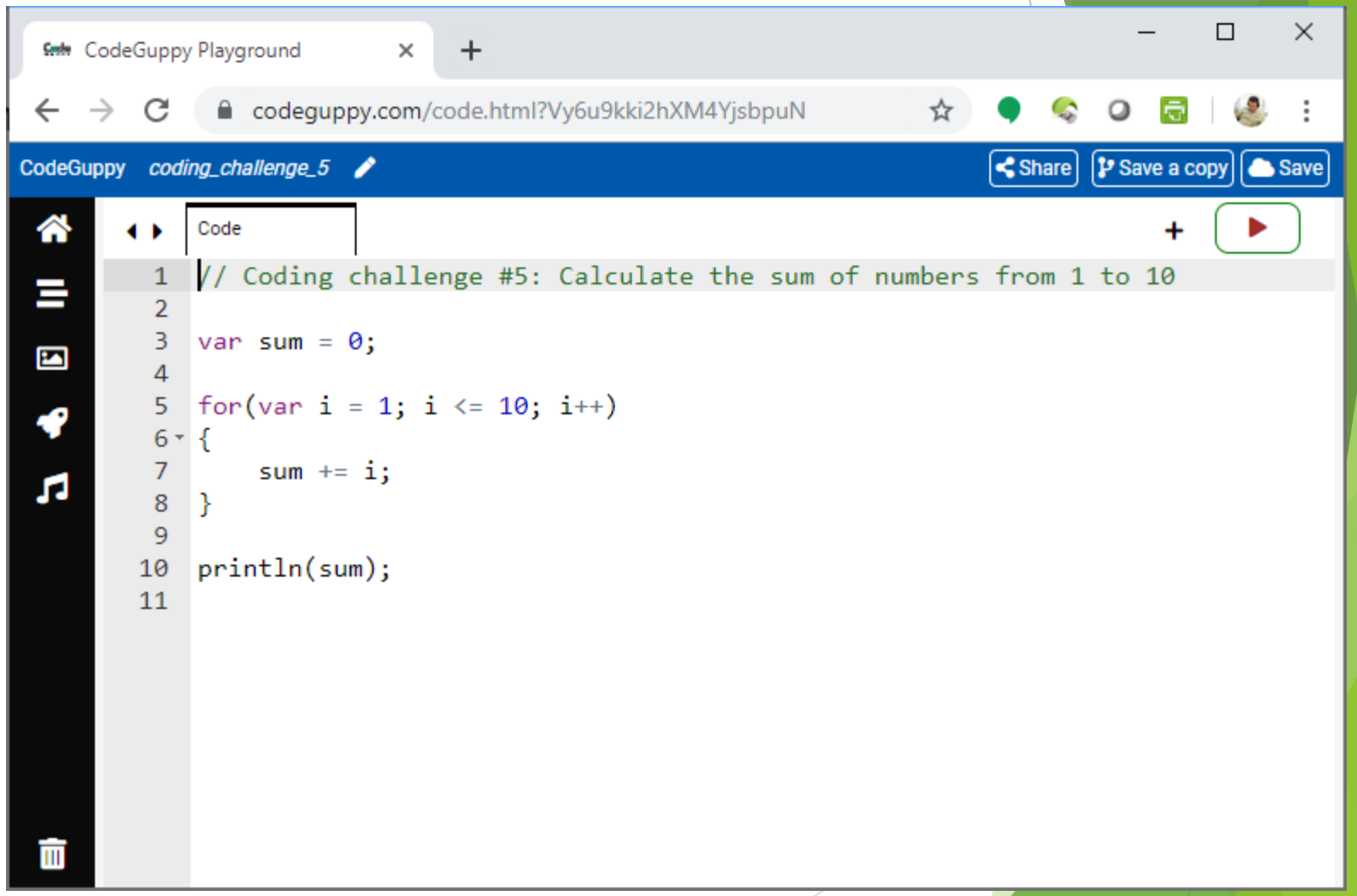
Coding challenge #4

Print the multiplication tables with numbers from 1 to 10



Coding challenge #5

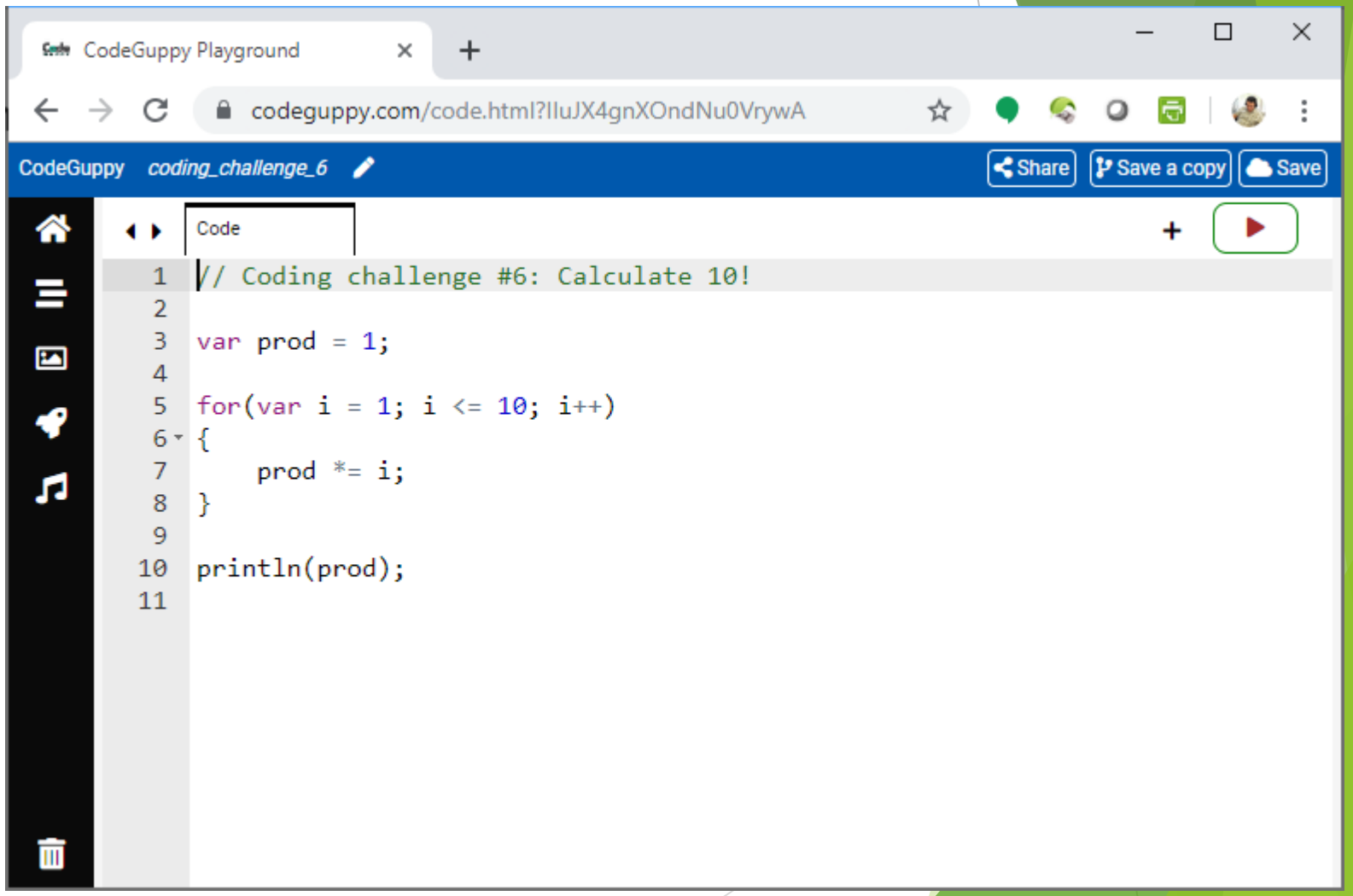
Calculate the sum of numbers from 1 to 10



Coding challenge #6

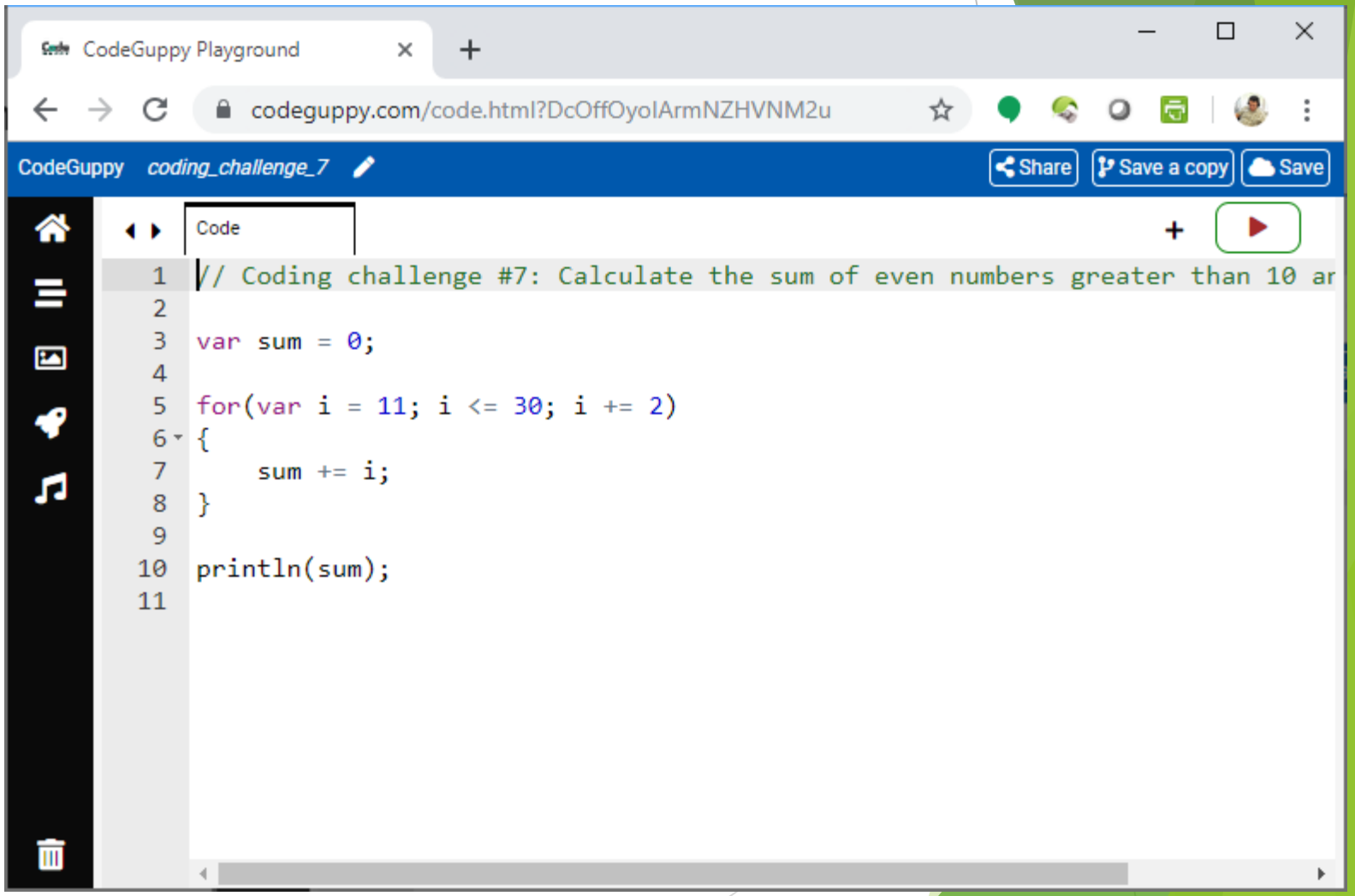
Calculate 10!

Reminder n! = 1 * 2 * … * n



Coding challenge #7

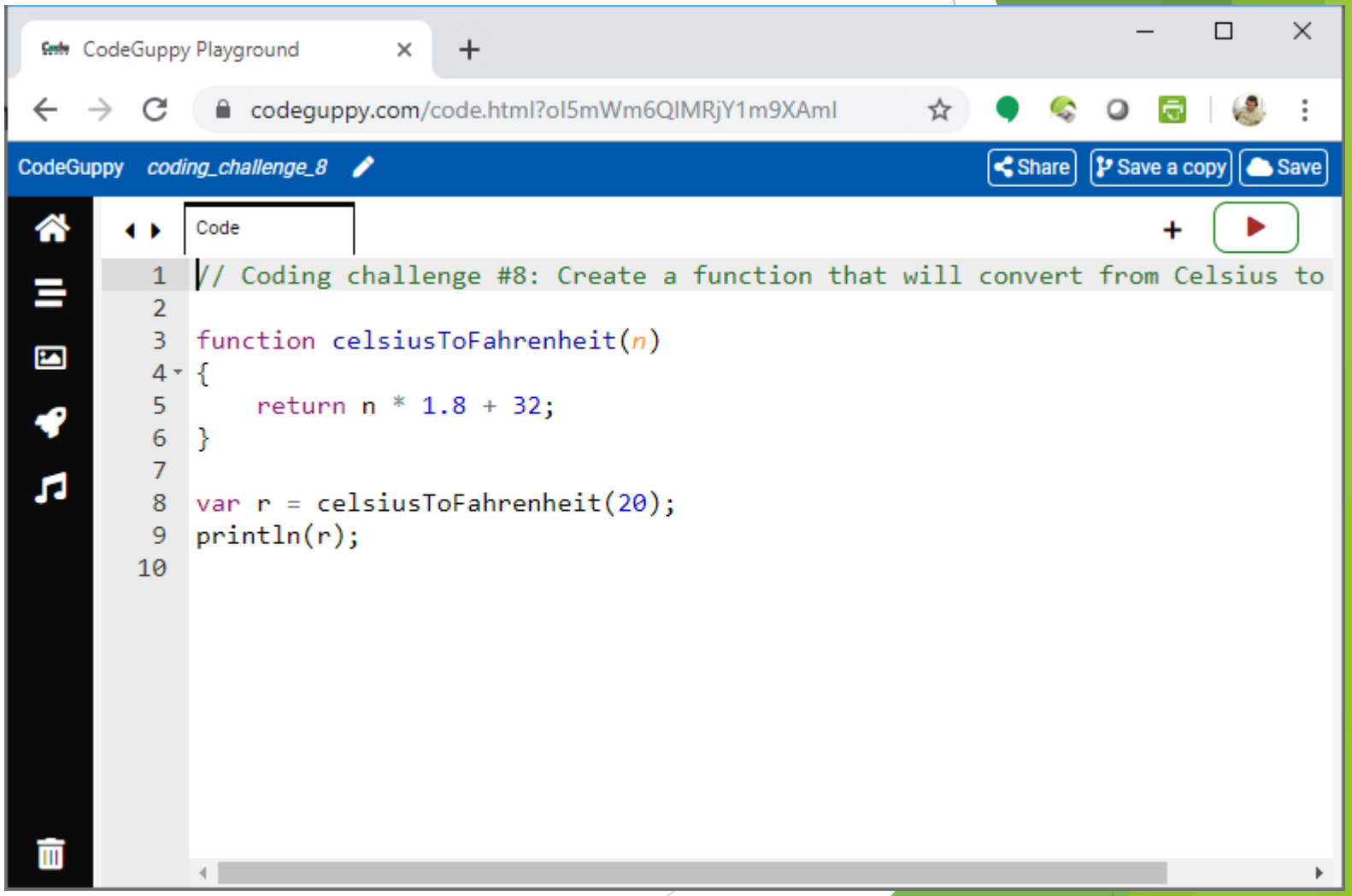
Calculate the sum of odd numbers greater than 10 and less or equal than 30



Coding challenge #8

Create a function that will convert from Celsius to Fahrenheit

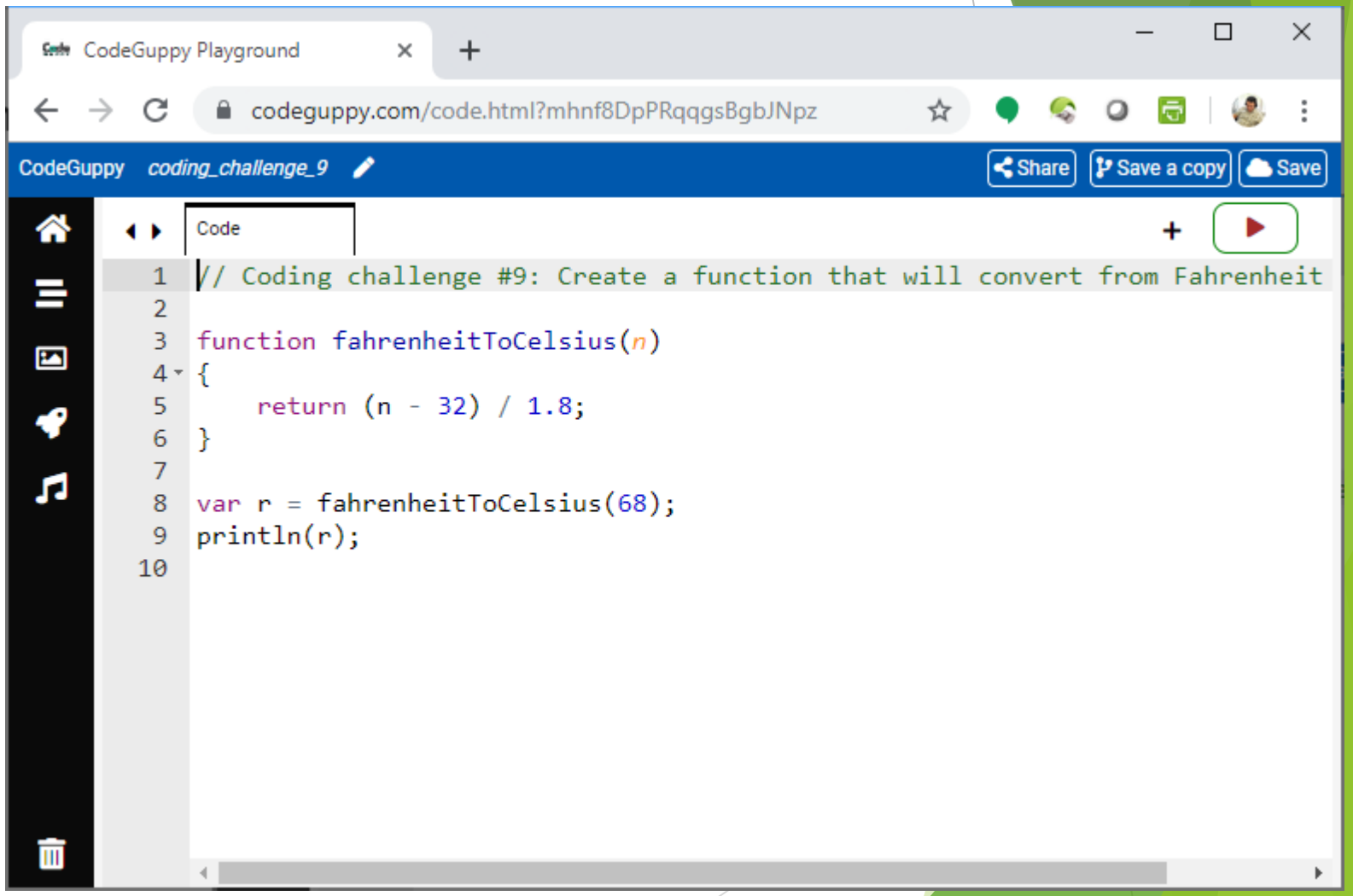
Reminder: C = F – 32 / 1.8



Coding challenge #9

Create a function that will convert from Fahrenheit to Celsius

Reminder: C = F – 32 / 1.8

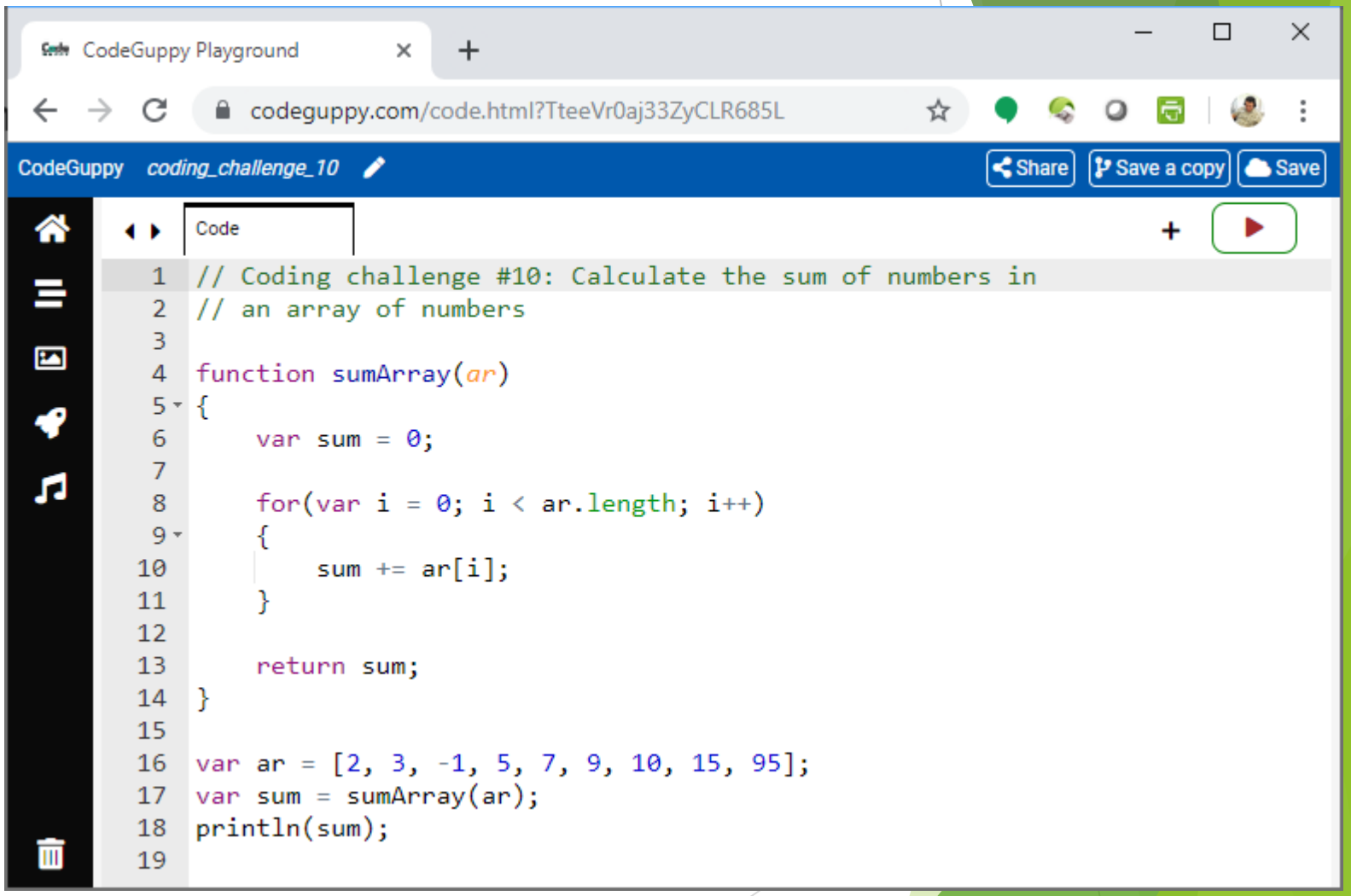


Coding challenge #10

Calculate the sum of numbers in an array of numbers.

Example array:
[2, 3, -1, 5, 7, 9, 10, 15, 95]

Expected output:
145

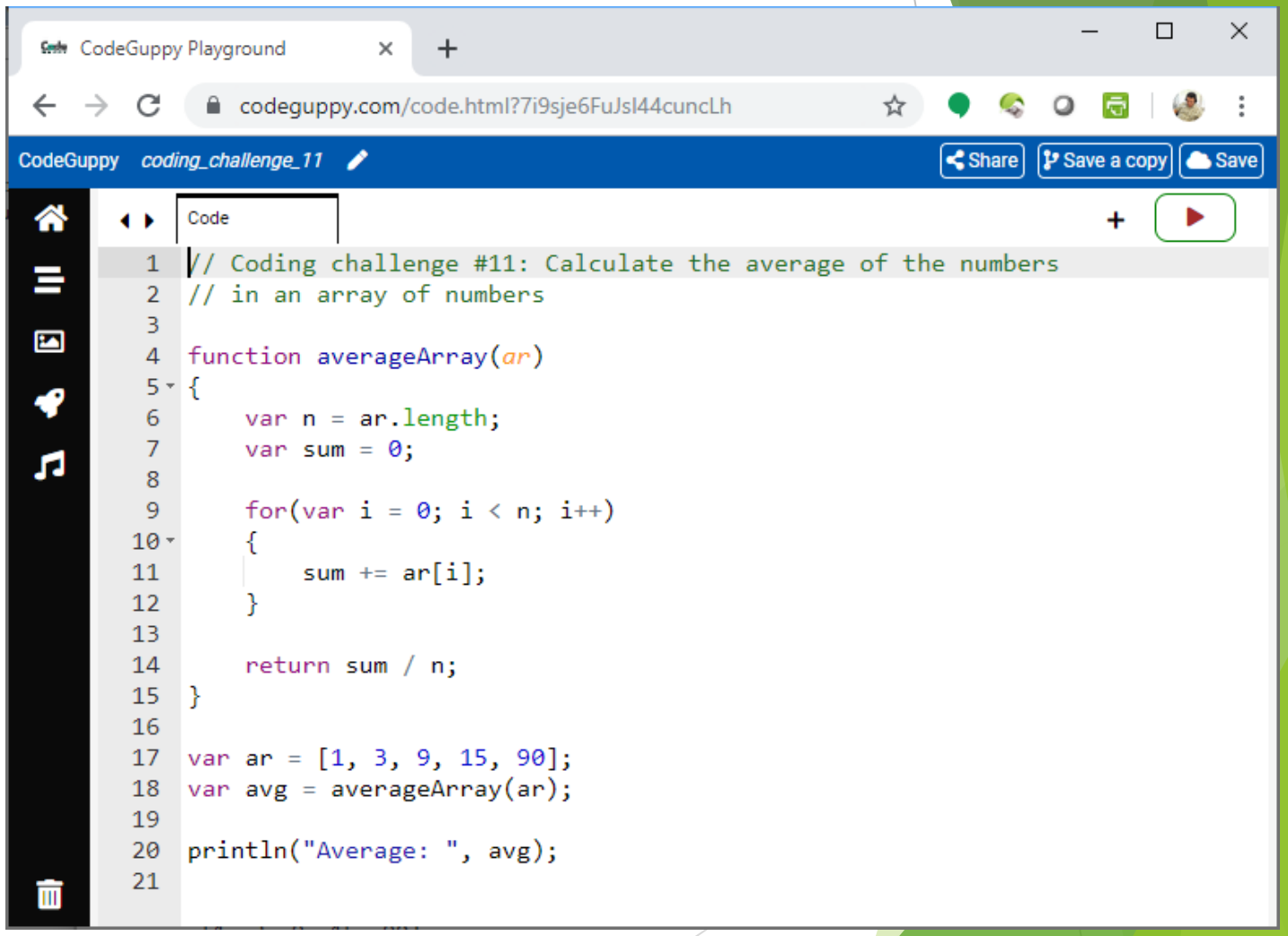


Coding challenge #11

Calculate the average of the numbers in an array of numbers

Example array:
[1, 3, 9, 15, 90]

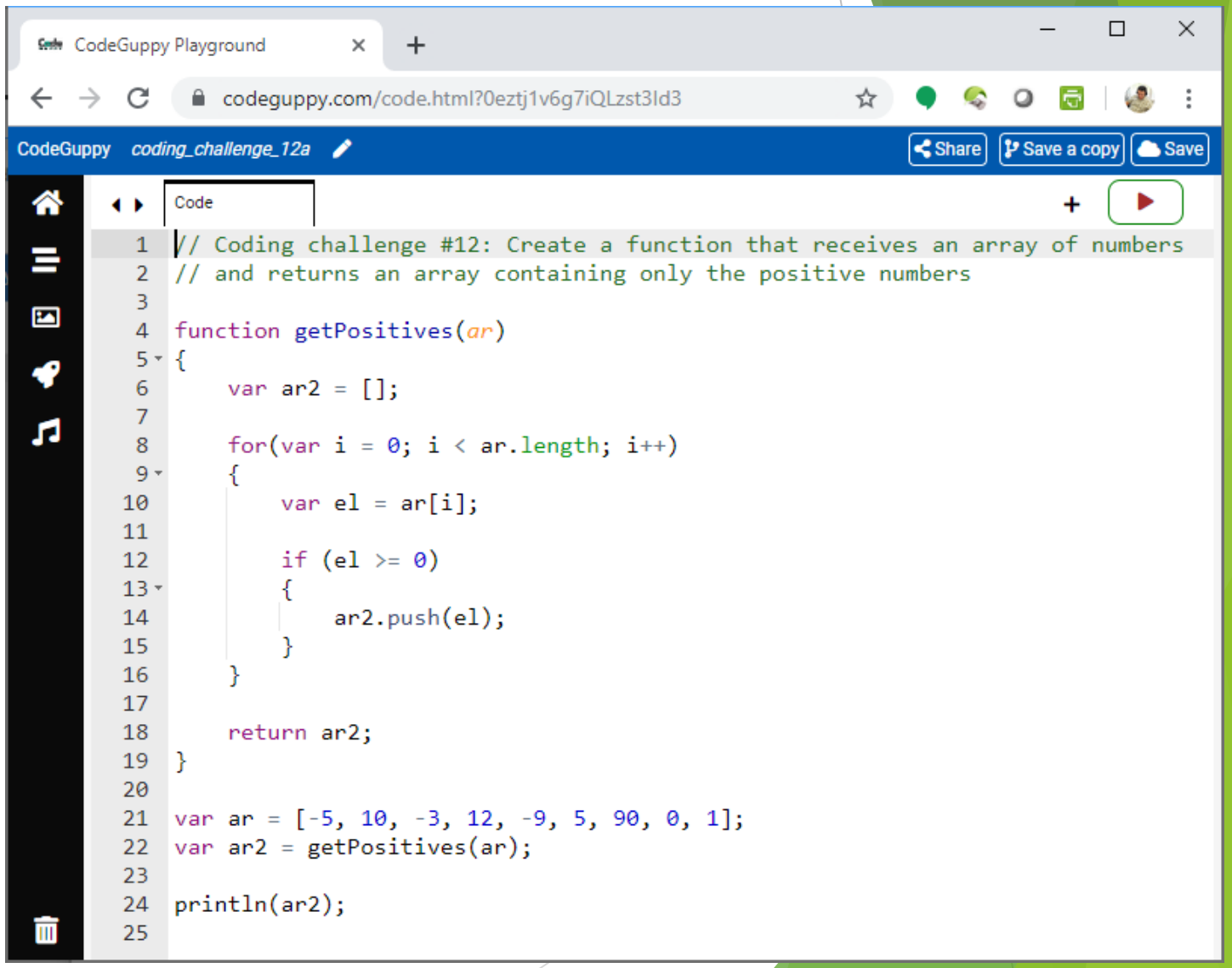
Expected output:
23.6



Coding challenge #12a

Create a function that receives an array of numbers and returns an array containing only the positive numbers.

