# *Java script => Code Challenge :*

1. The most common minimum age to vote is 18. Write a function `canIVote()` that takes in a number, representing the person's age, and returns the boolean `true` if they are 18 years old or older, and the boolean `false` if they are not.

```
function canIVote(age){
    if (age>=18)
    {
      console.log("you are eligble");
        return true;
    }
    else
      {
        console.log("you are not eligble");
        return false;
      }

  }
  canIVote(10);
```

2. Write a function, `agreeOrDisagree()`, that takes in two strings, and returns 'You agree!' if the two strings are the same and 'You disagree!' if the two strings are different.

```
const agreeOrDisagree = (first, second) => {

    if (first === second) {

        return 'You agree!'

    } else {

        return 'You disagree!'

    }

}

console.log(agreeOrDisagree("yep", "yep"))
```

3. Write a function, `lifePhase()`, that takes in a person's `age`, as a number, and returns which phase of life they are in.
   Here are the classifications:
   0-3 should return 'baby'
   4-12 should return 'child'
   13-19 should return 'teen'
   20-64 should return 'adult'
   65-140 should return 'senior citizen'
   If the number is less than 0 or greater than 140, the program should return 'This is not a valid age'

## Use if - else-if condition:

```
  const lifePhase = age => {

    if (age < 0 || age > 140) {

        return 'This is not a valid age'

    } else if (age < 4) {
        return 'baby'

    } else if (age < 13) {

        return 'child'

    } else if (age < 20) {

        return 'teen'

    } else if (age < 65) {

        return 'adult'

    } else {

        return 'senior citizen'

    }

}

console.log(lifePhase(5))
```

## Use of Switch Case:-

```
function lifePhase(age){

    switch(true)

      {

        case (age<=3):

        console.log("baby");

        break;


        case (age<=12):

        console.log("child");

        break;


        case (age<=19):

        console.log("teen");
        break;


        case (age<=64):
        console.log("adult");
        break;


        case (age<=140):
        console.log("senior citizen");
        break;


        default:
        console.log('This is not a valid age');
        break;
      }  }


  lifePhase(22);
```

4. Write a function, `finalGrade()`. It should:

- take three arguments of type number
- find the `average` of those three numbers
- return the letter grade (as a string) that the `average` corresponds to
- return 'You have entered an invalid grade.' if any of the three grades are less than 0 or greater than 100

0-59 should return: 'F'
60-69 should return: 'D'
70-79 should return: 'C'

80-89 should return: 'B'
90-100 should return: 'A'

```
//Example 1


function finalGrade(a, b, c)

{

  let total=a+b+c;

  let average=total * 100 / 300;

  if(average<=59)

    { console.log("fail");}

  else if(average<=69)

    { console.log("D"); }

  else if(average<=79)

    { console.log("C"); }

  else if(average<=89)

    { console.log("B");}

else if(average<=100)

    { console.log("A");}

else if(average>100)

  { console.log('You have entered an invalid grade.');}

  else{ console.log('HELLO '); }

}

finalGrade(70, 70, 70);
```

## Example 2:

```javascript
//Example 2

const finalGrade = (midterm, final, homework) => {

    if ((midterm < 0 || midterm > 100) || (final < 0 || final > 100) || (homework < 0 || homework > 100)) {

        return 'You have entered an invalid grade.'

    }

    let average = (midterm + final + homework) / 3

    if (average < 60) {

        return 'F'

    }

    else if (average < 70) {

        return 'D'

    }

    else if (average < 80) {

        return 'C'

    }

    else if (average < 90) {

        return 'B'

    } else {

        return 'A'

    }

}
```

5. Write a function, `reportingForDuty()`, that has two string parameters, `rank`and `lastName`, and returns a string in the following format: 'rank lastName reporting for duty!'

```
//Example 1

function reportingForDuty(rank,lastName){

console.log(`${rank} ${lastName} reporting for duty!,Sir!`);

}

  reportingForDuty(111,"mathur");
```

Example 2:

```
//Example :2

const reportingForDuty = (rank, lastName) => `${rank} ${lastName} reporting for
duty!`
```

6. We wrote a function, `rollTheDice()`, which is supposed to simulate two dice being rolled and totalled. It's close to doing what we want, but there's something not quite right. Can you fix our code, please?

```
const rollTheDice = () => {


    let die1 = Math.floor(Math.random() * 6  ) ;

   let die2 = Math.floor(Math.random()  * 6  ) ;

    return die1 + die2

}


console.log(rollTheDice());
```

7. Though an object's mass remains consistent throughout the universe, weight is determined by the force of gravity on an object. Since different planets have different gravity, the same object would weigh different amounts on each of those planets! Cool, huh?

Write a function, `calculateWeight()`. It should:

- have two parameters: `earthWeight`and `planet`
- expect `earthWeight` to be a number
- expect `planet` to be a string
- return a number representing what that Earth-weight would equate to on the `planet` passed in.

Handle the following cases:
`'Mercury'` weight = `earthWeight` * 0.378
`'Venus'` weight = `earthWeight` * 0.907
`'Mars'` weight = `earthWeight` * 0.377
`'Jupiter'` weight = `earthWeight` * 2.36
`'Saturn'` weight = `earthWeight` * 0.916
For all other inputs, return `'Invalid Planet Entry. Try: Mercury, Venus, Mars, Jupiter, or Saturn.'`

```
const calculateWeight =(planet,earthWeight) =>{


    switch(planet)

           {

             case 'Mercury':

              return earthWeight * 0.378

               break;


                 case 'Venus':

              return earthWeight * 0.907
```

```
               break;

                 case 'Mars':
              return earthWeight * 0.377
               break;

                 case 'Jupiter':
              return earthWeight * 2.36
               break;

                 case 'Saturn':
              return earthWeight * 0.916
               break;

             default:
               return 'Invalid Planet Entry. Try: Mercury, Venus, Mars, Jupiter, or Saturn.';

           }

  }
  console.log(calculateWeight('Jupiter', 100))
```

8. It can be hard to keep track of what's **truthy** or **falsy** in JavaScript.( https://developer.mozilla.org/en-US/docs/Glossary/Falsy) .Write a

function, `truthyOrFalsy()`, that takes in any value and returns `true` if that value is **truthy** and `false` if that value is **falsy**.

```
function truthyOrFalsy(ab){


    if(ab==10)

      {

        return true;

      }

    else

      {

        return false;

      }

  }

  let a=truthyOrFalsy(10);

  console.log(a);
```

9. A person's number of imaginary friends are always 33% of their total friends.

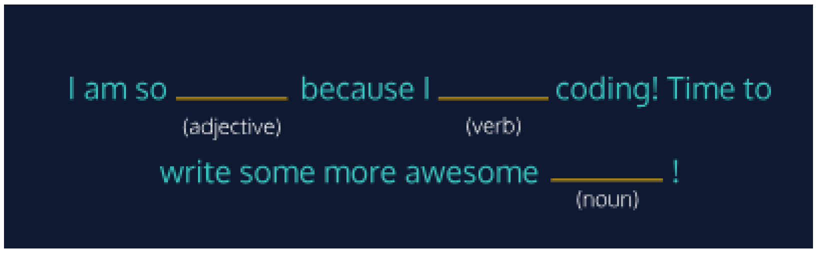
Write a function, `numImaginaryFriends()`, that takes in the total number of friends a person has and returns the number of imaginary friends they have.Since friends can only come in whole numbers, be sure to round your result before returning it.The JavaScript `Math.round()` function will come in handy.

```
const numImaginaryFriends=(totalFriends) =>{

    let imgFriends=Math.floor(Math.round(totalFriends*33/100 ));

    return imgFriends;  }
```

```
10.console.log(numImaginaryFriends(100));
```

10.

10. Write a function, `sillySentence()`, that has 3 string parameters and returns the following silly sentence with the blanks filled in by the arguments passed into the function:


I am so ______ because I ______ coding! Time to
(adjective) (verb)
write some more awesome ______ !
(noun)

```js
const sillySentence = (adjective, verb, noun) => `I am so ${adjective} because I ${verb} coding! Time to write some more awesome ${noun}!`
```

```js sillySentence('excited', 'love', 'functions') // Should return 'I am so excited because I love coding! Time to write some more awesome functions!' ```

11. Write a function, `howOld()`, that has two number parameters, `age` and `year`, and returns how old someone who is currently that `age` was (or will be) during that `year`.

