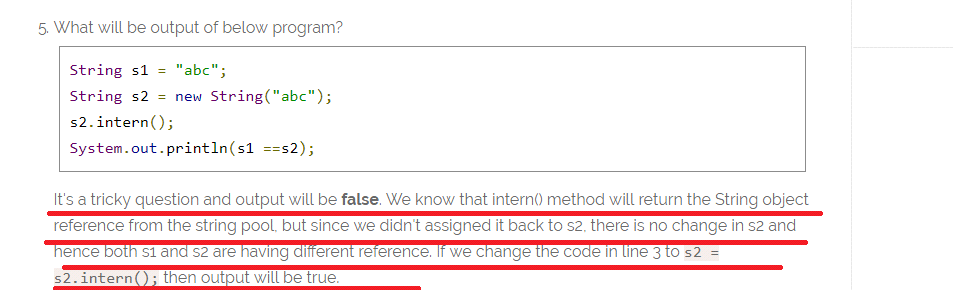
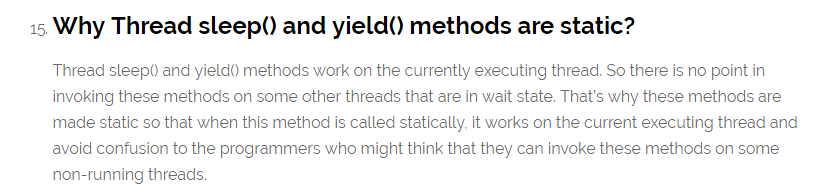
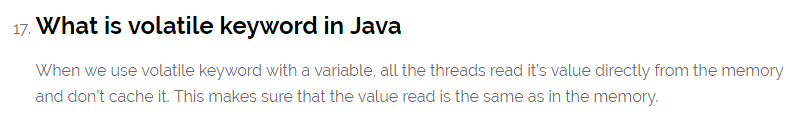
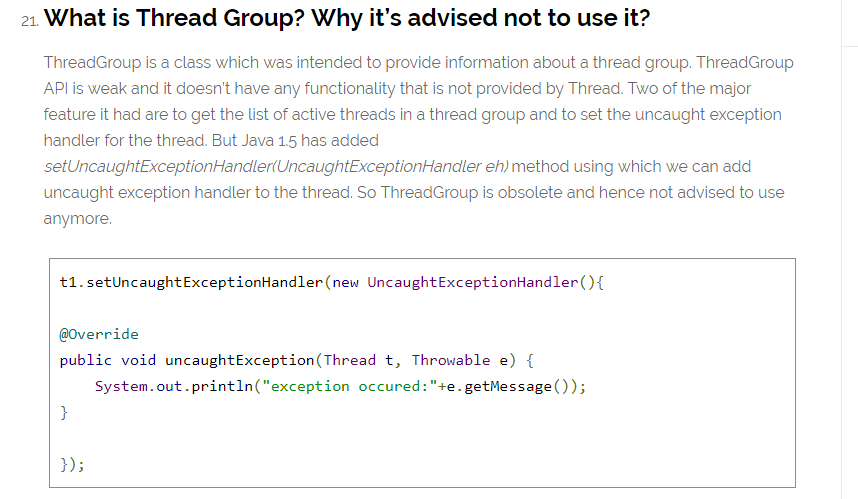
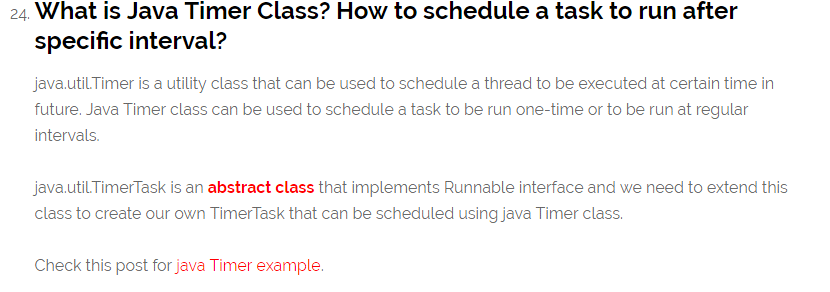
Journal Dev- String

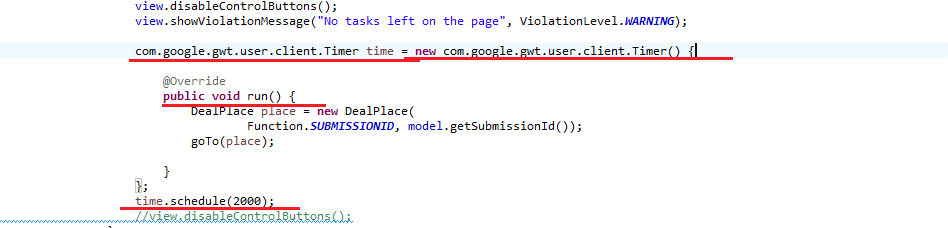
* 
* System.out.println("substring equals subSequence ? " + (str.substring(4, 14).equals(str.subSequence(4, 14))));

For the point 2 output is “**true”**

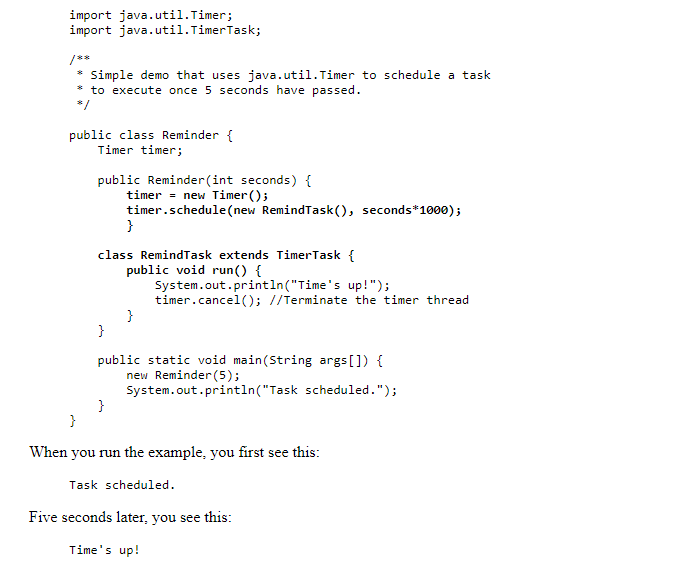
Journal Dev- Thread

Part -1 Basic

* 
* 
* Java ThreadLocal is used to create thread-local variables.
* 
* Avoid Nested Locks, Lock Only What is Required and Avoid waiting indefinitely are common ways to avoid deadlock situation, read this post to learn how to [analyze deadlock in java](https://www.journaldev.com/1058/deadlock-in-java-example) with sample program.  **needed practical example on how deadlock created and how it can be avoid**
* 

An Example on GWT Timer 

An Example on Timer and TimerTAsk



**java.util.TimerTask** is an [**abstract class**](https://www.journaldev.com/1582/abstract-class-in-java) that implements Runnable interface and we need to extend this class to create our own **TimerTask** that can be scheduled using *java Timer* class.

package com.journaldev.threads;

import java.util.Date;

import java.util.Timer;

import java.util.TimerTask;

public class MyTimerTask extends TimerTask {

@Override

public void run() {

System.out.println("Timer task started at:"+new Date());

completeTask();

System.out.println("Timer task finished at:"+new Date());

}

private void completeTask() {

try {

//assuming it takes 20 secs to complete the task

Thread.sleep(20000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

public static void main(String args[]){

TimerTask timerTask = new MyTimerTask();

//running timer task as daemon thread

Timer timer = new Timer(true);

timer.scheduleAtFixedRate(timerTask, 0, 10\*1000);

System.out.println("TimerTask started");

//cancel after sometime

try {

Thread.sleep(120000);

} catch (InterruptedException e) {

e.printStackTrace();

}

timer.cancel();

System.out.println("TimerTask cancelled");

try {

Thread.sleep(30000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

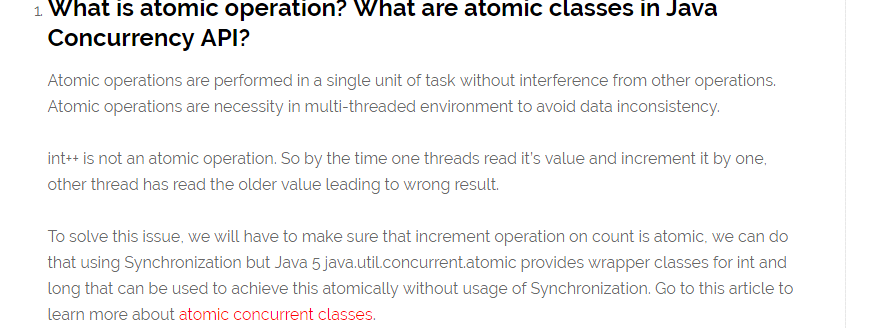
Notice that one thread execution will take 20 seconds but Java Timer object is scheduled to run the task every 10 seconds. Here is the output of the program:

Timer *cancel()* method is used to terminate the timer and discard any scheduled tasks, however it doesn’t interfere with the currently executing task and let it finish. If the timer is run as [daemon thread](https://www.journaldev.com/1072/daemon-thread-in-java), whether we cancel it or not, it will terminate as soon as all the user threads are finished executing.

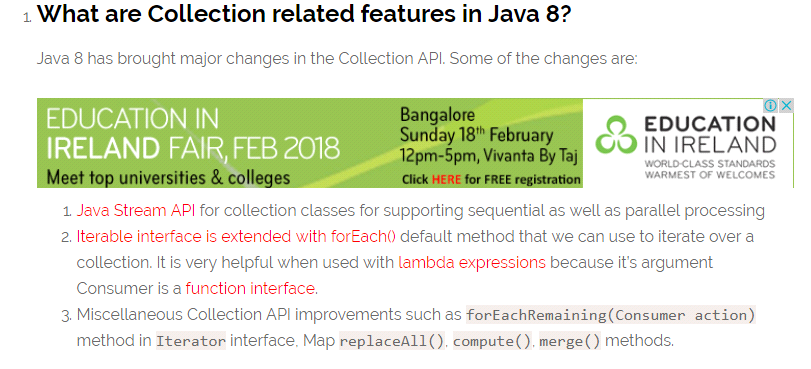
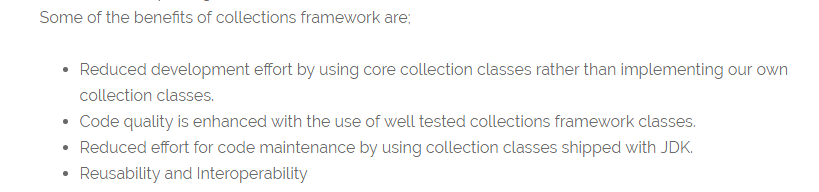
Timer class contains several **schedule**() methods to schedule a task to run once at given date or after some delay. There are several **scheduleAtFixedRate**() methods to run a task periodically with certain interval

While scheduling tasks using Timer, you should make sure that time interval is more than normal thread execution, otherwise tasks queue size will keep growing and eventually task will be executing always. That’s all for a quick roundup on Java Timer and Java TimerTask.

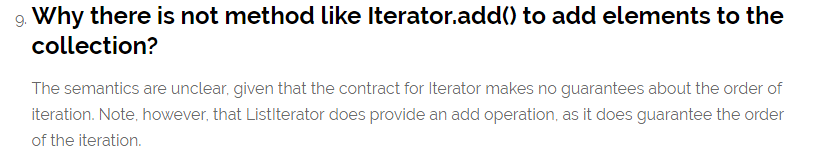
**Part 2: Java Concurrency Interview Questions and Answers**

* 
* **NOTE FEW MORE QUESTIONS ARE PRESENT ON CONCURRENCY BUT ITS QUITE HIGH LEVEL**

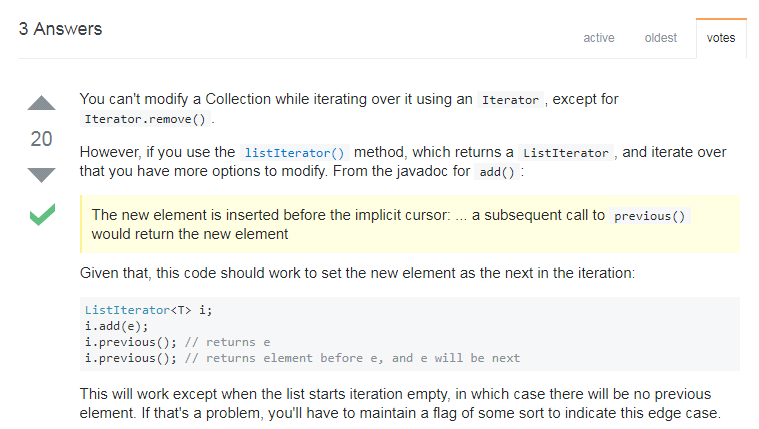
JOURNALdEV-Collection

* 
* 
* What is the purpose of  Cloneable and Serializable interfaces?

That said, the one thing it does is "enable" the default implementation of clone() in Object . If you don't implement Cloneable then invoking **super.clone**() **will** throw a**CloneNotSupportedException** . Purpose of **clone() method** is create a new instance (**copy**) of object on which it is called

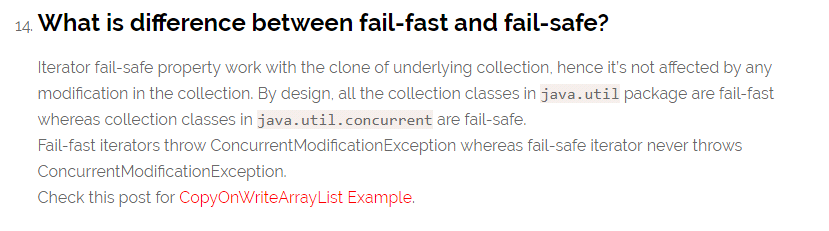
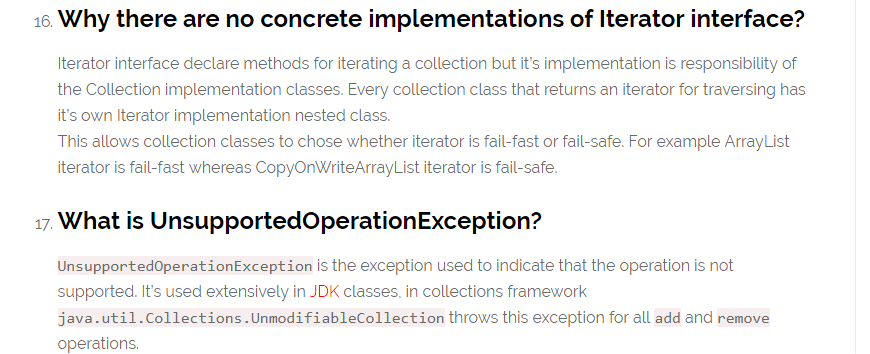
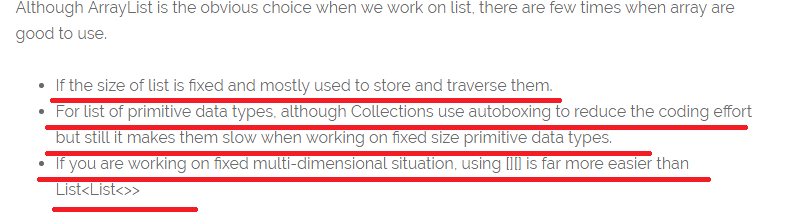
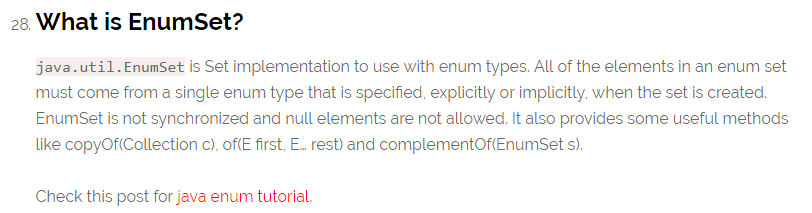
* 

Note from Stack Overflow



* What are different ways to iterate over a list?

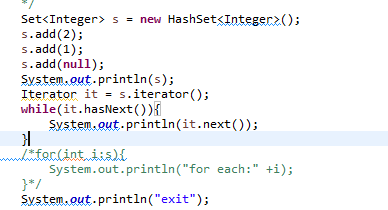
We can iterate over a list in two different ways – using iterator and using for-each loop.

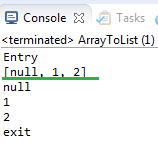
* 
* 
* 
* 
* 

WILL THE SET ALLOWS NULL?

We have two scenarios normatlly set will allow one null(in case of more null then overriding will happen)

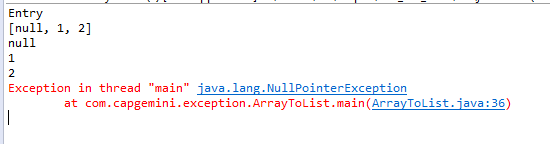
Case 1:



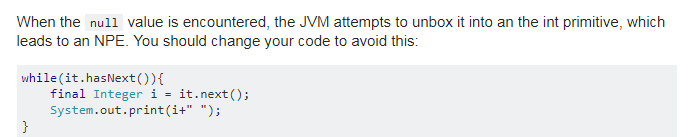
OUTPUT NO EXCEPTION 

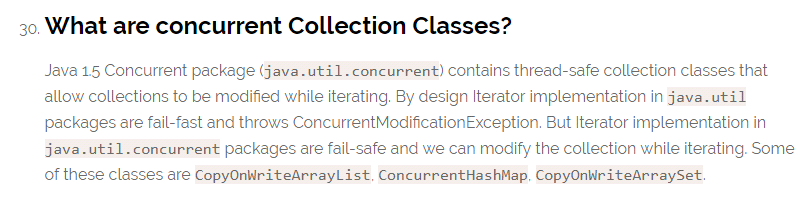
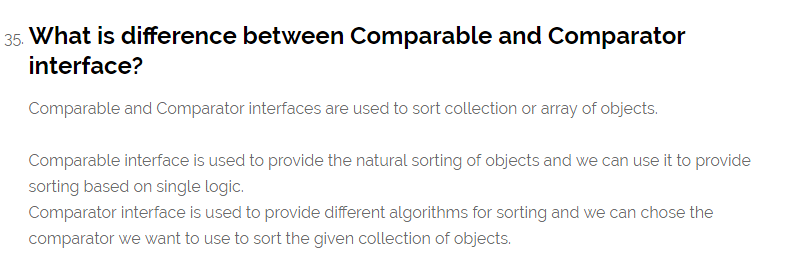
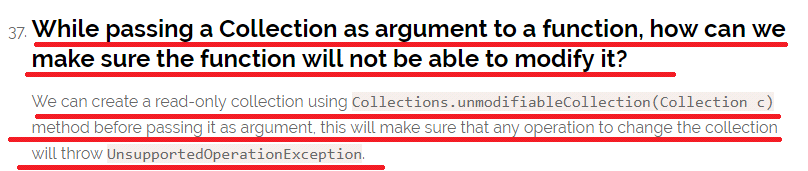
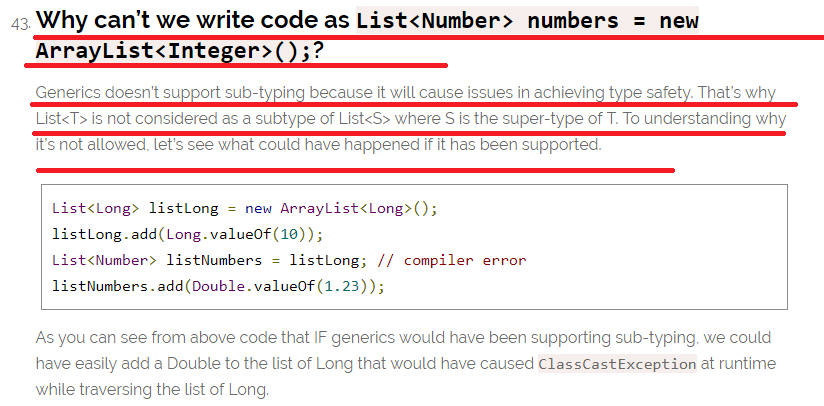
CASE 2: NOW LETS SEE THE ERROR SCENARIO BY UNCOMMING THE ABOVE SCREEN SHOT CODE

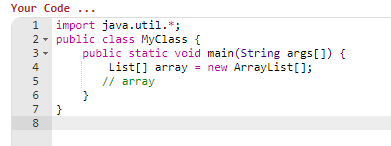


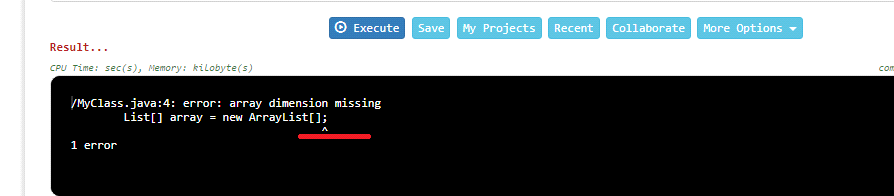
OUTPUT 

THE REASON FOR THE EXCEPTION IS FROM STACK OVERFLOW IS

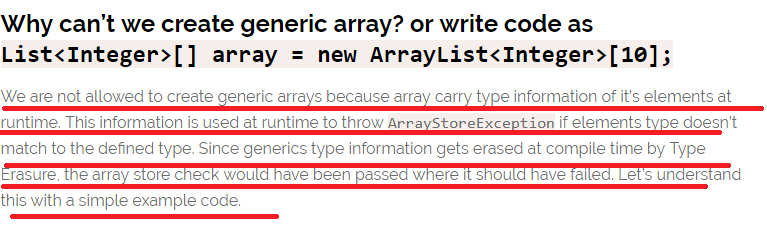


* 
* 
* 
* 
* What is the Output of the following code?

