**Javascript Kullanarak Yapay Zeka Hesap Makinesi Örnek Uygulaması**

<div id="hesapMakinesi">

  <div id="ekran">

    <input type="text" id="sonuc">

  </div>

  <div id="sayilar">

    <button>7</button>

    <button>8</button>

    <button>9</button>

    <button>+</button>

    <button>4</button>

    <button>5</button>

    <button>6</button>

    <button>-</button>

    <button>1</button>

    <button>2</button>

    <button>3</button>

    <button>/</button>

    <button>0</button>

    <button>.</button>

    <button>=</button>

    <button>\*</button>

  </div>

</div>

<style>

  #hesapMakinesi {

    margin: 0 auto;

    width: 350px;

    height: auto;

    background-color: slategray;

    border-radius: 5px;

    padding: 10px;

  }

  #ekran {

    padding: 10px 0px 10px 0px;

    text-align: center;

  }

  #sonuc {

    padding: 10px;

    width: 90%;

    border: none;

    border-radius: 5px;

    text-align: right;

    font-size: 50px;

    word-wrap: break-word;

  }

  #sayilar {

    text-align: center;

  }

  button {

    width: 70px;

    height: 50px;

    margin: 4px;

    border-radius: 5px;

    background-color: whitesmoke;

    border: none;

    outline: none;

  }

</style>

<script>

"use strict";

  var \_sayilar = document.getElementsByTagName("button"),

    \_sayiUzunluk = \_sayilar.length,

    i = 0;

  for (i = 0; i < \_sayiUzunluk; i++) {

    \_sayilar[i].onclick = hesapla;

  }

</script>

<script>

"use strict";

  var \_sonuc = document.getElementById("sonuc");

  function hesapla() {

    var \_deger = this.innerHTML;

    if (\_deger == "=") {

      try {

        \_sonuc.value = eval(\_sonuc.value);

      } catch (e) {

        \_sonuc.value = "0";

      }

      return;

    }

    \_sonuc.value += \_deger;

  }

</script>

<script>

"use strict";

  var \_sayilar = document.getElementsByTagName("button"),

    \_sayiUzunluk = \_sayilar.length,

    i = 0,

    \_sonuc = document.getElementById("sonuc");

  for (i = 0; i < \_sayiUzunluk; i++) {

    \_sayilar[i].onclick = hesapla;

  }

  function hesapla() {

    var \_deger = this.innerHTML;

    if (\_deger == "=") {

      try {

        \_sonuc.value = eval(\_sonuc.value);

      } catch (e) {

        \_sonuc.value = "0";

      }

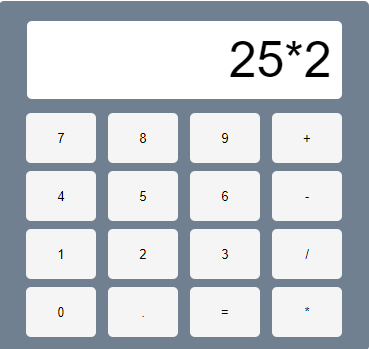
      return;

    }

    \_sonuc.value += \_deger;

  }

</script>



**Project Euler**

**Problem No:** 01

**Problem Adı:** Multiples of 3 and 5

**Açıklama:**

1. Bir sayı verildiğinde, 3'ün katı olup olmadığına bakın
2. Doğruysa, toplam sayıya ekleyin
3. Bir sayı verildiğinde, 5'in katı olup olmadığına bakın
4. Doğruysa, toplam sayıya ekleyin

## KOD: function sumOfMultiple(limit) {

## var i, sum = 0;

## for (i = 3; i < limit; i += 1) {

## if (i % 3 === 0 || i % 5 === 0) {

## sum += i;

## }

## }

## return sum;

## }

## console.log(sumOfMultiple(1000));

**Sonuç:** 233168

**Problem No:** 11

## Problem Adı: Largest product in a grid

**Açıklama:** Köşegenler için aynı sınırlar doğrudur ve ilk köşegen buna göre inşa edilebilir. İkincisi ters çevrilmelidir. Böylece çözüm, tam matrisi taradığımız ve geçerli bir aralığa düştüğünde maksimize edilecek.