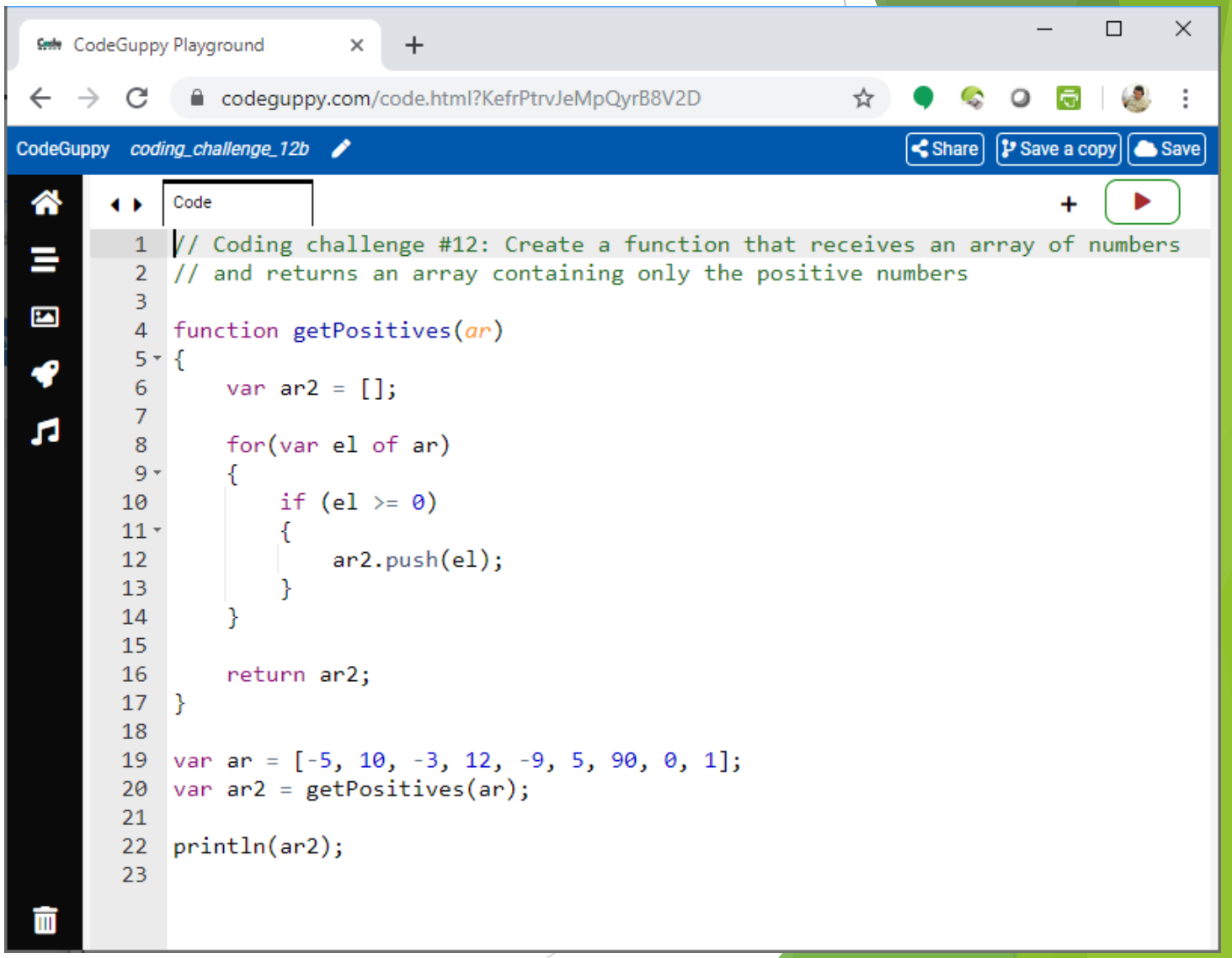
Requirement: Use a “for” loop



Coding challenge #12b

Create a function that receives an array of numbers and returns an array containing only the positive numbers.

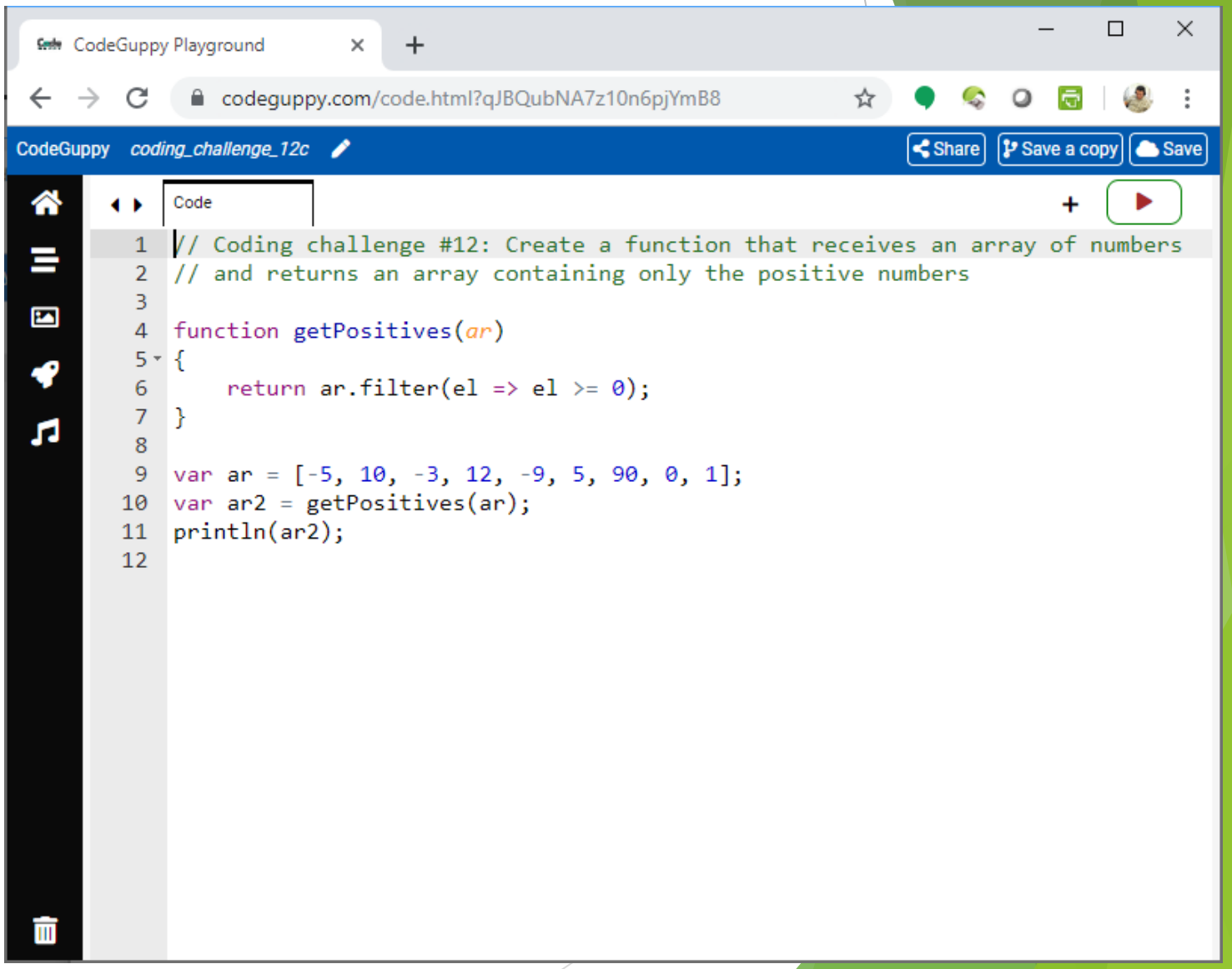
Requirement: Use a “for … of” loop



Coding challenge #12c

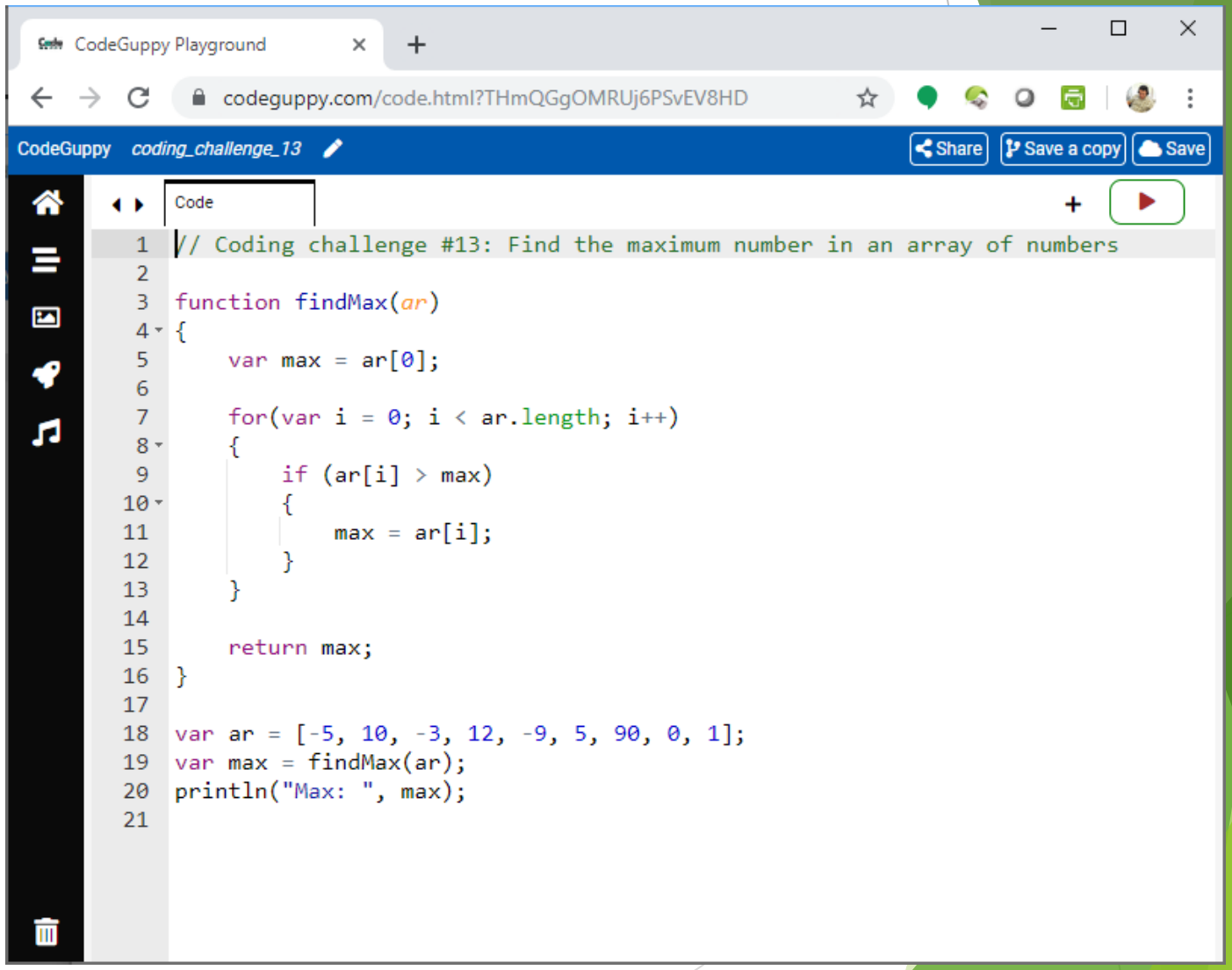
Create a function that receives an array of numbers and returns an array containing only the positive numbers.

Requirement: Use .filter() array method



Coding challenge #13

Find the maximum number in an array of numbers

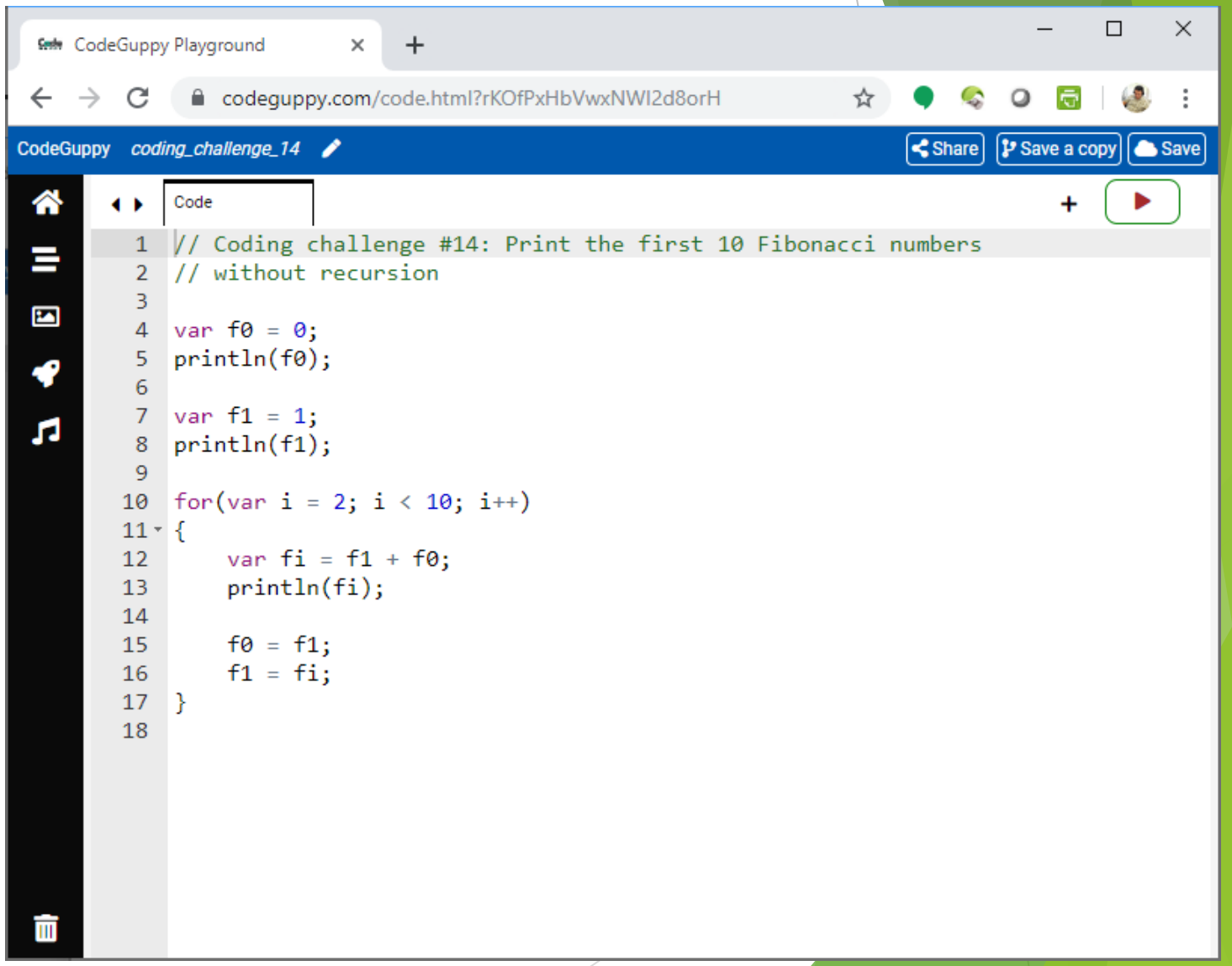


Coding challenge #14

Print the first 10 Fibonacci numbers without using recursion.

Reminder:

F(0) = 0
F(1) = 1
F(n) = F(n-1) + F(n-2)

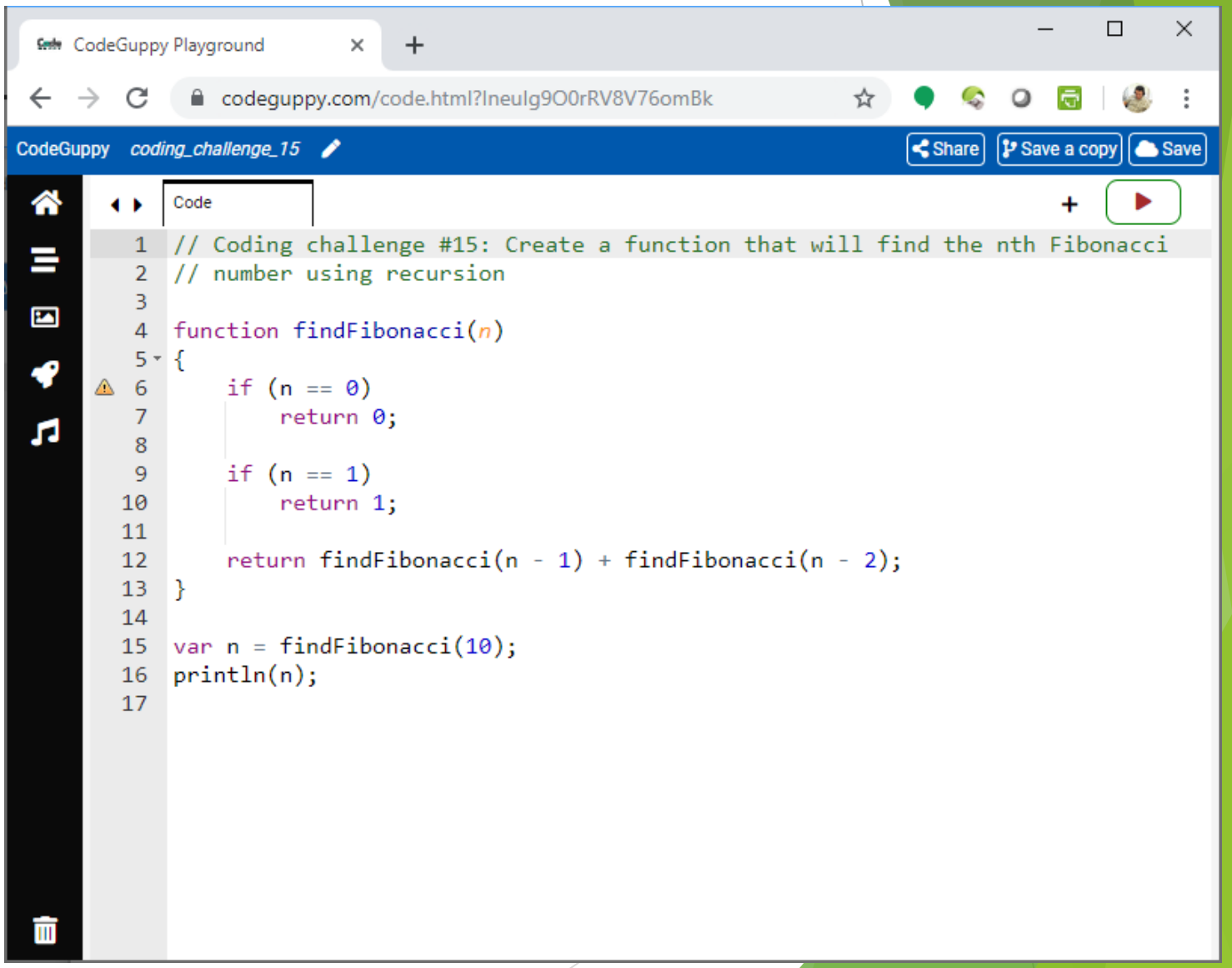


Coding challenge #15

Create a function that will find the nth Fibonacci number using recursion.

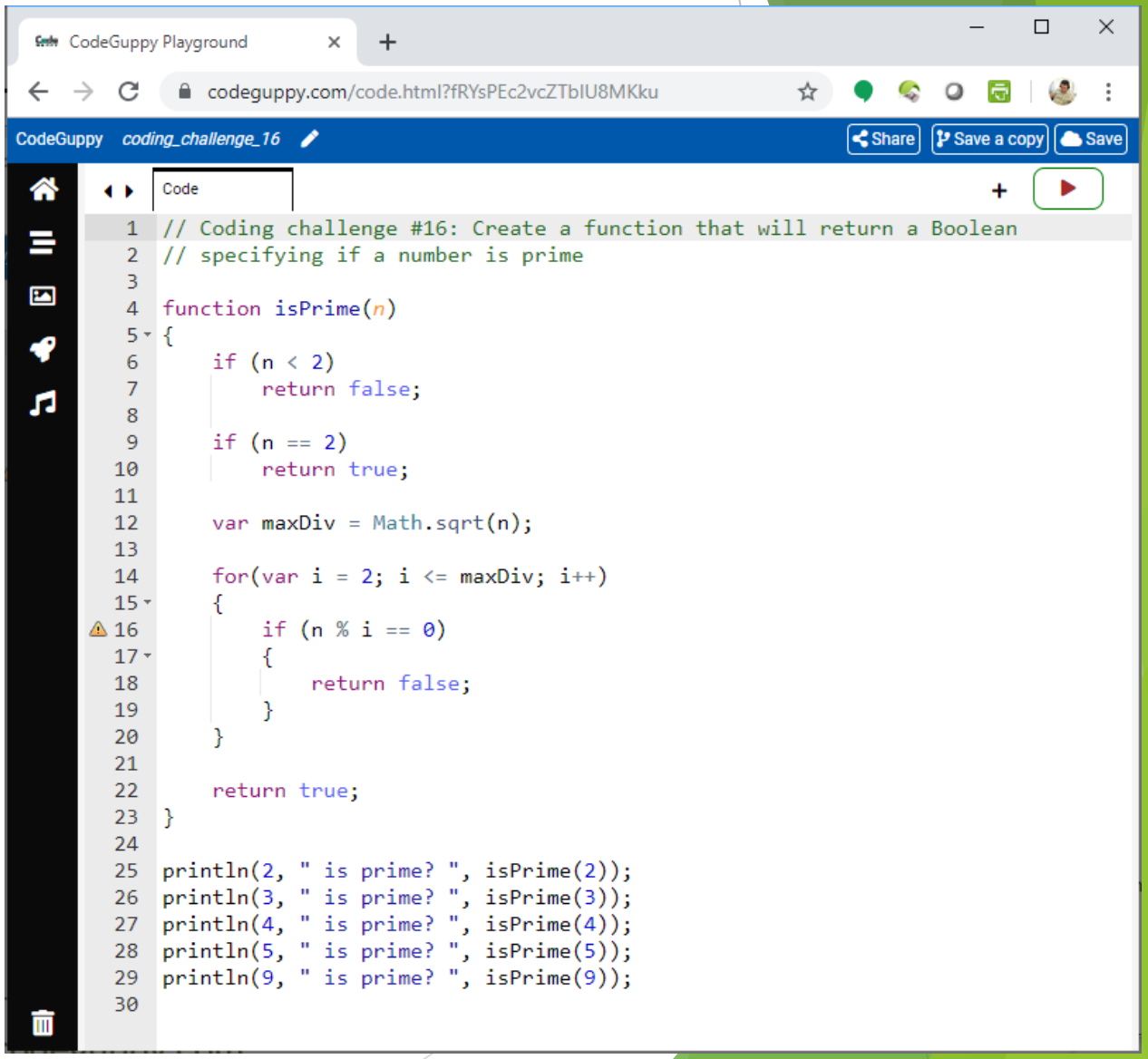
Reminder:

F(0) = 0
F(1) = 1
F(n) = F(n-1) + F(n-2)



