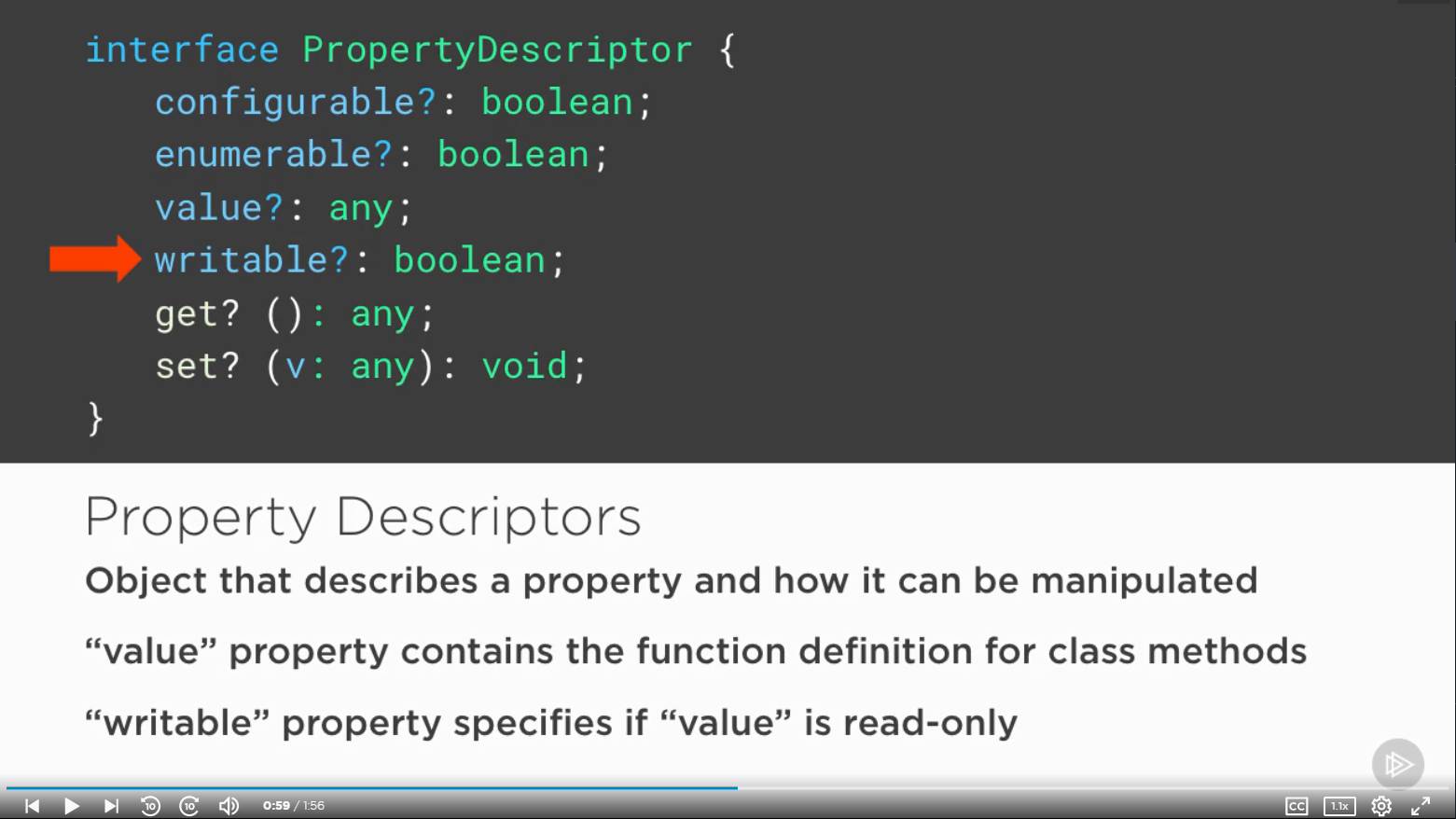
Advanced Course

* Polymorphic this types :-
  + Polymorphic this type is represents a type that is the subtype of containing class or interface.
  + Help build fluent api in ts.
  + Help build hirerachical api’s if the methods return object instance calling methods on the object created.
  + Eg:- Library Book is polymorphic as it can be library or children book depending on instance returned.
  + Able to call removeFromCustDevice() as checkin returns instance of electronic book
* Declaration Merging :-
  + Compiler merges two separate declaration declared with same name into single definition.
  + Allowed merges :- Interfaces,Enums, namespace, namspaces with functions, namspaces with classes, namspaces with enums,
  + Disallowed merges :- classes with classes(this can be done by mixins but)
  + Interface merging example can be seen in eg project for interface of book.
  + Module merging is augmentation and allows new properties with existing module.
  + When using a third party library missing method required by you you can extend by merging declaration eg:- universitylibrarian in extenson file.
* Type Guards :-
  + Uses js typeof operator. Compare result of typeof operator to a type name, where type name is only string,number,Boolean or symbol
  + Uses instanceof operator. We have used to check when var is Union o two types this instance can tell whose instance is it.
  + Custom guards can also be created look in demo for app.ts having custom guard example. Interfaces are good place where both of them don’t work i.e. typeof and instanceof.
  + Symbols created in ES2015, Immutable, Unique and fall in primitive types. Used in js as unique constant as enums, computed property declarations can be string but uniqueness is when you don’t have to deal wth name collisons, to customize internal language behavior.So target in compilation to be ES2015.
* Decorators :-
  + Proposed feature of js. Form of decalartive programming to tell what to do and not how. implemented as functions. They can be attached to classes, methods, accessors, properties, parameters.
  + Still experimental so as experimentalDecorators compiler options to it.
  + Method decorators have 3 parameters 1st is constructor function for static or prototype of class if decorats for instance method ,2nd is decorated property, 3rd is propertydescriptor for property.
  + Class decorator don’t need any parameter as class constructor passes value as the input to decorator function. Also for function decorator parameters will be directly passed directly by ts.
  + Decorator factories helps applying additional properties when decorator is applied.
  + PropertyDecorator parameters are different like 1st constructor function of prototype instance of mem the class belongs to, 2nd is name of decorated property.
  + ParameterDecorator are same as property with additional 3rd param which is index of property.
  + 
  + 