Search task

rem vs em:

1. em Unit: The em unit allows setting the font size of an element relative to the font size of its parent. When the size of the parent element changes, the size of the child changes automatically.
Note: When em units are used on font-size property, the size is relative to the font-size of the parent. When used on other properties, it's relative to the font-size of that element itself. Here, only the first declaration takes the reference of the parent.

2. rem Unit: The rem is based upon the font-size value of the root element, which is the <html> element. And if the <html> element doesn't have a specified font-size, the browser default value of 16px is used. So here only the value of the root is considered, and there is no relation with a parent element.
Unlike em, here size is relative for all declarations, not only first. Let's understand this with our previous example.

css position:

The `position` property specifies the type of positioning method used for an element.

There are five different position values:

- `static`
- `relative`
- `fixed`
- `absolute`
- `sticky`

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the `position` property is set first. They also work differently depending on the position value.

position: static;

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page

position: relative;

An element with `position: relative;` is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

position: fixed;

An element with `position: fixed;` is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

position: absolute;

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

position: sticky;

An element with `position: sticky;` is positioned based on the user's scroll position.

A sticky element toggles between `relative` and `fixed`, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

for vs while loop:

Here are few differences:

For loop	While loop
Initialization may be either in loop statement or outside the loop.	Initialization is always outside the loop.
Once the statement(s) is executed then after increment is done.	Increment can be done before or after the execution of the statement(s).
It is normally used when the number of iterations is known.	It is normally used when the number of iterations is unknown.
Condition is a relational expression.	Condition may be expression or non-zero value.
It is used when initialization and increment is simple.	It is used for complex initialization.
For is entry controlled loop.	While is also entry controlled loop.
for (init ; condition ; iteration) { statement(s); }	while (condition) { statement(s); }
used to obtain the result only when number of iterations is known.	used to satisfy the condition when the number of iterations is unknown

Object method :

An object is a collection of key/value pairs or properties. When the value is a function, the property becomes a method. Typically, you use methods to describe the object behaviors.

For example, the following adds the greet method to the person object

- When a function is a property of an object, it becomes a method.

- Objects in JavaScript are collections of **key/value** pairs. The values can consist of **properties** and **methods**, and may contain all other JavaScript data types, such as strings, numbers, and Booleans.
- All objects in JavaScript descend from the parent `Object` constructor. `Object` has many useful built-in methods we can use and access to make working with individual objects straightforward. Unlike Array prototype methods like `sort()` and `reverse()` that are used on the array instance, Object methods are used directly on the `Object` constructor, and use the object instance as a parameter. This is known as a static method.
- This tutorial will go over important built-in object methods, with each section below dealing with a specific method and providing an example of use.

regular vs arrow function js:

usual way, is by using the `function` keyword

available starting ES2015, is the *arrow function* syntax

Arrow functions – a new feature introduced in ES6 – enable writing concise functions in JavaScript. While both regular and arrow functions work in a similar manner, yet there are certain interesting differences between them, as discussed below.

Regular functions created using function declarations or expressions are ‘constructible’ and ‘callable’. Since regular functions are constructible, they can be called using the ‘new’ keyword. However, the arrow functions are only ‘callable’ and not constructible. Thus, we will get a run-time error on trying to construct a non-constructible arrow functions using the new keyword.

objects vs instance oop:

Instance: instance means just creating a reference(copy).

object: means when memory location is associated with the object (is a run-time entity of the class) by using the new operator.

In simple words, Instance refers to the copy of the object at a particular time whereas object refers to the memory address of the class.

- **Object** : *physical presence of the class in memory*
- **Instance** : *an unique copy of the object (same structure, different data)*

Object

An Object is created from a Class, like a house is created from a blueprint. You could use a blueprint of a 3 bedroom home to build multiple 3 bedroom homes - 1 blueprint to create multiple 3 bedroom homes. This is what happens with a Class. You can create multiple Objects from your single Class - remember, a Class is just the blueprint for creating Objects of the same type.

Instance

So we've taken our blueprint (Class), and created multiple 3 bedroom homes (Objects), but how do we refer to one particular home (Object) we've built? We refer to a particular Object as an Instance. If I were to paint the frontdoor of one of the homes, I would be painting the door of a particular Instance of the Objects (home) created from my Class (blueprint).