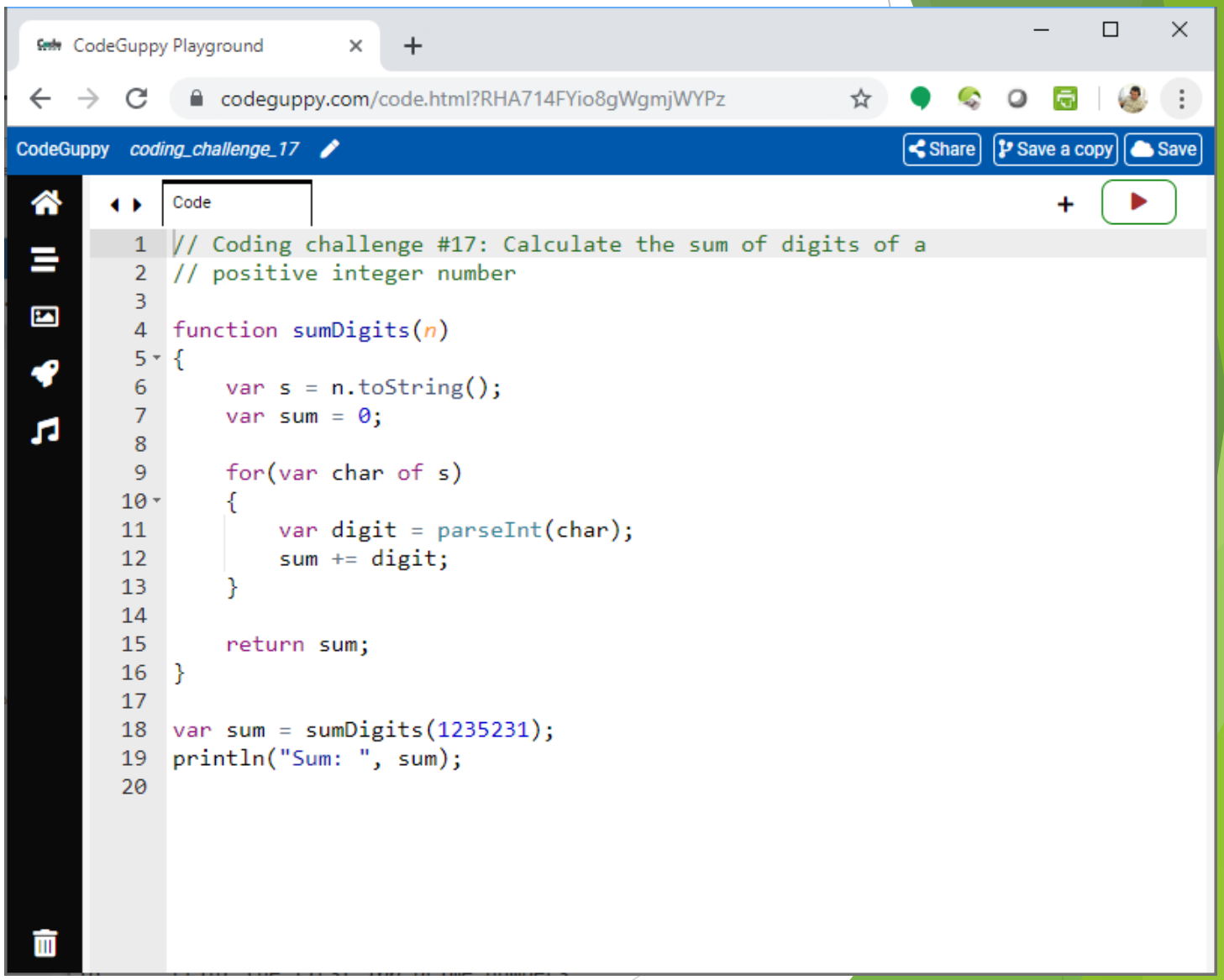
Coding challenge #16

Create a function that will return a Boolean specifying if a number is prime



Coding challenge #17

Calculate the sum of digits
of a positive integer number



Coding challenge #18: Print the first 100 prime numbers

CodeGuppy Playground

codeguppy.com/code.html?gnMVeOZXN6VhLekyvui8

CodeGuppy *coding_challenge_18* Share Save a copy Save

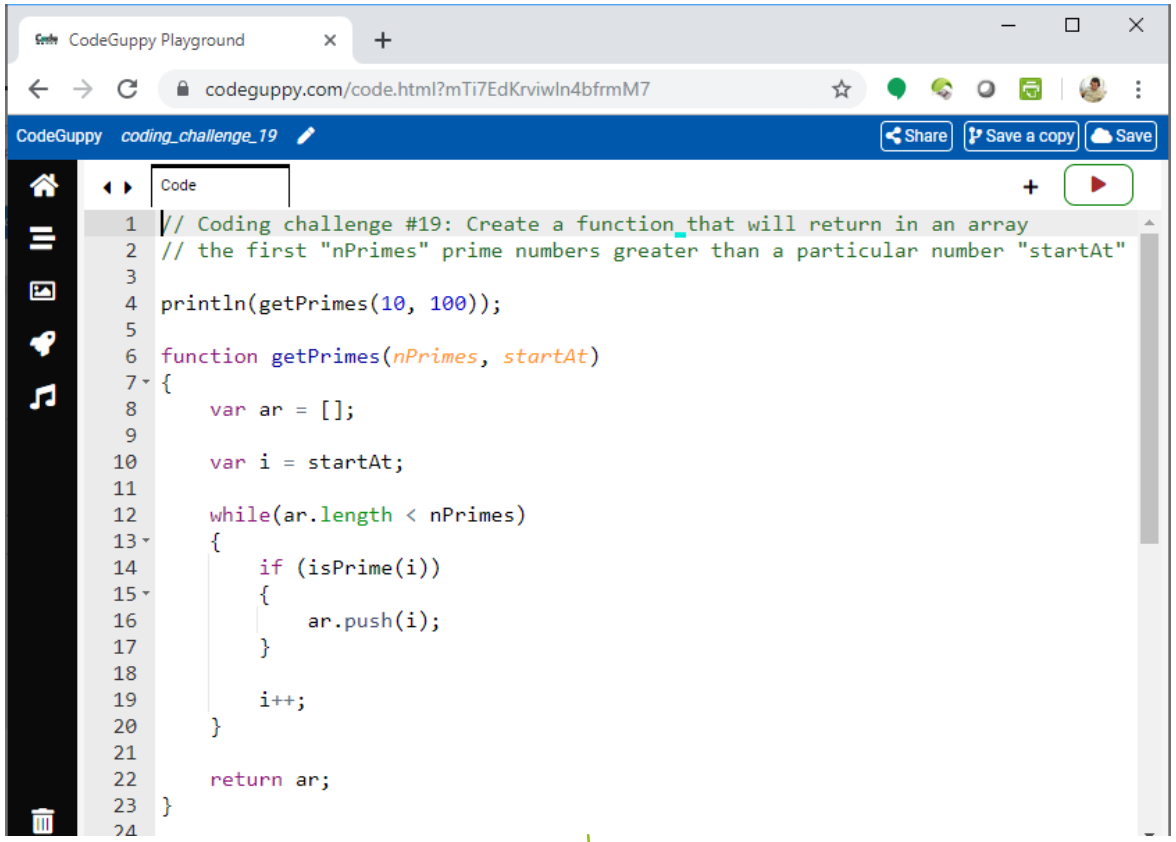
Code

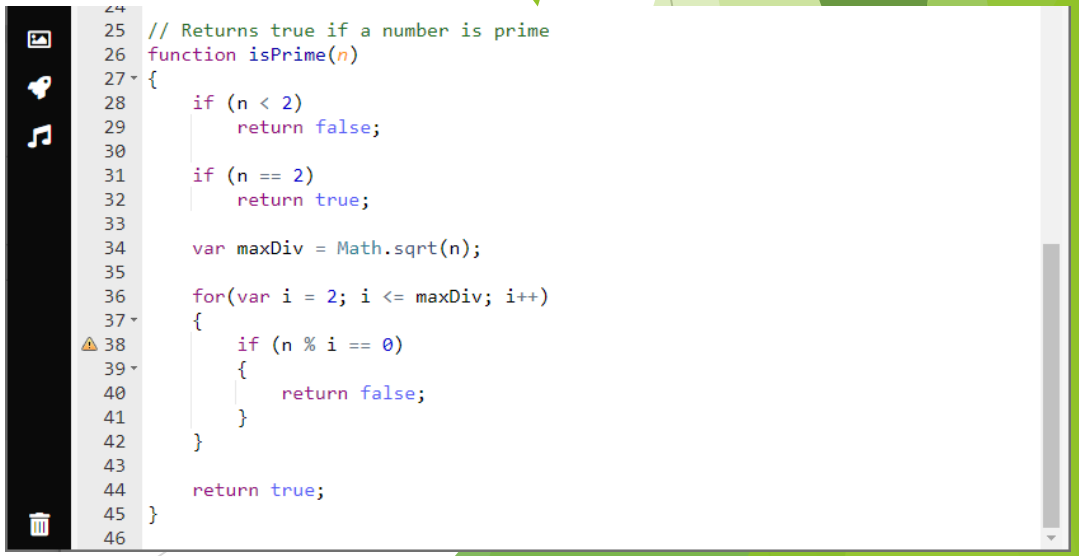
```
1  // Coding challenge #18: Print the first 100 prime numbers
2
3
4  printPrimes(100);
5
6  // Function prints the first nPrimes numbers
7  function printPrimes(nPrimes)
8  {
9      var n = 0;
10     var i = 2;
11
12     while(n < nPrimes)
13     {
14         if (isPrime(i))
15         {
16             println(n, " --> ", i);
17             n++;
18         }
19
20         i++;
21     }
22 }
23
```

```
24
25 // Returns true if a number is prime
26 function isPrime(n)
27 {
28     if (n < 2)
29         return false;
30
31     if (n == 2)
32         return true;
33
34     var maxDiv = Math.sqrt(n);
35
36     for(var i = 2; i <= maxDiv; i++)
37     {
38         if (n % i == 0)
39         {
40             return false;
41         }
42     }
43
44     return true;
45 }
46
```

codeguppy.com

Coding challenge #19: Create a function that will return in an array the first "nPrimes" prime numbers greater than a number "startAt"

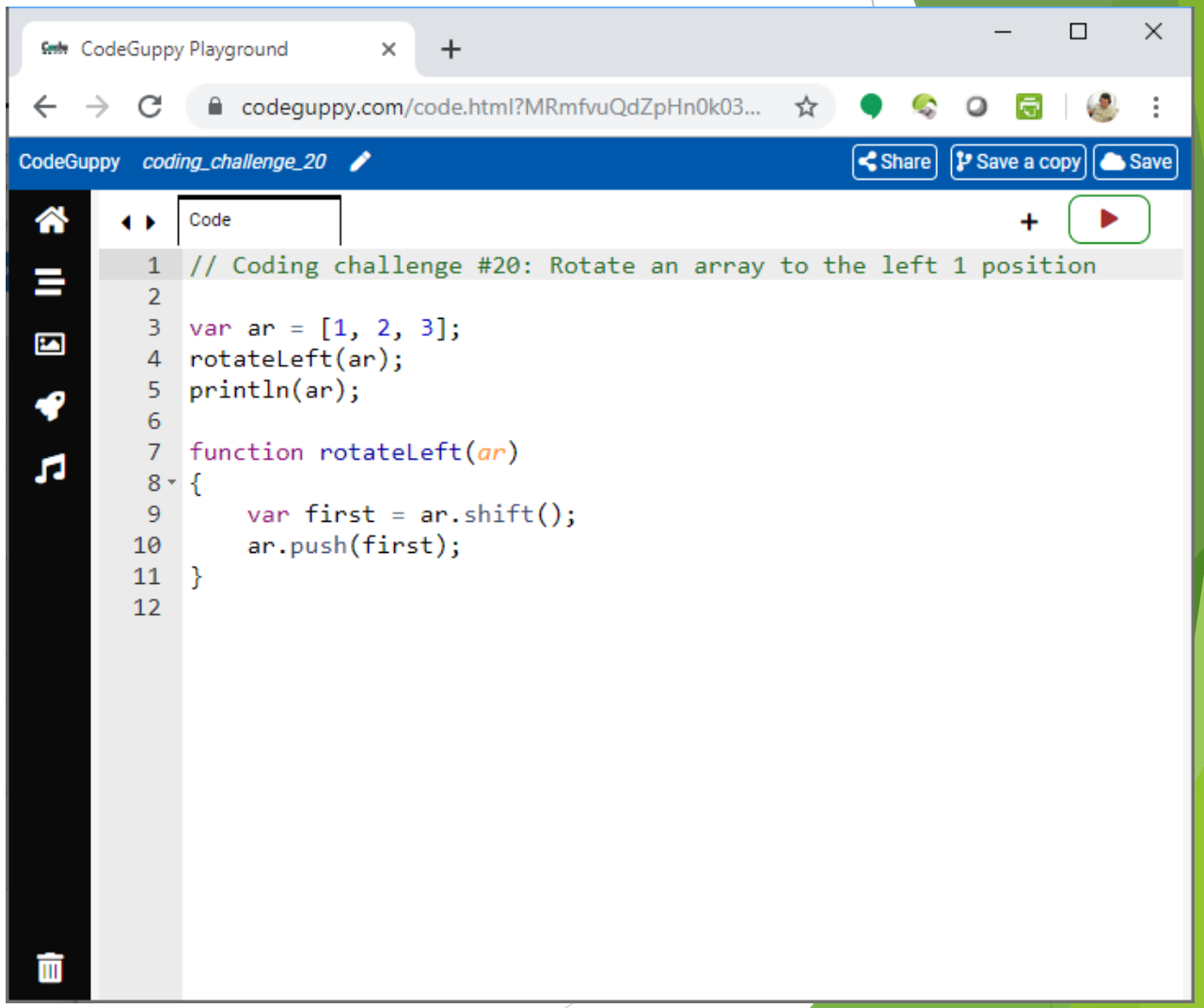




codeguppy.com

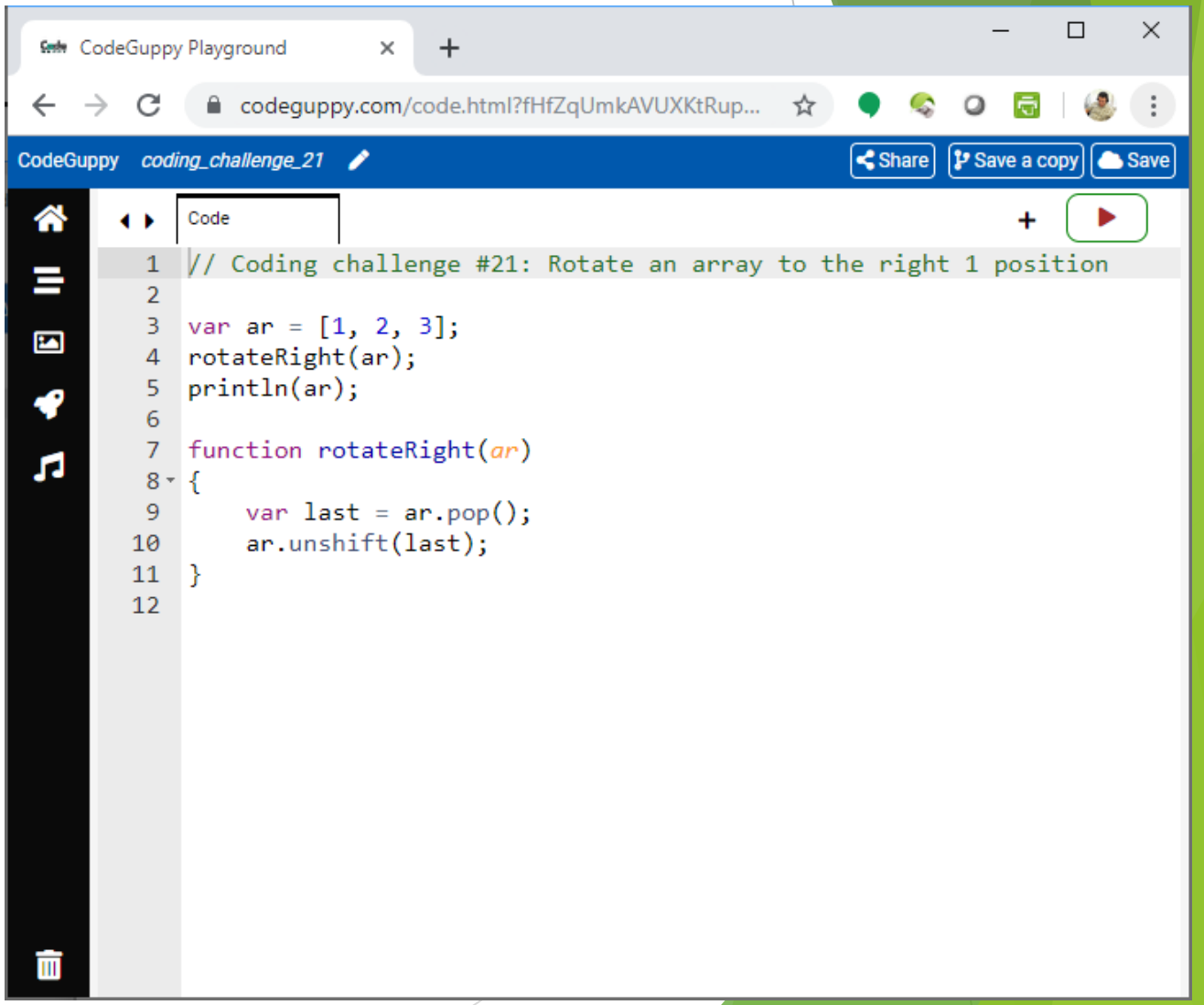
Coding challenge #20

Rotate an array to the left 1 position



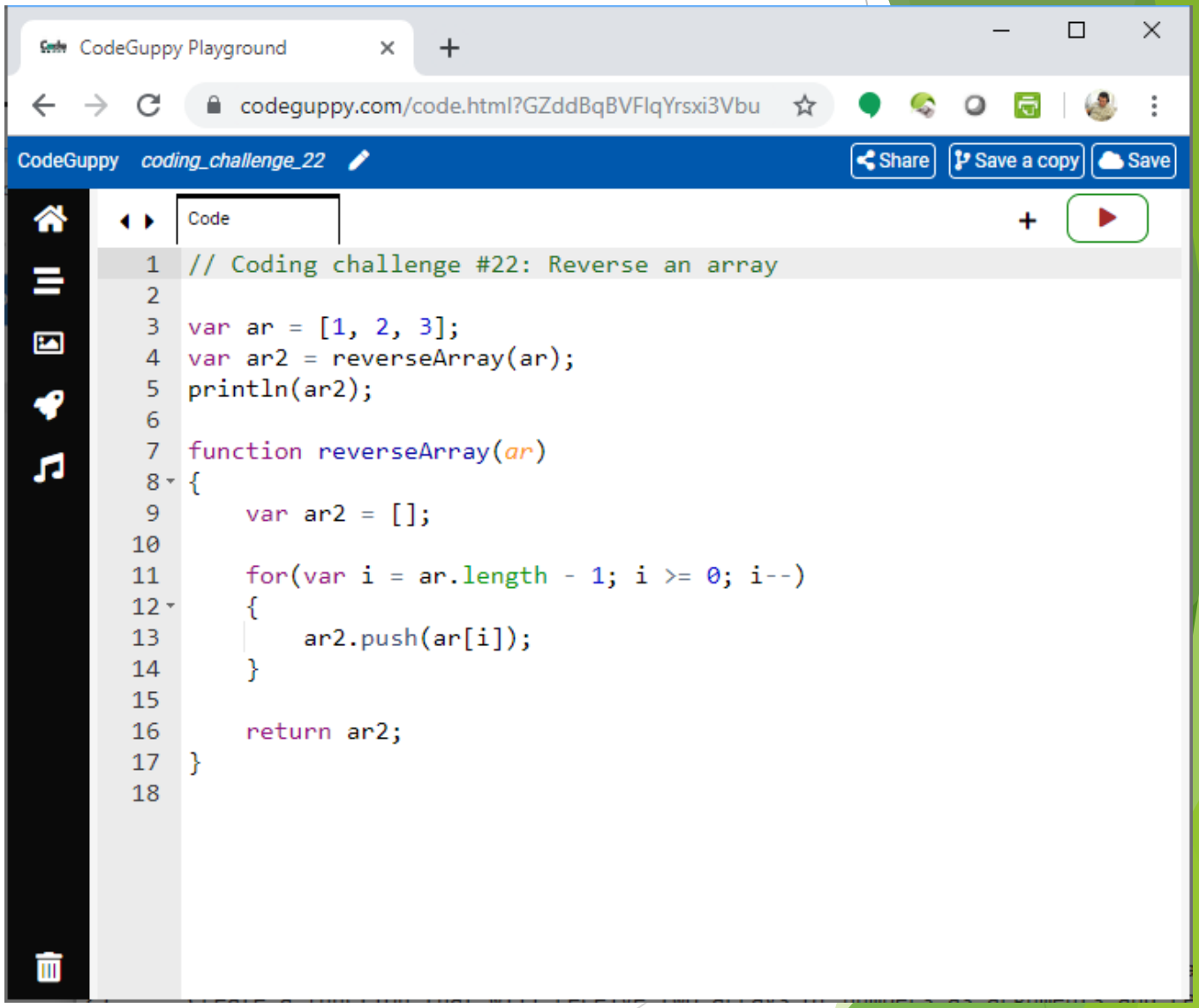
Coding challenge #21

Rotate an array to the right 1 position



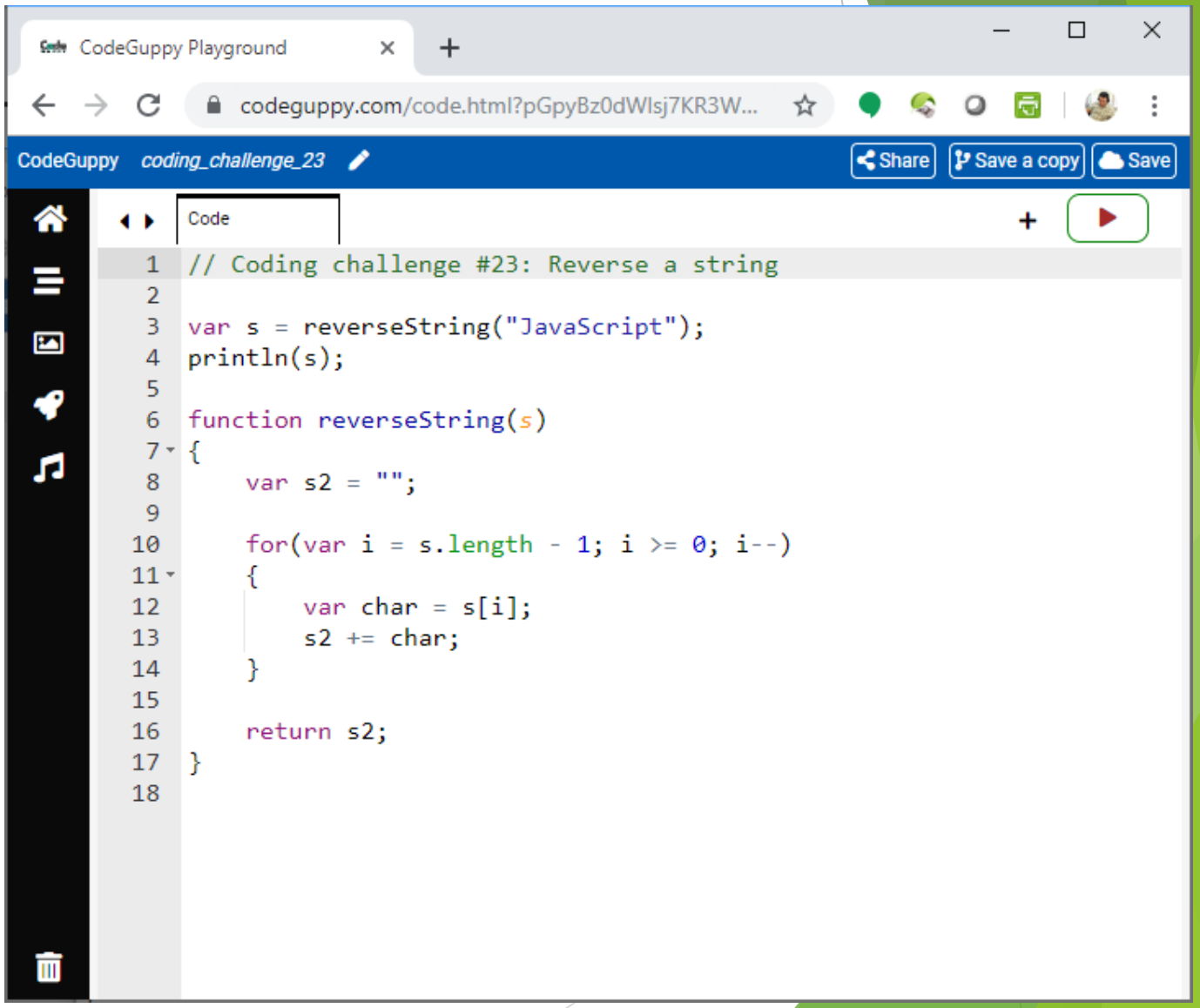
Coding challenge #22

Reverse an array



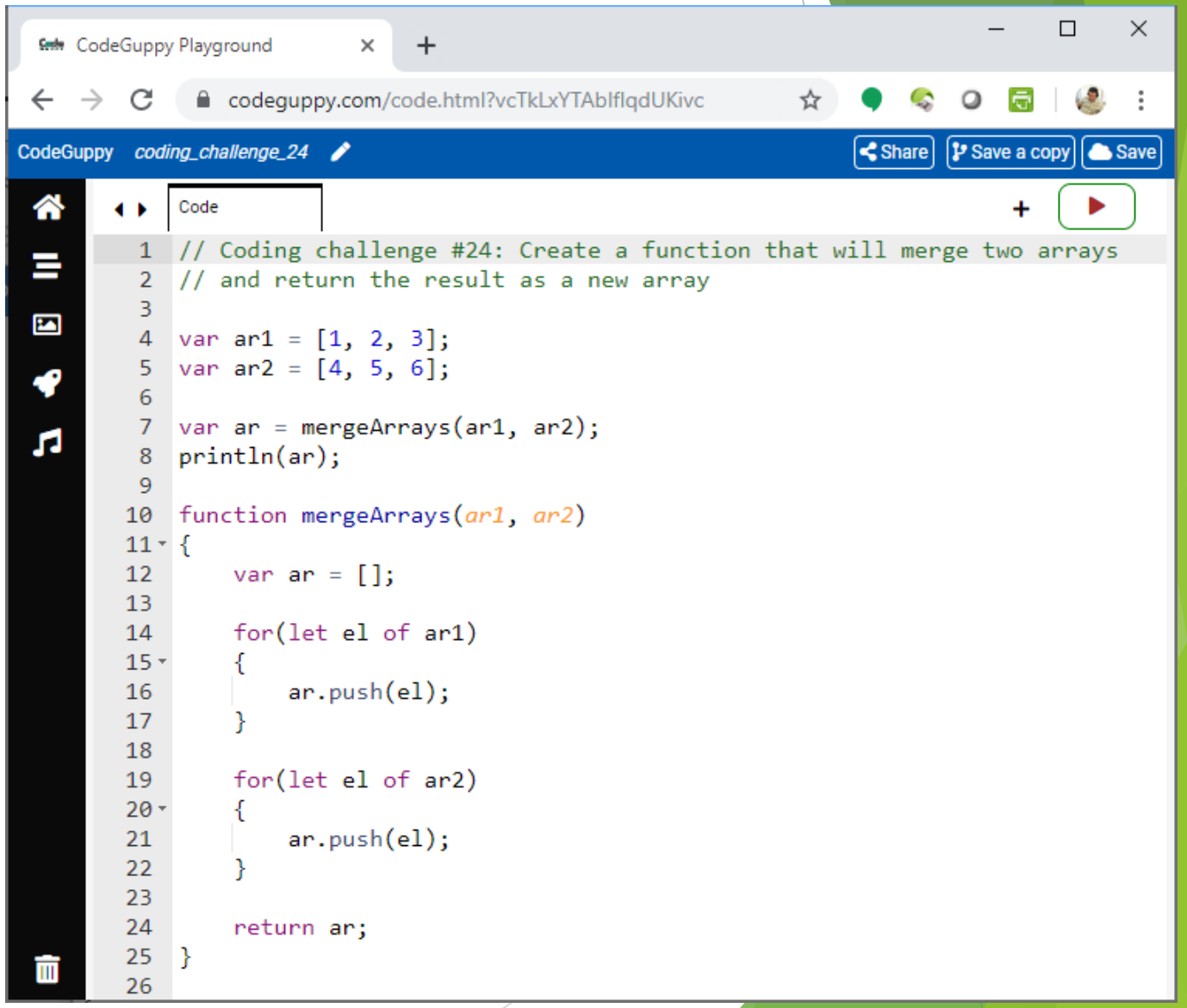
Coding challenge #23

Reverse a string



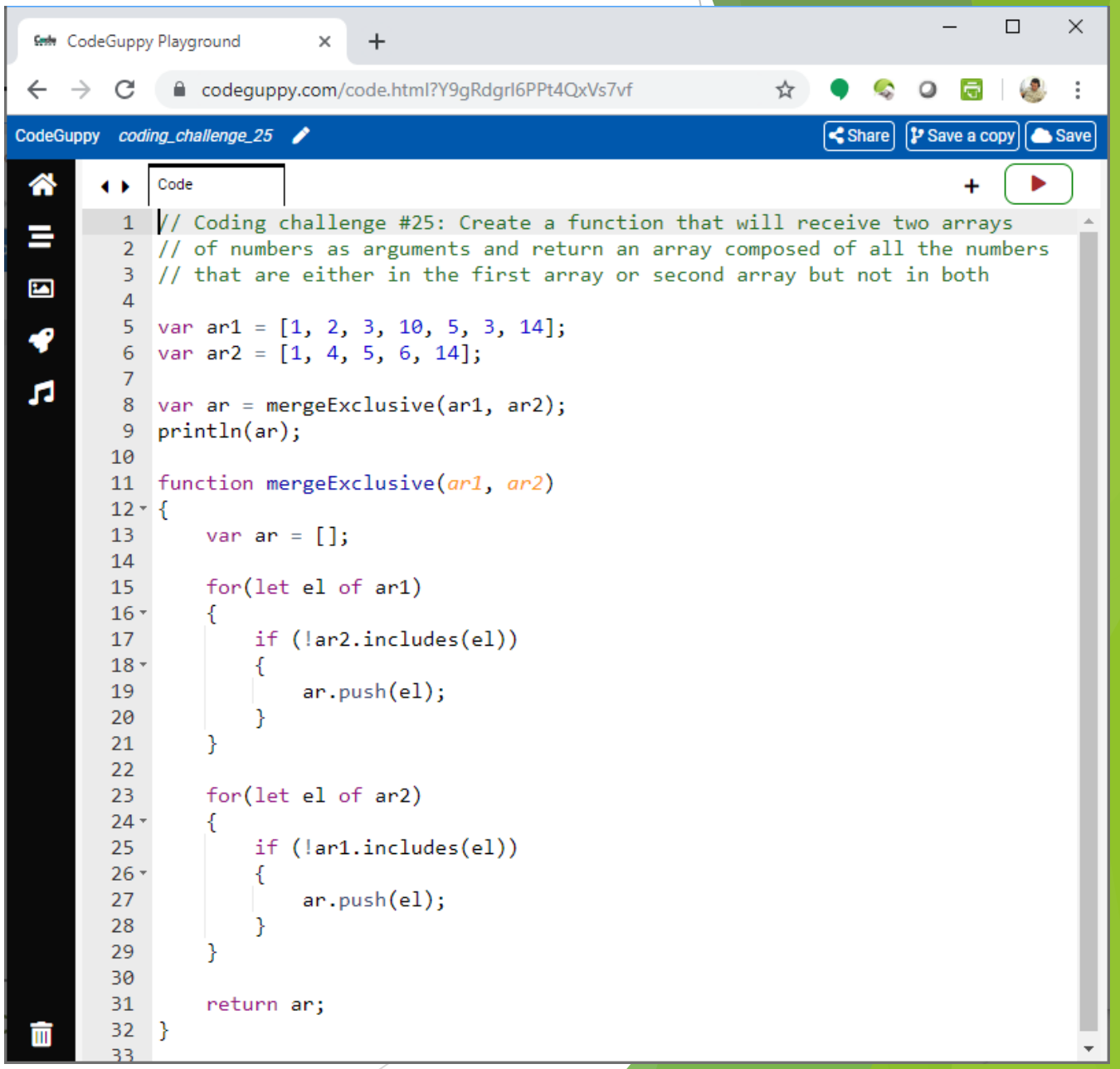
Coding challenge #24

Create a function that will merge two arrays and return the result as a new array