**KOD:** var grid = [

[08, 02, 22, 97, 38, 15, 00, 40, 00, 75, 04, 05, 07, 78, 52, 12, 50, 77, 91, 08],

[49, 49, 99, 40, 17, 81, 18, 57, 60, 87, 17, 40, 98, 43, 69, 48, 04, 56, 62, 00],

[81, 49, 31, 73, 55, 79, 14, 29, 93, 71, 40, 67, 53, 88, 30, 03, 49, 13, 36, 65],

[52, 70, 95, 23, 04, 60, 11, 42, 69, 24, 68, 56, 01, 32, 56, 71, 37, 02, 36, 91],

[22, 31, 16, 71, 51, 67, 63, 89, 41, 92, 36, 54, 22, 40, 40, 28, 66, 33, 13, 80],

[24, 47, 32, 60, 99, 03, 45, 02, 44, 75, 33, 53, 78, 36, 84, 20, 35, 17, 12, 50],

[32, 98, 81, 28, 64, 23, 67, 10, 26, 38, 40, 67, 59, 54, 70, 66, 18, 38, 64, 70],

[67, 26, 20, 68, 02, 62, 12, 20, 95, 63, 94, 39, 63, 08, 40, 91, 66, 49, 94, 21],

[24, 55, 58, 05, 66, 73, 99, 26, 97, 17, 78, 78, 96, 83, 14, 88, 34, 89, 63, 72],

[21, 36, 23, 09, 75, 00, 76, 44, 20, 45, 35, 14, 00, 61, 33, 97, 34, 31, 33, 95],

[78, 17, 53, 28, 22, 75, 31, 67, 15, 94, 03, 80, 04, 62, 16, 14, 09, 53, 56, 92],

[16, 39, 05, 42, 96, 35, 31, 47, 55, 58, 88, 24, 00, 17, 54, 24, 36, 29, 85, 57],

[86, 56, 00, 48, 35, 71, 89, 07, 05, 44, 44, 37, 44, 60, 21, 58, 51, 54, 17, 58],

[19, 80, 81, 68, 05, 94, 47, 69, 28, 73, 92, 13, 86, 52, 17, 77, 04, 89, 55, 40],

[04, 52, 08, 83, 97, 35, 99, 16, 07, 97, 57, 32, 16, 26, 26, 79, 33, 27, 98, 66],

[88, 36, 68, 87, 57, 62, 20, 72, 03, 46, 33, 67, 46, 55, 12, 32, 63, 93, 53, 69],

[04, 42, 16, 73, 38, 25, 39, 11, 24, 94, 72, 18, 08, 46, 29, 32, 40, 62, 76, 36],

[20, 69, 36, 41, 72, 30, 23, 88, 34, 62, 99, 69, 82, 67, 59, 85, 74, 04, 36, 16],

[20, 73, 35, 29, 78, 31, 90, 01, 74, 31, 49, 71, 48, 86, 81, 16, 23, 57, 05, 54],

[01, 70, 54, 71, 83, 51, 54, 69, 16, 92, 33, 48, 61, 43, 52, 01, 89, 19, 67, 48]

];

var maxProduct = 0;

for (var i = 0; i < grid.length; i++) {

var row = grid[i];

for (var j = 0; j < row.length; j++) {

//check the left-to-right sum

if (j < row.length - 3) {

var product = row[j] \* row[j + 1] \* row[j + 2] \* row[j + 3];

if (product > maxProduct) {

maxProduct = product;

}

}

//check the top-to-bottom sum

if (i < grid.length - 3) {

var product = grid[i][j] \* grid[i + 1][j] \* grid[i + 2][j] \* grid[i + 3][j];

if (product > maxProduct) {

maxProduct = product;

}

}

//check the top-left-to-bottom-right sum

if ((j < row.length - 3) && (i < grid.length - 3)) {

var product = grid[i][j] \* grid[i + 1][j + 1] \* grid[i + 2][j + 2] \* grid[i + 3][j + 3];

if (product > maxProduct) {

maxProduct = product;

}

}

//check the top-right-to-bottom-left sum

if ((j > 2) && (i < grid.length - 3)) {

var product = grid[i][j] \* grid[i + 1][j - 1] \* grid[i + 2][j - 2] \* grid[i + 3][j - 3];

if (product > maxProduct) {

maxProduct = product;

}

}

}

}

console.log(maxProduct);

**Sonuç:** 70600674

**Problem No:** 21

**Problem Adı:** Amicable numbers

**Açıklama:** Kodun ilk kısmı, belirli bir x tamsayısı için tüm uygun bölenleri listeleyen bir işlev sağlar . Döngü 220 ila 10.000 sayıları için bölenleri belirler, toplamlarını hesaplar ve sonra bu sayıların dostane olup olmadığını kontrol eder. Kod dostane bir sayı bulduğunda, sayaç bir sonrakini kontrol etmek için bölenlerin toplamına atlar.