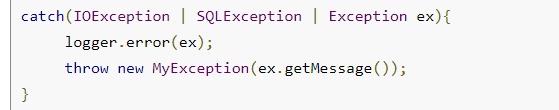
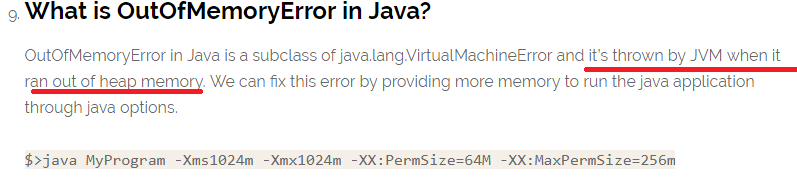
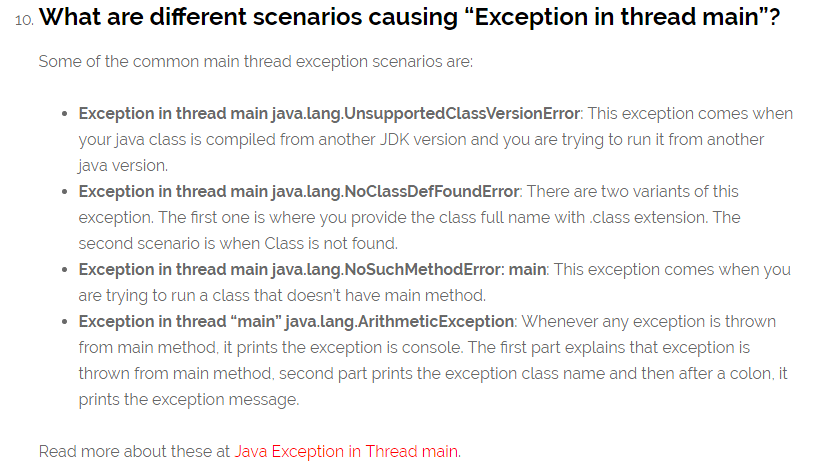
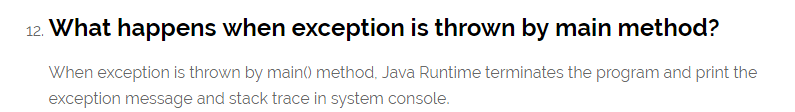


Output 

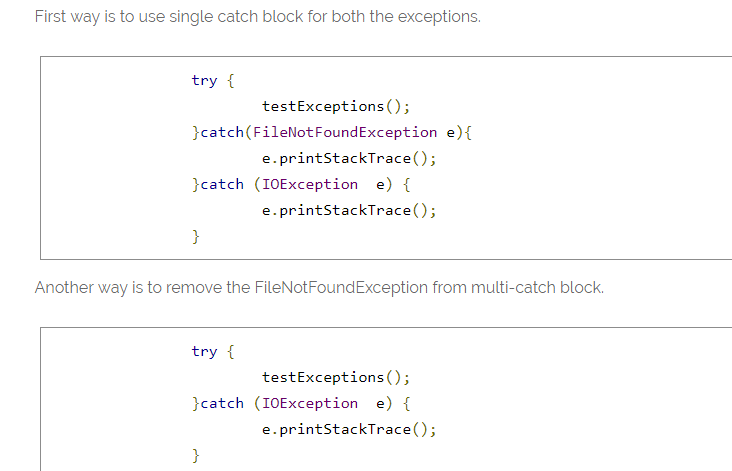
**See here though array created is of Collection type it requires fixed Size.**

* 

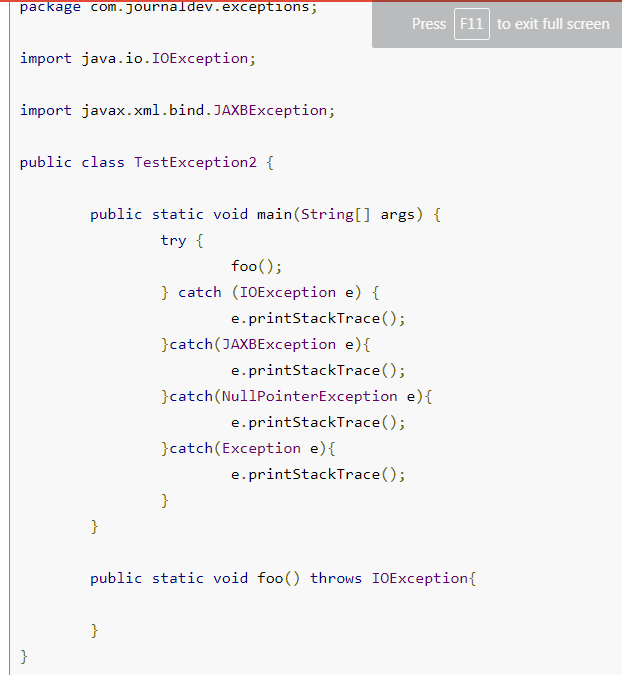
Journal Dev- Exception Interview Questions

* Exception is an error event that can happen during the execution of a program and disrupts it’s normal flow.
* Explain Java 7 ARM Feature and multi-catch block? 
* 
* What about StackOverFlow?
* 
* final keyword can be used with class variables so that they can’t be reassigned
* 
* 

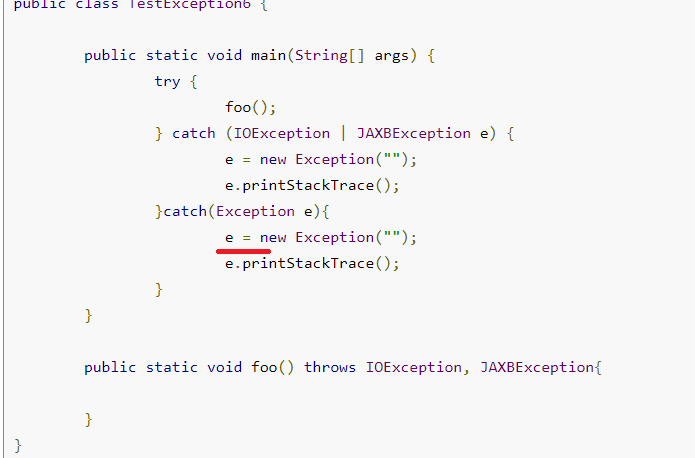
Above program won’t compile and you will get error message as “The exception FileNotFoundException is already caught by the alternative IOException”. This is because FileNotFoundException is subclass of IOException, there are two ways to solve this problem.



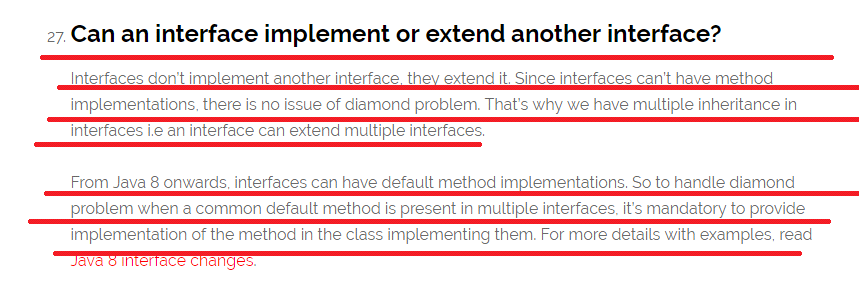
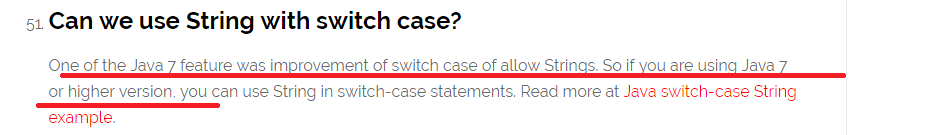
* What is the output for the below program?

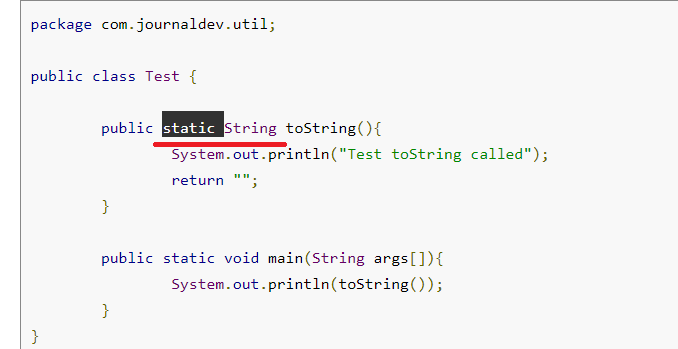


**The program won’t compile because JAXBException is a checked exception and foo() method should throw this exception to catch in the calling method. You will get error message as “Unreachable catch block for JAXBException. This exception is never thrown from the try statement body”. To solve this issue, you will have to remove the catch block of JAXBException.**

* **exception object in multi-catch block is final and we can’t change it’s value.  is from Java 7**
* What is the output for the below code? the red colour underlined line throws an exception

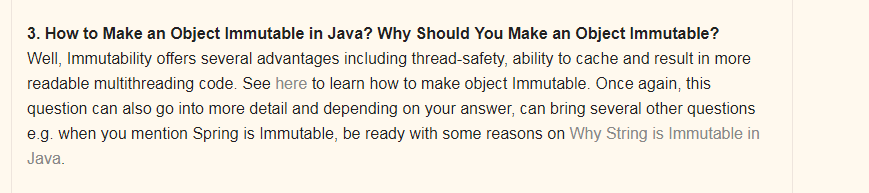
Now Let’s see the Journal Dev general Code Java Questions

* 
* Java Reflection API provides ability to inspect and modify the runtime behavior of java application
* How to sort a collection of custom Objects in Java? without using the build function
* 
* What is the output for the following code

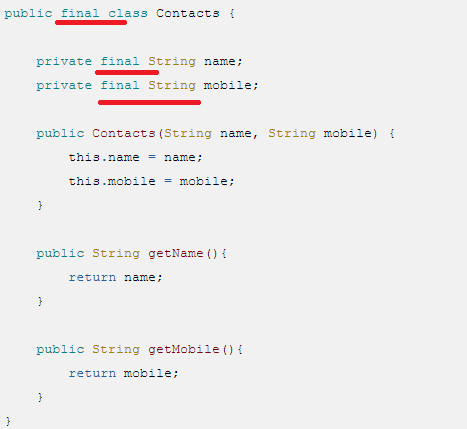


Answer**: The code won’t compile because we can’t have an Object class method with static keyword. Note that Object class has toString() method. You will get compile time error as “This static method cannot hide the instance method from Object”. The reason is that static method belongs to class and since every class base is Object, we can’t have same method in instance as well as in class. You won’t get this error if you change the method name from toString() to something else that is not present in super class Object.**

Interview Questions from <http://www.java67.com/2013/07/15-advanced-core-java-interview-questions-answers-senior-experienced-5-6-years-programmers-developers.html>

* 

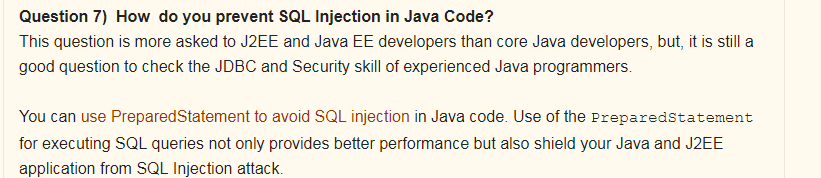
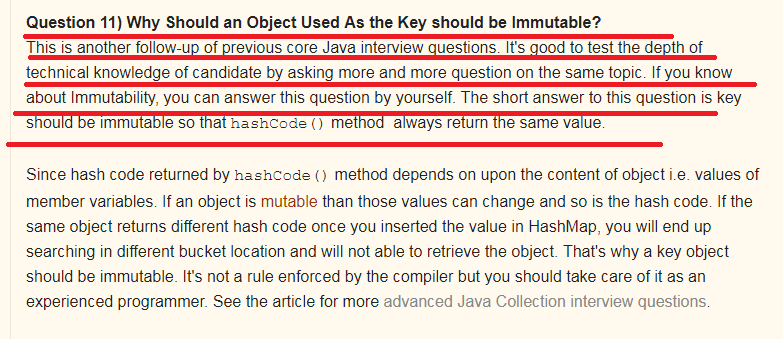
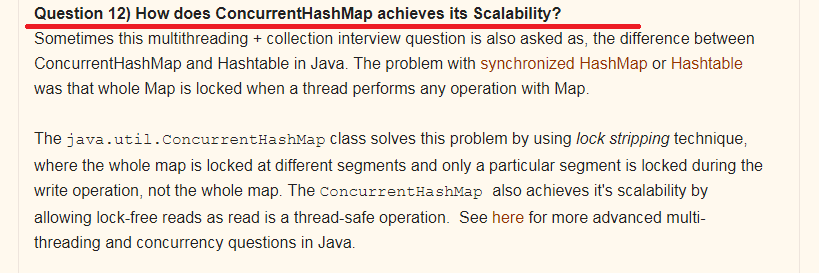
NOW LETS SEE HOW TO MAKE AN OBJECT IMMUTABLE WHAT IS THE DIFFERENCE BETWEEN MAKING A CLASS AS IMMUTABLE OR MAKING AN OBJECT AS IMMUTABLE

* Syntax to create a immutable class

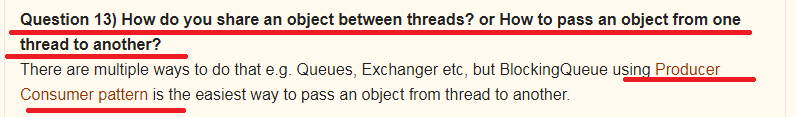
Explanation

* This Java class is immutable, because its state can not be changed once created.
* You can see that all of its fields are final. NOTE WE HAD MARKED IN THE CODE AS FINAL BUT I DID HANDS ON THAT IT WAS NOT FINAL BY DEFAULT,   
    
  Read more: [http://javarevisited.blogspot.com/2013/03/how-to-create-immutable-class-object-java-example-tutorial.html#ixzz53s9qcmAf](http://javarevisited.blogspot.com/2013/03/how-to-create-immutable-class-object-java-example-tutorial.html)

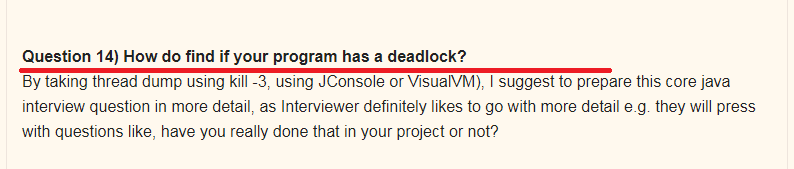
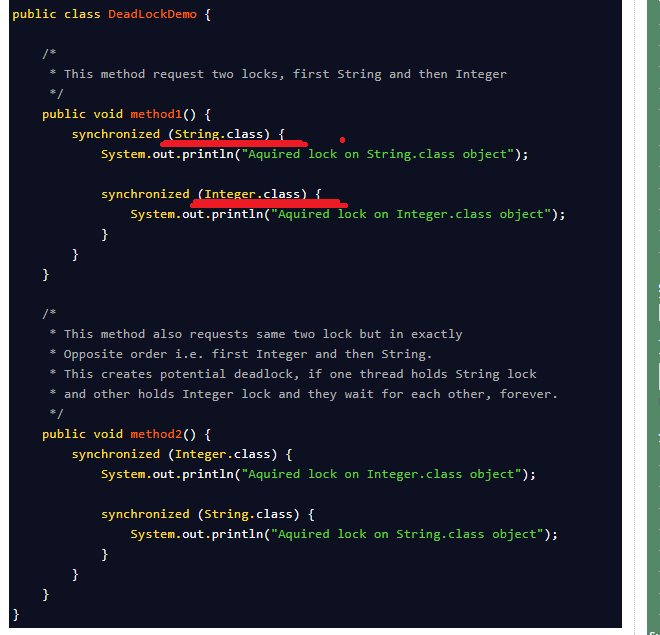
Doubt

* Which Design Patterns have You Used in Your Java Project?  
    
  Always expect some design patterns related question for Core Java Interview of senior developer position. It's a better strategy to mention any GOF design pattern rather than Singleton or MVC, which almost every other Java developer use it.   
    
  Your best bet can be [Decorator pattern](http://java67.blogspot.sg/2013/07/decorator-design-pattern-in-java-real-life-example-tutorial.html) or may be [**Dependency Injection Pattern**](http://javarevisited.blogspot.sg/2012/12/inversion-of-control-dependency-injection-design-pattern-spring-example-tutorial.html)**, NOTE WE CAN TALK ABOUT THE Dependency Injection Design Pattern.**
* 
* 
* 

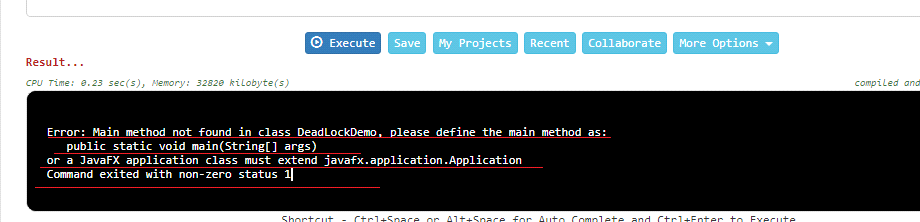
Explanation

* 

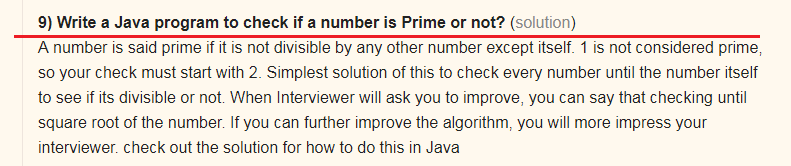
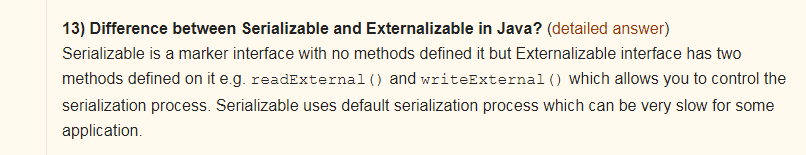
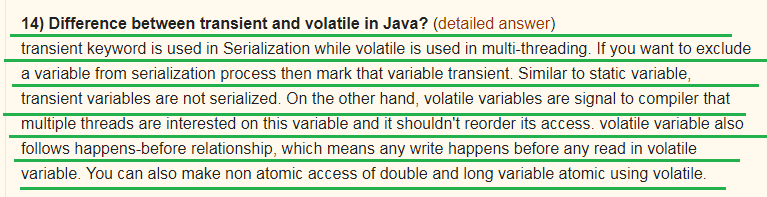
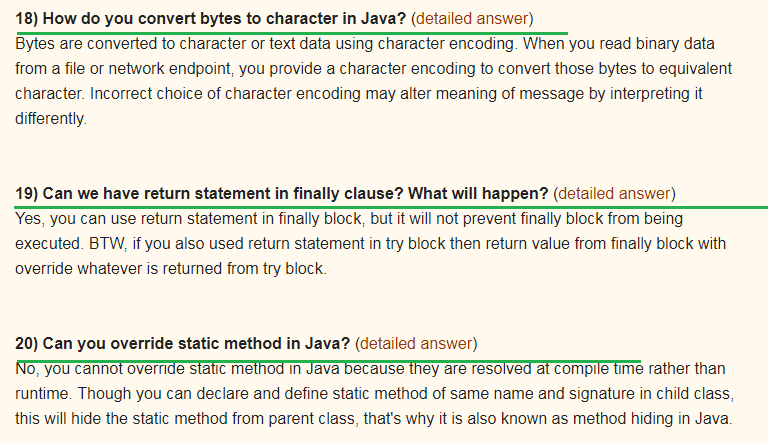
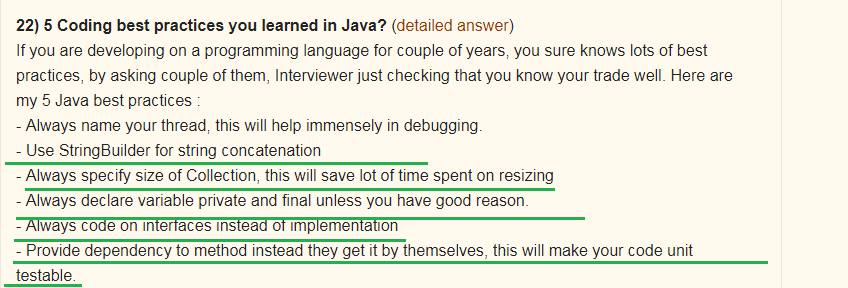
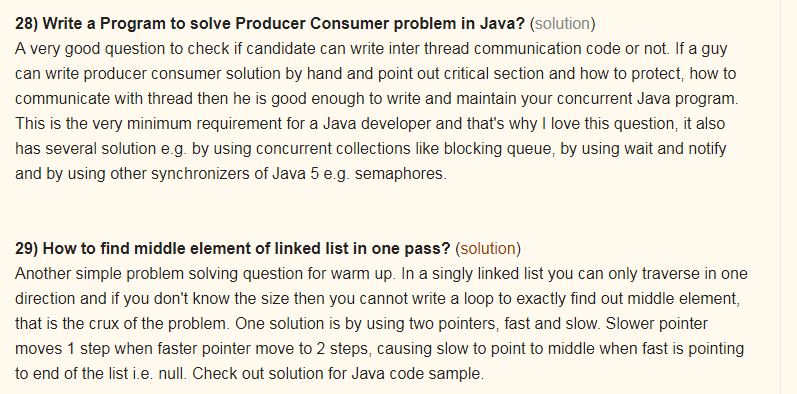
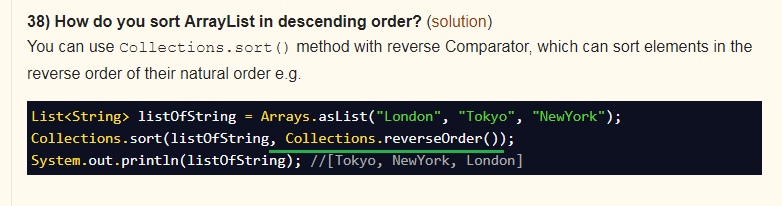
Explanation

*  Explanation:
* Write a Java program which will result in deadlock?  
  