Coding challenge #25

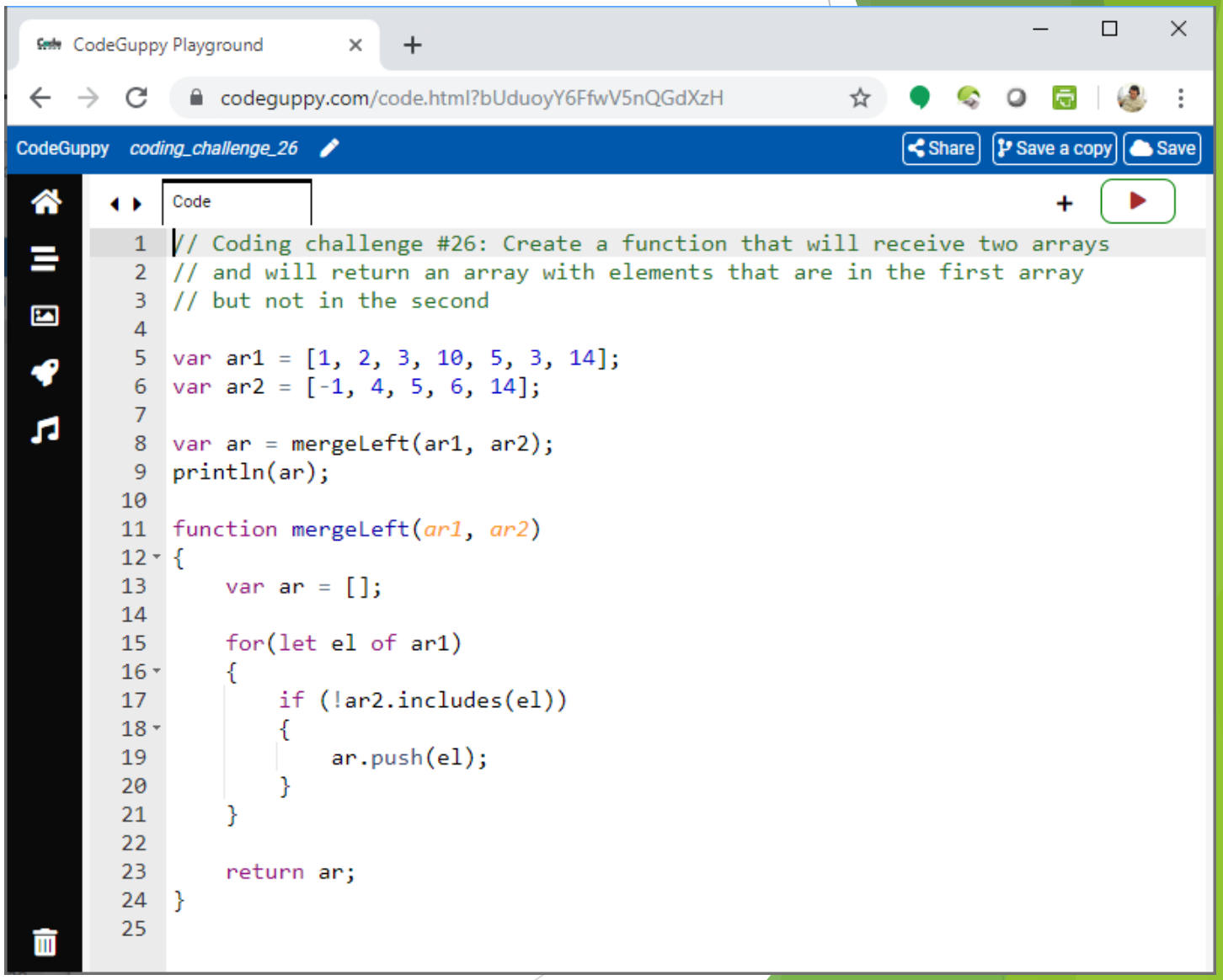
Create a function that will receive two arrays of numbers as arguments and return an array composed of all the numbers that are either in the first array or second array but not in both



codeguppy.com

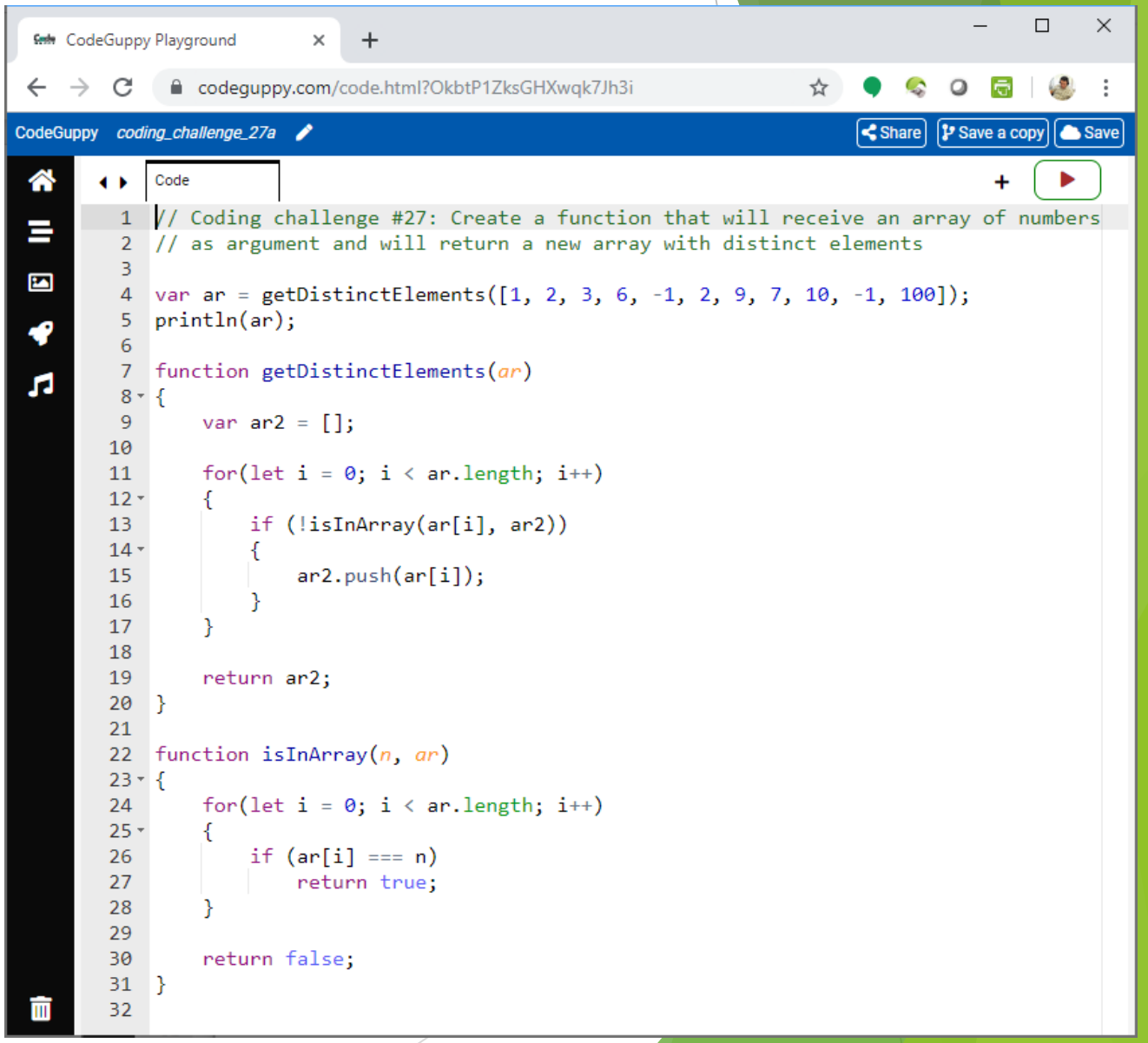
Coding challenge #26

Create a function that will receive two arrays and will return an array with elements that are in the first array but not in the second



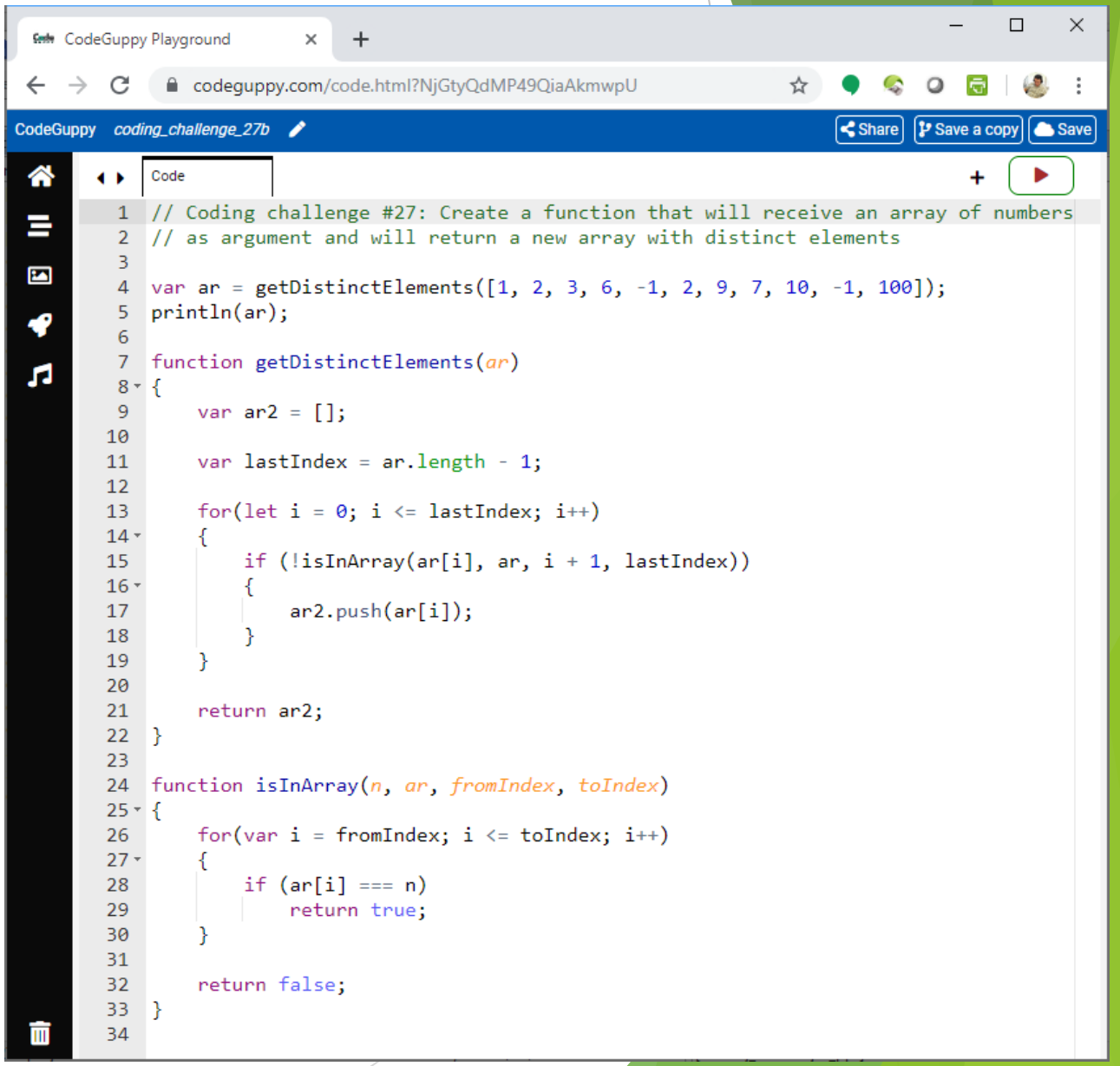
Coding challenge #27a

Create a function that will receive an array of numbers as argument and will return a new array with distinct elements

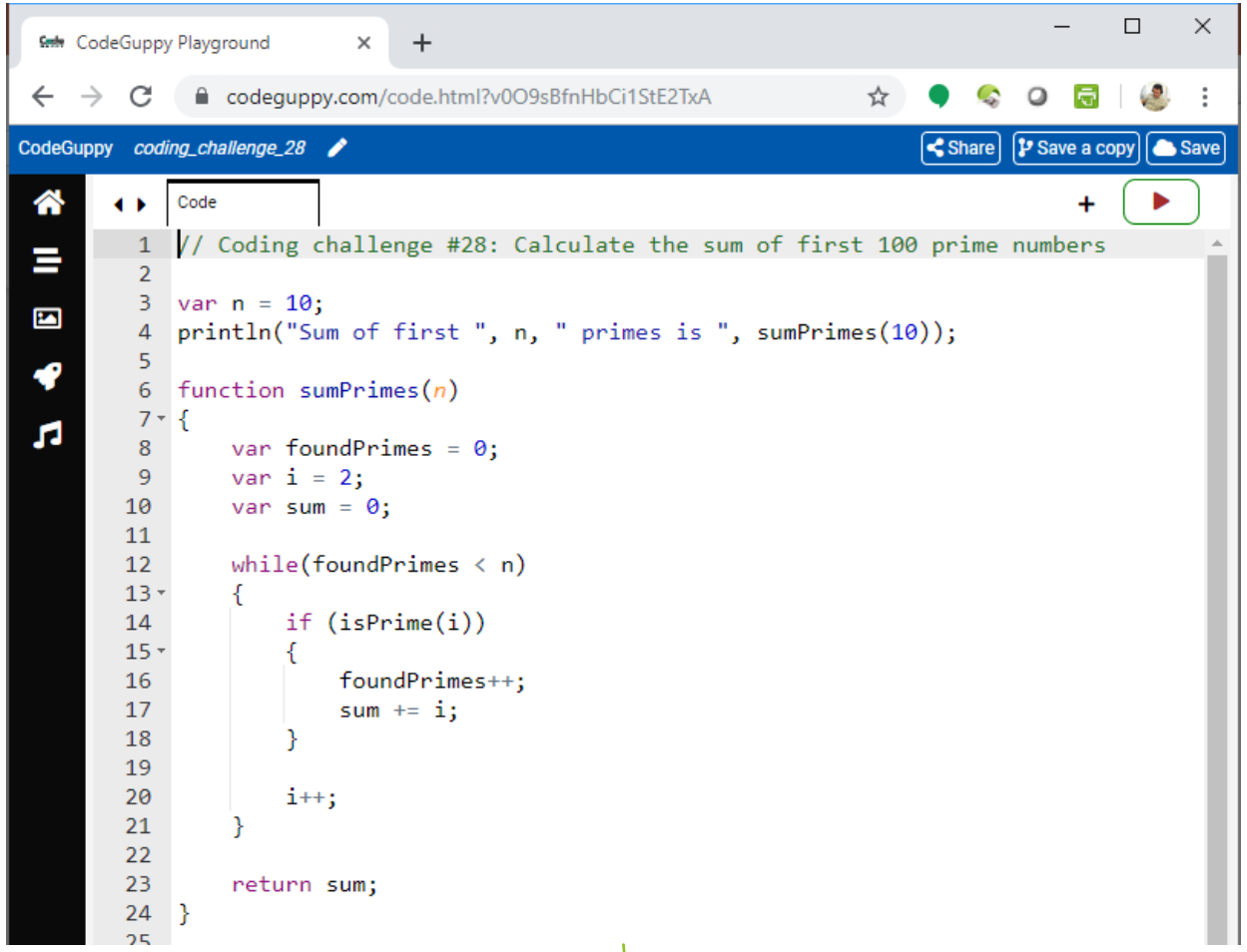


Coding challenge #27b

Create a function that will receive an array of numbers as argument and will return a new array with distinct elements



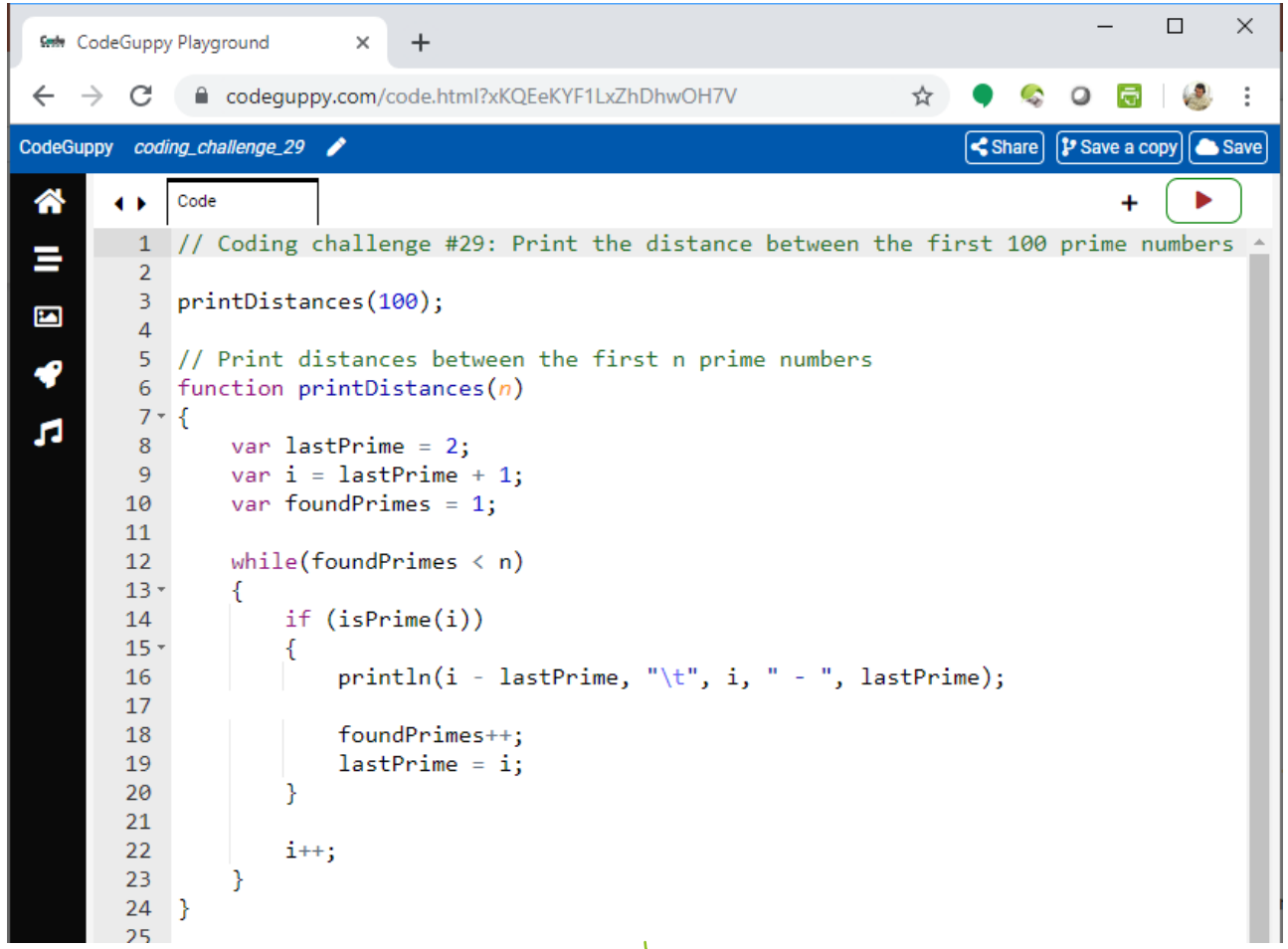
Coding challenge #28: Calculate the sum of first 100 prime numbers




```
// Coding challenge #28: Calculate the sum of first 100 prime numbers

var n = 10;
println("Sum of first ", n, " primes is ", sumPrimes(10));

function sumPrimes(n)
{
    var foundPrimes = 0;
    var i = 2;
    var sum = 0;

    while(foundPrimes < n)
    {
        if (isPrime(i))
        {
            foundPrimes++;
            sum += i;
        }

        i++;
    }

    return sum;
}
```

```
// Returns true if number n is prime
function isPrime(n)
{
    if (n < 2)
        return false;

    if (n == 2)
        return true;

    var maxDiv = Math.sqrt(n);

    for(var i = 2; i <= maxDiv; i++)
    {
        if (n % i === 0)
        {
            return false;
        }
    }

    return true;
}
```

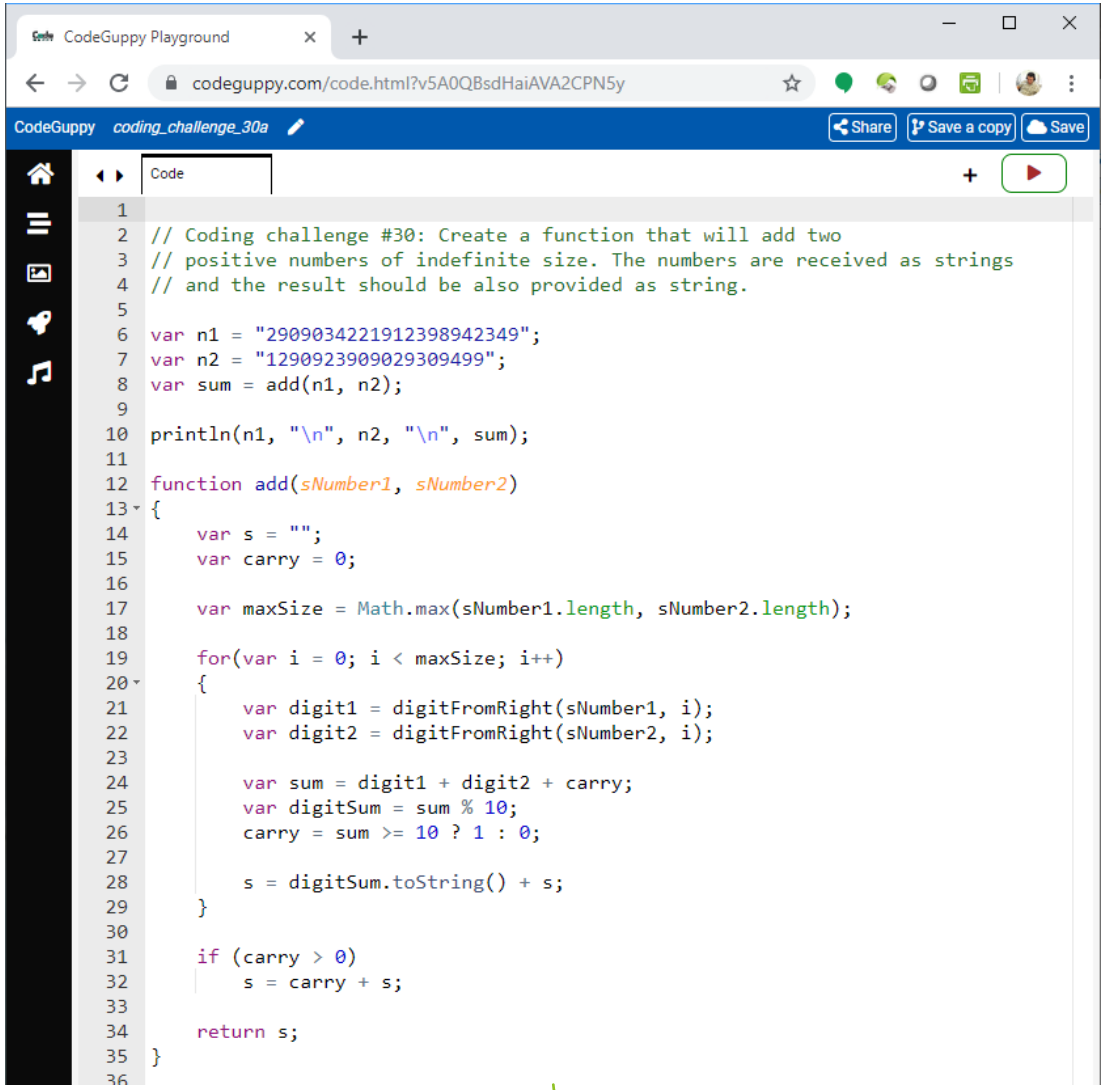
Coding challenge #29: Print the distance between the first 100 prime numbers




```
// Coding challenge #29: Print the distance between the first 100 prime numbers

printDistances(100);

// Print distances between the first n prime numbers
function printDistances(n)
{
    var lastPrime = 2;
    var i = lastPrime + 1;
    var foundPrimes = 1;

    while(foundPrimes < n)
    {
        if (isPrime(i))
        {
            println(i - lastPrime, "\t", i, " - ", lastPrime);

            foundPrimes++;
            lastPrime = i;
        }

        i++;
    }
}
```

```
// Returns true if number n is prime
function isPrime(n)
{
    if (n < 2)
        return false;

    if (n == 2)
        return true;

    var maxDiv = Math.sqrt(n);

    for(var i = 2; i <= maxDiv; i++)
    {
        if (n % i === 0)
        {
            return false;
        }
    }

    return true;
}
```

Coding challenge #30a: Create a function that will add two indefinite size numbers.