**KOD:** function sumAmicableNum(n) {

function d(n){

let output = []

for (let i = 1; i < n; i++){

if (n%i===0){

output.push(i)

}

}

if (output.length === 0){

return 0

}

else {

return output.reduce((a,b)=> a+=b)

}

}

let amicNums = [

for (let i = 1; i < n; i++){

let r1 = d(i)

let r2 = d(r1)

if (i === r2 && i !== r1 && amicNums.indexOf(i) === -1){

amicNums.push(i, r1)

}

}

return amicNums.reduce((a,b)=> a+=b)

}

sumAmicableNum(10000);

**Sonuç:** 31626

**Problem No:** 31

## Problem Adı: Coin sums

**Açıklama:** İngiltere'de para birimi pound, £ ve pence, p'den oluşur ve genel dolaşımda sekiz para vardır:

1p, 2p, 5p, 10p, 20p, 50p, £ 1 (100p) ve £ 2 (200p).

Aşağıdaki şekilde £ 2 kazanmanız mümkündür:

1 × £ 1 + 1 × 50p + 2 × 20p + 1 × 5p + 1 × 2p + 3 × 1p

Herhangi bir sayıda jeton kullanarak kaç farklı yol yapılabilir?

**KOD:** var makeChange = function(total){

var count = 0;

var coins = [1, 2, 5, 10, 20, 50, 100, 200];

var changer = function(index, value){

var currentCoin = coins[index];

if( index === 0){

if( value % currentCoin === 0){

count++;

}

return;

}

while( value >= 0 ){

changer(index-1, value);

value -= currentCoin;

}

}

changer(coins.length-1, total);

return count;

};

makeChange(200);

**Sonuç:** 73682

**Problem No:** 51

## Problem Adı: Prime digit replacements

## Açıklama: 2 basamaklı sayının \* 3'ünün 1. basamağını değiştirerek, dokuz olası değerin altısının: 13, 23, 43, 53, 73 ve 83'ün en önemli olduğu ortaya çıkıyor.

## **Sayının bir kısmını (mutlaka bitişik basamaklar değil) aynı rakamla değiştirerek, sekiz asal değerli bir ailenin parçası olan en küçük üssü bulun.**Bu yüzden temelde bir rakam bulmalıyız ki, bazı rakamlarını bir rakamla değiştirirsek (yani 1 \*\* 1: 1111, 1221, 1331, 1441), 8 tane daha primer alabiliriz.

## KOD: function isPrime(number){

## if(number <= 1){return false;}

## if(number == 2){return true;}

## if(!(number%2)){return false;}

## var maxCheck = Math.floor(Math.sqrt(number));

## for(var i = 3; i <= maxCheck; i+=2){

## if(!(number%i)){return false;}

## }

## return true;

## }

## function getNumPrimes(family){

## var length = family.length;

## var numPrimes = 0;

## // Replace every wild card with 0 to 9

## // and test primality

## // According to example, firt wild card can't be zero

## for(var j = 0; j < 10; j++){

## var number = 0;

## for(var i = 0; i < length; i++){

## if(family[i] < 0 && !(i == 0 && j == 0)){

## number += j\*Math.pow(10,length-1-i);

## }else{

## number += family[i]\*Math.pow(10,length-1-i);

## }

## }

## if(isPrime(number)){

## //console.log(number);

## numPrimes++;

## }

## }

## return numPrimes;

## }

## //Search up until hit family with 8

## var limit = 1000000;

## for(var j = 2; j < limit; j++){

## if(isPrime(j)){

## //Construct initial family without wildcards

## var tmpNumber = j;

## var family = [];

## while(tmpNumber > 0){

## family.push(tmpNumber%10);

## tmpNumber = tmpNumber/10 | 0;

## }

## family.reverse();

## //console.log(family);

## //Seach every possible wildcard combo

## var length = family.length;

## for(var i = 0; i < 10; i++){

## var tmpFamily = family.slice();

## var found = false;

## for(k = 0; k < length; k++){

## if(tmpFamily[k] == i){

## tmpFamily[k] = -1;

## found = true;

## }

## }

## if(found){

## if(getNumPrimes(tmpFamily) == 8){

## console.log(j);

## j = limit;

## break;

## }

## }

## }

## }

## }

## Sonuç: 121313

**LeetCode**

**Problem No:** 21

**Problem Adı:** Merge Two Sorted Lists

**Açıklama:** Sıralı iki bağlantılı listeyi birleştir ve yeni bir liste olarak döndür. Yeni liste, ilk iki listenin düğümlerini birleştirerek yapılmalıdır.