**See here usually we have seen the syntax of methodName(Datatype variable) but here it is in the format methodName(ClassNAme.class)**

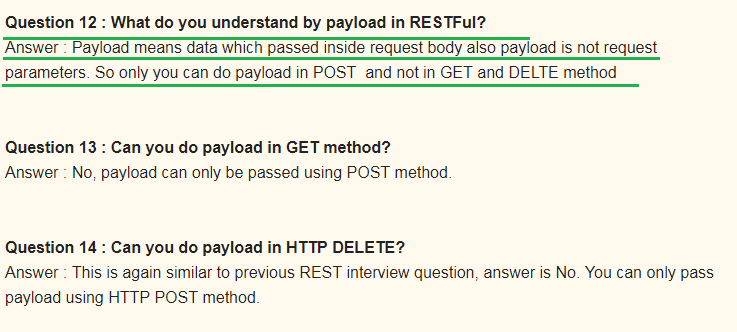
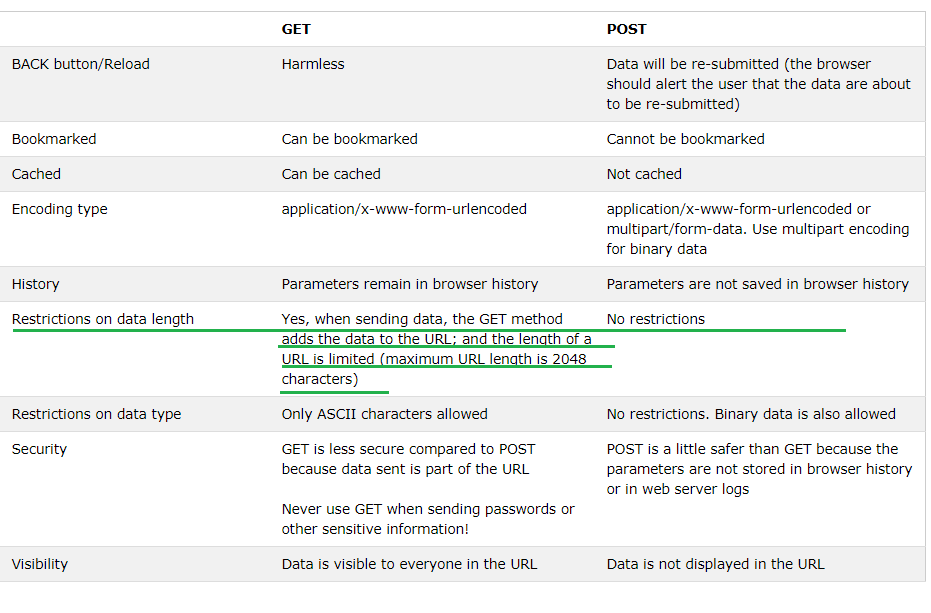
* What will be the Output if we run the Application Without Main Method

Core Java Interview Questions from <http://www.java67.com/2015/03/top-40-core-java-interview-questions-answers-telephonic-round.html>

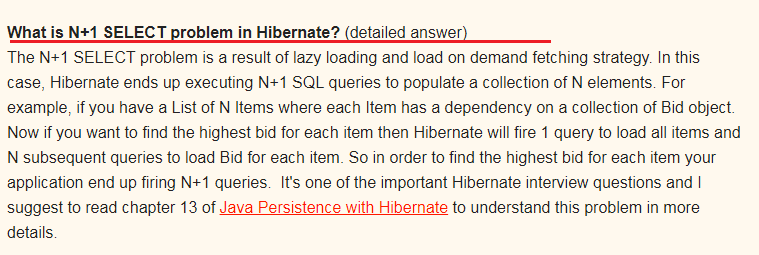
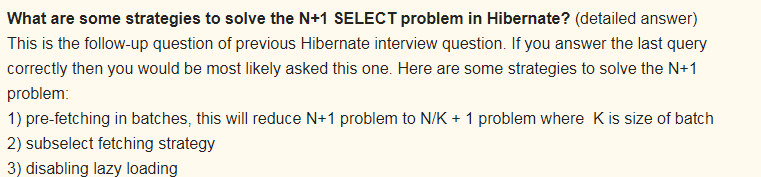
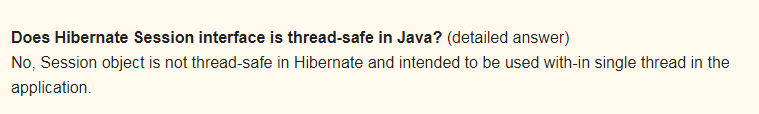
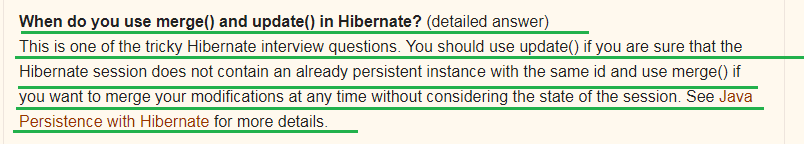
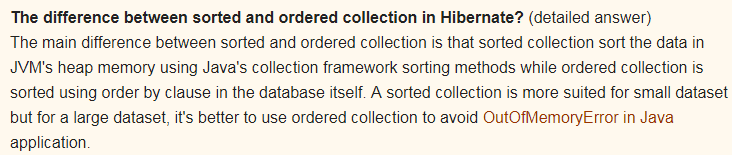
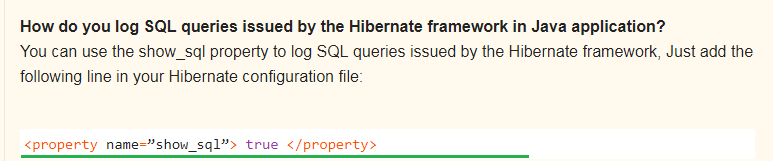
* Logic for Fibonacci series sum of previous two numbers i.e. f(n) = f(n-1) + f(n-2).
* 
* **11) How to check if linked list contains loop in Java?** ([solution](http://javarevisited.blogspot.sg/2013/05/find-if-linked-list-contains-loops-cycle-cyclic-circular-check.html))
* 
* 
* 
* 
* 
* **28) Write a Program to solve Producer Consumer problem in Java?** ([solution](http://java67.blogspot.sg/2012/12/producer-consumer-problem-with-wait-and-notify-example.html))
* **29) How to find middle element of linked list in one pass?** ([solution](http://javarevisited.blogspot.sg/2012/12/how-to-find-middle-element-of-linked-list-one-pass.html))
* 

Rest Based Interview Question from <http://www.java67.com/2015/09/top-10-restful-web-service-interview-questions-answers.html>

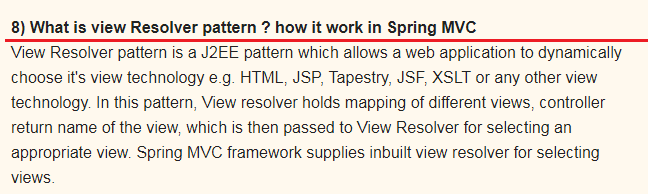
Rest Service Questions

* Question 3: What is HTTP Basic Authentication and how it works?
* Question 4: Can you tell me which API can be used to develop RESTFul web service in Java? There are many framework and libraries out there which helps to develop RESTful web services in Java including JAX-RS which is standard way to develop REST web services. Jersey is one of the popular implementation of JAX-RS which also offers more than specification recommends. Then you also have RESTEasy, RESTlet and Apache CFX. If you like Scala then you can also use Play framework to develop RESTful web services
* Question 5 : How do you configure RESTFul web service?
* Question 8 : How you maintain session in RESTful services?
* 
* 

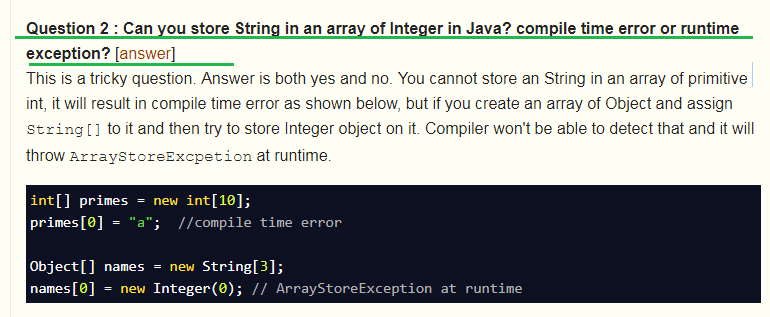
Top 20 Hibernate Interview Questions for Java J2EE Programmers <http://www.java67.com/2016/02/top-20-hibernate-interview-questions.html>

* 
* 
* 
* 
* 
* 

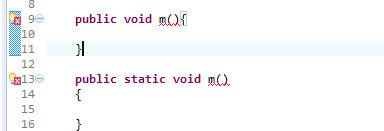
Top 23 Spring MVC Framework Interview Questions Answers - Java JEE <http://www.java67.com/2012/08/spring-interview-questions-answers.html>

* 
* 15) Can we use more than one configuration file for our Spring project?
* 18) Can you use Spring MVC framework along with Struts ? I have an existing Java MVC application which is based in Struts, Can I migrate that to use Spring MVC ? **How ?**
* 19) What is the advantage of Spring MVC framework over Struts 1.0 or Struts 2.0 ? is it worth to convert an existing Struts application to Spring MVC ?

22 Array Concepts Interview Questions Answers in Java <http://www.java67.com/2015/07/array-concepts-interview-questions-answers-java.html>

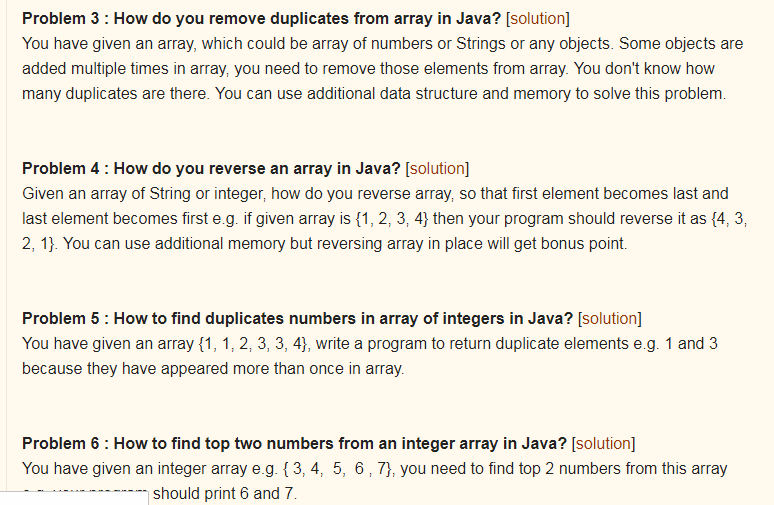
* 

Above Question is Very Important

 reason for CE is duplicate method

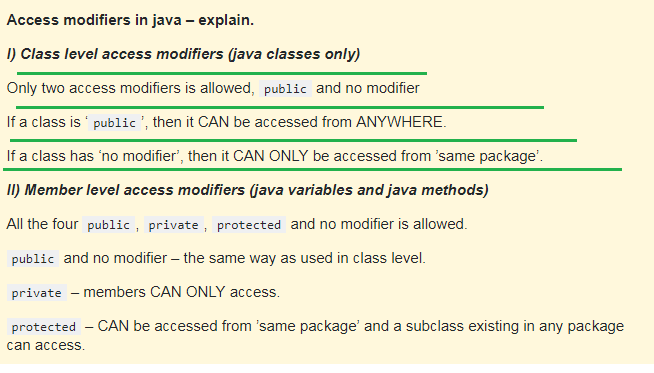
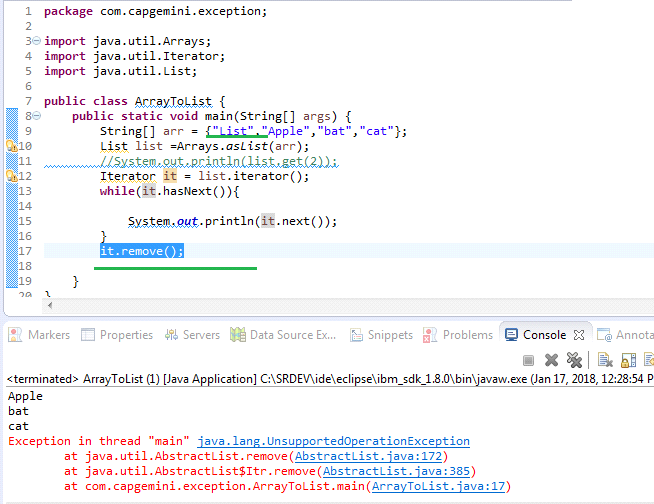
* **Question 4 : Can you use Generics with array?** [[answer](http://javarevisited.blogspot.sg/2011/09/generics-java-example-tutorial.html)]  
  No, you cannot use Generic with array, that's why sometime List is better choice over array in Java.

Array Based Programs

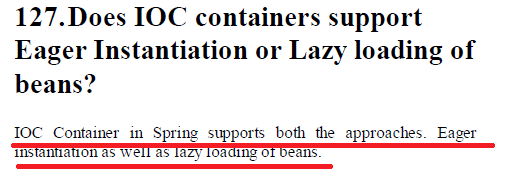
* Problem 2 : How do you find all pair whose sum is equal to given number from integer array in Java?
* Problem 2 : **How do you find all pair whose sum is equal to given number from integer array in Java?** [[solution](http://javarevisited.blogspot.sg/2014/08/how-to-find-all-pairs-in-array-of-integers-whose-sum-equal-given-number-java.html)]
*   
    
    
  16/1/2018
* Can we have more than one class @ted with @Configuration? YES
* Not only the add but also remove is not possible with the array converted into list
* Stack overflow [**Why are interface variables static and final by default?**](https://stackoverflow.com/questions/2430756/why-are-interface-variables-static-and-final-by-default)

Since interface doesn't have a direct object, the only way to access them is by using a class/interface and hence that is why if interface variable exists, it should be static otherwise it wont be accessible at all to outside world. Now since it is static, it can hold only one value and any classes that implements it can change it and hence it will be all mess.

Hence if at all there is an interface variable, it will be implicitly static, final and obviously public!!!

* **A constructor can have any of the Access specifiers like public, protected, private , none but unlike methods constructor can take up only the access specifier, therefore constructor cannot be abstract, final, static, synchronized, native**
* **Ms but outer class can have final and inner class can be static**
* 
* **SEE THE BELOW IMAGE ITERATOR HAS REMOVED THE Object but after removing it had thrown the “UnSupportedOperation” Exception**;

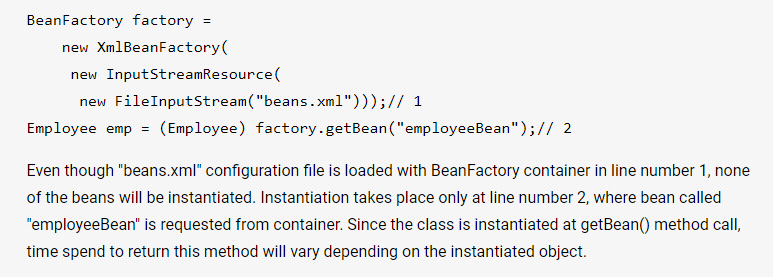
**ARRAY CONVERTED TO LIST DOES NOT ALLOW BOTH ADDITION OR REMOVAL OF OBJECTS**

* Single-Page Applications (SPAs) are Web apps that load a single HTML page and dynamically update that page as the user interacts with the app. SPAs use AJAX and HTML5 to create fluid and responsive Web apps, without constant page reloads. However, this means much of the work happens on the client side, in JavaScript.
* REGEXP\_SUBSTR()
* 

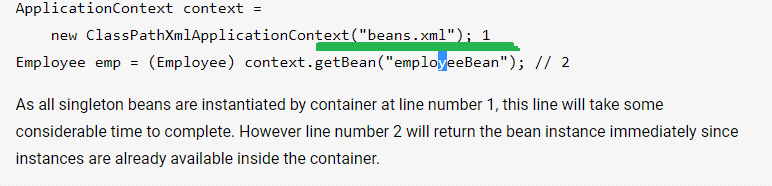
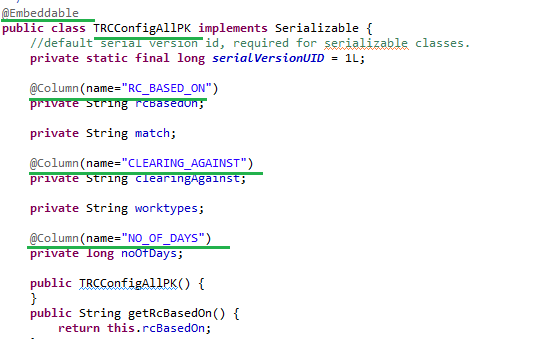
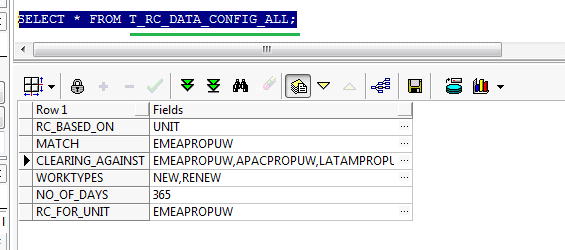
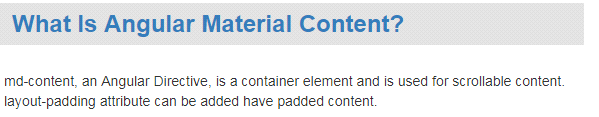
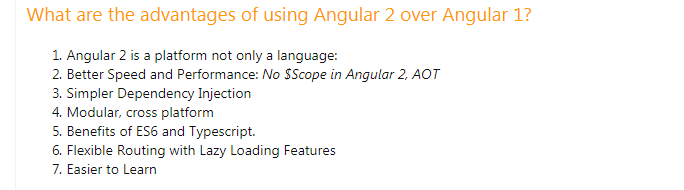
Explanation

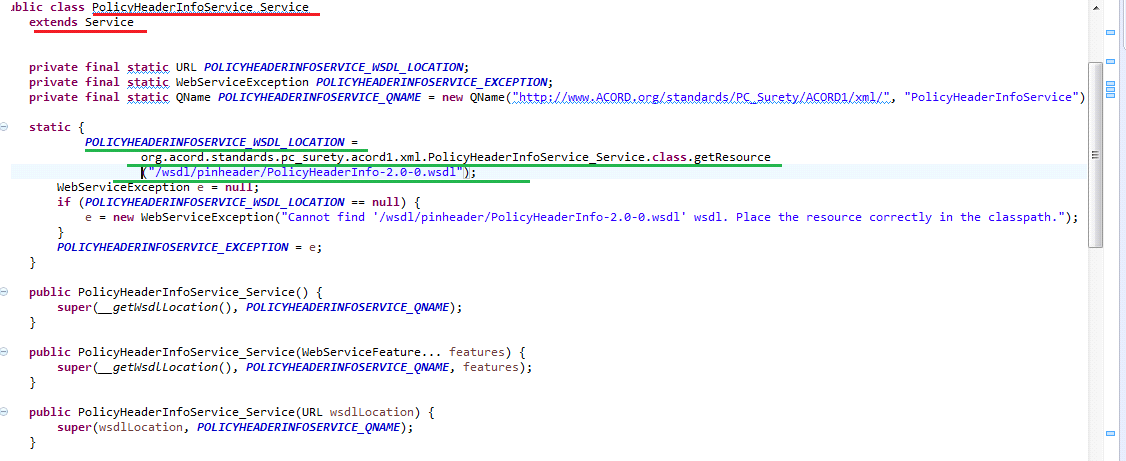
* Spring framework can instantiate and bind (called loading) related Java objects (called beans) according to a given configuration
* An XML file can easily be used to define these bindings.
* Spring framework supports two different types of loading methods; lazy loading and pre-loading respectively managed by BeanFactory and ApplicationContext containers.

**Lazy Loading**

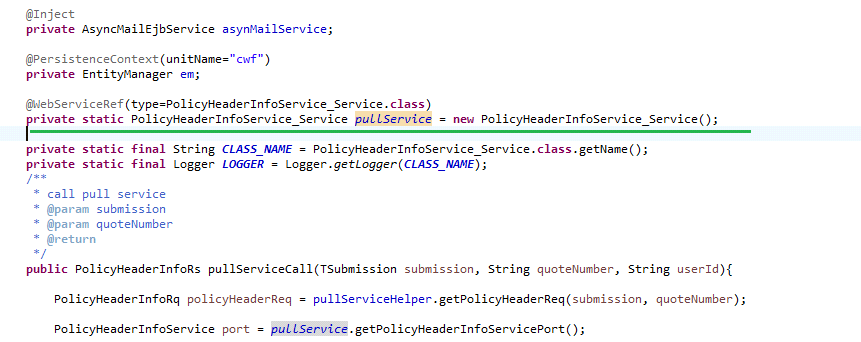
* A bean is loaded only when an instance of that Java class is requested by any other method or a class. org.springframework.beans.factory.BeanFactory (and subclasses) container loads beans lazily.
* 

**Pre-loading**

* All beans are instantiated as soon as the spring configuration is loaded by a container. org.springframework.context.ApplicationContext container follows pre-loading methodology.
* 
* **How will you call a stored procedure in Hibernate? needed information on this and also check how we had called in CNA**
* Let’s see an example for how EmbeddedId works
* A class which is going to act as a @EmbeddedId field
* A class which has @EmbeddedId and one of its own field called “ **RC\_FOR\_UNIT** ” 
* NOW LETS SEE THE TABLE STRUCTURE 
* See here an interface is having abstract keyword
*  needed more information on this
* **how will you handle errors in Angular2 application**
* Explain the life cycle hooks of Angular 2 application
* 
* Pending work in angular Custom filters/pipe, Auto completion, Event Emitter, input and output, Life cycle hooks, pagination, modal, and others see the notes.
* Few Information on Soap Service
* Auto generated Service class which is used to get the reference/port of out WS interface



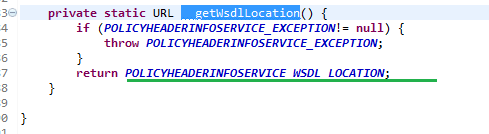
* Now lets see the flow for getting the Service Object first and then through service object how to get the Port or WS reference
* First lets see how to get the object of the auto generated Service class that extends Service



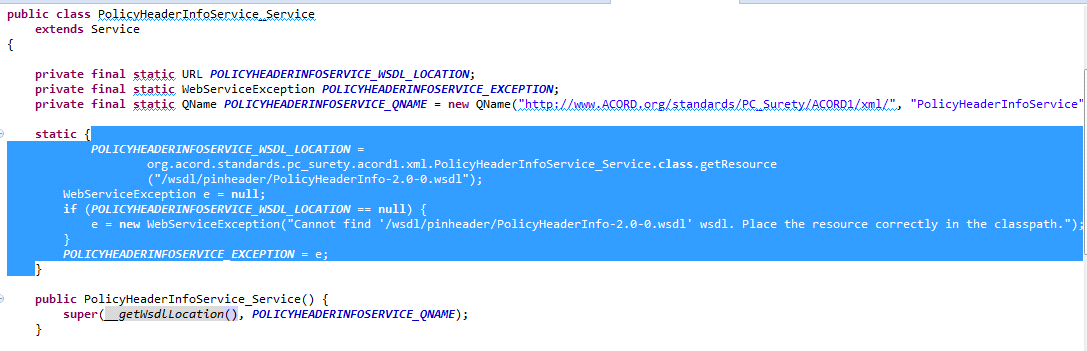
Now let’s see what happens in the PolicyHeaderInfoService\_Service() constructor



Now Let’s see what is this getWsdlLocation\_() method is doing



Now lets see Where and how POLICYHEADERINFO\_SERVICE\_EXCEPTION and POLICYHEADERINFOSERVICE\_WSDL\_LOCATION is set



* Now let’s see how is this port or WS reference is obtained using the auto generated service stub object
* 