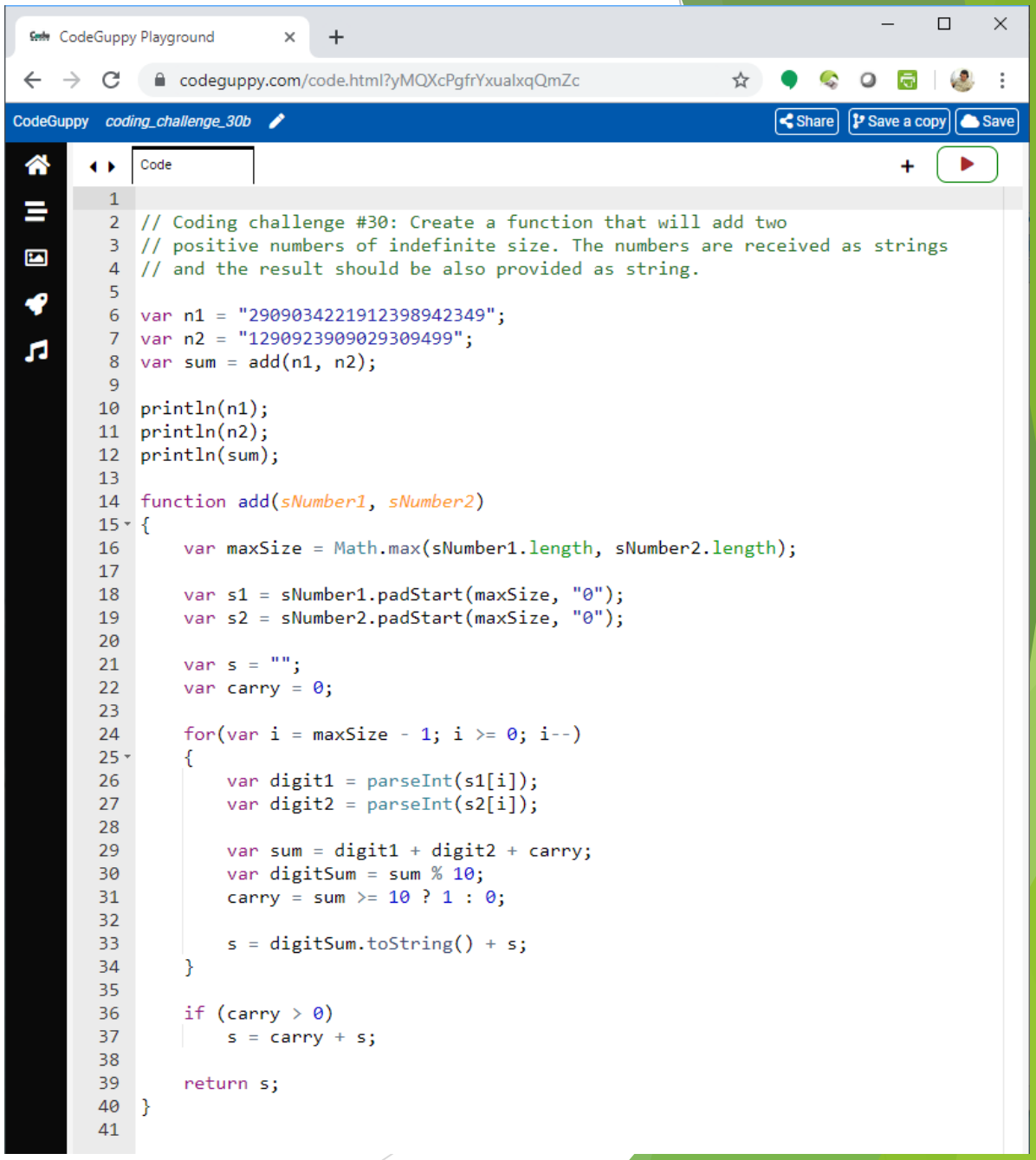



```
// Coding challenge #30: Create a function that will add two
// positive numbers of indefinite size. The numbers are received as strings
// and the result should be also provided as string.

var n1 = "2909034221912398942349";
var n2 = "1290923909029309499";
var sum = add(n1, n2);

println(n1, "\n", n2, "\n", sum);

function add(sNumber1, sNumber2)
{
    var s = "";
    var carry = 0;

    var maxSize = Math.max(sNumber1.length, sNumber2.length);

    for(var i = 0; i < maxSize; i++)
    {
        var digit1 = digitFromRight(sNumber1, i);
        var digit2 = digitFromRight(sNumber2, i);

        var sum = digit1 + digit2 + carry;
        var digitSum = sum % 10;
        carry = sum >= 10 ? 1 : 0;

        s = digitSum.toString() + s;
    }

    if (carry > 0)
        s = carry + s;

    return s;
}
```

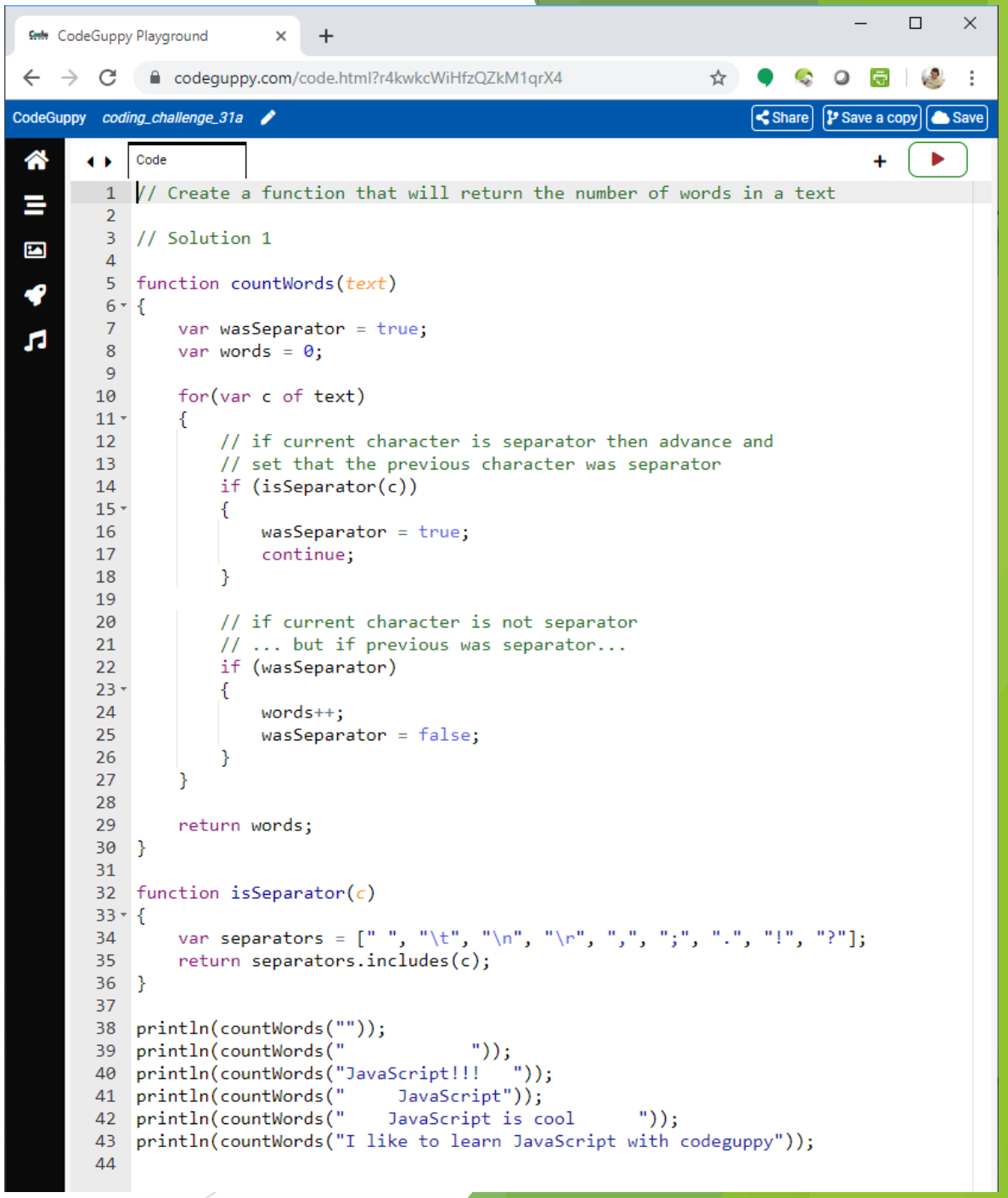
Requirements: Only positive numbers will be used and will be provided to the function as strings. The result should be also returned as a string.

```
function digitFromRight(s, digitNo)
{
    if (digitNo >= s.length)
        return 0;

    var char = s[ s.length - 1 - digitNo ];
    return parseInt(char);
}
```

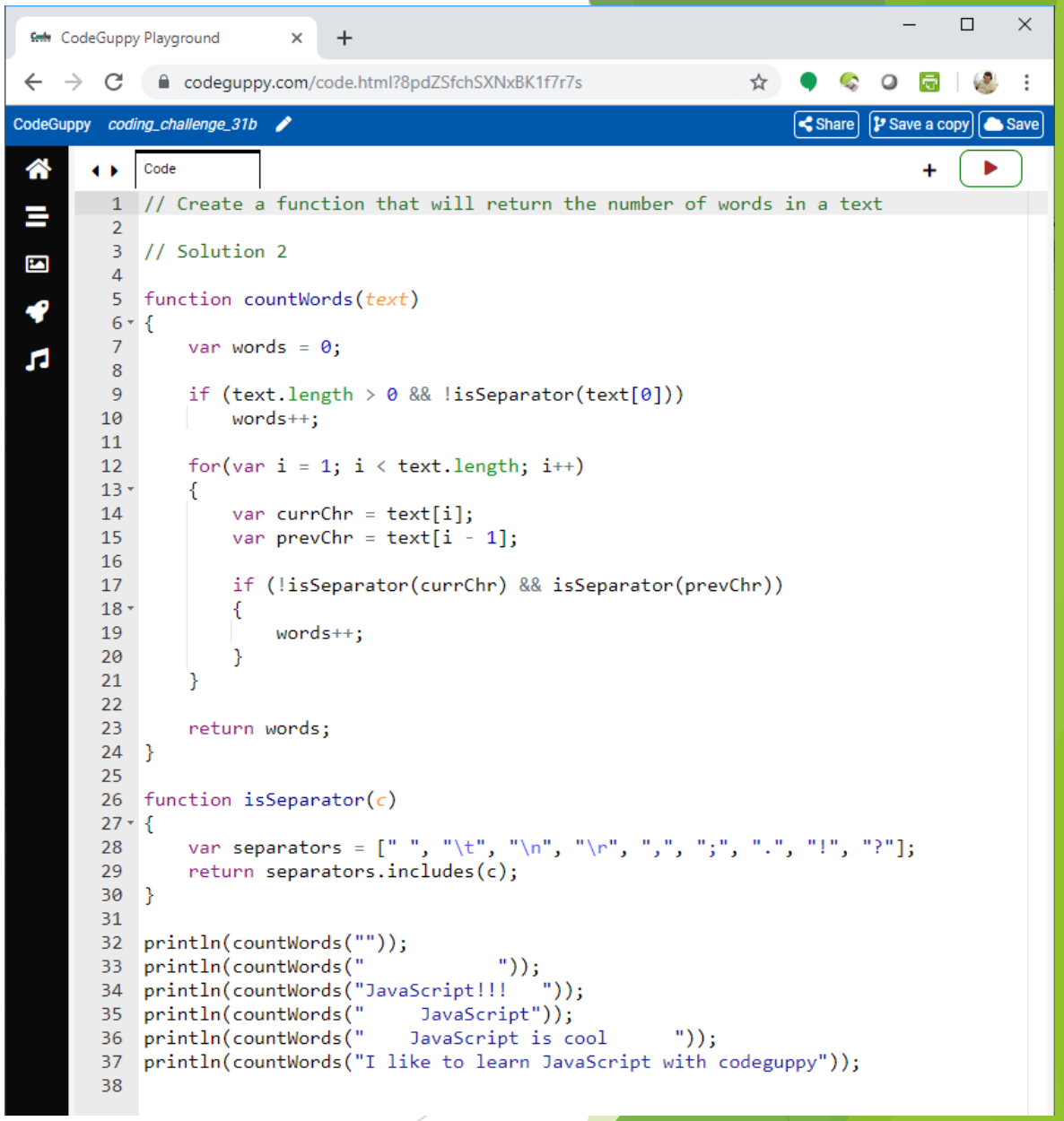
Coding challenge #30b: Create a function that will add two indefinite size numbers. Only positive numbers will be used and will be provided to the function as strings. The result should be also returned as a string.



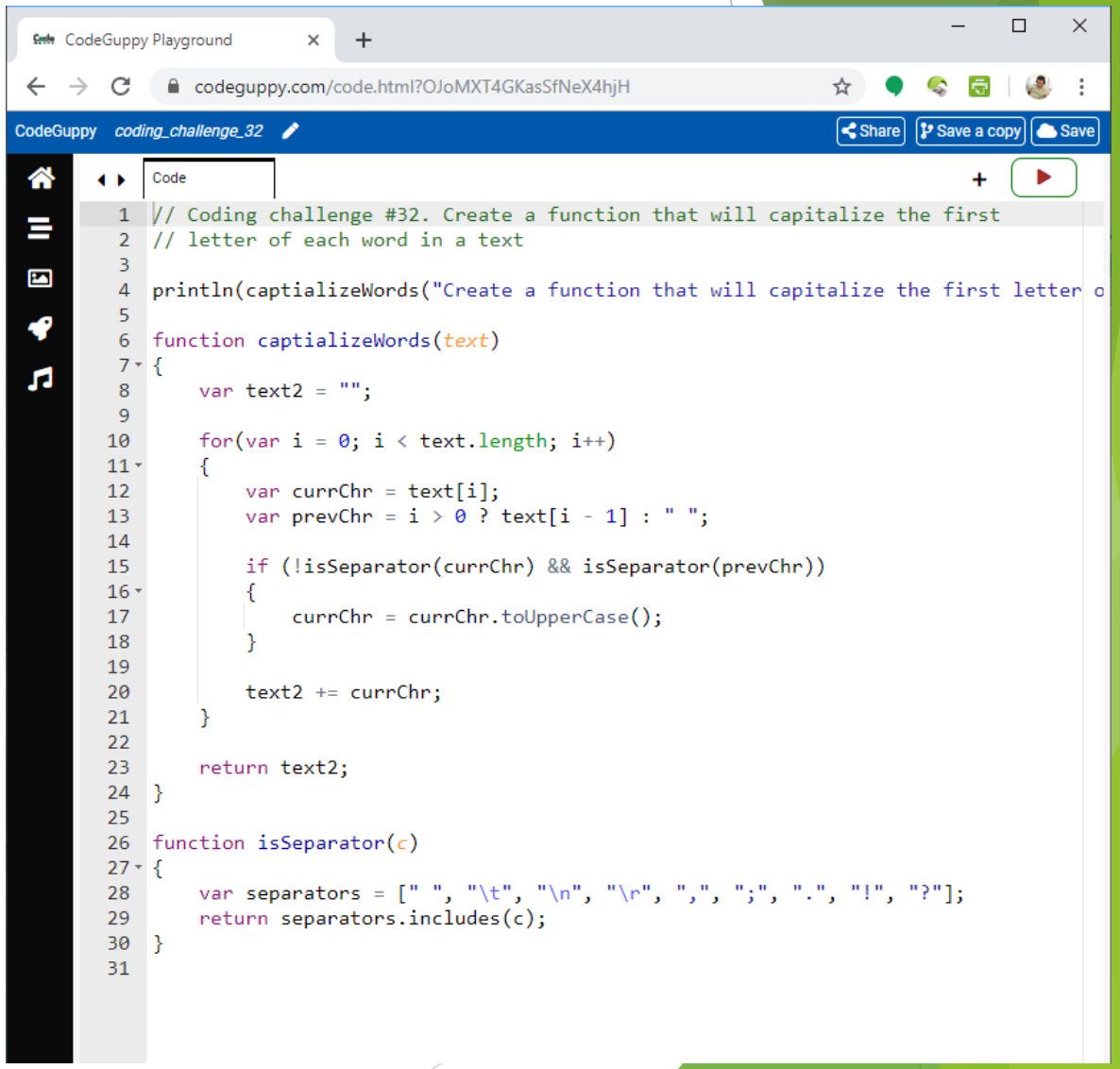
Coding challenge #31-a: Create a function that will return the number of words in a text



Coding challenge #31-b: Create a function that will return the number of words in a text

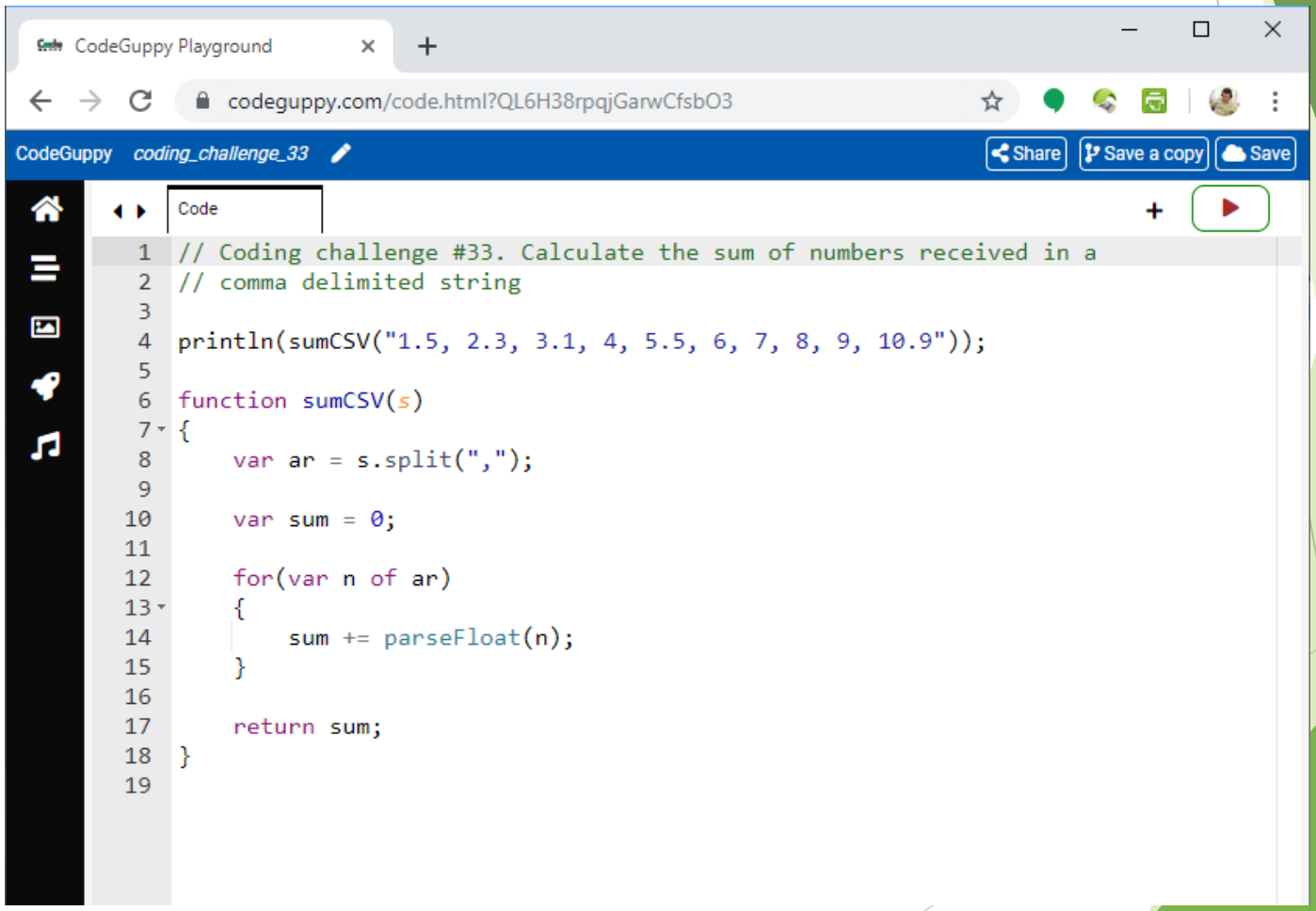


Coding challenge #32: Create a function that will capitalize the first letter of each word in a text

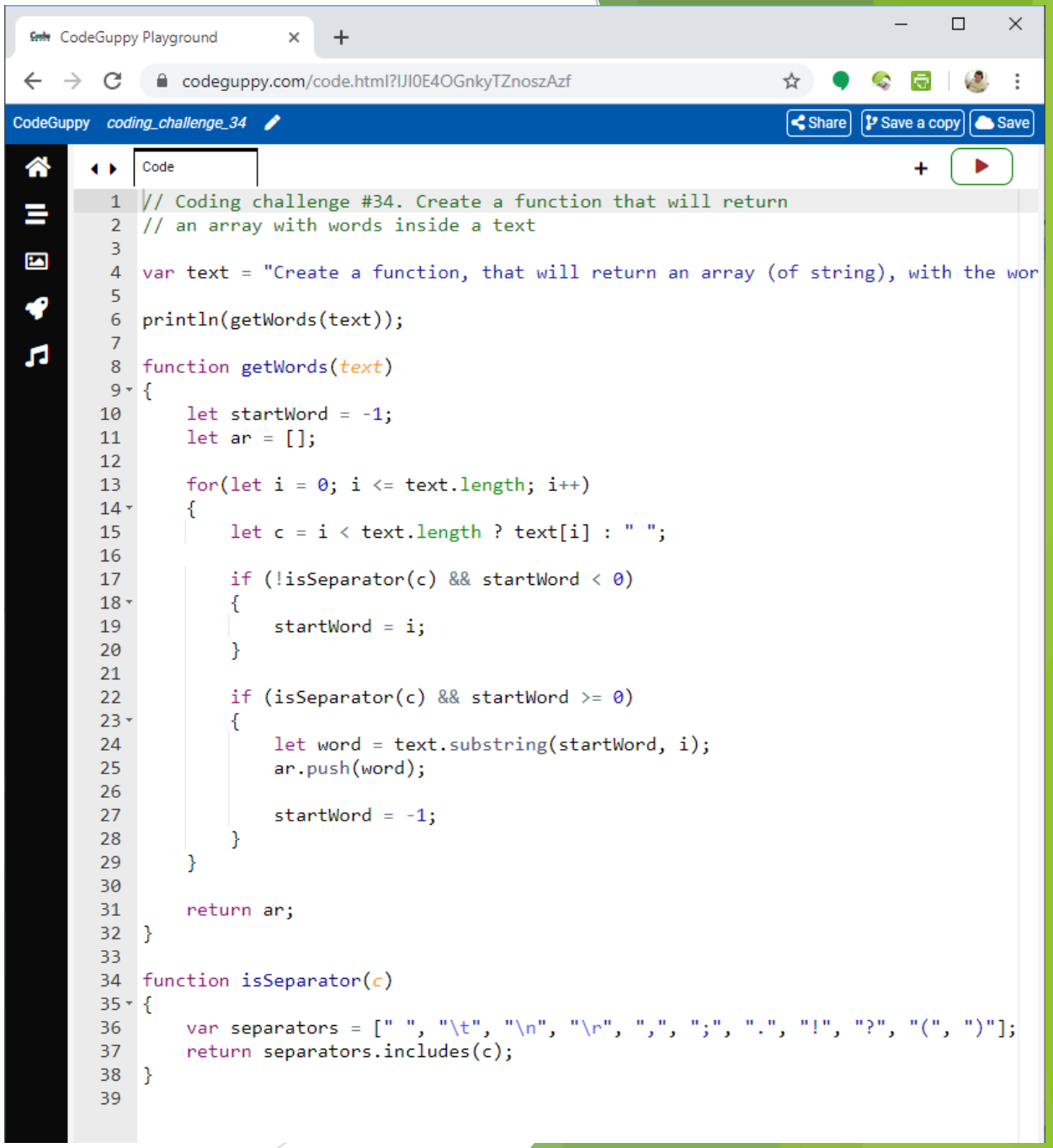


codeguppy.com

Coding challenge #33: Calculate the sum of numbers received in a comma delimited string

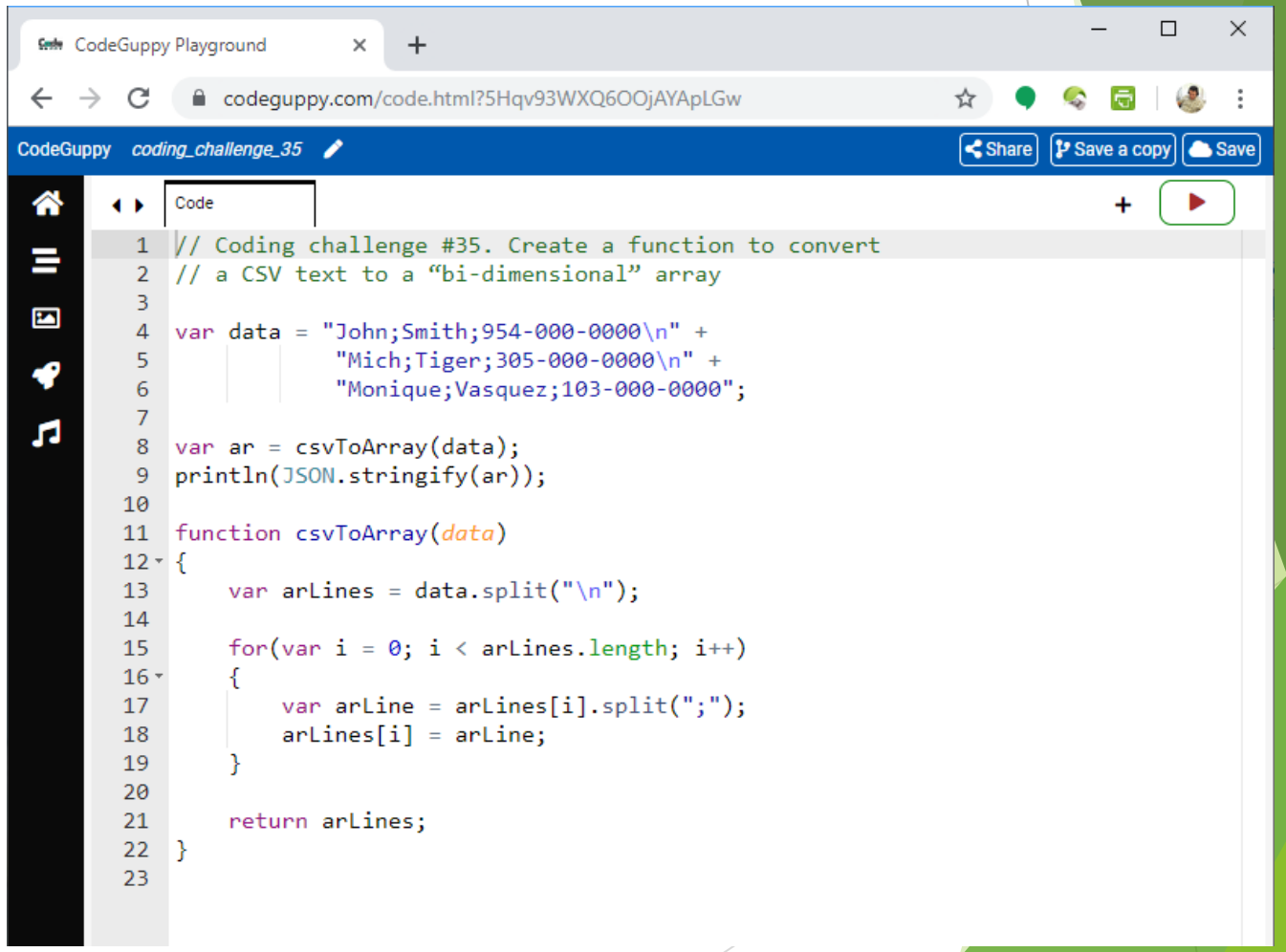


Coding challenge #34: Create a function that will return an array with words inside a text

