

Different between EM & REM

EM	REM
Is Relative to the parent element's font size, so If You wish to scale the element's size based on Its parent's size use EM	Is Relative to the root (HTML) font size, so if you wish to scale the element's size based on Its root size, no matter what is the parent size is use use REM

CSS position

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

STATIC: By default, HTML elements are static in position. The top, bottom, left, and right properties have no effect on static positioned elements.

RELATIVE: When an element is position relative, it stays in the flow of the document, and you can move it horizontally and vertically relative to its normal position.

Other content will not be adjusted to fit into any gap left by the element.

ABSOLUTE: This element will also be taken out of the normal flow, similar to fixed but it is affected by scroll, and you can have it be relative to the parent, unlike fixed which is relative to the document.

FIXED - This element will be taken out of the normal flow, and will be placed relative to the document and will not be affected by scrolling.

STICKY: This one behaves like relative until you scroll to a certain point and it will behave like a fixed position element.



FOR VS WHILE LOOP

FOR	WHILE
A for loop repeats until a specified condition evaluates too false.	A while statement executes its statements as long as a specified condition evaluates to true.
For loop defined when the number of iterative is known.	while loop defined when the number of iterative isn't know.
Absence of Condition: Goes into interaction mode for infinite repetitions	If the condition becomes false, statement within the loop stops executing and control.
Condition: Relational expression	Absence of condition displays error.
	Condition: An expression or non-zero value

OBJECT METHODS

The Object type represents one of data types of JavaScript. It is used to store various keyed collections and more complex entities.

- If the value is null or undefined, it will create and return an empty object.
- If the value is an object already, it will return the value.

STATIC METHODS

- **Object. assign()** Copies the values of all enumerable own properties from one or more source objects to a target object.
- **Object. create ()** Creates a new object with the specified prototype object and properties.
- **Object.defineProperty()** Adds the named property described by a given descriptor to an object.
- **Object.defineProperties()** Adds the named properties described by the given descriptors to an object.
- **Object. entries ()** Returns an array containing all of the [key, value] pairs of a given object's own enumerable string properties.
- **Object.keys()** Returns an array containing the names of all of the given object's own enumerable string properties.
- **Object.values()** Returns an array containing the values that correspond to all of a given object's own enumerable string properties.

- **Object.freeze()** Freezes an object. Other code cannot delete or change its properties.
- **Object.seal()** Prevents other code from deleting properties of an object.

OBJECT MAP

- **Sets** the value for a key in a Map.
- **Gets** the value for a key in a Map.
- **entries ()** Returns an iterator object with the [key, value] pairs in a Map.
- **Keys** Returns an iterator object with the keys in a Map.
- **Values** Returns an iterator object of the values in a Map.

REGULAR VS ARROW FUNCTION JS

ARROW FUNCTION	REGULAR FUNCTION
Arrow functions don't have their own bindings to this, arguments and shouldn't be used as methods.	The function keyword can be used to define a function inside an expression.
Arrow functions don't have access to the new.target keyword , aren't suitable for call methods	Function must have a return statement that specifies the value to return. A function without a return statement will return a default value.
Arrow functions cannot be used as constructors and yield, within its body.	In the case of a constructor called with the new keyword, the default value is the value of its this parameter. For all other functions, the default return value is undefined.
The main benefit is that it removes the several pain points associated with the this operator.	One of the benefits of creating a named function expression is that in case we encountered an error, the stack trace will contain the name of the function, making it easier to find the origin of the error.

OBJECT VS INSTANCE OOP

OBJECT OOP	INSTANCE OOP
<p>Object: means when memory location is associated with the object (is a run-time entity of the class) by using the new operator.</p> <p>Once using new class, that instantiated thing becomes an object. An object is something that can adhere to encapsulation, polymorphism, abstraction principles of OOP and the real thing a program interacts with to consume the instance members defined in class. Object contains instance members (non-static members).</p>	<p>On its own, a class doesn't do anything: it's a kind of template for creating concrete objects of that type. Each concrete professor we create is called an instance of the (CLASSNAME). The process of creating an instance is performed by a special function called a constructor. We pass values to the constructor for any internal state that we want to initialize in the new instance.</p> <p>Instance refers to a unique copy of the object (same structure, different data). An instance is a specific representation of an object. An object is a generic thing while an instance is a single object that has been created in memory. Usually, an instance will have values assigned to its properties that differentiates it from other instances of the type of object.</p>