**What is interface?**

Interfaces are generally an executable format for classes. By defining the general specifications of the methods, without implementing them, it requires the derived classes to fully implement those methods. Therefore, only the properties of the methods are defined once in the Interface, and where necessary, after inheriting, their methods are implemented. In all .NET versions, interfaces begin with the letter I and are separated from other components by this attribute. Its definition is very similar to classes; But with the same difference as above, their methods have no code. Interfaces do not have a constructor or a field, and no instance can be created from them.

**What is Abstract class?**

The Abstract class is one of the most important OOP tools that cannot create instance. In other words, we can’t define a variable of the Abstract class. An Abstract class is similar to the Interface but it's more expanded. These classes can have Abstract methods that are similar to the Interface declared only and must be rewritten in derived classes. Of course, you can have methods in these classes that are not Abstract and do not need to be implemented in derived classes.

It should be noted that only abstract class methods are required to implement when the word abstract is explicitly mentioned in the definition of that method. In fact, these methods do not need to be implemented. That is, they can also be mentioned in the subclass as abstract. Provided that the subclass is also defined as abstract. The abstract class can also have simple or non-abstract methods. As you know, non-abstract methods must have a body and do not need to be implemented. So, the abstract class can have both methods that need to be implemented and methods that do not need to be implemented. According to the above definitions, the Abstract class is a state between ordinary classes and interfaces, and it is a class that is indefinite and incomplete and must be completed at the level of its children.

### ****Difference between Abstract and interface:****

1. A regular class can only inherit from one Abstract class, but it can inherit from multiple interfaces.

2. An interface can only declare methods and properties; But an Abstract class in addition to them can have methods and properties with full code.

3- The elements in the Abstract class can have an access level like a normal class; But interfaces do not have this feature.

4. When you add a method to the Abstract class, it is automatically applied to all subclasses; But in the Interface, if you add a method, you must apply it to all subclasses.

5. Abstract classes, like regular classes, can have fields and other elements (such as constants); While an interface does not have this feature. The abstract class can also contain a constructor, but the interface cannot.

6- Abstract is one of the types of classes; But Interface is not a class.

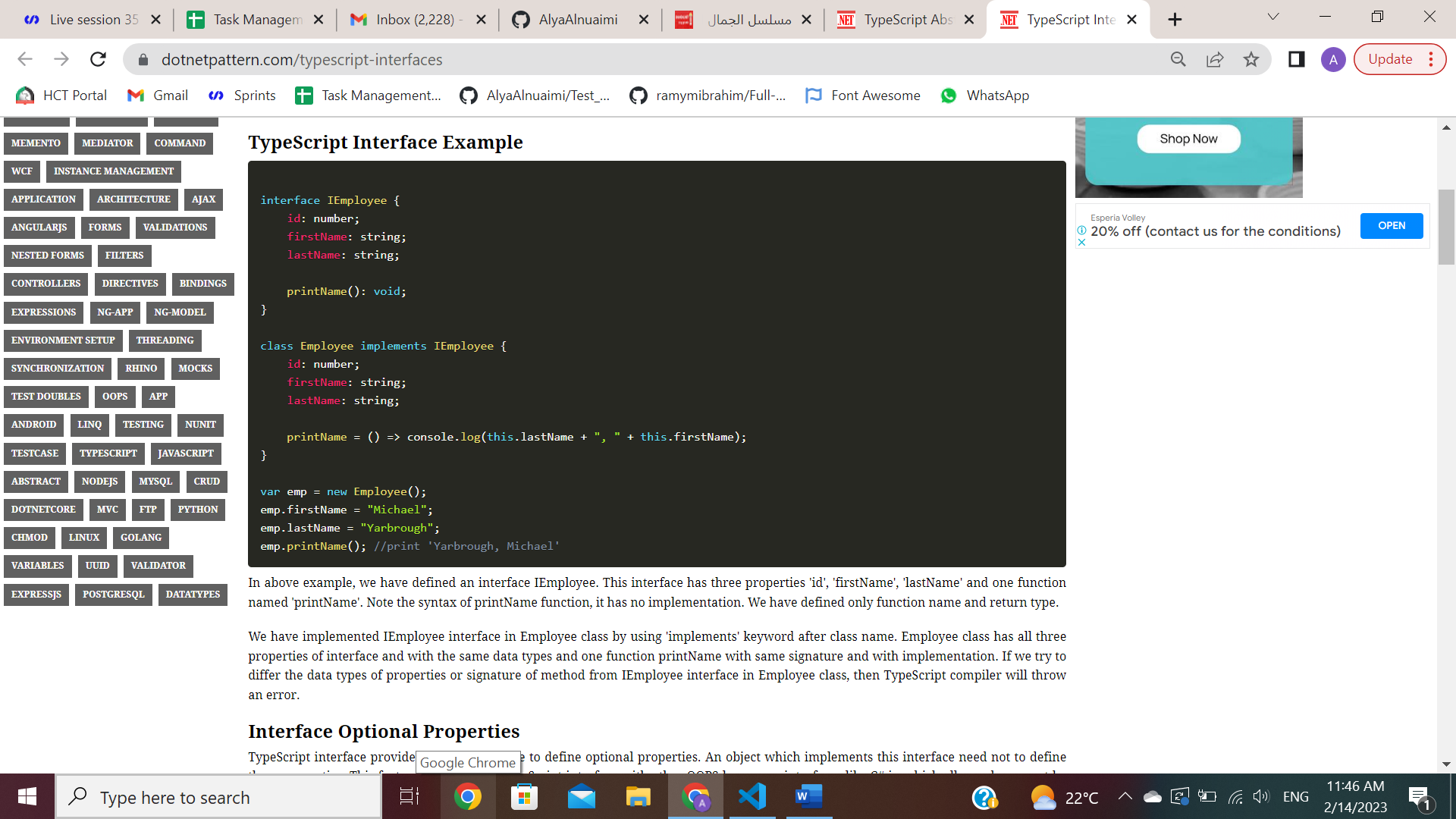
7. The interface can only inherit from the interface, but the abstract class can inherit from the interface, the Abstract class, or other classes.

**Example of Abstract class:**



In above example, we have created an abstract class BaseEmployee. We have one abstract method doWork which we do not provide any implementation. Employee class extends BaseEmployee class. In Employee class constructor we call BaseEmployee constructor using super method. super method is used to call base class constructor. super method takes the same parameters as defined in base class constructor. In last two lines, we have created an instance of Employee class and call doWork method.

**Example of interface:**



In above example, we have defined an interface IEmployee. This interface has three properties 'id', 'firstName', 'lastName' and one function named 'printName'. Note the syntax of printName function, it has no implementation. We have defined only function name and return type. We have implemented IEmployee interface in Employee class by using 'implements' keyword after class name. Employee class has all three properties of interface and with the same data types and one function printName with same signature and with implementation. If we try to differ the data types of properties or signature of method from IEmployee interface in Employee class, then TypeScript compiler will throw an error.