22/1/2018

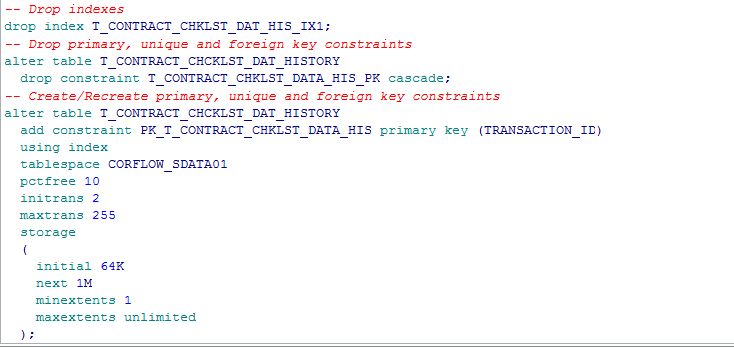
* **evict()** To detach the object from **session** cache,
* what is the difference between evict and delete methods in Hibernate Session

[clear(), evict() and close() methods in Hibernate](https://www.connect2java.com/tutorials/hibernate/clear-evict-and-close-methods-in-hibernate/)

* public void clear()Completely clear the session and is used to dissociate/disconnect all the objects from the session. ms clears entire session
* **evict():**  Removes the object from the session. This method is used to dissociate/disconnect the specified object from the session **public void evict(Object object)** throws HibernateException ms removes only particular object

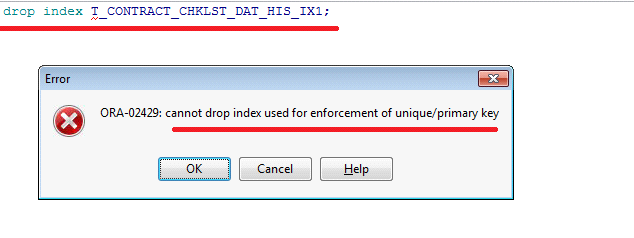
1/2/2018

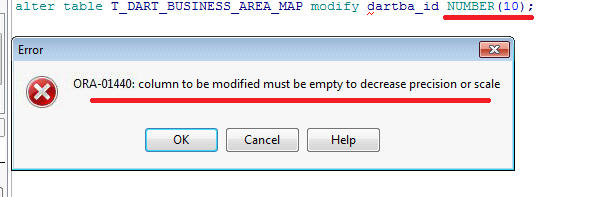
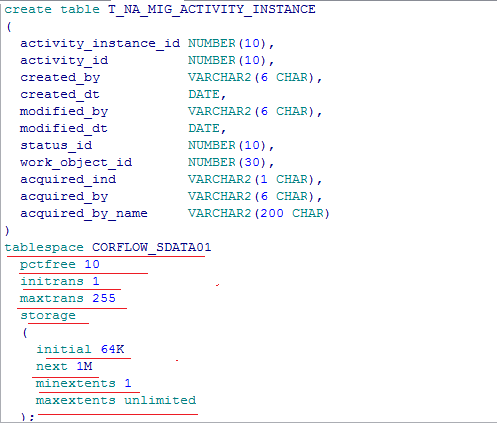
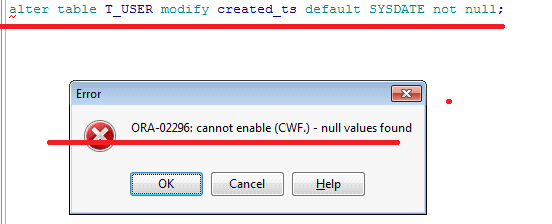
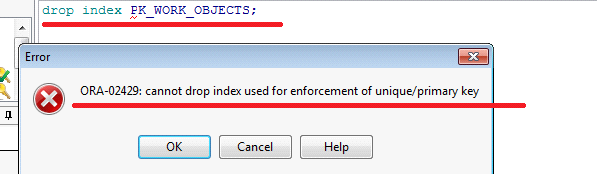
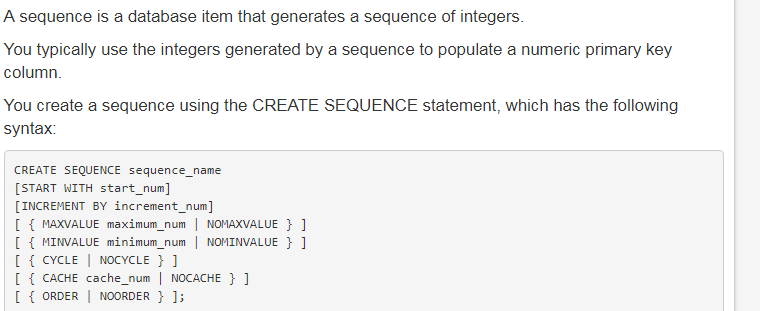
* While Comparing the two schemas I found that



There are dropping aa index and a primary key constraint and after that they again creating a primary key with index as shown above note actually they are doing all this circus just to rename the PKey constraint and index

Output on executing the drop index command I got the following error



* While trying to reduce the column size the error was 
* Ms Difference between Normal and Unique index normal index allows duplicates(not unique values) and Unique index does not allows unique index
* How to apply Unique constraint on 2 columns that is combination of two columns should be a unique value
* Ms to create a unique index then I need to add Unique keyword in the “Create Index” query otherwise it will create a normal index except for the primary key column that is if I create a index for a primary key column then even without adding the the Unique Keyword to “Create index” by default Unique index gets created
* 
* 
* NOTE WHEN WE TRY TO DROP AN INDEX WHICH IS CRESTED DUE TO THE PRIMARY KEY COLUMN (or) created on primary key column then we get the following error message
* Syntax to create a sequence
* The default start\_num is 1.
* The default increment number is 1.
* NOMINVALUE is the default.
* NOMAXVALUE is the default.
* CYCLE specifies the sequence generates integers even after reaching its maximum or minimum value.
* NOCYCLE specifies the sequence cannot generate any more integers after reaching its maximum or minimum value. NOCYCLE is the default.
* CACHE cache\_num specifies the number of integers to keep in memoryThe default number of integers to cache is 20
* NOCACHE specifies no integers are to be stored.
* ORDER guarantees the integers are generated in the order of the request.
* NOORDER doesn't guarantee the integers are generated in the ordeNOORDER is the default

**ALTER SEQUENCE**

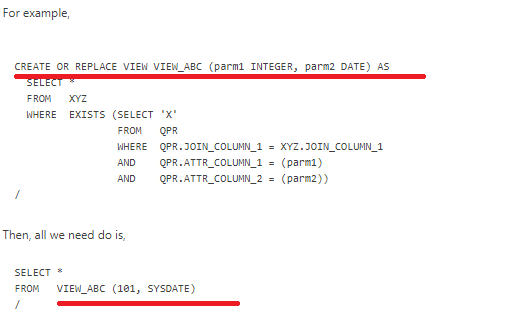
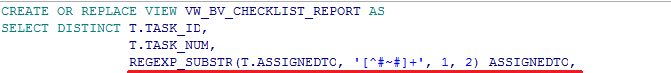
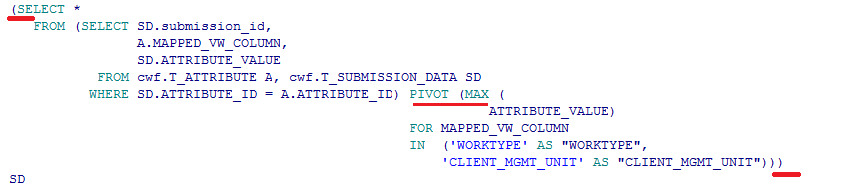
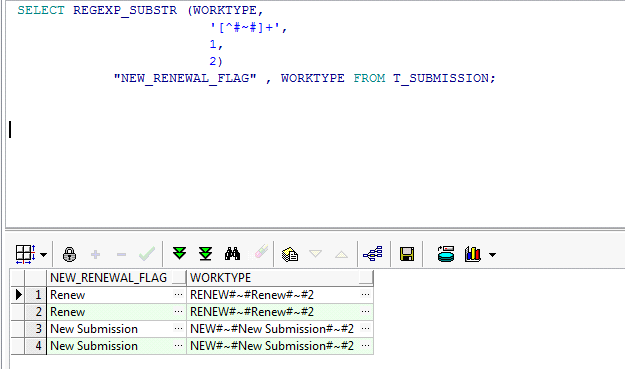
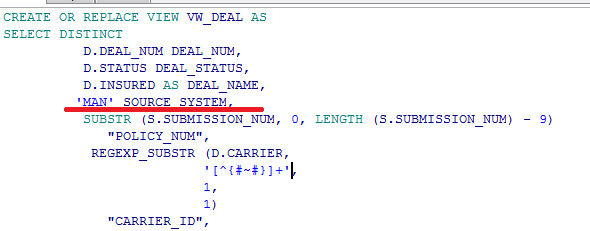
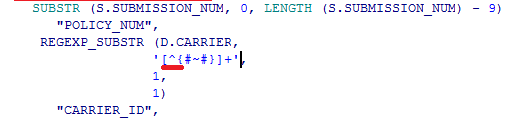
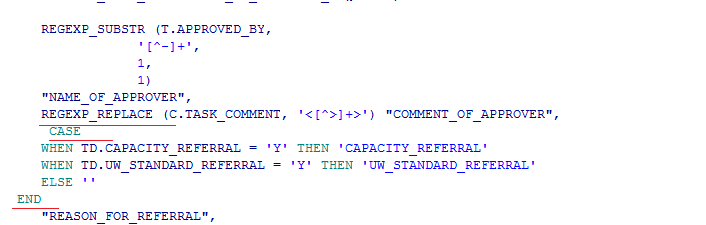
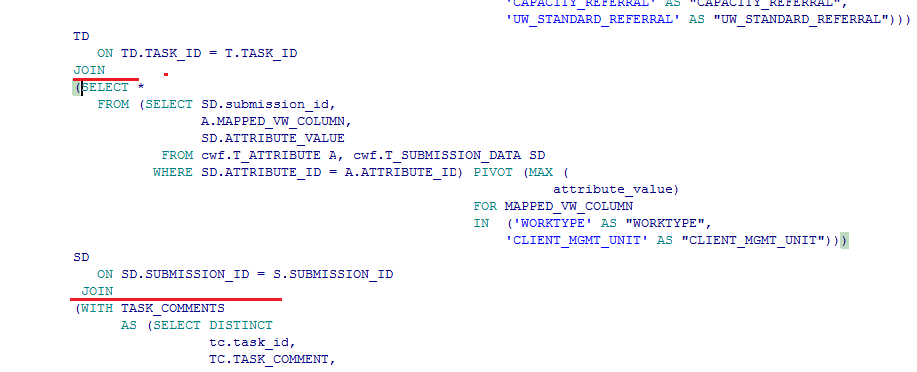
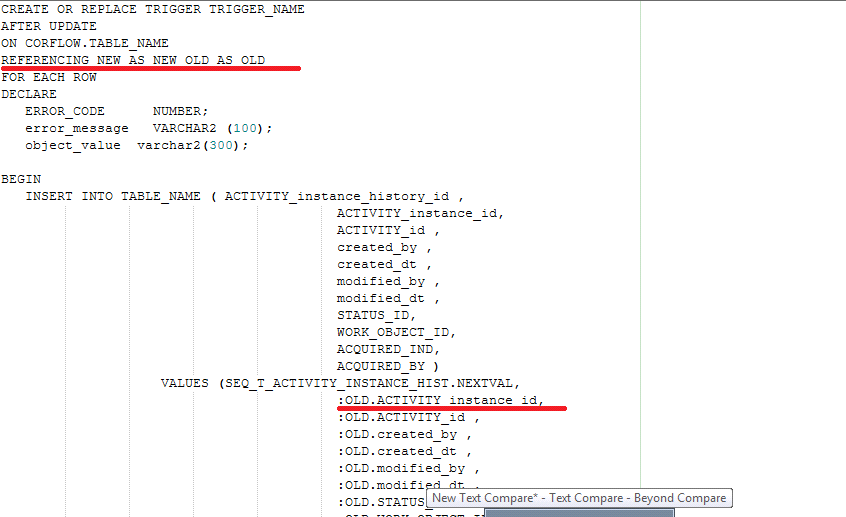
* Use the ALTER SEQUENCE statement to change the increment, minimum and maximum values, cached numbers, and behavior of an existing sequence. This statement affects only future sequence numbers.
* ALTER SEQUENCE customers\_seq MAXVALUE 1500;

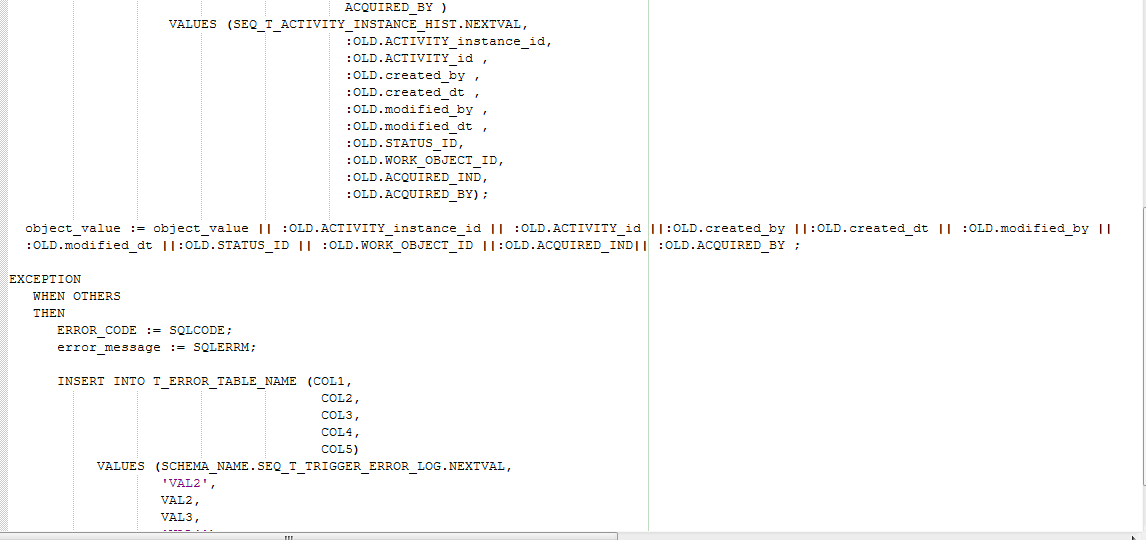
This statement sets a new maximum value for the customers\_seq

* ALTER SEQUENCE customers\_seq CYCLE CACHE 5;

This statement turns on CYCLE and CACHE for the customers\_seq sequence:

2/2/2018

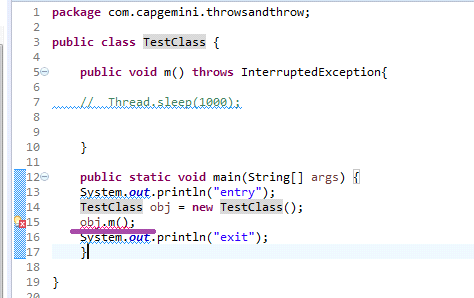
* Syntax to create a parameterized view in oracle
* Syntax for Substring in view 
* Needed more information on the PIVOT 
* I SHOULD LEARN SOME OF THE PATTERN USED IN REQEXP\_SUBSTR to understand it completely 
* View analysis
* While doing any changes in a view run it once so that my changes will get saved otherwise my changes will be lost
* In this case a column with name SOURCE\_SYSTEM will be created and it will have only “MAN” as the value
* Needed more information on this pattern used here for REGEXP\_SUBSTR 
* 
* Example for Rregexp\_REPLACE , CASE 
* WHAT KIND OF JOIN IS THIS 
* NLSSORT?
* TRIM FUNCTION
* 



8/2/2018

Exception Coding

* See her we will not get CE here if we don’t call m() which is throwing the Checked exception



* Core java Interview Questions from <http://www.baeldung.com/java-exceptions-interview-questions>

Q2. What is the purpose of the *throw* and *throws*keywords?

The *throws* keyword is used to specify that a method may raise an exception during its execution. It enforces explicit exception handling when calling a method:

|  |  |
| --- | --- |
| 1  2  3 | public void simpleMethod() throws Exception {      // ...  } |

The *throw* keyword allows us to throw an exception object to interrupt the normal flow of the program. This is most commonly used when a program fails to satisfy a given condition:

|  |  |
| --- | --- |
| 1  2  3 | if (task.isTooComplicated()) {      throw new TooComplicatedException("The task is too complicated");  } |

Q3. How can you handle an exception?

By using a *try-catch-finally* statement:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | try {      // ...  } catch (ExceptionType1 ex) {      // ...  } catch (ExceptionType2 ex) {      // ...  } finally {      // ...  } |

The block of code in which an exception may occur is enclosed in a *try* block. This block is also called “protected” or “guarded” code.

If an exception occurs, the *catch* block that matches the exception being thrown is executed, if not, all *catch* blocks are ignored.

The *finally* block is always executed after the *try* block exits, whether an exception was thrown or not inside it.

Q4. How can you catch multiple exceptions?

There are three ways of handling multiple exceptions in a block of code.

The first is to use a *catch* block that can handle all exception types being thrown:

|  |  |
| --- | --- |
| 1  2  3  4  5 | try {      // ...  } catch (Exception ex) {      // ...  } |

You should keep in mind that the recommended practice is to use exception handlers that are as accurate as possible.

Exception handlers that are too broad can make your code more error-prone, catch exceptions that weren’t anticipated, and cause unexpected behavior in your program.

The second way is implementing multiple catch blocks:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | try {      // ...  } catch (FileNotFoundException ex) {      // ...  } catch (EOFException ex) {      // ...  } |

Note that, if the exceptions have an inheritance relationship; the child type must come first and the parent type later. If we fail to do this, it will result in a compilation error.

The third is to use a multi-catch block:

|  |  |
| --- | --- |
| 1  2  3  4  5 | try {      // ...  } catch (FileNotFoundException | EOFException ex) {      // ...  } |

This feature, first introduced in Java 7; reduces code duplication and makes it easier to maintain.

Q5. What is the difference between a checked and an unchecked exception?

A checked exception must be handled within a *try-catch* block or declared in a *throws* clause; whereas an unchecked exception is not required to be handled nor declared.

Checked and unchecked exceptions are also known as compile-time and runtime exceptions respectively.

All exceptions are checked exceptions, except those indicated by *Error*, *RuntimeException*, and their subclasses.

Q6. What is the difference between an exception and error?

An exception is an event that represents a condition from which is possible to recover, whereas error represents an external situation usually impossible to recover from.

All errors thrown by the JVM are instances of *Error* or one of its subclasses, the more common ones include but are not limited to:

* *OutOfMemoryError* – thrown when the JVM cannot allocate more objects because it is out memory, and the garbage collector was unable to make more available
* *StackOverflowError* – occurs when the stack space for a thread has run out, typically because an application recurses too deeply
* *ExceptionInInitializerError* – signals that an unexpected exception occurred during the evaluation of a static initializer
* *NoClassDefFoundError* – is thrown when the classloader tries to load the definition of a class and couldn’t find it, usually because the required *class*files were not found in the classpath
* *UnsupportedClassVersionError* – occurs when the JVM attempts to read a *class* file and determines that the version in the file is not supported, normally because the file was generated with a newer version of Java

Although an error can be handled with a *try* statement, this is not a recommended practice since there is no guarantee that the program will be able to do anything reliably after the error was thrown.

Q7. What exception will be thrown executing the following code block?

|  |  |
| --- | --- |
| 1  2 | Integer[][] ints = { { 1, 2, 3 }, { null }, { 7, 8, 9 } };  System.out.println("value = " + ints[1][1].intValue()); |

It throws an *ArrayIndexOutOfBoundsException*since we’re trying to access a position greater than the length of the array.

Q8. What is exception chaining?

Occurs when an exception is thrown in response to another exception. This allows us to discover the complete history of our raised problem:

|  |  |
| --- | --- |
| 1  2  3  4  5 | try {      task.readConfigFile();  } catch (FileNotFoundException ex) {      throw new TaskException("Could not perform task", ex);  } |

Q9. What is a stacktrace and how does it relate to an exception?

A stack trace provides the names of the classes and methods that were called, from the start of the application to the point an exception occurred.

It’s a very useful debugging tool since it enables us to determine exactly where the exception was thrown in the application and the original causes that led to it.

Q10. Why would you want to subclass an exception?

If the exception type isn’t represented by those that already exist in the Java platform, or if you need to provide more information to client code to treat it in a more precise manner, then you should create a custom exception.

Deciding whether a custom exception should be checked or unchecked depends entirely on the business case. However, as a rule of thumb; if the code using your exception can be expected to recover from it, then create a checked exception otherwise make it unchecked.

Also, you should inherit from the most specific *Exception* subclass that closely relates to the one you want to throw. If there is no such class, then choose *Exception*as the parent.

Q11. What are some advantages of exceptions?

Traditional error detection and handling techniques often lead to spaghetti code hard to maintain and difficult to read. However, exceptions enable us to separate the core logic of our application from the details of what to do when something unexpected happens.

Also, since the JVM searches backward through the call stack to find any methods interested in handling a particular exception; we gain the ability to propagate an error up in the call stack without writing additional code.

Also, because all exceptions thrown in a program are objects, they can be grouped or categorized based on its class hierarchy. This allows us to catch a group exceptions in a single exception handler by specifying the exception’s superclass in the *catch* block.

Q12. Can you throw any exception inside a lambda expression’s body?

When using a standard functional interface already provided by Java, you can only throw unchecked exceptions because standard functional interfaces do not have a “throws” clause in method signatures:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | List<Integer> integers = Arrays.asList(3, 9, 7, 0, 10, 20);  integers.forEach(i -> {      if (i == 0) {          throw new IllegalArgumentException("Zero not allowed");      }      System.out.println(Math.PI / i);  }); |

However, if you are using a custom functional interface, throwing checked exceptions is possible:

|  |  |  |
| --- | --- | --- |
| 1  2  3  4 | @FunctionalInterface  public static interface CheckedFunction<T> {      void apply(T t) throws Exception;  } | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | | public void processTasks(    List<Task> taks, CheckedFunction<Task> checkedFunction) {      for (Task task : taks) {          try {              checkedFunction.apply(task);          } catch (Exception e) {              // ...          }      }  }    processTasks(taskList, t -> {      // ...      throw new Exception("Something happened");  }); |

Q13. What are the rules we need to follow when overriding a method that throws an exception?

Several rules dictate how exceptions must be declared in the context of inheritance.

When the parent class method doesn’t throw any exceptions, the child class method can’t throw any checked exception, but it may throw any unchecked.