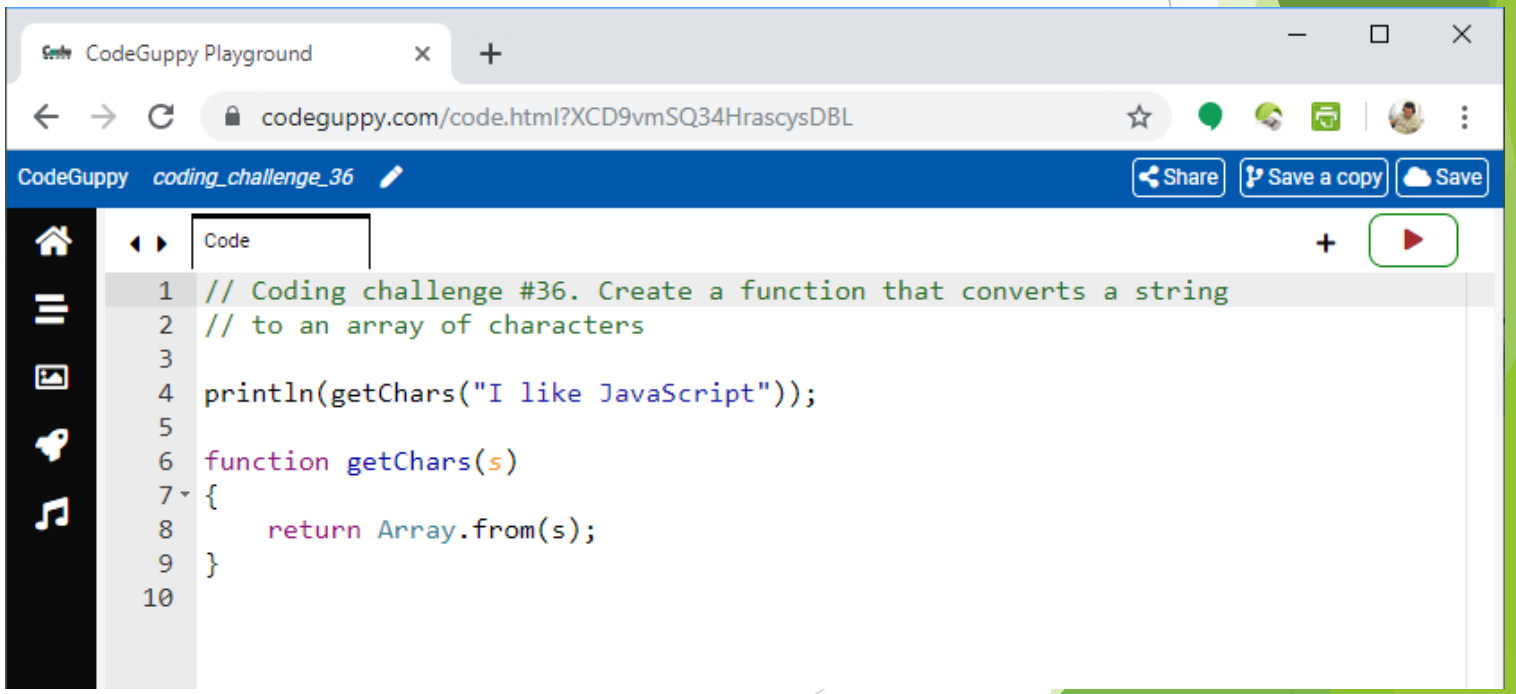


codeguppy.com

Coding challenge #35: Create a function to convert a CSV text to a “bi-dimensional” array

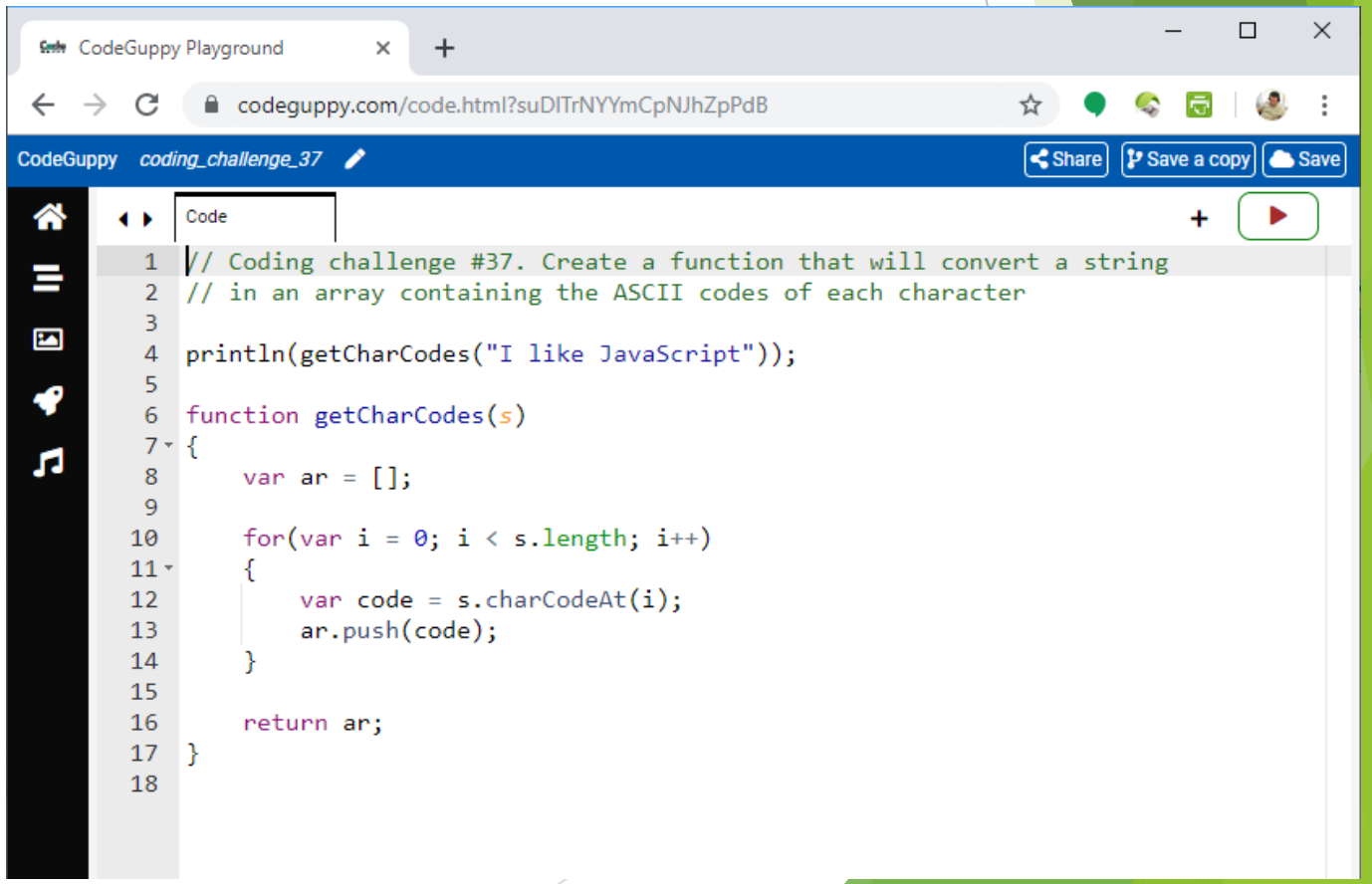


Coding challenge #36: Create a function that converts a string to an array of characters



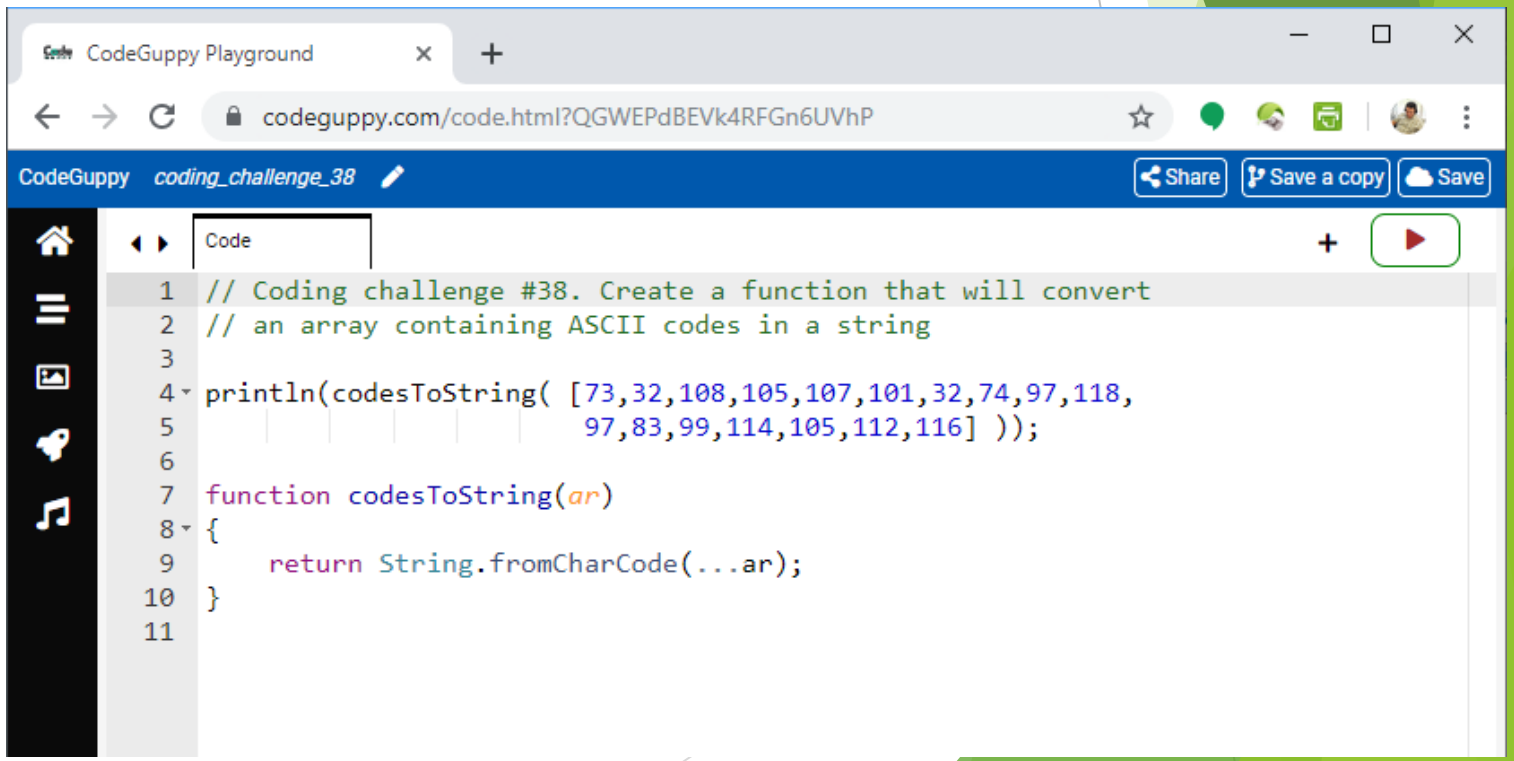
codeguppy.com

Coding challenge #37: Create a function that will convert a string in an array containing the ASCII codes of each character



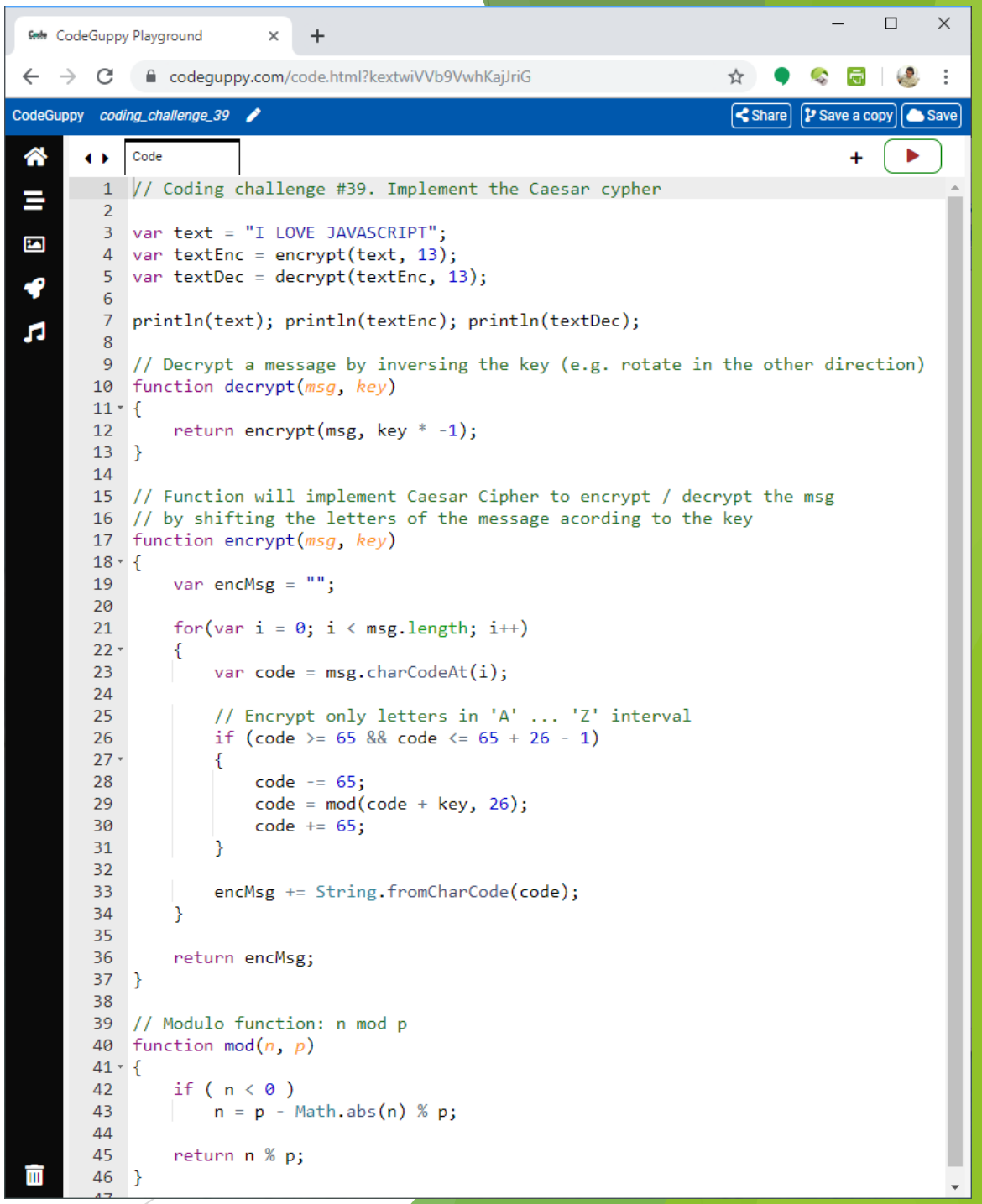
codeguppy.com

Coding challenge #38: Create a function that will convert an array containing ASCII codes in a string



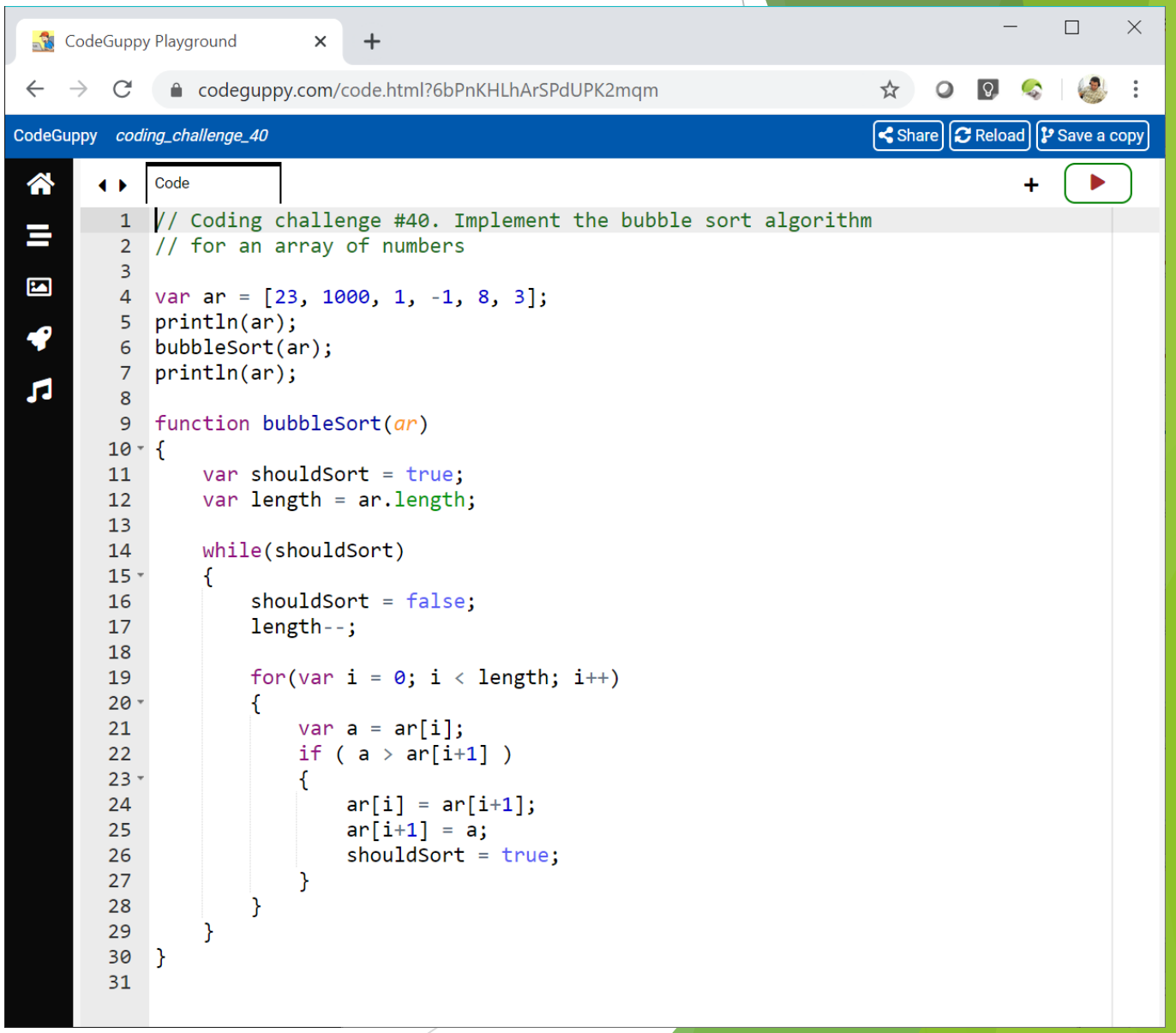
codeguppy.com

Coding challenge #39: Implement the Caesar cypher



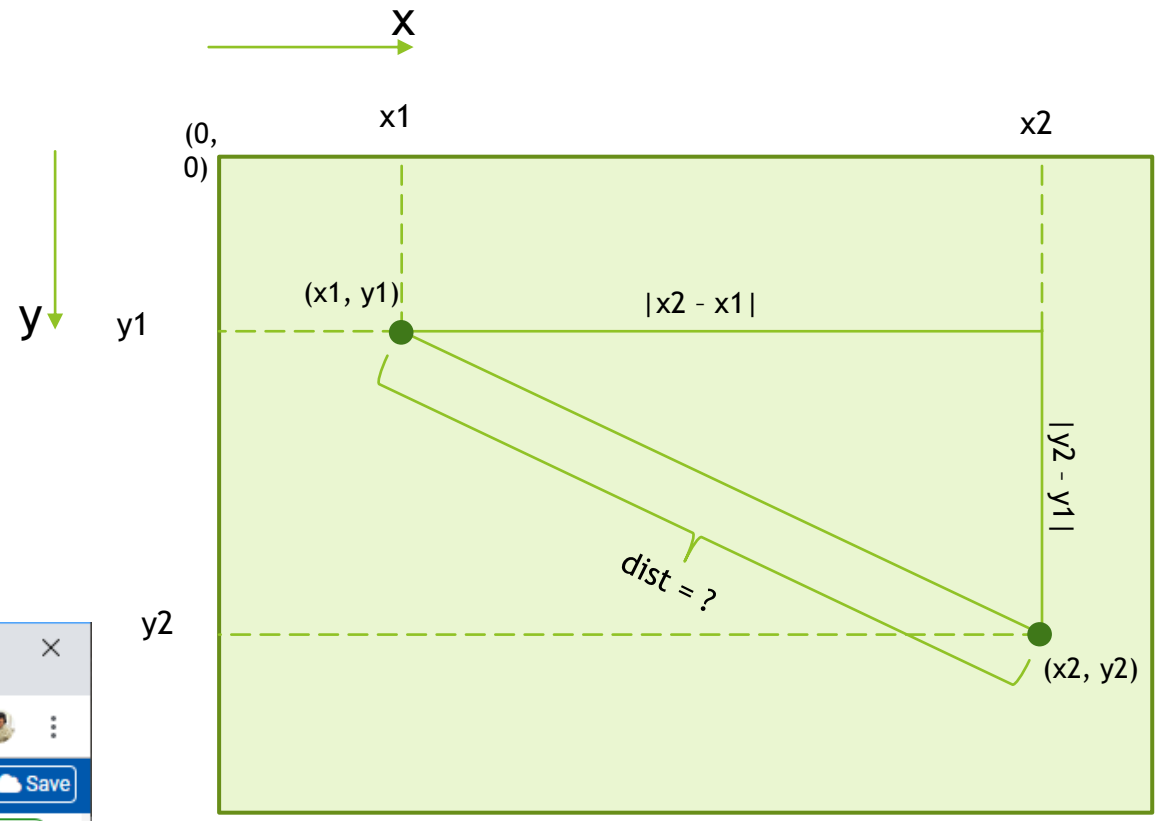
codeguppy.com

Coding challenge #40: Implement the bubble sort algorithm for an array of numbers



codeguppy.com

Coding challenge #41: Create a function to calculate the distance between two points defined by their x, y coordinates



CodeGuppy Playground

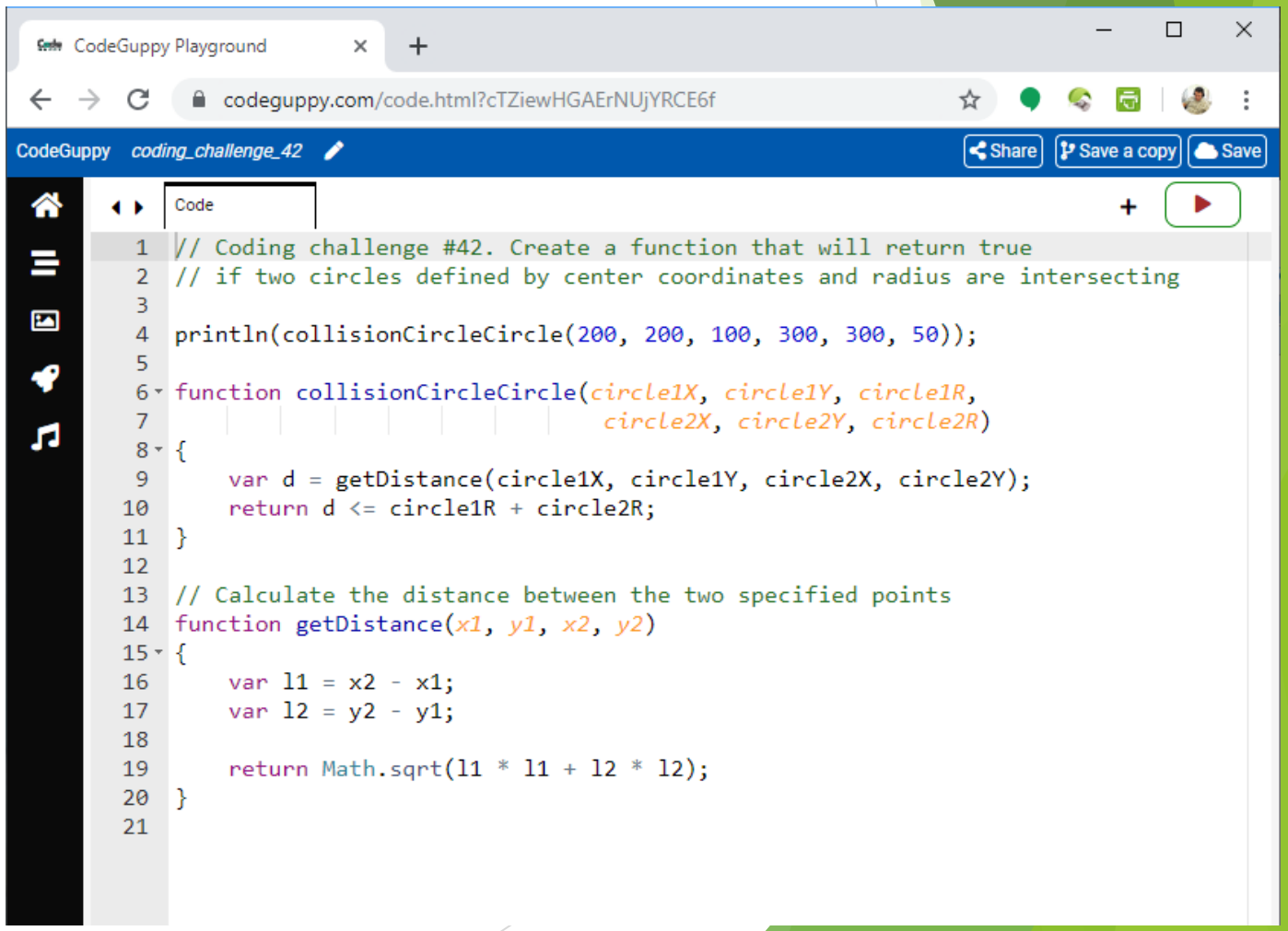
codeguppy.com/code.html?mnAuF3BjhDaFwBtDUnl4

CodeGuppy *coding_challenge_41* Share Save a copy Save

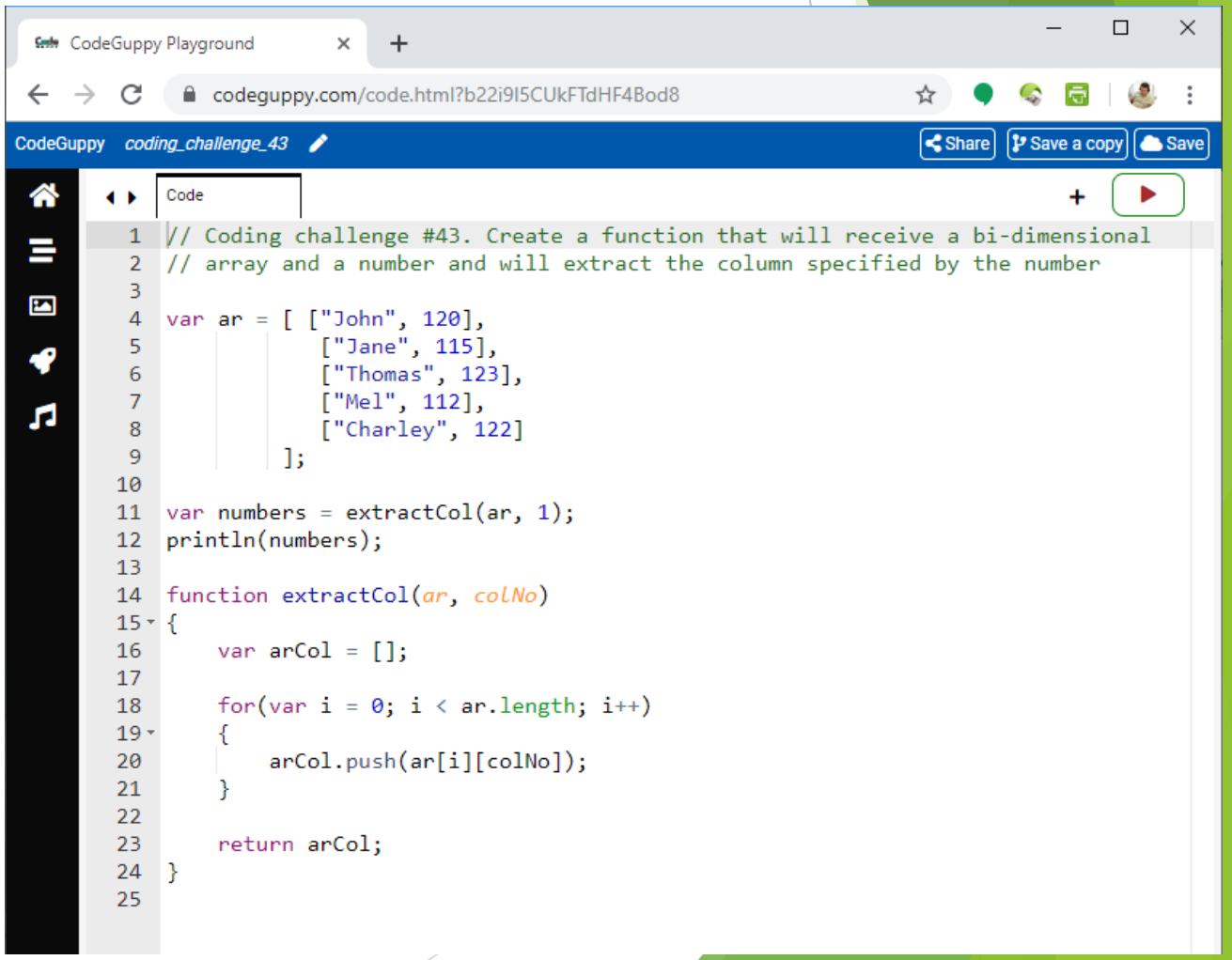
Code

```
// Coding challenge #41. Create a function to calculate the
// distance between two points defined by their x, y coordinates

println(getDistance(100, 100, 400, 300));

function getDistance(x1, y1, x2, y2)
{
    var l1 = x2 - x1;
    var l2 = y2 - y1;

    return Math.sqrt(l1 * l1 + l2 * l2);
}
```