**KOD:**

var mergeTwoLists = function(l1, l2) {

if (!l1) return l2;

if (!l2) return l1;

if (l1.val < l2.val) {

l1.next = mergeTwoLists(l1.next, l2);

return l1;

}

l2.next = mergeTwoLists(l2.next, l1);

return l2;

};

**Sonuç:** [1,1,2,3,4,4]

**Problem No:** 101

**Problem Adı:** Symmetric Tree

**Açıklama:** İkili bir ağaç verildiğinde, kendisinin bir ayna olup olmadığını kontrol edin (yani, merkezi etrafında simetrik).

**KOD:**

var isSymmetric = function(root) {

if (root == null) return true;

let list = [];

let queue = [];

queue.unshift(root);

while(queue.length > 0) {

let size = queue.length;

for (let i = 0; i < size; i++) {

let node = queue.pop();

if (node) {

list.push(node.val);

node.left ? queue.unshift(node.left) : queue.unshift(null);

node.right ? queue.unshift(node.right) : queue.unshift(null);

} else {

list.push(null);

}

}

for (let k = 0, j = list.length - 1; k <= j; k++, j--) {

if (list[k] !== list[j]) {

return false;

}

}

list = [];

}

return true;

};

**Sonuç:** [1,2,2,3,4,4,3] Doğru

**Problem No:** 171

**Problem Adı:** Excel Sheet Column Number

**Açıklama:** Bir Excel sayfasında göründüğü gibi bir sütun başlığı verildiğinde, karşılık gelen sütun numarasını döndürün.

**KOD:**

var titleToNumber = function(s) {

var result = 0;

for (var i=0; i<s.length; i++) {

result \*= 26;

result += ((s.charCodeAt(i) - 'A'.charCodeAt(0)) + 1);

}

return result;

};

**Sonuç:** "A”: 1 "Z”: 26

**Problem No:** 221

**Problem Adı:** Maximal Square

**Açıklama:** 0'lar ve 1'ler ile dolu bir 2D ikili matris verildiğinde, 1'lerin tümünü içeren en büyük kareyi bulun ve alanını döndürün.

**KOD:** var maximalSquare = function(matrix) {

if (matrix == null || matrix.length == 0 || matrix[0].length == 0)

return 0;

var m = matrix.length;

var n = matrix[0].length;

var t = [];

for (var i = 0; i < m; i++) {

var temp = [];

for (var j = 0; j < n; j++) {

temp.push(0);

}

t.push(temp);

}

//left column

for (var i = 0; i < m; i++) {

t[i][0] = matrix[i][0];

}

//top row

for (var j = 0; j < n; j++) {

t[0][j] = matrix[0][j];

}

//cells inside

for(var i = 1; i < m; i++) {

for (var j = 1; j < n; j++) {

if (matrix[i][j] == '1') {

var min = Math.min(t[i - 1][j], t[i - 1][j - 1]);

min = Math.min(min,t[i][j - 1]);

t[i][j] = min + 1;

} else {

t[i][j] = 0;

}

}

}

var max = 0;

//get maximal length

for (var i = 0; i < m; i++) {

for (var j = 0; j < n; j++) {

if (t[i][j] > max) {

max = t[i][j];

}

}

}

return max \* max;

};

**Sonuç:** 4

**Problem No:** 401

**Problem Adı:** Binary Watch

**Açıklama:** İkili bir saatte üstte saatleri temsil eden 4 LED bulunur ( 0-11 ) ve alttaki 6 LED'in dakikası ( 0-59 ) temsil eder .

Her LED sıfır veya bir değeri temsil eder, en az anlamlı bit sağdadır.

**KOD:** var readBinaryWatch = function(num) {

let res =[];

function bitCount (n) {

n = n - ((n >> 1) & 0x55555555)

n = (n & 0x33333333) + ((n >> 2) & 0x33333333)

return ((n + (n >> 4) & 0xF0F0F0F) \* 0x1010101) >> 24

}

for (let h = 0; h < 12;h++){

for (let m = 0; m < 60; m++){

if (bitCount(h) + bitCount(m) == num){

let str = "";

str += h;

str += ":";

str += (m < 10 ? "0" : "");

str += m;

res.push(str);

}

}

}

return res;

};

**Sonuç:** 0