Here’s an example code to demonstrate this:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | class Parent {      void doSomething() {          // ...      }  }    class Child extends Parent {      void doSomething() throws IllegalArgumentException {          // ...      }  } |

The next example will fail to compile since the overriding method throws a checked exception not declared in the overridden method:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | class Parent {      void doSomething() {          // ...      }  }    class Child extends Parent {      void doSomething() throws IOException {          // Compilation error      }  } |

When the parent class method throws one or more checked exceptions, the child class method can throw any unchecked exception; all, none or a subset of the declared checked exceptions, and even a greater number of these as long as they have the same scope or narrower.

Here’s an example code that successfully follows the previous rule:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | class Parent {      void doSomething() throws IOException, ParseException {          // ...      }        void doSomethingElse() throws IOException {          // ...      }  }    class Child extends Parent {      void doSomething() throws IOException {          // ...      }        void doSomethingElse() throws FileNotFoundException, EOFException {          // ...      }  } |

Note that both methods respect the rule. The first throws fewer exceptions than the overridden method, and the second, even though it throws more; they’re narrower in scope.

However, if we try to throw a checked exception that the parent class method doesn’t declare or we throw one with a broader scope; we’ll get a compilation error:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | class Parent {      void doSomething() throws FileNotFoundException {          // ...      }  }    class Child extends Parent {      void doSomething() throws IOException {          // Compilation error      }  } |

When the parent class method has a throws clause with an unchecked exception, the child class method can throw none or any number of unchecked exceptions, even though they are not related.

Here’s an example that honors the rule:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | class Parent {      void doSomething() throws IllegalArgumentException {          // ...      }  }    class Child extends Parent {      void doSomething()        throws ArithmeticException, BufferOverflowException {          // ...      }  } |

Q14. Will the following code compile?

|  |  |
| --- | --- |
| 1  2  3  4 | void doSomething() {      // ...      throw new RuntimeException(new Exception("Chained Exception"));  } |

Yes. When chaining exceptions, the compiler only cares about the first one in the chain and, because it detects an unchecked exception, we don’t need to add a throws clause.

Q15. Is there any way of throwing a checked exception from a method that does not have a *throws* clause?

Yes. We can take advantage of the type erasure performed by the compiler and make it think we are throwing an unchecked exception, when, in fact; we’re throwing a checked exception:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public <T extends Throwable> T sneakyThrow(Throwable ex) throws T {      throw (T) ex;  }    public void methodWithoutThrows() {      this.<RuntimeException>sneakyThrow(new Exception("Checked Exception"));  } |

**3. Conclusion**

* Core Java Exception Interview Question from <http://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/>
* [**30 Java Exception Handling Interview Questions And Answers**](http://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/)
* The url for the above 30 Questions I s <http://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/>
* Interview Questions from Tpoint

8/2/2018

from Library

TPoint

1. **1. If I don't provide any arguments on the command line, then the String array of Main method will be empty or null?**It is empty. But not null.
2. No, constructor is not inherited
3. **Why Java does not support pointers?**Pointer is a variable that refers to the memory address. They are not used in java because they are unsafe(unsecured) and complex to understand.  
     
   but while studying about NullPointerPointer exception we saw taht all obhetc reference in java is a pointer
4. **51) Can we intialize blank final variable?**Yes, only in constructor if it is non-static. If it is static blank final variable, it can be initialized only in the static block.
5. **When can an object reference be cast to an interface reference?**An object reference can be cast to an interface reference when the object implements the referenced interface.
6. 1.finally block will not be executed if program exits(either by calling System.exit() or by causing a fatal error that causes the process to abort)

7. String s=new String("Welcome");//creates two objects and one reference variable

In such case, JVM will create a new string object in normal(non pool) heap memory and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in heap(non pool).

1. what is the differene between gc() and finalize() ?

2. Reflection is the process of examining or modifying the runtime behaviour of a class at runtime.

**Can you access the private method from outside the class?**

Yes, by changing the runtime behaviour of a class if the class is not secured.

**What is singleton class?**

Singleton class means that any given time only one instance of the class is present, **in one JVM**.

A Locale object represents a specific geographical, political, or cultural region.

preemptive scheduling ==> is priority based

time slicing ==>a task executes for a predefined slice of time

HashSet contains only values whereas HashMap contains entry(key,value)**. HashSet can be iterated but HashMap need to convert into Set to be iterated**



**What is the advantage of generic collection?**

If we use generic class, we don't need typecasting. It is typesafe and checked at compile time.

The Dictionary class provides the capability to store key-value pairs.

ArrayList doesn't have length() method, the size() method of ArrayList provides the number of objects available in the collection.Array has length property which provides the length or capacity of the Array. It is the total space allocated during the intialization of the array.



**How can we execute stored procedures and functions?**

By using **Callable statement** interface, we can execute procedures and functions ==> see this again in our application so taht we get a clear idea

Servlets Interview Question from Javatpoint

1. **How many objects of a servlet is created?**

Only one object at the time of first request by servlet or web container.

**1) How many objects of a servlet is created?**

Only one object at the time of first request by servlet or web container.

**2) What is the life-cycle of a servlet?**

1. Servlet is loaded
2. servlet is instantiated
3. servlet is initialized
4. service the request
5. servlet is destroyed

[more details...](file:///D:\Users\ushap\Downloads\life-cycle-of-a-servlet)

**3) What are the life-cycle methods for a servlet?**

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void init(ServletConfig config) | It is invoked only once when first request comes for the servlet. It is used to initialize the servlet. |
| public void service(ServletRequest request,ServletResponse)throws ServletException,IOException | It is invoked at each request.The service() method is used to service the request. |
| public void destroy() | It is invoked only once when servlet is unloaded. |

[more details...](file:///D:\Users\ushap\Downloads\life-cycle-of-a-servlet)

**4) Who is responsible to create the object of servlet?**

The web container or servlet container.

**5) When servlet object is created?**

At the time of first request.

**6) What is difference between Get and Post method?**

|  |  |
| --- | --- |
| **Get** | **Post** |
| 1) Limited amount of data can be sent because data is sent in header. | Large amount of data can be sent because data is sent in body. |
| 2) Not Secured because data is exposed in URL bar. | Secured because data is not exposed in URL bar. |
| 3) Can be bookmarked | Cannot be bookmarked |
| 4) Idempotent | Non-Idempotent |
| 5) It is more efficient and used than Post | It is less efficient and used |

[more details...](file:///D:\Users\ushap\Downloads\http-request)

**7) What is difference between PrintWriter and ServletOutputStream?**

PrintWriter is a character-stream class where as ServletOutputStream is a byte-stream class. The PrintWriter class can be used to write only character-based information whereas ServletOutputStream class can be used to write primitive values as well as character-based information.

**8) What is difference between GenericServlet and HttpServlet?**

The GenericServlet is protocol independent whereas HttpServlet is HTTP protocol specific. HttpServlet provides additional functionalities such as state management etc.

**9) What is servlet collaboration?**

When one servlet communicates to another servlet, it is known as servlet collaboration. There are many ways of servlet collaboration:

* RequestDispacher interface
* sendRedirect() method etc.

[more details...](file:///D:\Users\ushap\Downloads\requestdispatcher)

**10) What is the purpose of RequestDispatcher Interface?**

The RequestDispacher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interceptor can also be used to include the content of antoher resource.

[more details...](file:///D:\Users\ushap\Downloads\requestdispatcher)

**11) Can you call a jsp from the servlet?**

Yes, one of the way is RequestDispatcher interface for example:

\l "\l "\l "\l "

1. RequestDispatcher rd=request.getRequestDispatcher("/login.jsp");
2. rd.forward(request,response);

[more details...](file:///D:\Users\ushap\Downloads\requestdispatcher)

**12) Difference between forward() method and sendRedirect() method ?**

|  |  |
| --- | --- |
| **forward() method** | **sendRedirect() method** |
| 1) forward() sends the same request to another resource. | 1) sendRedirect() method sends new request always because it uses the URL bar of the browser. |
| 2) forward() method works at server side. | 2) sendRedirect() method works at client side. |
| 3) forward() method works within the server only. | 3) sendRedirect() method works within and outside the server. |

**13) What is difference between ServletConfig and ServletContext?**

The container creates object of ServletConfig for each servlet whereas object of ServletContext is created for each web application.

**14) What is Session Tracking?**

**Session** simply means a particular interval of time.

Session Tracking is a way to maintain state of an user.Http protocol is a stateless protocol.Each time user requests to the server, server treats the request as the new request.So we need to maintain the state of an user to recognize to particular user.

[more details...](file:///D:\Users\ushap\Downloads\session-tracking-in-servlets)

**15) What are Cookies?**

A cookie is a small piece of information that is persisted between the multiple client requests. A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

[more details...](file:///D:\Users\ushap\Downloads\cookies-in-servlet)

**16) What is difference between Cookies and HttpSession?**

Cookie works at client side whereas HttpSession works at server side.

**17) What is filter?**

A filter is an object that is invoked either at the preprocessing or postprocessing of a request. It is pluggable.

[more details...](file:///D:\Users\ushap\Downloads\servlet-filter)

**18) How can we perform any action at the time of deploying the project?**

By the help of ServletContextListener interface.

**19) What is the disadvantage of cookies?**

It will not work if cookie is disabled from the browser.

[more details...](file:///D:\Users\ushap\Downloads\cookies-in-servlet)

**20) How can we upload the file to the server using servlet?**

One of the way is by MultipartRequest class provided by third party.

[more details...](file:///D:\Users\ushap\Downloads\example-of-uploading-file-to-the-server-in-servlet)

**21) What is load-on-startup in servlet?**

The load-on-startup element of servlet in web.xml is used to load the servlet at the time of deploying the project or server start. So it saves time for the response of first request.

[more details...](file:///D:\Users\ushap\Downloads\load-on-startup)

**22) What if we pass negative value in load-on-startup?**

It will not affect the container, now servlet will be loaded at first request.

[more details...](file:///D:\Users\ushap\Downloads\load-on-startup)

**23) What is war file?**

A war (web archive) file specifies the web elements. A servlet or jsp project can be converted into a war file. Moving one servlet project from one place to another will be fast as it is combined into a single file.

[more details...](file:///D:\Users\ushap\Downloads\war-file)

**24) How to create war file?**

The war file can be created using jar tool found in jdk/bin directory. If you are using Eclipse or Netbeans IDE, you can export your project as a war file.

To create war file from console, you can write following code.

\l "\l "\l "\l "

1. jar -cvf abc.war \*



Now all the files of current directory will be converted into abc.war file.

[more details...](file:///D:\Users\ushap\Downloads\war-file)

**25) What are the annotations used in Servlet 3?**

There are mainly 3 annotations used for the servlet.

1. @WebServlet : for servlet class.
2. @WebListener : for listener class.
3. @WebFilter : for filter class.

**26) Which event is fired at the time of project deployment and undeployment?**

ServletContextEvent.

[more details...](file:///D:\Users\ushap\Downloads\ServletContextEvent)

**27) Which event is fired at the time of session creation and destroy?**

HttpSessionEvent.

[more details...](file:///D:\Users\ushap\Downloads\HttpSessionEvent)

**28) Which event is fired at the time of setting, getting or removing attribute from application scope?**

ServletContextAttributeEvent.

**29) What is the use of welcome-file-list?**

It is used to specify the welcome file for the project.

[more details...](file:///D:\Users\ushap\Downloads\welcome-file-list)

**30) What is the use of attribute in servlets?**

Attribute is a map object that can be used to set, get or remove in request, session or application scope. It is mainly used to share information between one servlet to another.

Ejb Interview Questions from TPoint

1. **1) What is EJB?**

EJB stands for Enterprise Java Bean. It is a server-side component to develop scalable, robust and secured enterprise applications in java. [More details...](file:///D:\Users\ushap\Downloads\what-is-ejb)

**2) What are the types of Enterprise Bean?**

There are 3 types of enterprise bean in java.

1. Session Bean
2. Message Driven Bean
3. Entity Bean

[More details...](file:///D:\Users\ushap\Downloads\what-is-ejb)

**3) What is session bean?**

Session Bean encapsulates business logic. It can be invoked by local, remote or web service client.

There are 3 types of session bean.

1. Stateless Session Bean
2. Stateful Session Bean
3. Singleton Session Bean

[More details...](file:///D:\Users\ushap\Downloads\session-bean)

**4) What is stateless session bean?**

Stateless session bean is a business object that doesn't maintain conversational state with the client. [More details...](file:///D:\Users\ushap\Downloads\stateless-session-bean)

**5) What is stateful session bean?**

Stateful session bean is a business object that maintains conversational state with the client. [More details...](file:///D:\Users\ushap\Downloads\stateful-session-bean)

**6) What is singleton session bean?**

Singleton session bean is instantiated only once for the application. It exists for the life cycle of the application.

**7) What is JMS?**

Java Message Service is a messaging service to create, send and receive messages asynchronously. [More details...](file:///D:\Users\ushap\Downloads\jms-tutorial)

**8) What are the advantages of JMS?**

* Asynchronous
* Reliable

[More details...](file:///D:\Users\ushap\Downloads\jms-tutorial)

**9) What is PTP model?**

In Point to Point model, one message is delivered to one receiver only. Here, Queue is used as a message oriented middleware. [More details...](file:///D:\Users\ushap\Downloads\jms-tutorial)

**10) What is Pub/Sub model?**

In Publisher/Subscriber model, one message is delivered to all subscribers. Here, Topic is used as a message oriented middleware. [More details...](file:///D:\Users\ushap\Downloads\jms-tutorial)

**11) What is MDB?**

Message Driven Bean (MDB) encapsulates business logic. It is invoked by passing message. It is like JMS receiver. [More details...](file:///D:\Users\ushap\Downloads\message-driven-bean)

**12) What is Entity Bean?**

Entity Bean is a server side component that represents the persistent data. Since EJB 3.x, it is replaced by JPA. [More details...](file:///D:\Users\ushap\Downloads\entity-bean)

Note

Entity bean represents the persistent data stored in the database. It is a server-side component.

In EJB 2.x, there was two types of entity beans: **bean managed persistence** (BMP) and container managed persistence (CMP).

Since EJB 3.x, it is deprecated and replaced by JPA (Java Persistence API) that is covered in the hibernate tutorial.

In hibernate tutorial, there are given hibernate with annotation examples where we are using JPA annotations. The JPA with Hibernate is widely used today.

**13) What is Session Facade?**

Session Facade is a design pattern to access enterprise bean through local interface. It abstracts the business object interactions and provides a service layer. It makes the performance fast over network.

Hibernate Interview Questions from Tpoint

1. No, Session is not a thread-safe object, many threads can access it simultaneously. In other words, you can share it between threads.

update() should be used if session doesn't contain an already persistent state with same id. It means update should be used inside the session only. After closing the session it will throw error.

merge() should be used if you don't know the state of the session, means you want to make modification at any time.

**lets see an example for upate and merge ==> later**

**Doubt ==> when session si closed what about its cache ? ==> is it cleared or not?**

**12) What are the states of object in hibernate?**

There are 3 states of object (instance) in hibernate.

1. **Transient**: The object is in transient state if it is just created but has no primary key (identifier) and not associated **with session**.
2. **Persistent**: The object is in persistent state if session is open, and you just saved the instance in the database or retrieved the instance from the database.
3. **Detached**: The object is in detached state if session is closed. After detached state, object comes to persistent state if you call lock() or update() method.

**15) What is automatic dirty checking in hibernate?**

**The automatic dirty checking feature of hibernate, calls update statement automatically on the objects that are modified in a transaction.**

**17) Is it possible to perform collection mapping with One-to-One and Many-to-One?**

No, collection mapping can only be performed with One-to-Many and Many-to-Many

Spring Interview Questions

**11) In which scenario, you will use singleton and prototype scope?**

Singleton scope should be used with EJB **stateless session bean** and prototype scope with EJB **stateful session bean**.

**22) Does spring framework support all JoinPoints?**

No, spring framework supports method execution joinpoint only.

**31) Does spring perform weaving at compile time?**

No, spring framework performs weaving at runtime.

Accenture Interview Question

1. why do we need or what is teh difference between @Service, @Controlelr and @Repository annotations

2. EJB life Cycle

L & T Infotech interview Questions

1. in resume

PL/SQL interview Questions from tutorials point

**10) Does PL/SQL support CREATE command?**

No. PL/SQL doesn't support the data definition commands like CREATE.

doubt ==> tehn we have create or replace in PL/SQL tehn what does this Answer means>

**13) What is the main reason behind using an index?**

Faster access of data blocks in the table.

**18) What is the maximum number of triggers, you can apply on a single table?**

12 triggers.

**20) What is the difference between execution of triggers and stored procedures?**

A trigger is automatically executed without any action required by the user, while, a stored procedure is explicitly invoked by the user.

**23) How to disable a trigger name update\_salary?**

ALTER TRIGGER update\_salary DISABLE;

**24) Which command is used to delete a trigger?**

DROP TRIGGER command.

**27) What are the different schemas objects that can be created using PL/SQL?**

* Stored procedures and functions
* Packages
* Triggers

**36) How to execute a stored procedure?**

There are two way to execute a stored procedure.

From the SQL prompt, write EXECUTE or EXEC followed by procedure\_name.

1. EXECUTE or [EXEC] procedure\_name;

2. Simply use the procedure name

example ==>procedure\_name;

Note ==> **%ISOPEN, %ROWCOUNT, %FOUND, %NOTFOUND** ==>all this are called as **cursor attributes**

Oracle Interview Questions from tutorials Point

**2) What are the components of logical database structure in Oracle database?**

Components of logical database structure.

* Tablespaces
* Database's schema objects

**3) What is a tablespace?**

A database contains Logical Storage Unit called tablespaces. A tablespace is a set of related logical structures. Actually a tablespace groups related logical structures together.

**8) What is the relationship among database, tablespace and data file?**

An Oracle database contains one or more logical storage units called tablespaces. These tablespaces collectively store whole data of databases and each tablespace in Oracle database consists of one or more files called datafiles. These datafiles are physical structure that confirm with the operating system in which Oracle is running.

NOTE==> NOT COVERED FULLY , I HAD JUST TAKEN THE TABLEPACE DEFINITION THAT'S ALL

maven Interview Questions From TPoint

**Maven Interview Question**

A list of top frequently asked **maven interview questions** and answers are given below.

**1) What is Maven?**

Maven is a project management tool. It is based on POM (Project Object Model). [More details...](file:///D:\Users\ushap\Downloads\maven-tutorial)

**2) What aspects are managed by Maven?**

* Builds
* Documentation
* Reporing
* SCMs
* Releases
* Distribution

[More details...](file:///D:\Users\ushap\Downloads\maven-tutorial)

**3) What are the advantages of Maven?**

* No need to add jar file in each project
* Creates right directory structure
* Builds and deploys the project

[More details...](file:///D:\Users\ushap\Downloads\maven-tutorial)

**4) What is the command to check the maven version?**

Type the following command on console to know the maven version.

\l "\l "\l "\l "

1. mvn -version

[More details...](file:///D:\Users\ushap\Downloads\how-to-install-maven)

**5) What does the build tool?**

* Generates source code (if auto-generated code is used)
* Generates documentation from source code
* Compiles source code
* Packages compiled code into JAR of ZIP file
* Installs the packaged code in local repository, server repository, or central repository

[More details...](file:///D:\Users\ushap\Downloads\maven-tutorial)

**6) What is the difference between Ant and Maven?**

|  |  |
| --- | --- |
| **Ant** | **Maven** |
| It is **a tool** box. | It is **a framework**. |
| It is **mainly a build tool**. | It is **mainly a project management tool**. |
| There is **no life cycle**. | There is **life cycle**. |
| Ant **doesn't have formal conventions**. | Maven **has a convention** to place source code, compiled code etc. |
| Ant is **procedural**. | Maven is **declarative**. |
| The ant scripts are **not reusable**. | The maven plugins are **reusable**. |

[More details...](file:///D:\Users\ushap\Downloads\difference-between-ant-and-maven)

**7) What is a MOJO?**

A MOJO stands for Maven plain Old Java Object. Each MOJO is an executable goal in Maven, and a plugin is a distribution of one or more related MOJOs.

**8) What is repository?**

A repository is a directory or place where all the jars and pom.xml file are stored. There are 3 types of repository in Maven:

1. Local Repository
2. Central Repository
3. Remote Repository

[More details...](file:///D:\Users\ushap\Downloads\maven-repository)

**9) What is local repository?**

Maven local repository is created by maven in your local system when you run any maven command. [More details...](file:///D:\Users\ushap\Downloads\maven-repository#local)

**10) What is central repository?**

Maven central repository is created by maven community on the web. [More details...](file:///D:\Users\ushap\Downloads\maven-repository#central)

**11) What is remote repository?**

Maven remote repository is located on the web by different vendors. So you need to define the dependency in pom.xml file manually. It is important because most of libraries are missing from the central repository. [More details...](file:///D:\Users\ushap\Downloads\maven-repository#remote)

**12) What is POM?**

POM stands for Project Object Model. The pom.xml file contains information of project and project configuration. [More details...](file:///D:\Users\ushap\Downloads\maven-pom-xml)

**13) What are the build phases in Maven?**

1. validate
2. compile
3. test
4. package
5. integration-test
6. verify
7. install
8. deploy

**14) What is the command to package maven project?**

\l "\l "\l "\l "

1. mvn -package



**15) What is fully qualified artifact name of maven project?**

1. <groupId>:<artifactId>:<version>



**16) What is archetype?**

Archetype is the maven plugin. It creates the project structure.

Interview Questions from tutorials point

1. List some Java keywords(unlike C, C++ keywords)?

Some Java keywords are import, super, finally, etc.

2. Default value of byte datatype is 0.

Default value of float and double datatype in different as compared to C/C++. For float its 0.0f and for double it’s 0.0d

What are use cases?

Is there any limitation of using Inheritance?

Yes, since inheritance inherits everything from the super class and interface, it may make the subclass too clustering and sometimes error-prone when dynamic overriding or dynamic overloading in some situation.

Integer / and % can result in the throwing of an ArithmeticException.

No, a top level class can not be private or protected. It can have either "public" or no modifier. ==> why a top level class cannot be protected?

Private constructor is used if you do not want other classes to instantiate the object and to prevent subclassing.T

Octal Numbers are denoted by leading zero in java, example: **0**6

Hexadecimal Numbers are denoted by leading 0x or 0X in java, example − **0X**F

Can a vector contain heterogenous objects?

Yes a Vector can contain heterogenous objects. Because a Vector stores everything in terms of Object.

9/2/2018

Questions from 🡺 <http://www.instanceofjava.com/2016/04/exception-handling-interview-questions.html>

1. **How to write user defined exception or custom exception in java?**  
     
   By extending Exception class we can define custom exceptions.
   1. We need to write a constructor for passing message .
   2. [User defined exception in java](http://www.instanceofjava.com/2015/07/user-defined-exceptions-in-java.html)
2. **What are the different ways to print exception message on console.**  
   In Java there are three ways to find the details of the exception .They are

Using an object of java.lang.Exception

Using public void printStackTrace() method

Using public String getMessage() method.