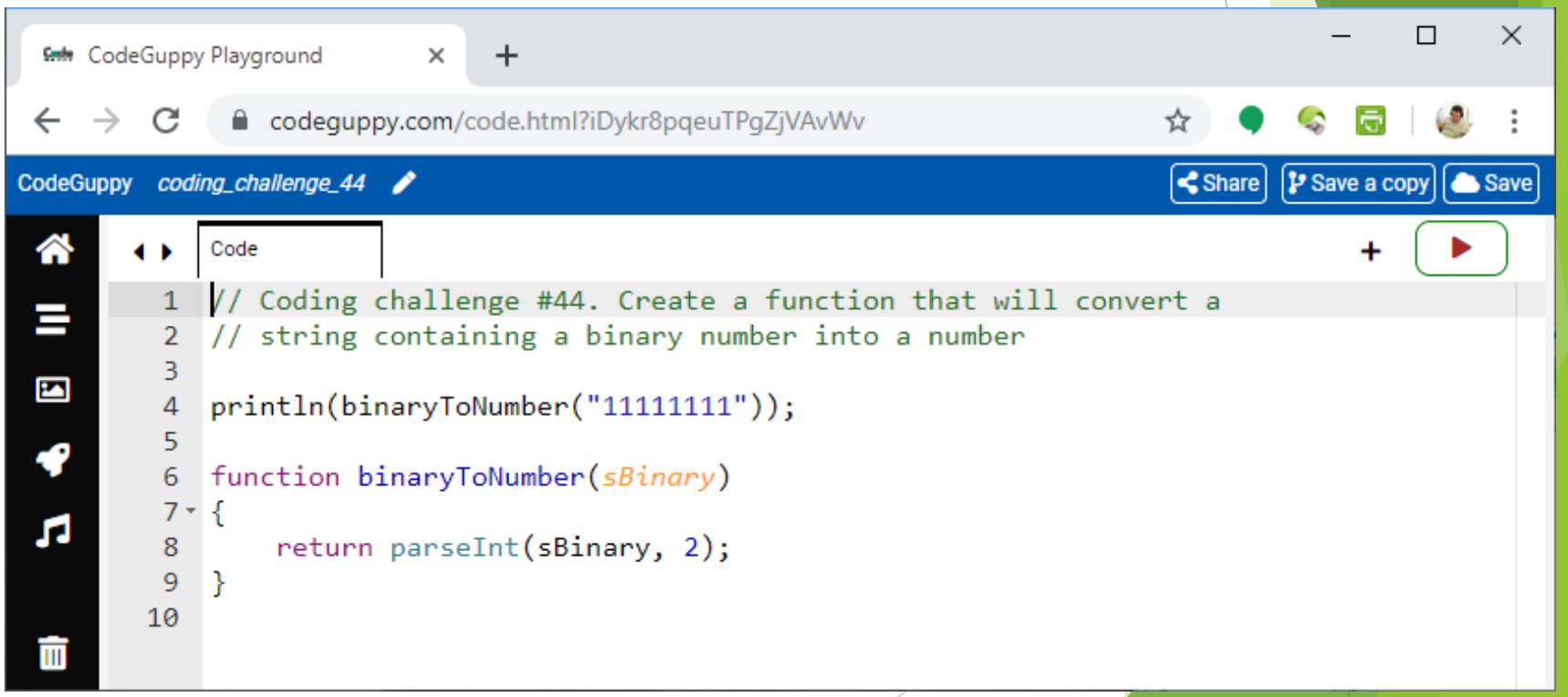
Coding challenge #42: Create a function that will return a Boolean value indicating if two circles defined by center coordinates and radius are intersecting



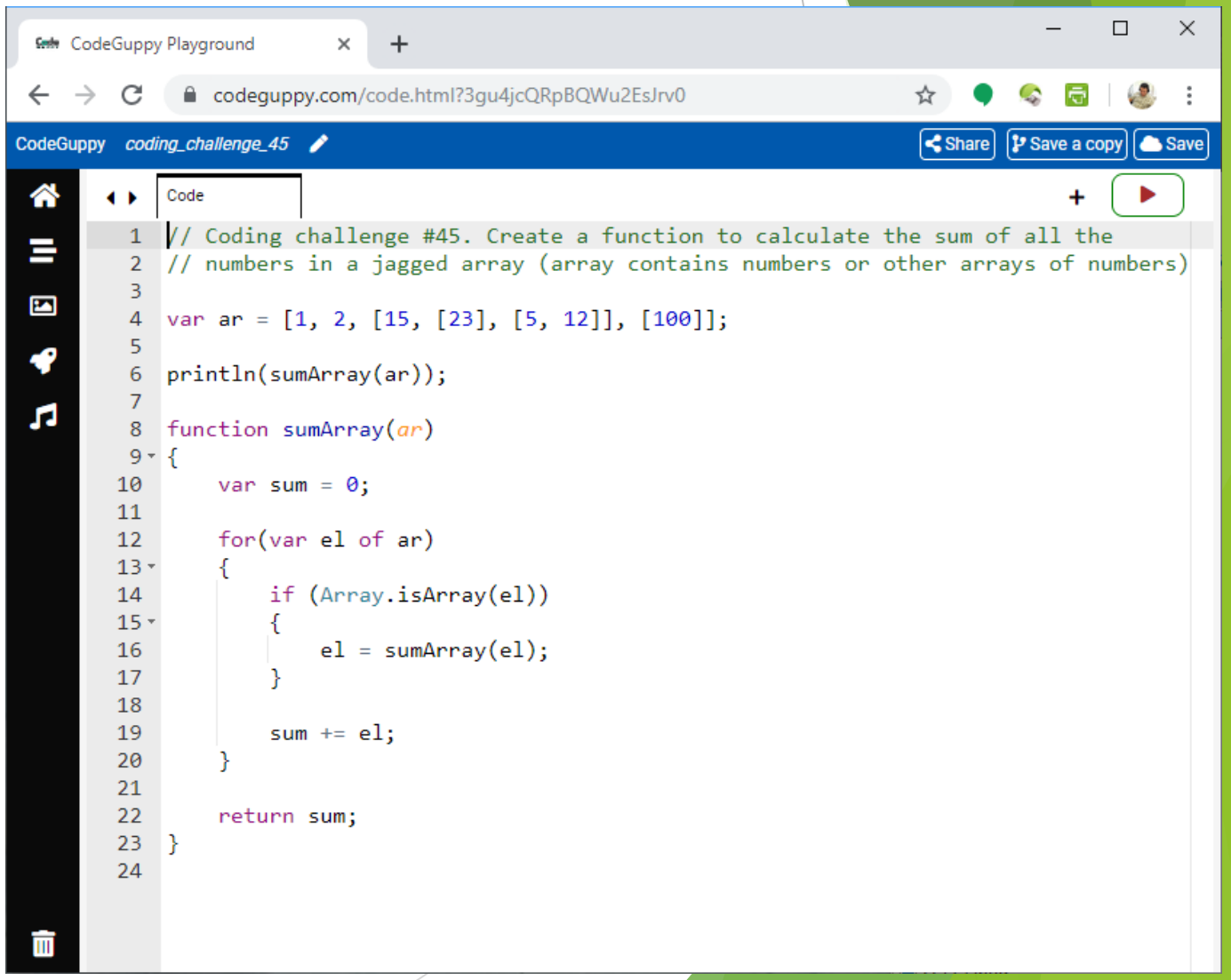
Coding challenge #43: Create a function that will receive a bi-dimensional array as argument and a number and will extract as a unidimensional array the column specified by the number



Coding challenge #44: Create a function that will convert a string containing a binary number into a number



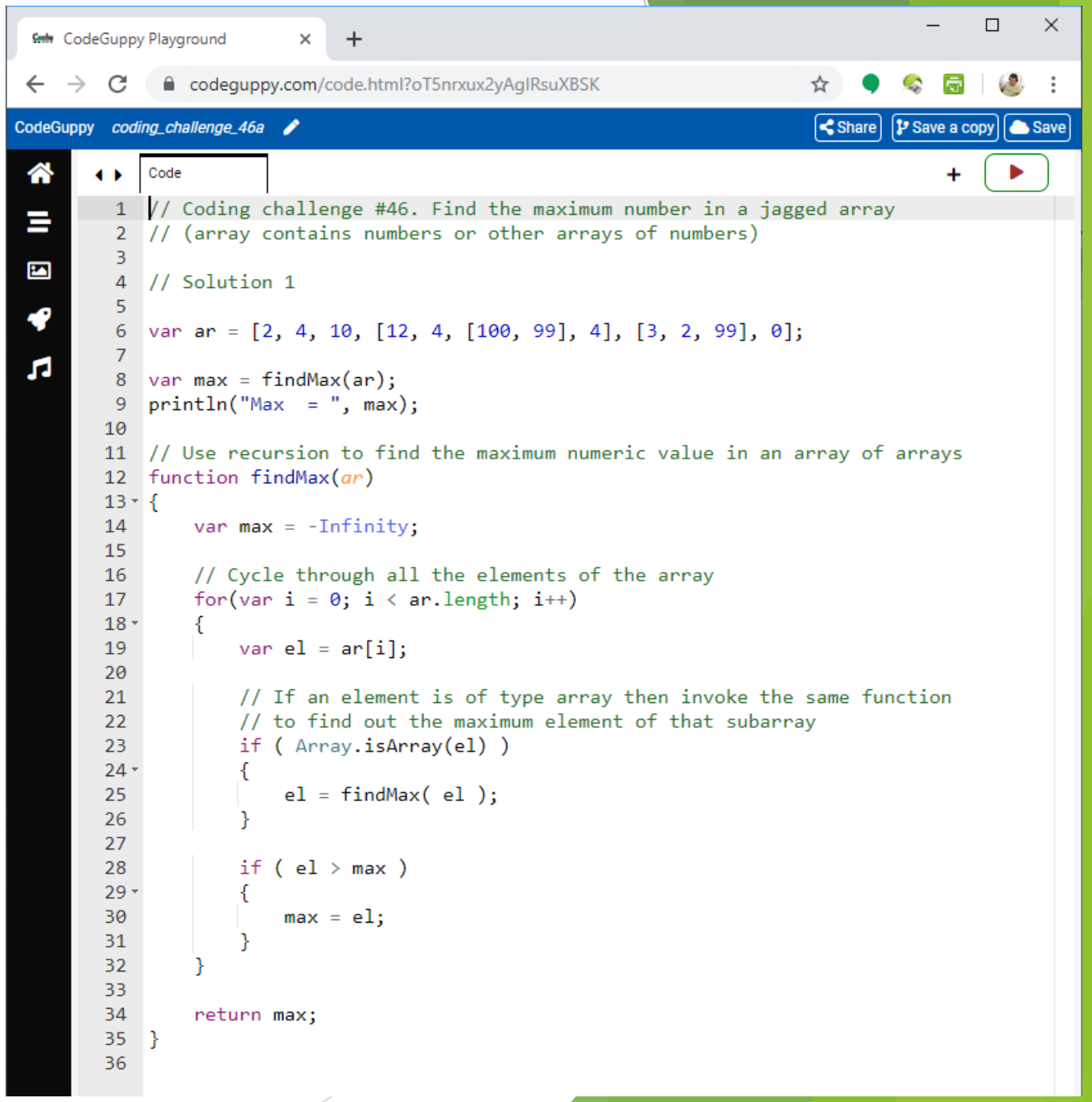
Coding challenge #45: Create a function to calculate the sum of all the numbers in a jagged array (array contains numbers or other arrays of numbers on an unlimited number of levels)



codeguppy.com

Coding challenge #46-a: Find the maximum number in a jagged array of numbers or array of numbers

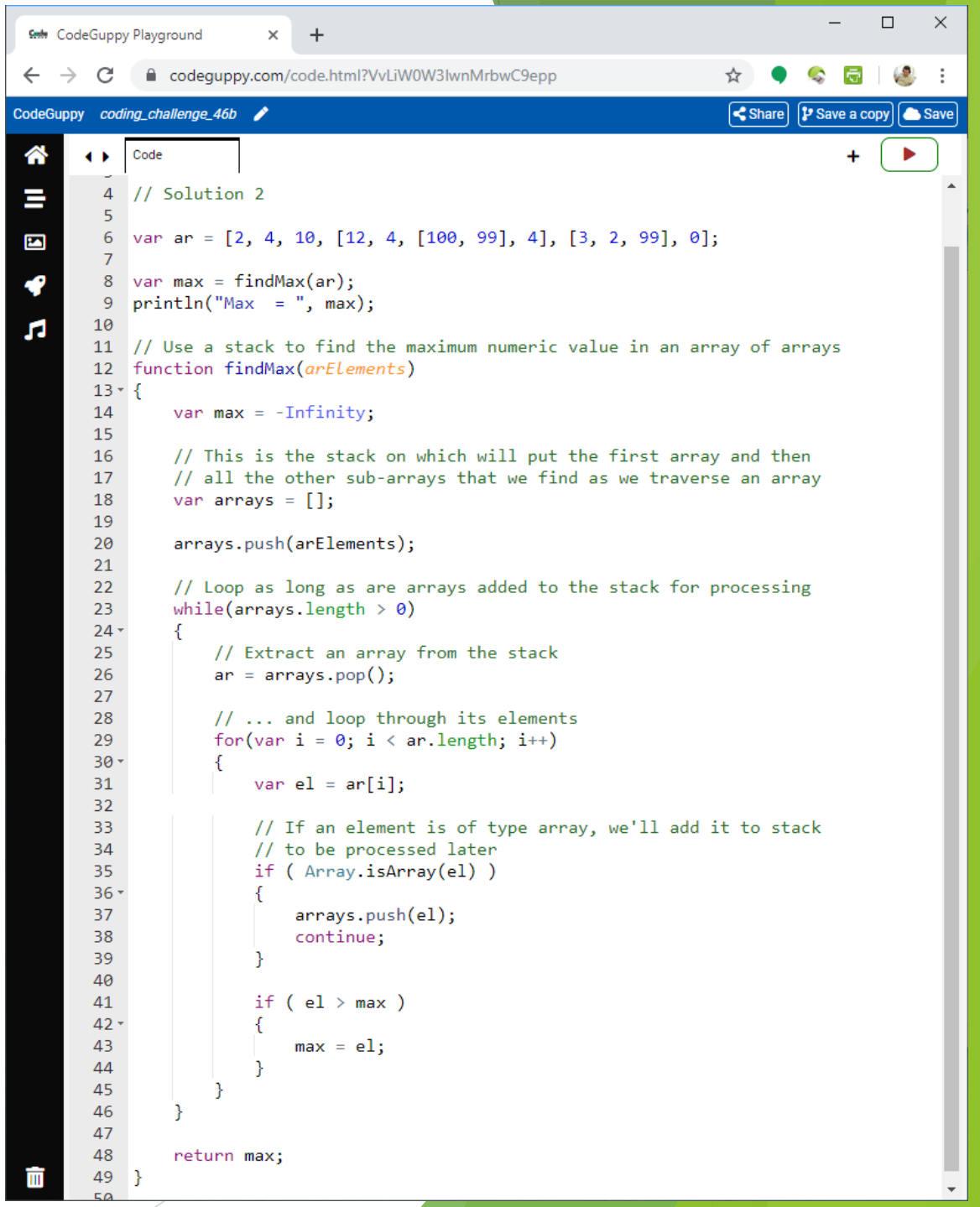
Requirements: Use recursion



codeguppy.com

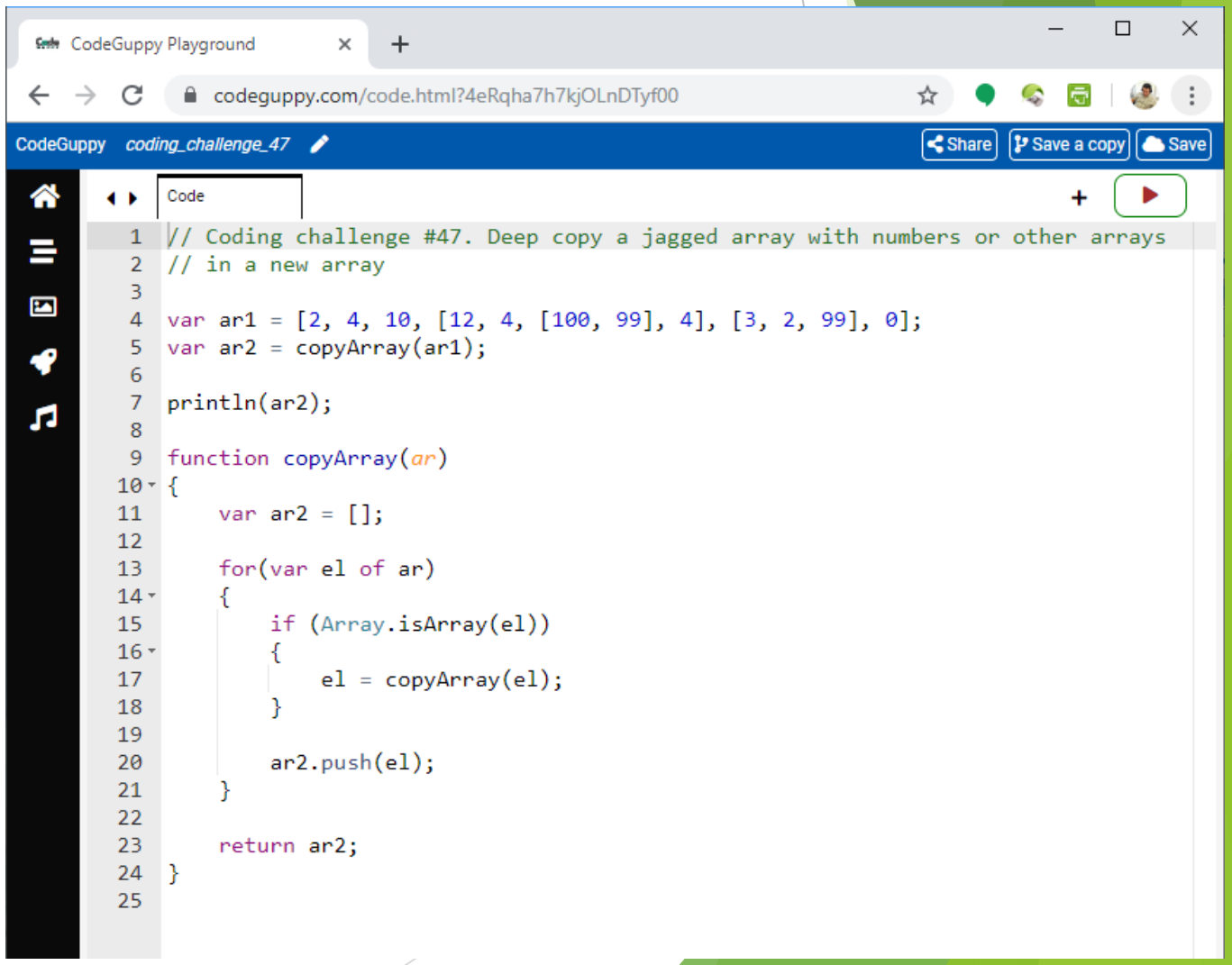
Coding challenge #46-b: Find the maximum number in a jagged array of numbers or array of numbers

Requirements: Do not use recursion



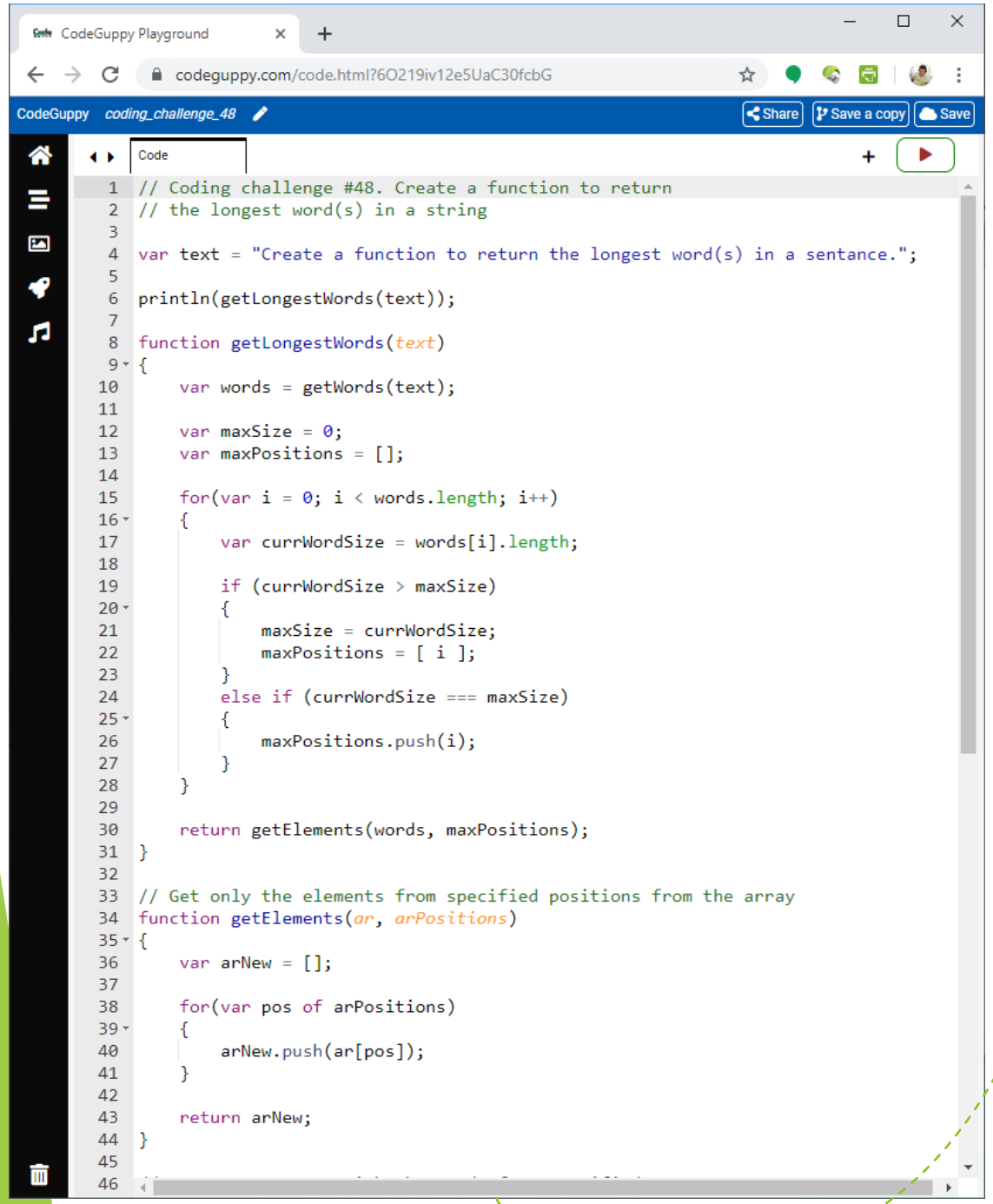
codeguppy.com

Coding challenge #47: Deep copy a jagged array with numbers or other arrays in a new array



codeguppy.com

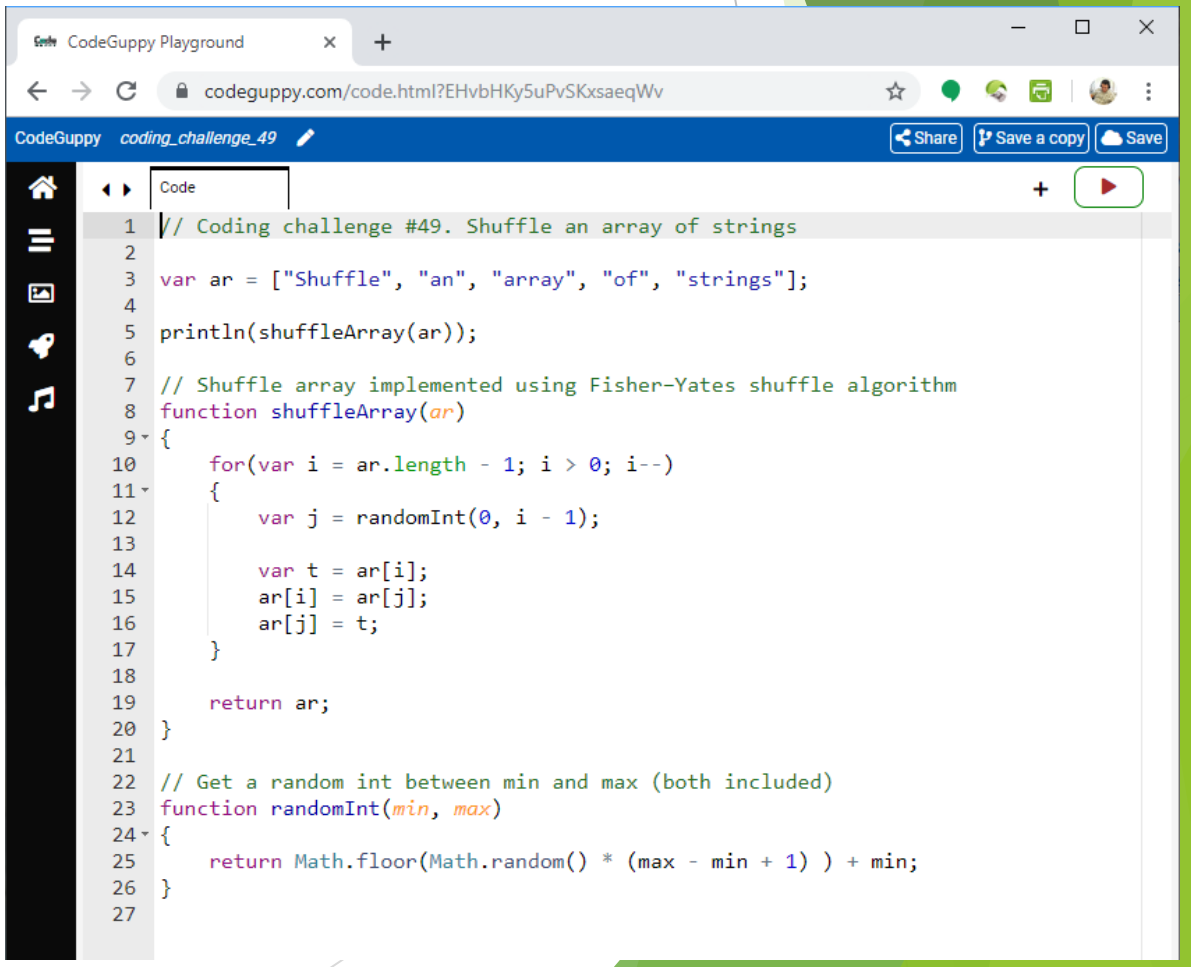
Coding challenge #48: Create a function to return the longest word(s) in a string




```
// Coding challenge #48. Create a function to return
// the longest word(s) in a string

var text = "Create a function to return the longest word(s) in a sentance.";

println(getLongestWords(text));

function getLongestWords(text)
{
    var words = getWords(text);

    var maxSize = 0;
    var maxPositions = [];

    for(var i = 0; i < words.length; i++)
    {
        var currWordSize = words[i].length;

        if (currWordSize > maxSize)
        {
            maxSize = currWordSize;
            maxPositions = [ i ];
        }
        else if (currWordSize === maxSize)
        {
            maxPositions.push(i);
        }
    }

    return getElements(words, maxPositions);
}

// Get only the elements from specified positions from the array
function getElements(ar, arPositions)
{
    var arNew = [];

    for(var pos of arPositions)
    {
        arNew.push(ar[pos]);
    }

    return arNew;
}
```