```js
const yearDifference = year - theCurrentYear
const newAge = age + yearDifference
```

Once you have `newAge`, you'll be able to handle the three difference cases.

If the `newAge` is less than 0, this means the year provided was before the person was born. If the `newAge` is greater than their current age, this means the year passed in is in the future. Otherwise, we know the year provided was in the past but not before they were born.

```
const howOld = (age, year) => {
    // The following two lines make it so that our function always knows the
current year.
        let dateToday = new Date();
        let thisYear = dateToday.getFullYear();
    // It is totally ok if your function used the current year directly!
      console.log(dateToday);
      console.log(thisYear);


        const yearDifference = year - thisYear
        const newAge = age + yearDifference


        if (newAge < 0) {
            return `The year ${year} was ${-newAge} years before you were born`
        } else if (newAge > age) {
            return `You will be ${newAge} in the year ${year}`
        } else {
            return `You were ${newAge} in the year ${year}`
        }
    }
    console.log(howOld(-24,2019));
```

12. We wrote a function, `whatRelation()`, that has one number parameter, `percentSharedDNA`, and returns the likely relationship. We expect the number passed in to always be an integer from 0 to 100, but for some reason it's not working!

Here's how it's supposed to calculate the relationship:
100 should return 'You are likely identical twins.'
35-99 should return 'You are likely parent and child or full siblings.'
14-34 should return 'You are likely grandparent and grandchild, aunt/uncle and neice/nephew, or half siblings.'
6-13 should return 'You are likely 1st cousins.'
3-5 should return 'You are likely 2nd cousins.'

1-2 should return 'You are likely 3rd cousins.'
0 should return 'You are likely not related.'

```
const whatRelation = function(percentSharedDNA){

    if (percentSharedDNA === 100)
    {
        return 'You are likely identical twins.'
    }

    if (percentSharedDNA > 34)
    {
        return 'You are likely parent and child or full siblings.'
    }

    if (percentSharedDNA > 13)
    {
        return 'You are likely grandparent and grandchild, aunt/uncle and 
niece/nephew, or half siblings.'
    }

    if (percentSharedDNA > 5)
    {
        return 'You are likely 1st cousins.'
    }

    if (percentSharedDNA > 2)
    {
        return 'You are likely 2nd cousins.'
    }

    if (percentSharedDNA > 0)
    {
        return 'You are likely 3rd cousins'
    }

    return 'You are likely not related.'
}

console.log(whatRelation(34))
// Should print 'You are likely grandparent and grandchild, aunt/uncle and 
niece/nephew, or half siblings.'

console.log(whatRelation(3))
```

```
// Should print 'You are likely 2nd cousins.'
```

13 . Create a function, `tipCalculator()`, that has two parameters, a string representing the `quality` of the service received and a number representing the `total` cost.

Return the tip, as a number, based on the following:
'bad' should return a 5% tip
'ok' should return a 15% tip
'good' should return a 20% tip
'excellent' should return a 30% tip
all other inputs should default to 18%

```
let result;
function tipCalculator(quality,total){
  if(quality==='bad')
    {
      result=total*5/100;
      return result;
    }

 else if(quality==='ok')
    {
      result=total*15/100;
      return result;
    }

   else if(quality==='good')
    {
      result=total*20/100;
      return result;
    }

     else if(quality==='excellent')
    {
      result=total*30/100;
      return result;
    }
  else{
  result=total*18/100;
```

```
      return result;
  }
}

const show=tipCalculator('loda',100);
console.log(show);
```

14. Write a function, `toEmoticon()`, that takes in a string and returns the corresponding emoticon as a string. Use a switch/case, and cover these cases:

`'shrug'` should return `'|_{"}_|'`
`'smiley face'` should return `':)'`
`'frowny face'` should return`':('`
`'winky face'` should return `';)'`
`'heart'` should return `'<3'`
any other input should return `'|_(* ~ *)_|'`

```
function toEmoticon(bro){
    switch(bro)
           {
           case 'shrug':
           return '|_{"}_|';
           break;

           case 'smiley face':
           return ':)';
           break;

           case 'frowny face':
           return ':(';
           break;


           case 'winky face':
           return ';)';
           break;

           case 'heart':
           return '<3';
           break;
```

```
         default:
         return '|_(* ~ *)_|';

         }
}

const obb=toEmoticon('heart');
console.log(obb);
```