**13.Can we write return statement in try and catch blocks**  
Yes we can write return statement of the method in try block and catch block

We need to follow some rules to use it.Please check below link

[Return statement in try and catch blocks](http://www.instanceofjava.com/2016/03/return-statement-in-try-catch-block-java.html)

**Can we write return statement in finally block**

Yes we can write return statement of the method in finally block

We need to follow some rules to use it.Please check below link

[Return statement in finally block](http://www.instanceofjava.com/2016/03/return-statement-in-finally-block-in.html)

**17.What are the rules we need to follow in overriding if super class method throws exception ?**

* If sub class throws checked exception super class should throw same or super class exception of this.
* **If super class method  throws checked or unchecked exceptions its not mandatory to put throws in sub class overridden method.**
* If super class method throws exceptions in sub class if you want to mention throws  then use  same class  or its  sub class exception.

Interview Questions from 🡺 <http://www.javainterview.in/p/exception-handling-interview-questions.html>

Is a finally block executed even when there is a return statement in the try block?

Yes. In the example below, connection.close() method is called even though there is a return in the catch block.

### How do you throw a Checked Exception from a Method?

Consider the example below. The method addAmounts throws a new Exception. However, it gives us a compilation error because Exception is a Checked Exception.

**All classes that are not RuntimeException or subclasses of RuntimeException but extend Exception are called CheckedExceptions. The rule for CheckedExceptions is that they should be handled or thrown. Handled means it should be completed handled - i.e. not throw out of the method. Thrown means the method should declare that it throws the exception**

Interview Question from <http://www.javamadesoeasy.com/2015/05/exceptions-top-60-interview-questions_16.html>

**Exception interview Question 13. What is Automatic resource management in java 7?**

**Exception interview Question 17. Is it allowed to use nested try-catch in java?**



**Exception interview Question 21. Can a catch or finally block throw exception in java?**

**Answer**. Yes, catch or finally block can throw checked or unchecked exception but it must be handled accordingly. Please refer this post for [handling checked and unchecked exceptions](http://www.javamadesoeasy.com/2015/05/checked-compile-time-exceptions-and.html) in java.

**Exception interview Question 24.  can a method be overloaded on basis of  exceptions in java ?**

Yes a method be overloaded on basis of  exceptions in java.

But now question which overloaded exception will be called.

Let’s take an example :

***Ques****. Let's say one method handles Exception and other handles ArithmeticException. Which method will be invoked when ArithmeticException is thrown?*

**Ans**. Method which handles more specific exception will be called

Another Exception interview Question.  **What are the differences between** [**between ClassNotFoundException and NoClassDefFoundError in java ?**](http://www.javamadesoeasy.com/2015/12/what-is-difference-between.html)

**Answer**.

|  |  |  |
| --- | --- | --- |
|  | ***ClassNotFoundException*** | ***NoClassDefFoundError*** |
| 1 | ClassNotFoundException is [Checked (compile time) **Exception** in java.](http://www.javamadesoeasy.com/2015/05/checked-compile-time-exceptions-and.html) | NoClassDefFoundError is a [Error](http://www.javamadesoeasy.com/2015/05/javalangerror-in-exception-handling-in.html) in java. Error and its subclasses are regarded as unchecked exceptions in java. |
| 2 | Here is the hierarchy of java.lang.ClassNotFoundException - -java.lang.Object  -java.lang.Throwable   -java.lang.[Exception](http://www.javamadesoeasy.com/2015/05/exception-handling-exception-hierarchy.html)    -java.lang.ReflectiveOperationException     -java.lang.ClassNotFoundException | Here is the hierarchy of java.lang.NoClassDefFoundError - -java.lang.Object  -java.lang.Throwable   -java.lang.[Error](http://www.javamadesoeasy.com/2015/05/javalangerror-in-exception-handling-in.html)    -java.lang.LinkageError     -java.lang.NoClassDefFoundError |
| 3 | **ClassNotFoundException** is thrown when JVM tries to class from classpath but it does not find that class. | **NoClassDefFoundError** is thrown when JVM tries to load class which >   * **was NOT** available at **runtime** but * **was** available at **compile** time. |
|  | **ExceptionInInitializerError** has got nothing to do with **ClassNotFoundException**. | You must ensure that class does not throws **java.lang.ExceptionInInitializerError** because that is likely to be followed by **NoClassDefFoundError**. |

For more read [differences between between ClassNotFoundException and NoClassDefFoundError in java](http://www.javamadesoeasy.com/2015/12/what-is-difference-between.html)

Interview Questions from 🡺

**Q12) How to create custom Exception? http://java-questions.com/Exceptions-interview-questions.html**

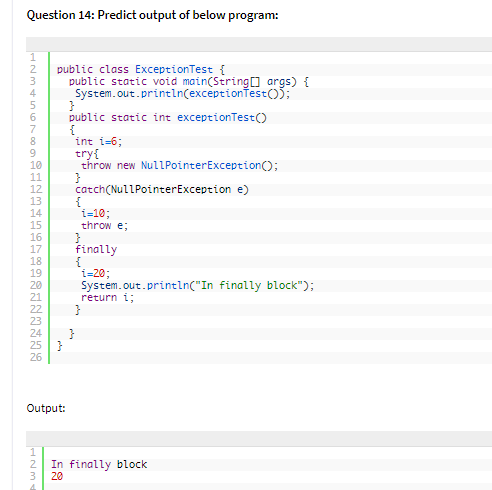
Ans) To create you own exception extend the Exception class or any of its subclasses.

* **class New1Exception extends Exception { } // this will create Checked Exception**
* class NewException extends IOExcpetion { } // this will create Checked exception
* class NewException extends NullPonterExcpetion { } // this will create UnChecked exception

<https://java2blog.com/java-exception-handling-interview-questions-and-answers/>

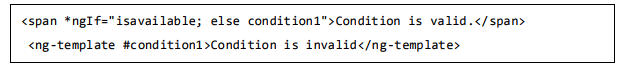
10/2/2018

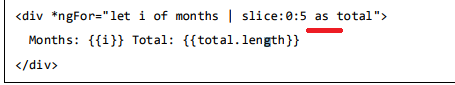
1. See here no Return in finally block in such case it had returned from catch 🡺
2. See here no exception is thrown by JVM which is re-thrown by catch bloc🡺

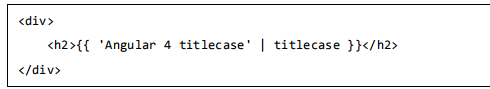


Angular Documentation

1. There are three major releases of Angular. The first version that was released is Angular1, which is also called AngularJS. Angular1 was followed by Angular2, which came in with a lot of changes when compared to Angular1. The structure of Angular is based on the components/services architecture. AngularJS was based on the model view controller. Angular 4 released in March 2017 proves to be a major breakthrough and is the latest release from the Angular team after Angular2. Angular 4 is almost the same as Angular 2. It has a backward compatibility with Angular 2. Projects developed in Angular 2 will work without any issues with Angular 4.
2. Angular2 supported only the “if” condition. However, Angular 4 supports the “if else” condition as well.



1. as keyword in for loop 🡺 With the help of as keyword you can store the value
2. Animation in Angular 4 is available as a separate package and needs to be imported from @angular/animations. In Angular2, it was available with @angular/core. It is still kept the same for its backward compatibility aspect.
3. Angular 4 uses as the tag instead of ; the latter was used in Angular2. The reason Angular 4 changed to is because of the name conflict of the tag with the html standard tag. It will deprecate completely going ahead. This is one of the major changes in Angular 4.
4. Angular 4 is updated to a recent version of TypeScript, which is 2.2. This helps improve the speed and gives better type checking in the project.
5. Angular 4 has added a new pipe title case, which changes the first letter of each word into uppercase.



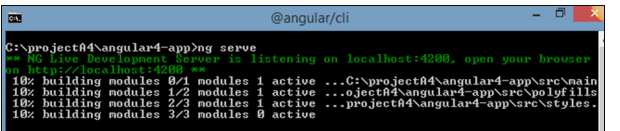
1. Search parameters to the http get api is simplified 🡺see the difference between both in detail later
2. Angular 4 applications are smaller and faster when compared to Angular2. It uses the TypeScript version 2.2, the latest version which makes the final compilation small in size.

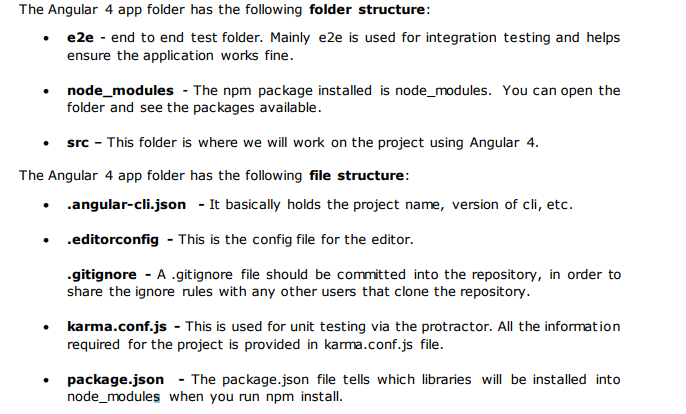
Angular 4 – Environment Setup

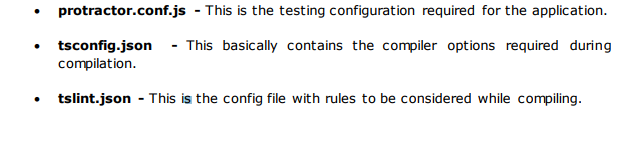
1. we require the following: • Nodejs • Npm • Angular CLI • IDE for writing your code
2. Nodejs has to be greater than 4 and npm has to be greater than 3.
3. C:\>node –v
4. Type npm install –g @angular/cli, to install angular cli on your system.
5. IDE of IDE WebStorm, Atom, Visual Studio Code, etc .

Angular 4 – Project Setup

1. AngularJS is based on the model view controller, whereas Angular 2 is based on the components structure. Angular 4 works on the same structure as Angular2 but is faster when compared to Angular2.
2. Angular4 uses TypeScript 2.1 version whereas Angular 2 uses TypeScript version 1.8. This brings a lot of difference in the performance.
3. To check whether Angular CLI is installed or not 🡺 ng –v
4. cmd to create a new Project 🡺 ng new projectname
5. compile our project with the command 🡺 ng serve
6. The ng serve command builds the application and starts the web server 🡺



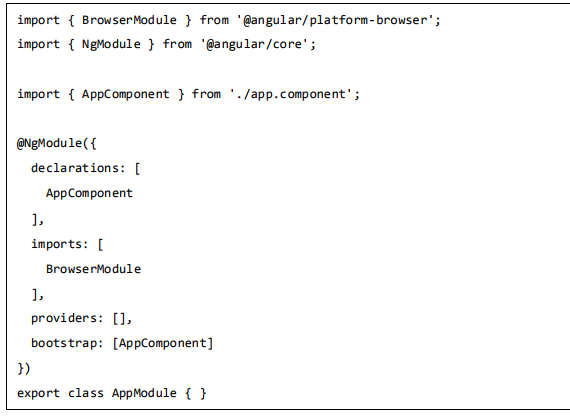
1. The web server starts on port 4200. Type the url http://localhost:4200/ in the browser and see the output.
2. Let us complete the project setup. If you see we have used port 4200, which is the default port that angular–cli makes use of while compiling. You can change the port if you wish using the following command🡺 
3. 



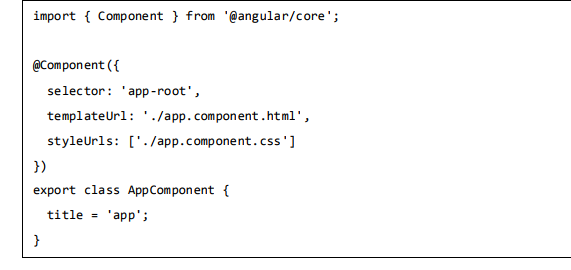
1. In The src folder 🡺 contains app folder and the app folder contains following files

* app.module.ts - If you open the file, you will see that the code has reference to different libraries, which are imported. 🡺

it has declarations, imports, providers , and bootstrap.



* app.component.css
* app.component.html 🡺 The html code will be available in this file.
* app.component.spec.ts - These are automatically generated files which contain unit tests for source component.
* app.component.ts 🡺



* Assets 🡺You can save your images, js files in this folder
* Environment This folder has the details for the production or the dev environment. The folder contains two files.

• environment.prod.ts

• environment.ts

Both the files have details of whether the final file should be compiled in the production environment or the dev environment.

* favicon.ico 🡺 This is a file that is usually found in the root directory of a website.
* main.ts 🡺 main.ts is the file from where we start our project development. It starts with importing the basic module which we need. Right now if you see angular/core, angular/platformbrowser-dynamic , app.module and environment is imported by default during angular-cli installation and project setup.



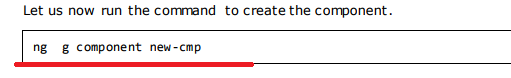
The platformBrowserDynamic().bootstrapModule(AppModule) has the parent module reference AppModule.

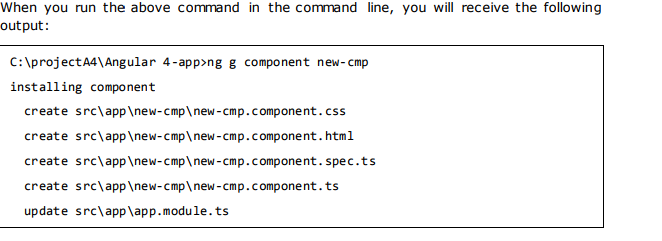
When AppModule is called, it calls app.module.ts which further calls the AppComponent based on the boostrap as follows🡺 

* polyfill.ts 🡺This is mainly used for backward compatibility.
* styles.css This is the style file required for the project
* test.ts Here, the unit test cases for testing the project will be handled.
* tsconfig.app.json 🡺 This is used during compilation, it has the config details that need to be used to run the application.
* typings.d.ts 🡺 It is used to manage the TypeScript definition

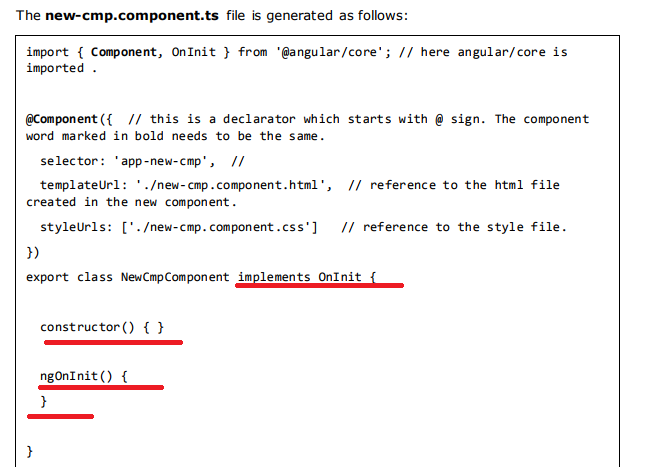
Angular 4 – Components

1. Components are basically classes that interact with the .html file of the component, which gets displayed on the browser.
2. Now, angular-cli has a command to create your own component. However, the app component which is created by default will always remain the parent and the next components created will form the child components.





1. Now, if we go and check the file structure, we will get the new-cmp new folder created under the src/app folder.
2. Now let’s go and see the newly generated ts file contents 🡺



ngOnInit is called by default when the class is executed.

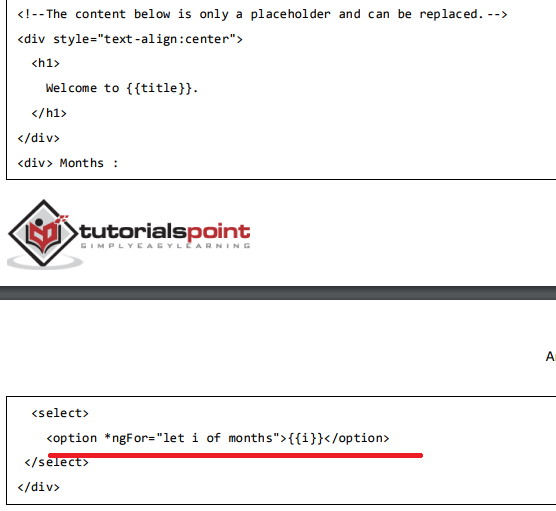
Angular 4 – Module

1. Module in Angular refers to a place where you can group the components, directives, pipes, and services, which are related to the application. 🡺 To define module, we can use the NgModule.

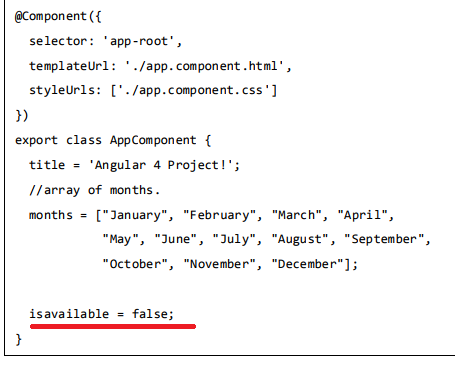
Angular 4 – Data Binding

1. Data binding is available right from **AngularJS**, Angular 2 and is now available in Angular 4 as well.
2. We use curly braces for data binding - {{}}; this process is called interpolation.
3. Now let’s code to create a dropdown of months in the browser.

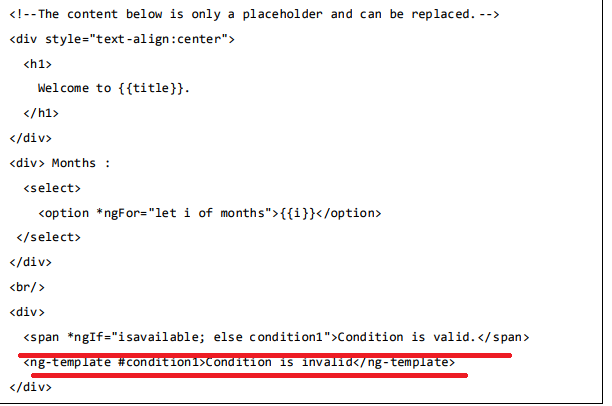




1. Example for IF –else

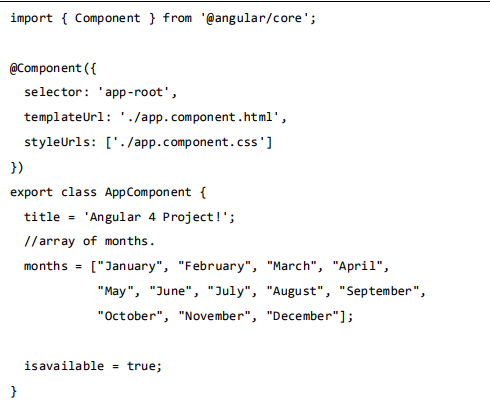


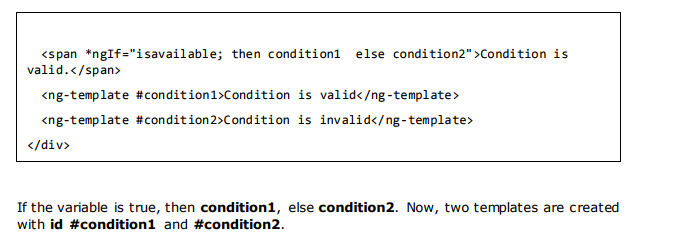
In this case, we have made the isavailable variable as false. To print the else condition, we will have to create the ng-template as follows: 🡺



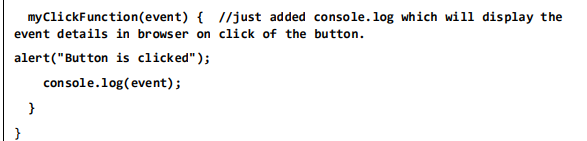
Is it possible to create if- else- if else ladder?

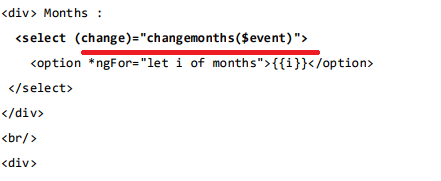
1. 🡺 Let us now use the if then else condition. 🡺 ACTUALLY THIS IS A KINGD OF SWITCH I FELT LIKE



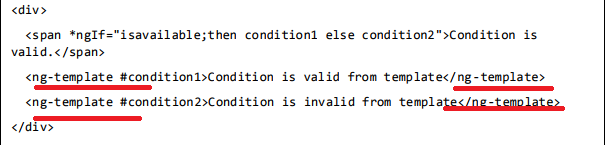


Angular 4 – Event Binding

1. When a user interacts with an application in the form of a keyboard movement, a mouse click, or a mouseover, it generates an event. These events need to be handled to perform some kind of action. This is where event binding comes into picture.
2. 
3. In the Component 🡺 
4. The following line of code will help you add the change event to the dropdown🡺



Angular 4 – Templates

1. Angular 4 uses the as the tag instead of which is used in Angular2. The reason Angular 4 changed to is because there is a name conflict between the tag and the html standard tag. It will deprecate completely going ahead. This is one of the major changes in Angular 4.
2. Ms 🡺 use the template along with the if else condition 🡺

For the Span tag, we have added the if statement with the else condition and will call template condition1, else condition2.

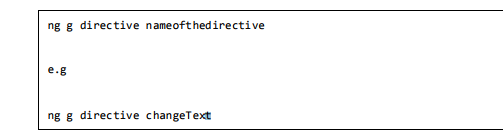
Angular 4 – Directives

1. **Directives in Angular is a js class, which is declared as @directive.** We have 3 directives in Angular. The directives are listed below:

* Component Directives 🡺These form the main class having details of how the component should be processed, instantiated and used at runtime.
* Structural Directives 🡺A structure directive basically deals with manipulating the dom elements. Structural directives have a \* sign before the directive. For example, \*ngIf and \*ngFor.
* Attribute Directives 🡺Attribute directives deal with changing the look and behavior of the dom element. You can create your own directives as shown below.

How to Create Custom Directives?

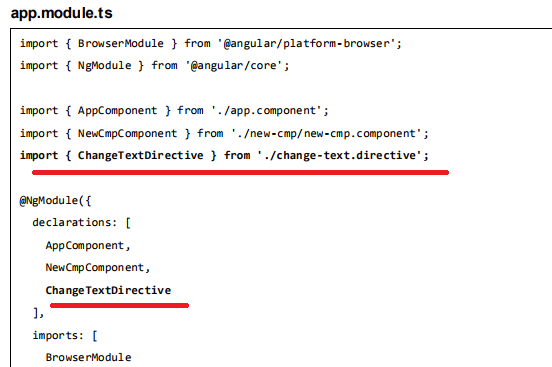
1. We will create the directive using the command line. The command to create the directive using the command line is:



On executing the above command change-text.directive.spec.ts and change-text.directive.ts get created and the app.module.ts file is updated.