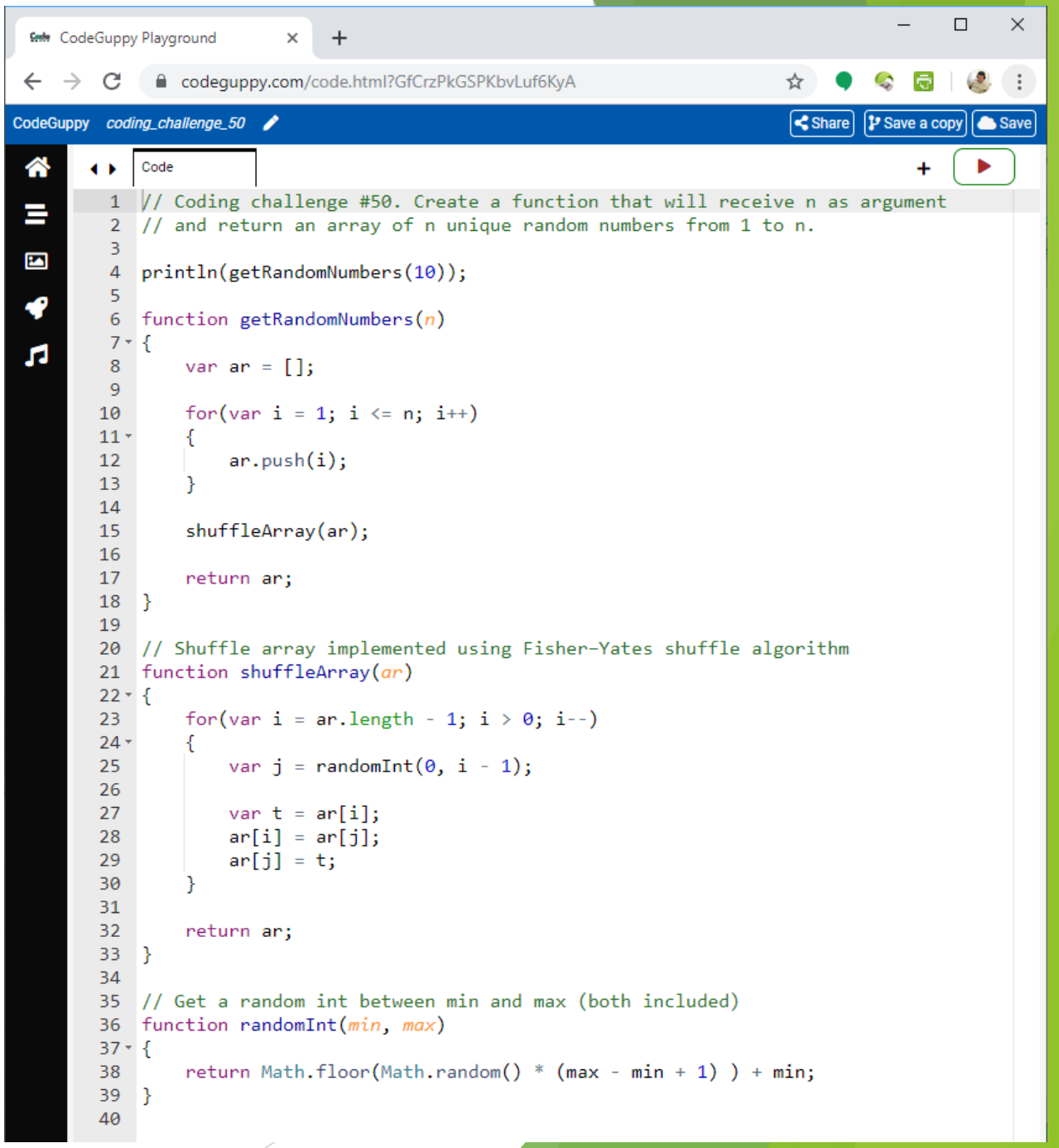
```
// Returns an array with the words from specified text
function getWords(text)
{
    let startWord = -1;
    let ar = [];

    for(let i = 0; i <= text.length; i++)
    {
        let c = i < text.length ? text[i] : " ";

        if (!isSeparator(c) && startWord < 0)
        {
            startWord = i;
        }

        if (isSeparator(c) && startWord >= 0)
        {
            let word = text.substring(startWord, i);
            ar.push(word);

            startWord = -1;
        }
    }

    return ar;
}

function isSeparator(c)
{
    var separators = [" ", "\t", "\n", "\r", ",", ";", ".", "!", "?", "(", ")"];
    return separators.includes(c);
}
```

Coding challenge #49: Shuffle an array of strings



codeguppy.com

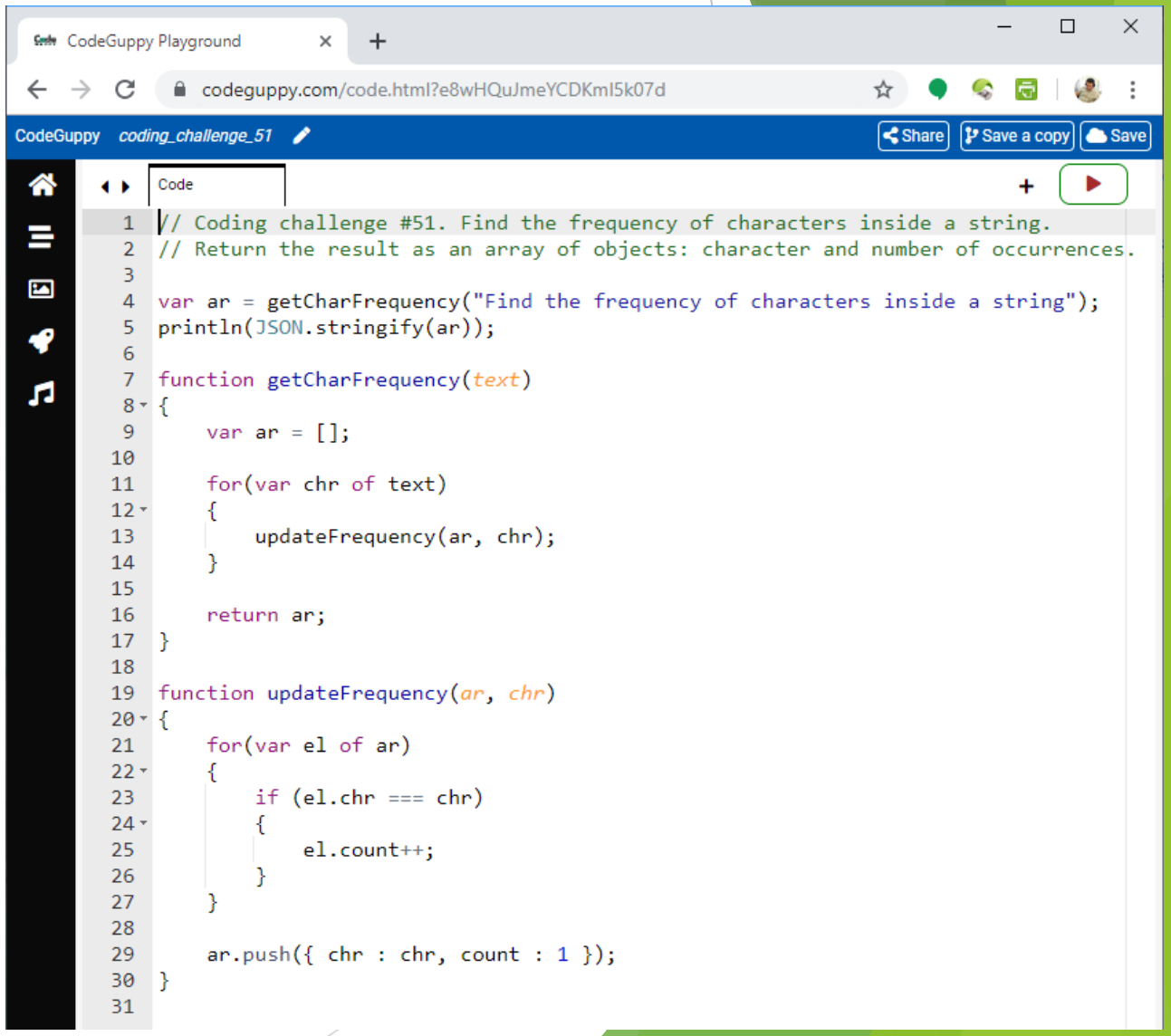
Coding challenge #50: Create a function that will receive n as argument and return an array of n random numbers from 1 to n



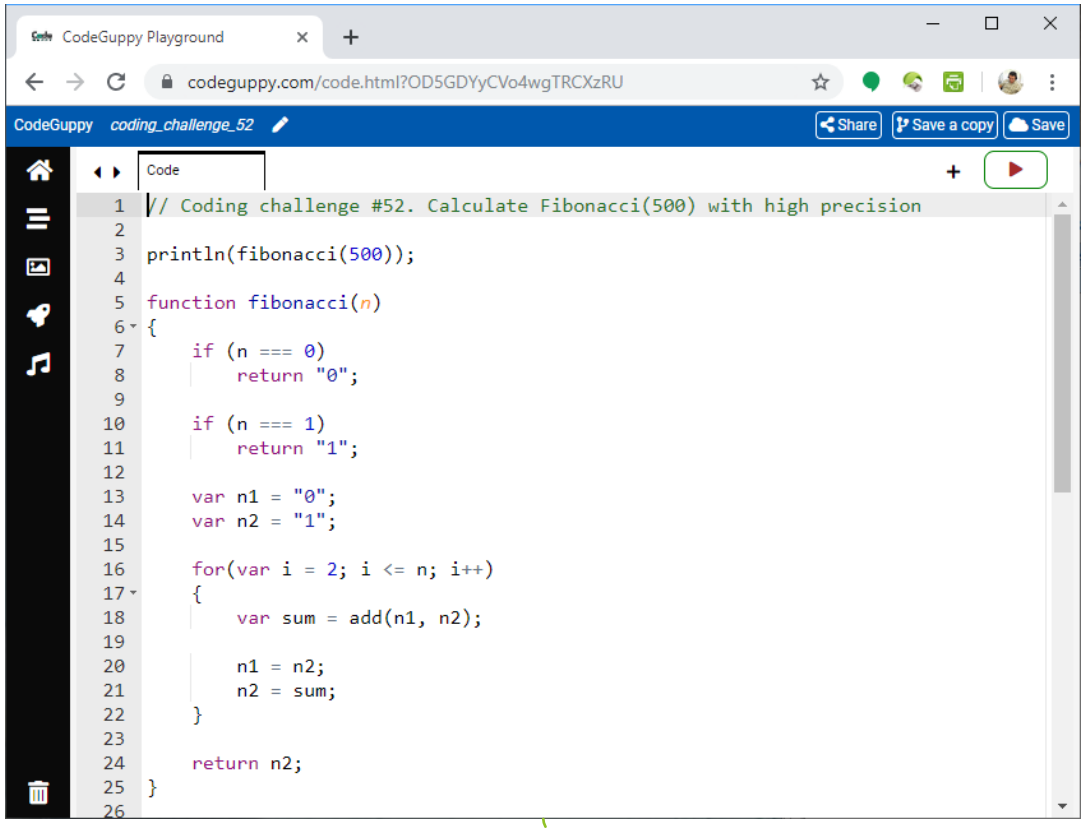
codeguppy.com

Coding challenge #51: Find the frequency of characters inside a string. Return the result as an array of objects.

Each object has 2 fields: character and number of occurrences.



Coding challenge #52: Calculate Fibonacci(500) with high precision (all digits)

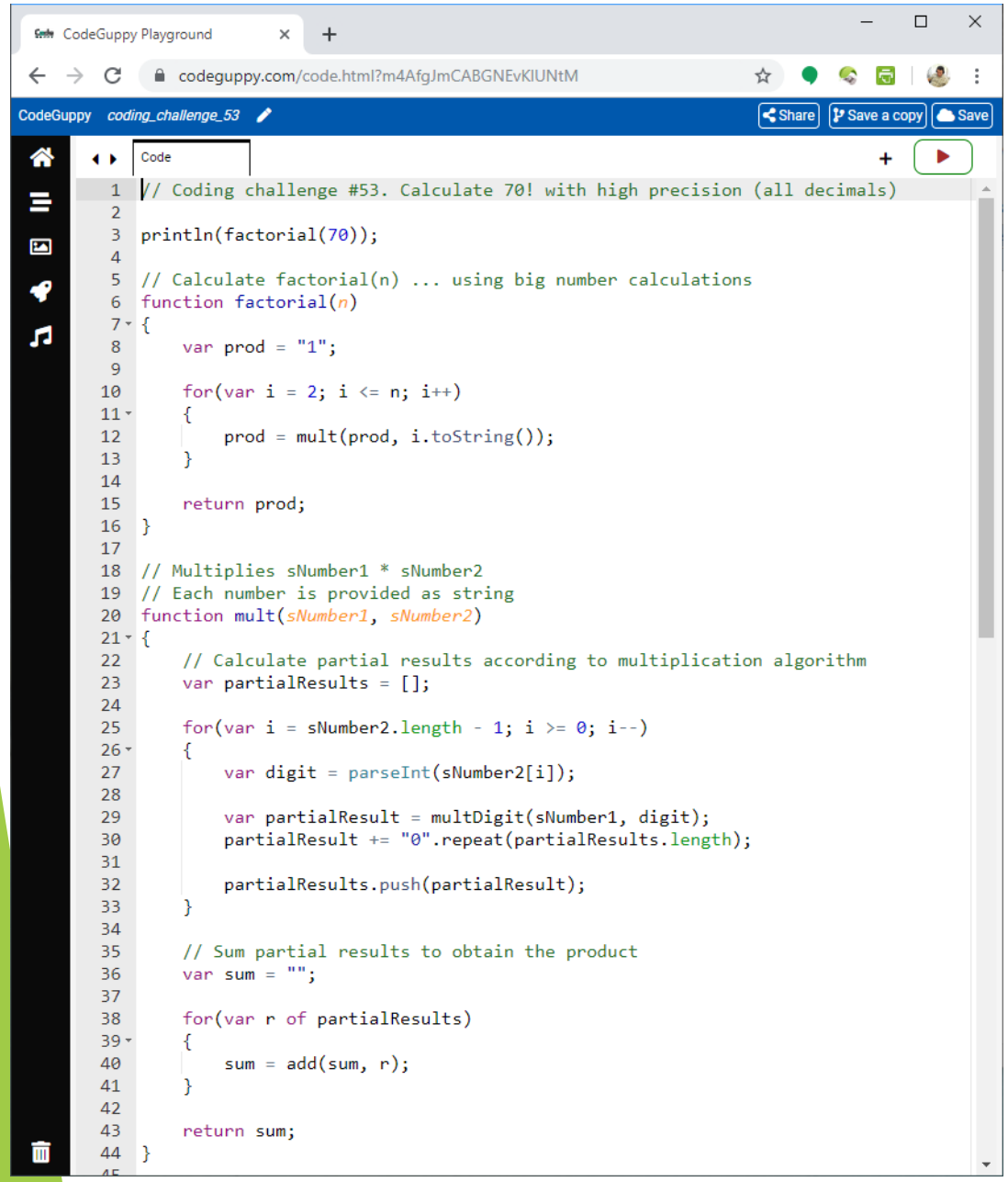



```
// Coding challenge #52. Calculate Fibonacci(500) with high precision

println(fibonacci(500));

function fibonacci(n)
{
    if (n === 0)
        return "0";

    if (n === 1)
        return "1";

    var n1 = "0";
    var n2 = "1";

    for(var i = 2; i <= n; i++)
    {
        var sum = add(n1, n2);

        n1 = n2;
        n2 = sum;
    }

    return n2;
}
```

```
function add(sNumber1, sNumber2)
{
    var maxSize = Math.max(sNumber1.length, sNumber2.length);

    var s1 = sNumber1.padStart(maxSize, "0");
    var s2 = sNumber2.padStart(maxSize, "0");

    var s = "";
    var carry = 0;

    for(var i = maxSize - 1; i >= 0; i--)
    {
        var digit1 = parseInt(s1[i]);
        var digit2 = parseInt(s2[i]);

        var sum = digit1 + digit2 + carry;
        var digitSum = sum % 10;
        carry = sum >= 10 ? 1 : 0;

        s = digitSum.toString() + s;
    }

    if (carry > 0)
        s = carry + s;

    return s;
}
```

codeguppy.com

Coding challenge #53: Calculate 70! with high precision (all digits)




```
// Coding challenge #53. Calculate 70! with high precision (all decimals)

println(factorial(70));

// Calculate factorial(n) ... using big number calculations
function factorial(n)
{
    var prod = "1";

    for(var i = 2; i <= n; i++)
    {
        prod = mult(prod, i.toString());
    }

    return prod;
}

// Multiplies sNumber1 * sNumber2
// Each number is provided as string
function mult(sNumber1, sNumber2)
{
    // Calculate partial results according to multiplication algorithm
    var partialResults = [];

    for(var i = sNumber2.length - 1; i >= 0; i--)
    {
        var digit = parseInt(sNumber2[i]);

        var partialResult = multDigit(sNumber1, digit);
        partialResult += "0".repeat(partialResults.length);

        partialResults.push(partialResult);
    }

    // Sum partial results to obtain the product
    var sum = "";

    for(var r of partialResults)
    {
        sum = add(sum, r);
    }

    return sum;
}
```

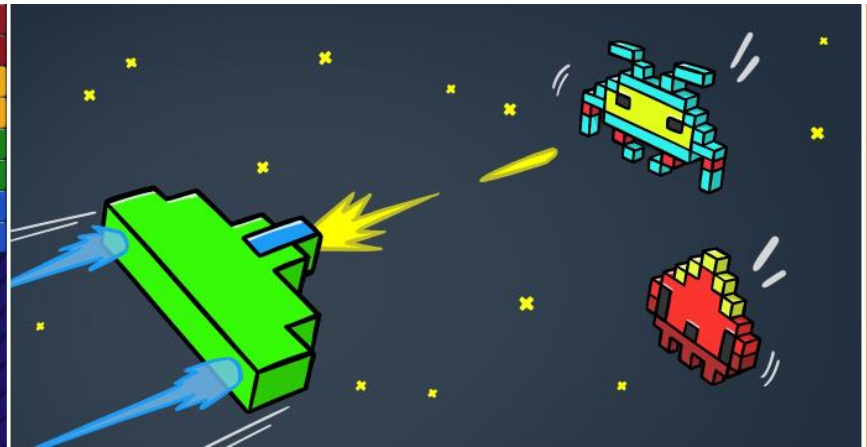
```
// Multiplies number sNumber (as string) with a single digit number
function multDigit(sNumber, digit)
{
    var p = "";
    var carry = 0;

    for(var i = sNumber.length - 1; i >= 0; i--)
    {
        var numberDigit = parseInt(sNumber[i]);

        var prod = digit * numberDigit + carry;
        var prodDigit = prod % 10;
        carry = Math.floor(prod / 10);

        p = prodDigit.toString() + p;
    }

    if (carry > 0)
        p = carry + p;

    return p;
}
```

```
function add(sNumber1, sNumber2)
{
    var maxSize = Math.max(sNumber1.length, sNumber2.length);

    var s1 = sNumber1.padStart(maxSize, "0");
    var s2 = sNumber2.padStart(maxSize, "0");

    var s = "";
    var carry = 0;

    for(var i = maxSize - 1; i >= 0; i--)
    {
        var digit1 = parseInt(s1[i]);
        var digit2 = parseInt(s2[i]);

        var sum = digit1 + digit2 + carry;
        var digitSum = sum % 10;
        carry = sum >= 10 ? 1 : 0;

        s = digitSum.toString() + s;
    }

    if (carry > 0)
        s = carry + s;

    return s;
}
```

These coding challenges were brought to you by codeguppy.com – the fun coding site for kids, teens and creative adults

Don’t forget to visit https://codeguppy.com for more fun projects!

For news and updates follow @codeguppy on Twitter!



Font

There are many properties related to the font, such as the face, weight, style, etc. These properties allow you to change the style or complete look of your text.

Font-Family

```
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
```

Font-Style

```
font-style: italic;
```

Font-Variant

```
font-variant: small-caps;
```

Font-Weight

```
font-weight: bold;
```

Font-Size

```
font-size: larger;
```

Font

```
font: style variant weight size family;
```

Text

Text properties allow one to manipulate alignment, spacing, decoration, indentation, etc., in the document.

Text-Align

Font

There are many properties related to the font, such as the face, weight, style, etc. These properties allow you to change the style or complete look of your text.

Font-Family

```
font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
```

Font-Style

```
font-style: italic;
```

Font-Variant

```
font-variant: small-caps;
```

Font-Weight

```
font-weight: bold;
```

Font-Size

```
font-size: larger;
```

Font

```
font: style variant weight size family;
```

Text

Text properties allow one to manipulate alignment, spacing, decoration, indentation, etc., in the document.

Text-Align

```
text-align: justify;
```

Letter-Spacing

```
letter-spacing: .15em;
```

Text-Decoration

```
text-decoration: underline;
```

Word-Spacing

```
word-spacing: 0.25em;
```

Text-Transform

```
text-transform: uppercase;
```

Text-Indent

```
text-indent: 0.5cm;
```

Line-Height

```
line-height: normal;
```

Background

As the name suggests, these properties are related to background, i.e., you can change the color, image, position, size, etc., of the document.

Background-Image

```
background-image: url("Path");
```

Background-Position

```
background-position: right top;
```

Background-Size

```
background-size: cover;
```

Background-Repeat

```
background-repeat: no-repeat;
```

Background-Attachment

```
background-attachment: scroll;
```

Background-Color

```
background-color: fuchsia;
```

Background

```
background: color image repeat attachment position;
```

Border

Border properties are used to change the style, radius, color, etc., of buttons or other items of the document.

Border-Width

```
border-width: 5px;
```

Border-Style

```
border-style: solid;
```

Border-Color

```
border-color: aqua;
```

Border-Radius

```
border-radius: 15px;
```

Border

```
border: width style color;
```

Box Model

In laymen's terms, the CSS box model is a container that wraps around every HTML element. It consists of margins, borders, padding, and the actual content. It is used to create the design and layout of web pages.

Float

```
float: none;
```

Clear

```
clear: both;
```

Display

```
display: block;
```

Height

```
height: fit-content;
```

Width

```
width: auto;
```

Margin

```
margin: top right bottom left;
```

Padding

```
padding: top right bottom left;
```

Overflow

```
overflow: hidden;
```

Visibility

```
visibility: visible;
```

Colors

With the help of the color property, one can give color to text, shape, or any other object.

Color

```
color: cornsilk;
```

Opacity

```
opacity: 4;
```

Template Layout

Specifies the visual look of the content inside a template

Box-Align

```
box-align : start;
```

Box-Direction

```
box-direction : normal;
```

Box-Flex

```
box-flex : normal;
```

Box-Flex-Group

```
box-flex-group : 2;
```

Box-Orient

```
box-orient : inline;
```

Box-Pack

```
box-pack : justify;
```

Box-Sizing

```
box-sizing : margin-box;
```

max-width

```
max-width: 800px;
```

min-width

```
min-width: 500px;
```

max-height

```
max-height: 100px;
```

min-height

```
min-height: 80px;
```

Table

Table properties are used to give style to the tables in the document. You can change many things like border spacing, table layout, caption, etc.

Border-Collapse

```
border-collapse: separate;
```

Empty-Cells

```
empty-cells: show;
```

Border-Spacing

```
border-spacing: 2px;
```

Table-Layout

```
table-layout: auto;
```

Caption-Side

```
caption-side: bottom;
```

Columns

These properties are used explicitly with columns of the tables, and they are used to give the table an incredible look.

Column-Count

```
column-count : 10;
```

Column-Gap

```
column-gap : 5px;
```

Column-rule-width

```
column-rule-width : medium;
```

Column-rule-style

```
column-rule-style : dotted ;
```

Column-rule-color

```
column-rule-color : black;
```

Column-width

```
column-width : 10px;
```

Column-span

```
column-span : all;
```

List & Markers

List and marker properties are used to customize lists in the document.

List-style-type

```
list-style-type: square;
```

List-style-position

```
list-style-position : 20px;
```

List-style-image

```
list-style-image : url(image.gif);
```

Marker-offset

```
marker-offset : auto;
```

Animations

CSS animations allow one to animate transitions or other media files on the web page.

Animation-name

```
animation-name : myanimation;
```

Animation-duration

```
animation-duration : 10s;
```

Animation-timing-function

```
animation-timing-function : ease;
```

Animation-delay

```
animation-delay : 5ms;
```

Animation-iteration-count

```
animation-iteration-count : 3;
```

Animation-direction

```
animation-direction : normal;
```

Animation-play-state

```
animation-play-state : running;
```

Animation-fill-mode

```
animation-fill-mode : both;
```

Transitions

Transitions let you define the transition between two states of an element.

Transition-property

```
transition-property: none;
```

Transition-duration

```
transition-duration : 2s;
```

Transition-timing-function

```
transition-timing-function: ease-in-out;
```

Transition-delay

```
transition-delay : 20ms;
```

CSS Flexbox

Flexbox is a layout of CSS that lets you format HTML easily. Flexbox makes it simple to align items vertically and horizontally using rows and columns. Items will "flex" to different sizes to fill the space. And overall, it makes the responsive design more manageable.

Parent Properties (flex container)

display

```
display: flex;
```

flex-direction

```
flex-direction: row | row-reverse | column | column-reverse;
```

flex-wrap

```
flex-wrap: nowrap | wrap | wrap-reverse;
```

flex-flow

```
flex-flow: column wrap;
```

justify-content

```
justify-content: flex-start | flex-end | center | space-between | space-arou
```

align-items

```
align-items: stretch | flex-start | flex-end | center | baseline | first bas
```

align-content

```
align-content: flex-start | flex-end | center | space-between | space-around
```

Child Properties (flex items)

order

```
order: 5; /* default is 0 */
```

flex-grow

```
flex-grow: 4; /* default 0 */
```

flex-shrink

```
flex-shrink: 3; /* default 1 */
```

flex-basis

```
flex-basis: | auto; /* default auto */
```

flex shorthand

```
flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
```

align-self

```
align-self: auto | flex-start | flex-end | center | baseline | stretch;
```

CSS Grid

Grid layout is a 2-Dimensional grid system to CSS that creates complex responsive web design layouts more easily and consistently across browsers.

Parent Properties (Grid container)

display

```
display: grid | inline-grid;
```

grid-template-columns

```
grid-template-columns: 12px 12px 12px;
```

grid-template-rows

```
grid-template-rows: 8px auto 12px;
```

grid-template

```
grid-template: none | <grid-template-rows> / <grid-template-columns>;
```

column-gap

```
column-gap: <line-size>;
```

row-gap

```
row-gap: <line-size>;
```

grid-column-gap

```
grid-column-gap: <line-size>;
```

grid-row-gap

```
grid-row-gap: <line-size>;
```

gap shorthand

```
gap: <grid-row-gap> <grid-column-gap>;
```

grid-gap shorthand

```
grid-gap: <grid-row-gap> <grid-column-gap>;
```

justify-items

```
justify-items: start | end | center | stretch;
```

align-items

```
align-items: start | end | center | stretch;
```

place-items

```
place-items: center;
```

justify-content

```
justify-content: start | end | center | stretch | space-around | space-betwe
```

align-content

```
align-content: start | end | center | stretch | space-around | space-between
```

place-content

```
place-content: <align-content> / <justify-content> ;
```

grid-auto-columns

```
grid-auto-columns: <track-size> ...;
```

grid-auto-rows

```
grid-auto-rows: <track-size> ...;
```

grid-auto-flow

```
grid-auto-flow: row | column | row dense | column dense;
```

Child Properties (Grid items)

grid-column-start

```
grid-column-start: <number> | <name> | span <number> | span <name> | auto;
```

grid-column-end

```
grid-column-end: <number> | <name> | span <number> | span <name> | auto;
```

grid-row-start

```
grid-row-start: <number> | <name> | span <number> | span <name> | auto;
```

grid-row-end

```
grid-row-end: <number> | <name> | span <number> | span <name> | auto;
```

grid-column shorthand

```
grid-column: <start-line> / <end-line> | <start-line> / span <value>;
```

grid-row shorthand

```
grid-row: <start-line> / <end-line> | <start-line> / span <value>;
```

grid-area

```
grid-area: <name> | <row-start> / <column-start> / <row-end> / <column-end>;
```

justify-self

```
justify-self: start | end | center | stretch;
```

align-self

```
align-self: start | end | center | stretch;
```

place-self

```
place-self: center;
```

CSS

cascading style sheets

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint