**CODE CHALLENGES: JAVASCRIPT FUNDAMENTALS**

**canIVote()**

**Instructions**

**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.

Hint

You might find a condition like this useful:

if (age >= 18)

// Write your function here:

const canIVote = age =>{

  //(age >= 18) ? (return true) : (return false);

  if (age >= 18){

    return true;

  }

  return false;

}

console.log(canIVote(18));

// Uncomment the line below when you're ready to try out your function

// console.log(canIVote(19)) // Should print true

// We encourage you to add more function calls of your own to test your code!

**agreeOrDisagree()**

**Instructions**

**1.**

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.

Hint

You might use a condition like:

if (first === second)

// Write your function here:

function agreeOrDisagree(one,two){

  if(one===two){

    return 'You agree!';

  }return 'You disagree!';

}

// Uncomment the line below when you're ready to try out your function

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

// Should print 'You agree!'

// We encourage you to add more function call of your own to test your code!

**lifePhase()**

**Instructions**

**1.**

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'

Hint

You’ll need to write multiple if statements.

// Write your function here:

function lifePhase(age){

  if(age>0&&age<=3){

    return 'baby';

  }else if(age>=4&&age<=12){

    return 'child';

}else if(age>=13&&age<=19){

    return 'teen';

}else if(age>=20&&age<=64){

    return 'adult';

}else if(age>=65&&age<140){

    return 'senior citizen';

}return 'This is not a valid age';

}

console.log(lifePhase(4));

**finalGrade()**

**Instructions**

**1.**

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’

Hint

It’s probably a good idea to check that all the grades are valid first.

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

Next, you’ll need to find the average of the arguments that were passed in ((midterm + final + homework)/3).

Now that you have the sum, you can write your control flow to handle returning the letter grade.

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'

    }

}

/\*

//as a function declaration:

function 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'

    }

}

\*/

# reportingForDuty()

**Instructions**

**1.**

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!’

reportingForDuty('Private', 'Fido')

// Should return 'Private Fido reporting for duty!'

Hint

You’ll need to return a single string. You can use concatenation (string1 + " " + string2) or string interpolation (`${string1} ${string2}`) to accomplish this.

// Write your function here:

function reportingForDuty(rank,lastName){

return `${rank} ${lastName} reporting for duty!`

}

// Uncomment the line below when you're ready to try out your function

// console.log(reportingForDuty('Private', 'Fido')) // Should return 'Private Fido reporting for duty!'

// We encourage you to add more function calls of your own to test your code!

# Fix The Broken Code

**Instructions**

**1.**

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?

Hint

We want our code to always return a whole number between 1 and 12. It’s so close! Try using console.log() to figure out what each individual die actually is:

console.log(die1)

You’ll need to use Math.floor to get the functionality we need.

const rollTheDice = () => {

  // Math.random() gives us a random number from 0 up to, but not including, 1

  // We multiplied that by 6 to get a number between 0 and up to, but not including, 6

  // But since we actually wanted numbers from 1 to 6, inclusive, we added 1

    let die1 = Math.random() \* 5 + 1

    let die2 = Math.random() \* 5 + 1

    return die1 + die2

}

**calculateWeight()**

**Instructions**

**1.**

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.'

Hint

For this problem, you may choose to use either a switch or multiple if’s. A switch case takes the format:

switch (expression) {

case "firstCase":

// Do something

case "secondCase":

// Do something

default:

// Do something

}

// Write your function here:

function calculateWeight(earthWeight,planet){

if(planet === 'Mercury'){

    return weight = earthWeight \* 0.378;

  }else if(planet === 'Venus'){

    return weight = earthWeight \* 0.907;

}else if(planet === 'Mars'){

    return weight = earthWeight \* 0.377;

}else if(planet === 'Jupiter'){

    return weight = earthWeight \* 2.36;

}else if(planet === 'Saturn'){

    return weight = earthWeight \* 0.916;

}else{

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

}

}

console.log(calculateWeight(22,'planet'));

// Uncomment the line below when you're ready to try out your function

// console.log(calculateWeight(100, 'Jupiter')) // Should print 236

// We encourage you to add more function calls of your own to test your code!

# truthyOrFalsy()

**Instructions**

**1.**

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**.

Hint

You might find this condition helpful:

if (value) {

// Do something here

}

const truthyOrFalsy = value => {

    if (value) {

        return true

    }

    return false

}

/\*

// As a function declaration:

function truthyOrFalsy(value) {

    if (value) {

        return true

    } else {

        return false

    }

}

// Using a ternary:

const truthyOrFalsy = value => value ? true : false

\*/

# numImaginaryFriends()

**Instructions**

**1.**

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. Check out [the documentation here](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/round) to figure out how it works.

Hint

You’ll need to calculate the number of imaginary friends:

Math.round(totalFriends \* .33)

// Write your function here:

function numImaginaryFriends(totalFriends){

  return Math.round(totalFriends \* .33);

}

// Uncomment the line below when you're ready to try out your function

console.log(numImaginaryFriends(18))

// We encourage you to add more function calls of your own to test your code!

# sillySentence()

**Instructions**

**1.**

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:

sillySentence('excited', 'love', 'functions');

// Should return 'I am so excited because I love coding! Time to write some more awesome functions!'

// Write your function here:

function sillySentence(adj,verb,noun){

  return `I am so ${adj} because I ${verb} coding! Time to write some more awesome ${noun}!`

}

// Uncomment the line below when you're ready to try out your function

console.log(sillySentence('excited', 'love', 'functions'))

// We encourage you to add more function calls of your own to test your code!

**howOld()**

**Instructions**

**1.**

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. Handle three different cases:

* If the year is in the future, you should return a string in the following format:

'You will be [calculated age] in the year [year passed in]'

* If the year is before they were born, you should return a string in the following format:

'The year [year passed in] was [calculated number of years] years before you were born'

* If the year is in the past but not before the the person was born, you should return a string in the following format:

'You were [calculated age] in the year [year passed in]'

Hint

You might find these two variables helpful:

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.

/\*

Our solution is written as a function expression and uses string interpolation, but it would be equally acceptable to use a function declaration and/or string concatenation

\*/

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!

    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}`

    }

}

**Fix the broken code (round 2)!**

**Instructions**

**1.**

Given the percentage of DNA shared between two people, you can [calculate their likely familial relationship](https://isogg.org/wiki/Autosomal_DNA_statistics).

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 niece/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.'

Unfortunately, it’s not working how we want!

whatRelation(34)

// Should return 'You are likely grandparent and grandchild, aunt/uncle and niece/nephew, or half siblings.'

// But instead it's returning 'You are likely 1st cousins.'

whatRelation(3)

// Should return 'You are likely 2nd cousins.'

// But instead it's returning 'You are likely grandparent and grandchild, aunt/uncle and niece/nephew, or half siblings.'

Can you fix our code, please?

Hint

You may think there’s a problem with using only if‘s and not else if‘s, but since we return from each block, that shouldn’t be a problem. But check all the conditions we wrote… there’s something not quite right there…

const whatRelation = percentSharedDNA => {

    if (percentSharedDNA === 100) {

        return 'You are likely identical twins.'

    }

    if (percentSharedDNA >= 35 && percentSharedDNA <= 99) {

        return 'You are likely parent and child or full siblings.'

    }

    if (percentSharedDNA >= 14 && percentSharedDNA <= 34) {

        return 'You are likely grandparent and grandchild, aunt/uncle and niece/nephew, or half siblings.'

    }

    if (percentSharedDNA >= 6 && percentSharedDNA <= 13) {

        return 'You are likely 1st cousins.'

    }

    if (percentSharedDNA >= 2 && percentSharedDNA <= 5) {

        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.'

# tipCalculator()

**Instructions**

**1.**

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%

tipCalculator('good', 100) // Should return 20

Hint

A good way to calculate a percentage is by multiplying it by the percentage divided by 100.

20% of X = X \* .20  
  
For this problem, you may choose to use either a switch or multiple if‘s. A switch case takes the format:

switch (expression) {

case "firstCase":

// Do something

case "secondCase":

// Do something

default:

// Do something

}

const tipCalculator = (quality, total) => {

    switch (quality) {

        case 'bad':

            return total \* .05;

        case 'ok':

            return total \* .15;

        case 'good':

            return total \* .20;

        case 'excellent':

            return total \* .30;

        default:

            return total \* .18;

    }

}

/\*

// As a function declaration:

function tipCalculator(quality, total) {

    switch (quality) {

        case 'bad':

            return total \* .05;

        case 'ok':

            return total \* .15;

        case 'good':

            return total \* .20;

        case 'excellent':

            return total \* .30;

        default:

            return total \* .18;

    }

}

\*/

console.log(tipCalculator('good', 100))

# toEmoticon()

**Instructions**

**1.**

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 '|\_(\* ~ \*)\_|'

Hint

A switch case takes the format:

switch (expression) {

case "firstCase":

// Do something

case "secondCase":

// Do something

default:

// Do something

}

const toEmoticon = meaning => {

    switch (meaning) {

        case 'shrug':

            return '|\_{"}\_|'

        case 'smiley face':

            return ':)';

        case 'frowny face':

            return ':(';

        case 'winky face':

            return ';)';

        case 'heart':

            return '<3';

        default:

            return '|\_(\* ~ \*)\_|';

    }

}

/\*

// As a function declaration:

function toEmoticon(meaning) {

    switch (meaning) {

        case 'shrug':

            return '|\_{"}\_|'

        case 'smiley face':

            return ':)';

        case 'frowny face':

            return ':(';

        case 'winky face':

            return ';)';

        case 'heart':

            return '<3';

        default:

            return '|\_(\* ~ \*)\_|';

    }

}

\*/

console.log(toEmoticon("whatever"))