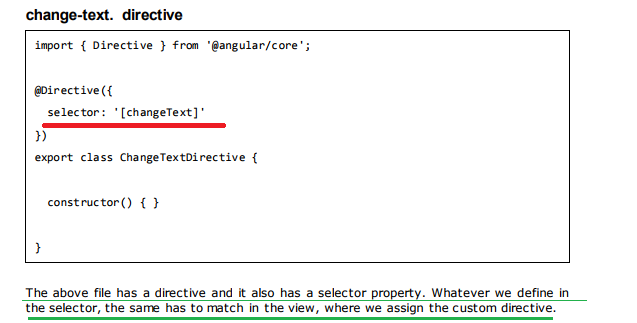
Note 🡺 I think in change-text.directive.spec.ts the name of the file is generated base don the directive name that we give.

And the changes that is MADE IN APP.MODULE.TS FILE IS 🡺

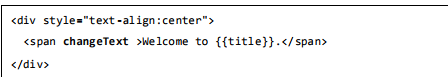


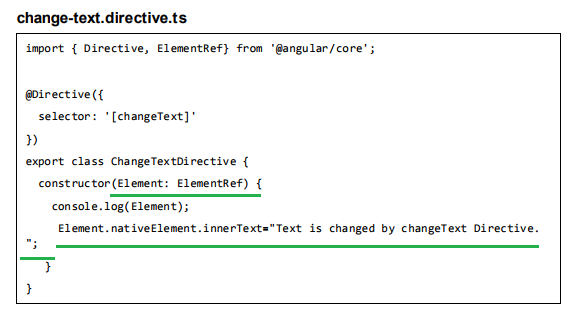
**The ChangeTextDirective class is included in the declarations in the above file. The class is also imported from the file given below.**

1. NOW LETS SEE change-text. Directive or **ChangeTextDirective class**



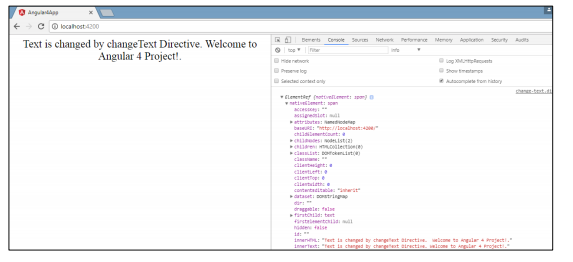
In the app.component.html view, let us add the directive as follows 🡺



1. Now let’s do the following changes in the 🡺 change-text.directive.ts

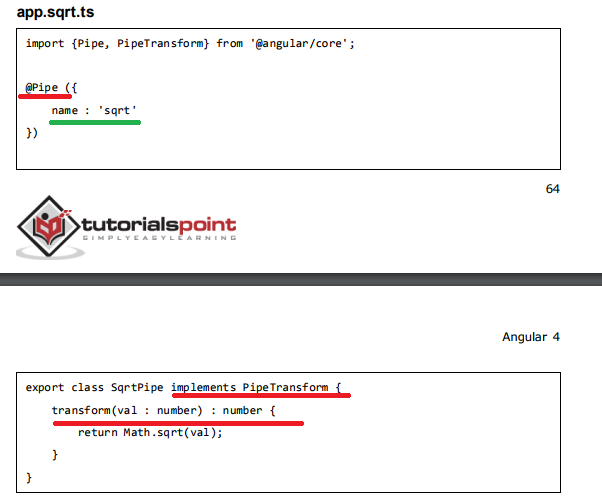
In the above file, there is a class called ChangeTextDirective and a constructor, which takes the element of type ElementRef, which is mandatory. The element has all the details to which the Change Text directive is applied.

Output 🡺

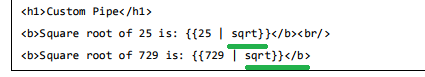


12/2/2018

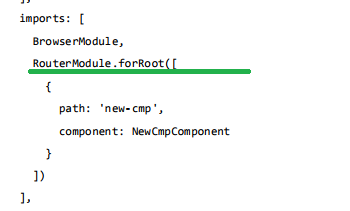
Angular 4 – Pipes

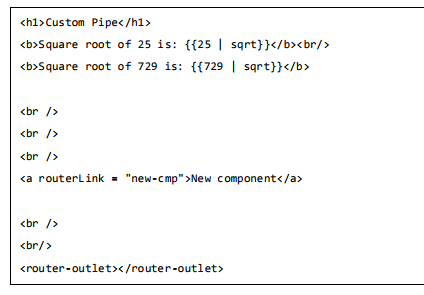
1. Pipes were earlier called filters in Angular1 and called pipes in Angular 2 and 4.
2. The | character is used to transform data.
3. Few of the build in pipes
4. {{ Welcome to Angular 4 | lowercase}}
5. **{{title | uppercase}}**
6. **{{6589.23 | currency:"USD"}}**
7. **{{6589.23 | currency:"USD":true}}** //Boolean true is used to get the sign of the currency.
8. **{{todaydate | date:'d/M/y'}} , {{todaydate | date:'shortTime'}}**
9. {{ 454.78787814 | number: '3.4-4' }} // 3 is for main integer, 4 -4 are for integers to be displayed.
10. **{{ jsonval | json }} 🡺 ms to display the data in the json format itself**
11. **{{00.54565 | percent}}**
12. **{{months | slice:2:6}} 🡺**here 2 and 6 refers to the start and the end index
13. How to Create a Custom Pipe?
14. To create a custom pipe, we have created a new ts file. Here, we want to create the sqrt custom pipe.
15. 

Note 🡺 Green color line indicates that we have to give same name in html file also where we do pipe

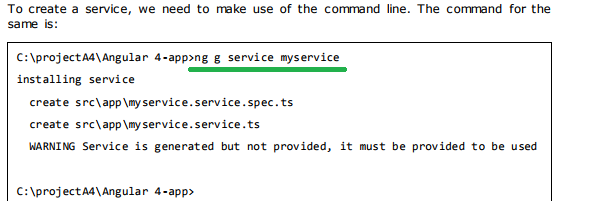
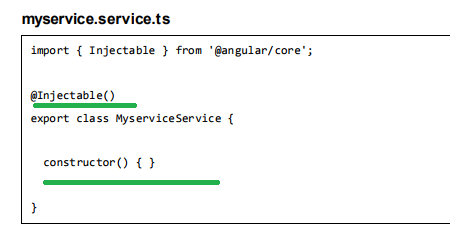
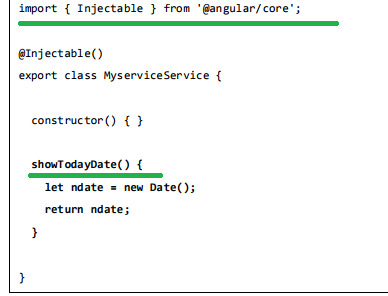
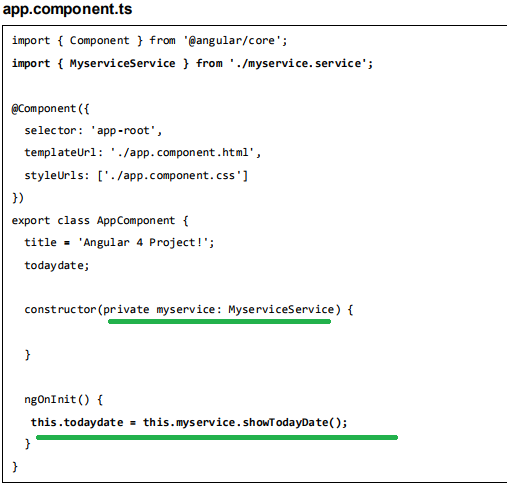
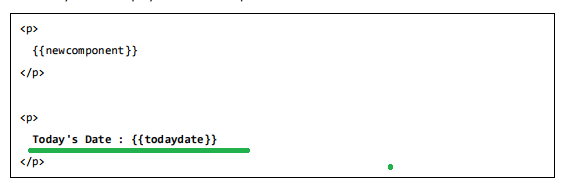
1. Since we have created a new file, we need to add the same in app.module.ts. 🡺 in the declarations array
2. Html file 🡺 

Angular 4 – Routing

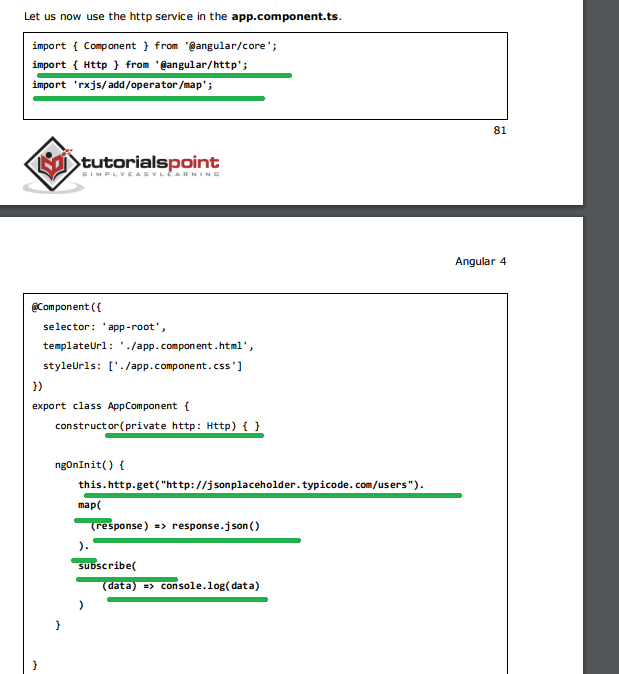
1. app.module.ts in 🡺 import { RouterModule} from '@angular/router' 🡺 



Angular 4 – Services

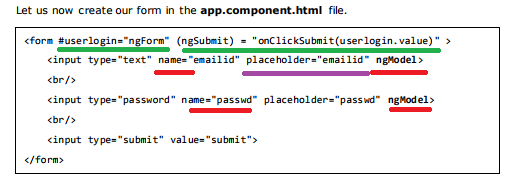
1. We might come across a situation where we need some code to be used everywhere on the page. It can be for data connection that needs to be shared across components, etc. Services help us achieve that.
2. With services, we can access methods and properties across other components in the entire project.
3. 
4. Now lest see the content of the generated file🡺
5. we need to include the service created in the main parent app.module.ts. 🡺 in the providers array
6. Implemented service class example🡺 
7. Syntax to call the service method🡺
8. Html🡺 
9. **If you change the property of the service in any component, the same is changed in other components too. Let us now see how this works.**
10. **We will define one variable in the service and use it in the parent and the new component. We will again change the property in the parent component and will see if the same is changed in the new component or not. 🡺 YES THE VALUE CHANGED IN THE PARENT COMPONENT WAS REFLECTING IN CHILD COMPONENT 🡺 TRY TO CHANGE IN CHILD AND SEE IN THE PARENT COMPONENT IF THIS SERVICE PROPERTY IS UPDATED OR NOT**

Angular 4 – Http Service

1. Http Service will help us fetch external data, post to it, etc. 🡺 Ms i.e. get, post, put, delete and etc.
2. We need to import the http module to make use of the http service.
3. we need to import the module in app.module.ts [in the imports array] 🡺import { HttpModule } from '@angular/http';
4. Example 🡺
5. **Similarly in the Documentation see for 🡺 post, put, delete, get with the parameter and etc.**

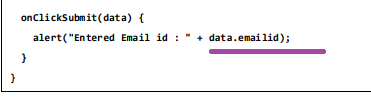
Angular 4 – Forms

1. We will discuss two ways of working with forms –
2. template driven form 🡺 need to import to FormsModule from @angular/core 🡺 ms , means most of the work is done in template or html

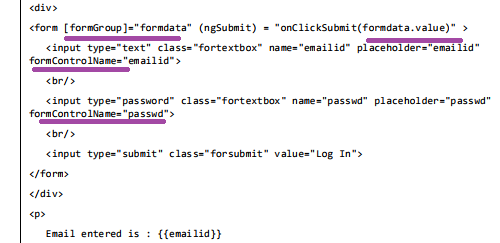


MS 🡺 It needs mandatorily name field and ngModel

So that if we want any field’s value we can access by using name attribute as shown below



1. model driven forms🡺In the model driven form, we need to import the ReactiveFormsModule from @angular/core and use the same in the imports array 🡺 ms means most of the work is done in model

* 
* Now let’s see the HTML🡺

MS🡺SEE HERE IN CASE OF MODEL DRIVEN FORMS NO NAME FIELD IS MANDATORY

1. Form Validation 🡺 later

Angular 4 – Animations

1. The difference with Angular 4 is that animation is no more a part of the @angular/core library, but is a separate package that needs to be imported in app.module.ts.
2. LATER

Angular 4 – Materials

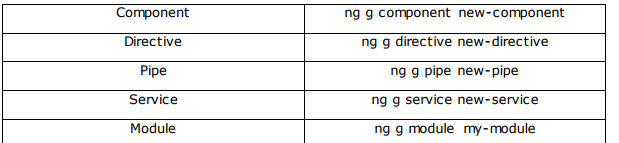
1. Angular 2 also has all the above features but they are available as part of the @angular/core module. Angular 4 has come up with a separate module @angular/materials
2. To start using materials, you need to install two packages - materials and **cdk**.
3. Material components depend on the animation module for advanced features, hence you need the animation package for the same, i.e., @angular/animations



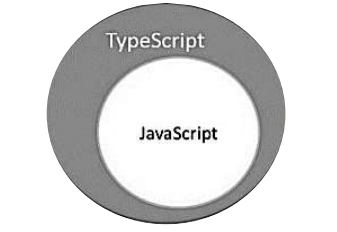
Angular 4 – CLI

1. Angular CLI comes with commands that help us create and start on our project very fast.
2. Now lets see the commands provided by CLI

* To work with Angular CLI, we need to have it installed on our system 🡺npm install -g @angular/cli
* we can run the following command in the command line and the project will be created🡺ng new PROJECT-NAME
* to compile and you can see the output of your project in the browser🡺 ng serve
* default port 🡺
* 4200 is the default port used when a new project is created. You can change the port with the following command: 🡺ng serve --host 0.0.0.0 --port 4201
* The following table lists down a few important commands required while working with Angular 4 projects.



TypeScript Notes

1. JavaScript was introduced as a language for the client side.
2. as JavaScript code grows, it tends to get messier, making it difficult to maintain and reuse the code
3. What is TypeScript? 🡺By definition, “TypeScript is JavaScript for application-scale development.”
4. TypeScript is a strongly typed, object oriented, compiled language.
5. It was designed by Anders Hejlsberg (designer of C#) at Microsoft.
6. In other words, TypeScript is JavaScript plus some additional features.
7. Features of TypeScript
8. **TypeScript is just JavaScript**🡺Hence; you only need to know JavaScript to use TypeScript.
9. **TypeScript supports other JS libraries**🡺Compiled TypeScript can be consumed from any JavaScript code. TypeScript-generated JavaScript can reuse all of the existing JavaScript frameworks, tools, and libraries.

22/5/2017

1. TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.
2. TypeScript is pure object oriented with classes, interfaces and statically typed like C# or Java.

Note🡺 **what is the difference between Client side Scripting language and Server side Scripting language?**

1. 

What is TypeScript?

TypeScript is a strongly typed, object oriented, compiled language. It was designed by **Anders Hejlsberg** (designer of C#) at Microsoft. TypeScript is both a language and a set of tools. TypeScript is a typed superset of JavaScript compiled to JavaScript.

## Features of TypeScript

1. **TypeScript is just JavaScript**.
2. **TypeScript supports other JS libraries**.
3. **JavaScript is TypeScript**. This means that any valid **.js** file can be renamed to **.ts** and compiled with other TypeScript files.
4. **TypeScript is portable**. TypeScript is portable across browsers, devices, and operating systems. It can run on any environment that JavaScript runs on. Unlike its counterparts, TypeScript doesn’t need a dedicated VM or a specific runtime environment to execute.

### TypeScript and ECMAScript

1. The ECMAScript specification is a standardized specification of a scripting language. There are six editions of ECMA-262 published.

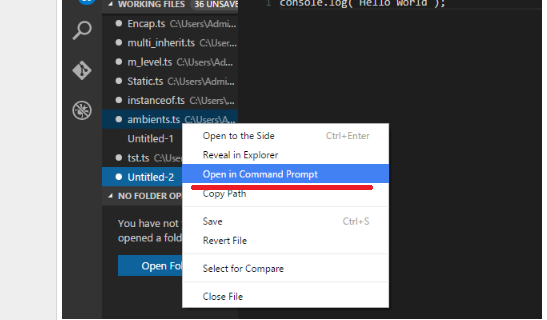


1. **TypeScript adopts its basic language features from the ECMAScript5 specification, i.e., the official specification for JavaScript. TypeScript language features like Modules and class-based orientation are in line with the EcmaScript 6 specification. Additionally, TypeScript also embraces features like generics and type annotations that aren’t a part of the EcmaScript6 specification.**
2. Why Use TypeScript?
3. **Compilation**
4. **Strong Static Typing**
5. **TypeScript supports type definitions for existing JavaScript libraries. TypeScript Definition file (with .d.ts extension) provides definition for external JavaScript libraries. Hence, TypeScript code can contain these libraries.**
6. **TypeScript supports Object Oriented Programming concepts like classes, interfaces, inheritance, etc.**

### Declaration Files

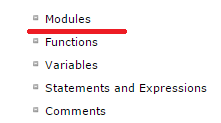
1. When a TypeScript script gets compiled, there is an option to generate a **declaration file** (with the extension **.d.ts**) that functions as an interface to the components in the compiled JavaScript.
2. The declaration files (files with **.d.ts** extension) provide intellisense for types, function calls, and variable support for JavaScript libraries like jQuery, MooTools, etc.

**Intellisense🡺Intelligent code completion**[**[1]**](https://en.wikipedia.org/wiki/Intelligent_code_completion#cite_note-1)[**[2]**](https://en.wikipedia.org/wiki/Intelligent_code_completion#cite_note-2)**is a context-aware**[**code completion**](https://en.wikipedia.org/wiki/Code_completion)**feature in some programming environments that speeds up the process of coding applications by reducing typos and other common mistakes**

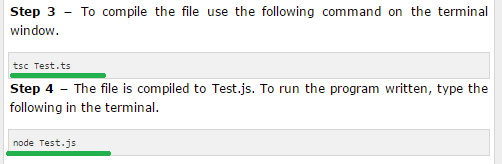


# TypeScript - Basic Syntax

1. A TypeScript program is composed of –



1. Steps to Compile and Run the TS file in Command Prompt is🡺



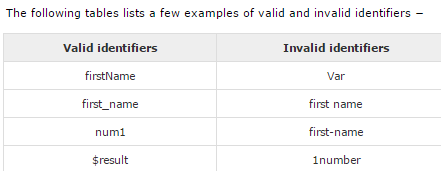
## 10. Compiler Flags

Compiler flags enable you to change the behavior of the compiler during compilation. Each compiler flag exposes a setting that allows you to change how the compiler behaves.

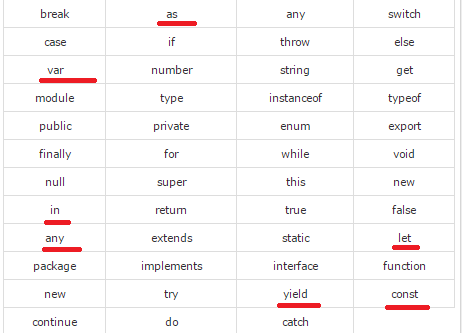


1. Multiple files can be compiled at once🡺
2. **Identifiers in TypeScript**
3. Identifiers are names given to elements in a program like variables, functions etc. The rules for identifiers are −

* Identifiers can include both, characters and digits. However, the identifier cannot begin with a digit.
* Identifiers cannot include special symbols except for underscore (\_) or a dollar sign ($).
* Identifiers cannot be keywords.
* They must be unique.
* Identifiers are case-sensitive.
* Identifiers cannot contain spaces.

1. 

## TypeScript ─ Keywords

1. 

### Whitespace and Line Breaks 🡺TypeScript ignores spaces, tabs, and newlines that appear in programs. You can use spaces, tabs, and newlines freely in your program

### TypeScript is Case-sensitive

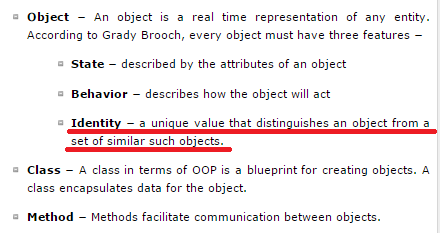
### Semicolons are optional

**Note🡺 a single line can contain multiple statements. However, these statements must be separated by a semicolon.**

### Comments in TypeScript🡺

## TypeScript and Object Orientation🡺

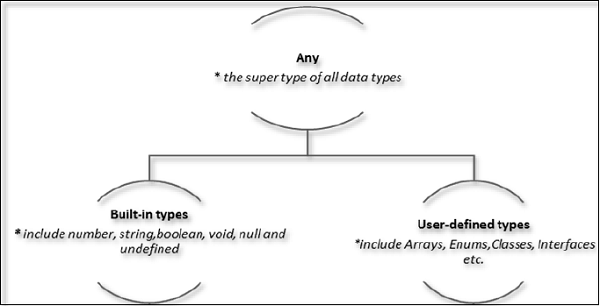
TypeScript is Object-Oriented JavaScript. Object Orientation is a software development paradigm that follows real-world modelling. Object Orientation considers a program as a collection of objects that communicate with each other via mechanism called methods. TypeScript supports these object oriented components too.



1. A simple type Script Example🡺



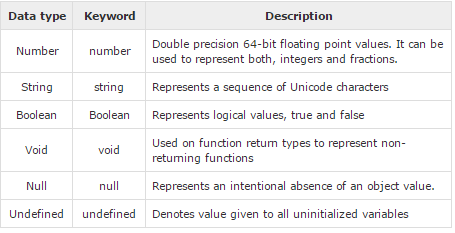
# TypeScript - Types

1. The Type System represents the different types of values supported by the language. The Type System checks the validity of the supplied values, before they are stored or manipulated by the program. This ensures that the code behaves as expected.
2. 

## The Any type

* 1. **The any data type is the super type of all types in TypeScript. It denotes a dynamic type. Using the any type is equivalent to opting out of type checking for a variable.**

## Built-in types

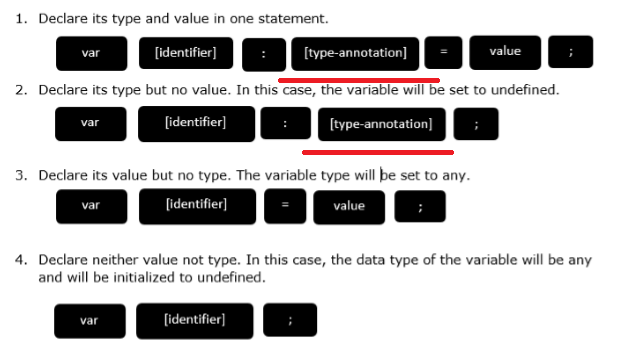
1. 
2. **Note − There is no integer type in TypeScript and JavaScript.**

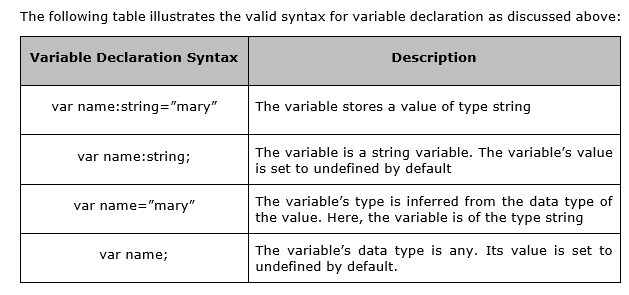
### Null and undefined ─ Are they the same?

1. The null and undefined cannot be used to reference the data type of a variable. They can only be assigned as values to a variable.
2. However, *null and undefined are not the same*. A variable initialized with undefined means that the variable has no value or object assigned to it while null means that the variable has been set to an object whose value is undefined.

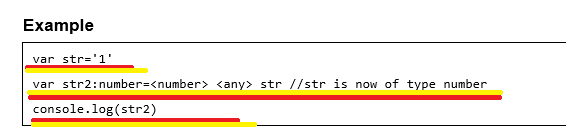
# TypeScript - Variables

1. A variable, by definition, is “a named space in the memory” that stores values. In other words, it acts as a container for values in a program
2. Variable Declaration in TypeScript🡺



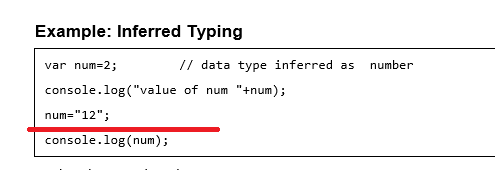
1. 
2. 

# Type Assertion in TypeScript

1. **TypeScript allows changing a variable from one type to another. TypeScript refers to this process as Type Assertion.**
2. The syntax is to put the target type between < > symbols and place it in front of the variable or expression
3. 

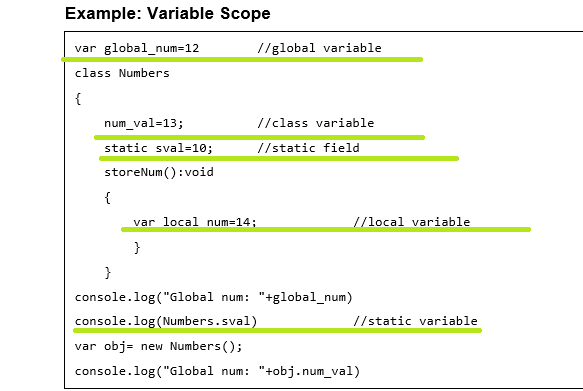
# Inferred Typing in TypeScript

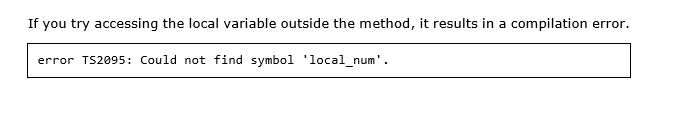
1. TypeScript also encourages dynamic typing of variables. This means that, TypeScript encourages declaring a variable without a type. In such cases, the compiler will determine the type of the variable on the basis of the value assigned to it. TypeScript will find the first usage of the variable within the code, determine the type to which it has been initially set and then assume the same type for this variable in the rest of your code block.



1. TypeScript Variable Scope🡺The availability of a variable within a program is determined by its scope.

* Global Scope ─ Global variables are declared outside the programming constructs. These variables can be accessed from anywhere within your code.
* Class Scope ─ These variables are also called fields. Fields or class variables are declared within the class but outside the methods. These variables can be accessed using the object of the class. Fields can also be static. Static fields can be accessed using the class name.
* Local Scope ─ Local variables, as the name suggests, are declared within the constructs like methods, loops etc. Local variables are accessible only within the construct where they are declared.

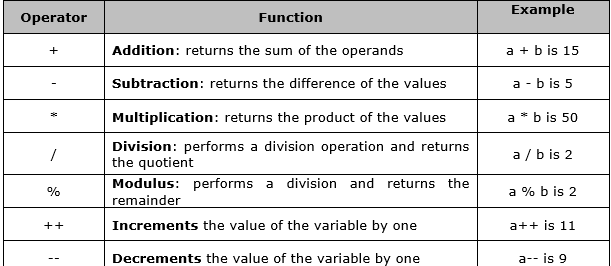
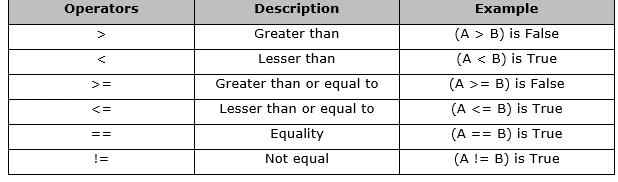
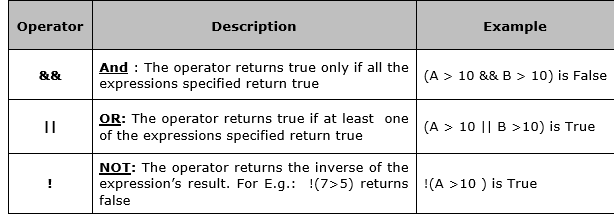


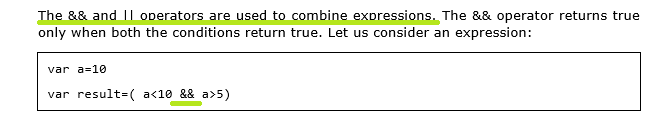
1. 

# TypeScript ─ Operators

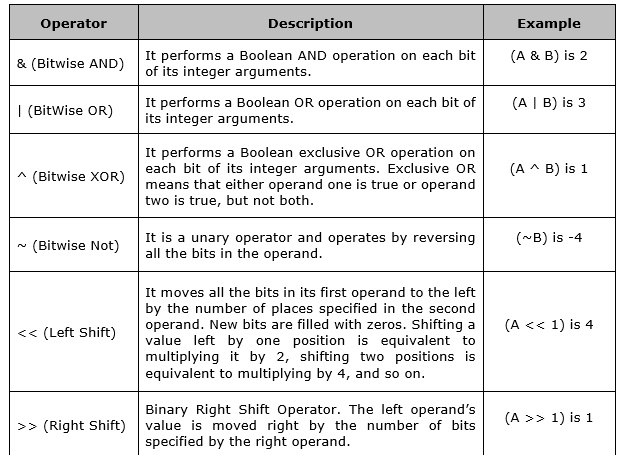
1. **An operator defines some function that will be performed on the data.** The data on which operators work are called operands.
2. The major operators in TypeScript can be classified as:

* Arithmetic operators
* Logical operators
* Relational operators
* Bitwise operators
* Assignment operators
* Ternary/conditional operator
* String operator
* Type Operator

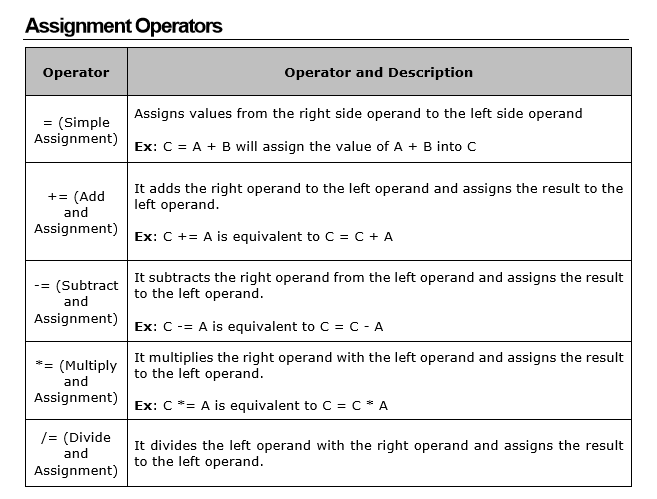
1. 
2. Relational Operators🡺
3. Logical Operators🡺
4. Short-circuit Operators (&& and ||) 🡺



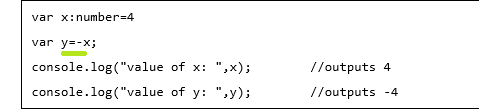
1. Bitwise Operators

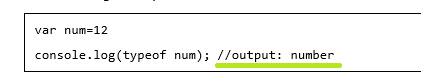


1. Assignment Operators



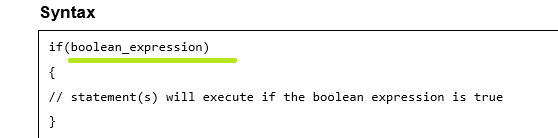
1. Miscellaneous Operators
2. The negation operator (-)

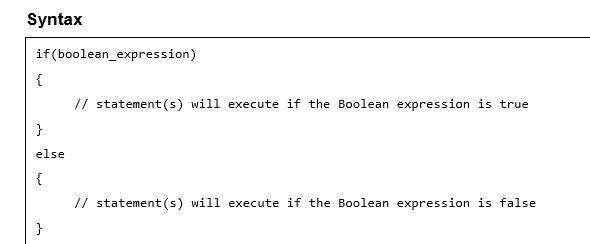


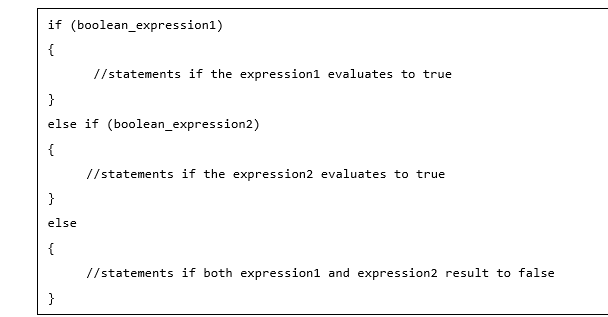
1. String Operators: Concatenation operator (+)
2. Conditional Operator (?) 🡺 Test\_Condition ? expr1 : expr2
3. Type Operators
4. typeof operator 🡺 It is a unary operator. This operator returns the data type of the operand. Take a look at the following example:
5. instanceof 🡺**This operator can be used to test if an object is of a specified type or not. The use of instanceof operator is discussed in the chapter classes.**

# TypeScript ─ Decision Making

1. **The if Statement** 🡺

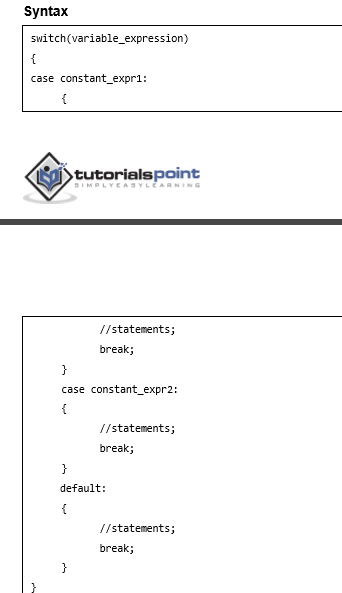


1. **The if…else Statement** 🡺
2. The else…if Ladder 🡺

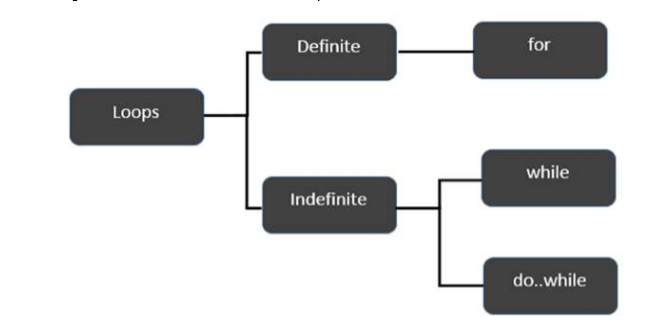
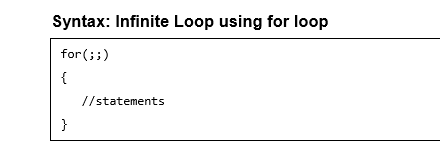
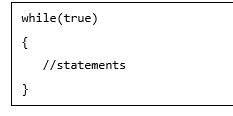


1. **The switch…case Statement**

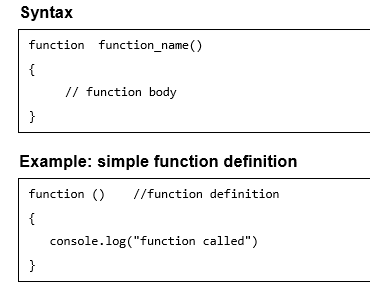
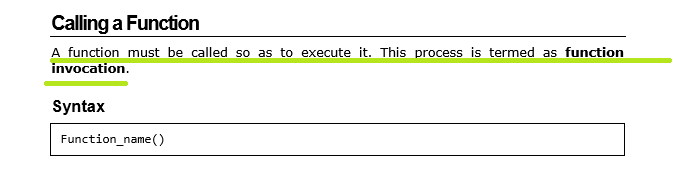
The switch statement evaluates an expression, matches the expression’s value to a case clause, and executes statements associated with that case.



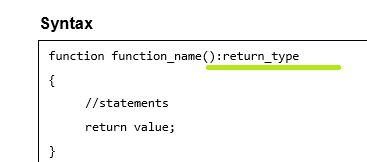
# TypeScript ─ Loops

1. 
2. The break Statement
3. The continue Statement
4. The Infinite Loop🡺An infinite loop is a loop that runs endlessly. The for loop and the while loop can be used to make an endless loop.
5. Example: Infinite loop using while loop🡺 

# TypeScript – Functions

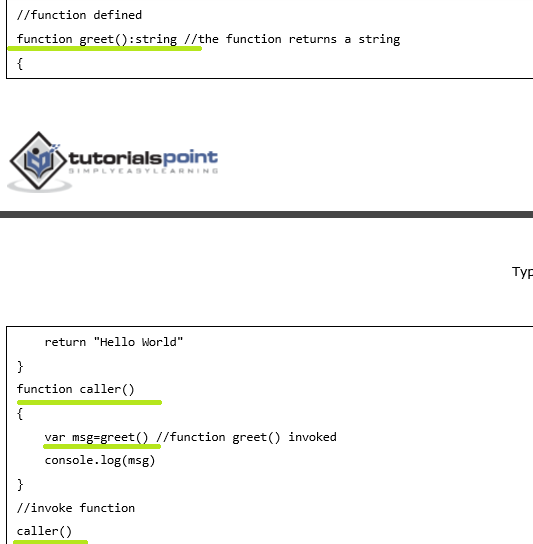
1. Functions are the building blocks of readable, maintainable, and reusable code.
2. A function is a set of statements to perform a specific task.
3. **A function declaration tells the compiler about a function's name, return type, and parameters. A function definition provides the actual body of the function.**
4. 
5. 

Returning Functions

1. Functions may also return value along with control, back to the caller. Such functions are called as returning functions.
2. 

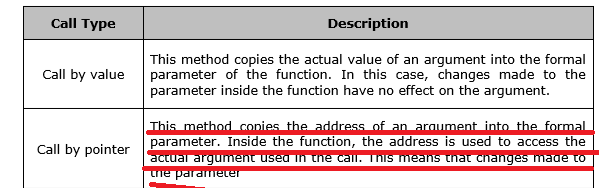
Syntax Explanation🡺

* The return\_type can be any valid data type.
* A returning function must end with a return statement.
* A function can return at the most one value. In other words, there can be only one return statement per function.
* The data type of the value returned must match the return type of the function.

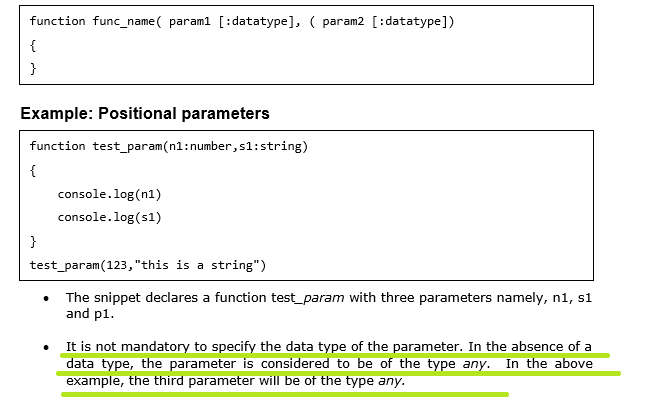
1. 

Parameterized Function

1. While calling a function, there are two ways that arguments can be passed to a function:

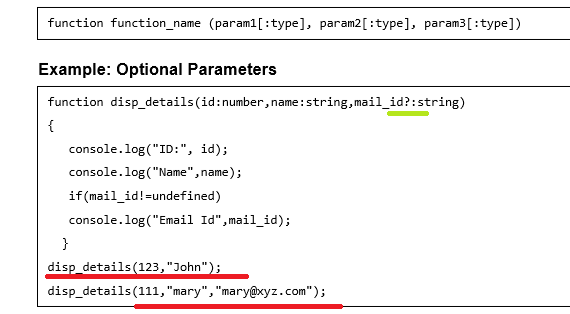


1. Following are the ways in which parameters can be used by functions:
   1. Positional Parameters



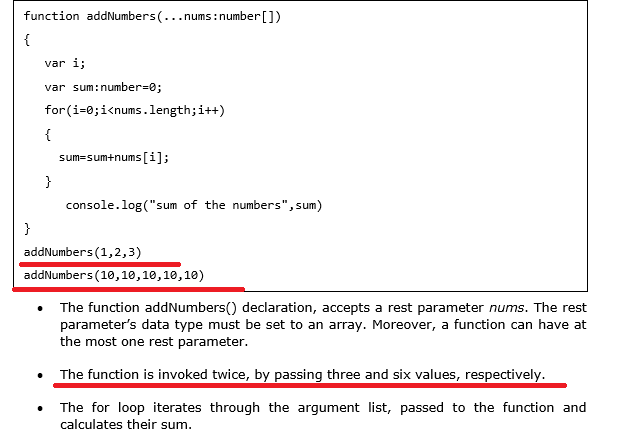
Note 🡺 In case of Positional parameters 🡺**The data type of the value passed must match the type of the parameter during its declaration. In case the data types don’t match, the compiler throws an error.**

* 1. **Optional Parameters**
* **Optional parameters can be used when arguments need not be compulsorily passed for a function’s execution. A parameter can be marked optional by appending a question mark to its name**
* **The optional parameter should be set as the last argument in a function.**
* **The syntax to declare a function with optional parameter is as given below**

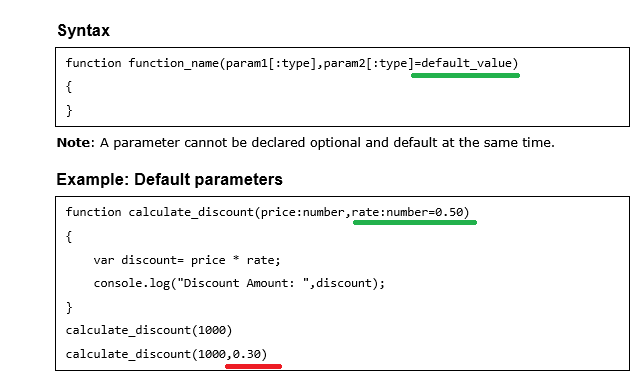


Explanation

* The above example declares a parameterized function. Here, the third parameter, i.e., mail\_id is an optional parameter.
* **If an optional parameter is not passed a value during the function call, the parameter’s value is set to undefined.**
* The function prints the value of mail\_id only if the argument is passed a value.
  1. **Rest Parameters**
* **Rest parameters are similar to variable arguments in Java. Rest parameters don’t restrict the number of values that you can pass to a function.** However, the values passed must all be of the same type. In other words, rest parameters act as placeholders for multiple arguments of the same type
* Example🡺

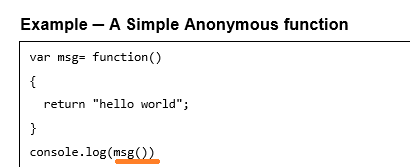


* 1. **Default Parameters**
* **Function parameters can also be assigned values by default.** However, such parameters can also be explicitly passed values.
* Example

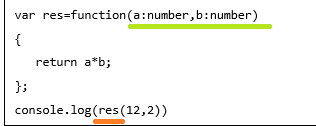


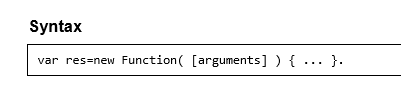
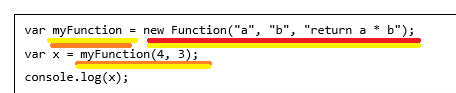
Explanation🡺**The value of the parameter rate is set to 0.50 by default. The same function is invoked, but with two arguments. The default value of rate is overwritten and is set to the value explicitly passed.**

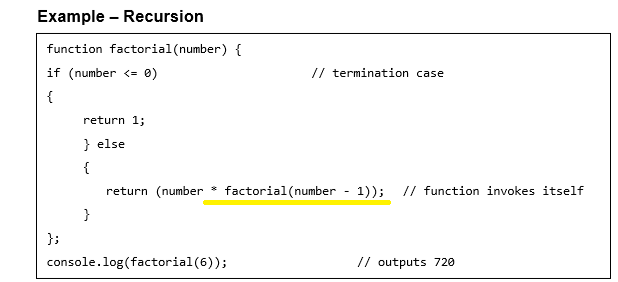
* 1. **Anonymous Function**
* Functions that are not bound to an identifier (function name) are called as anonymous functions. These functions are dynamically declared at runtime.
* Example🡺



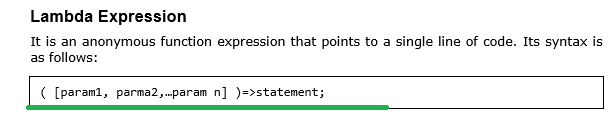
* Example 🡺Anonymous function with parameters



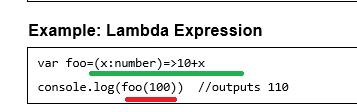
* 1. **The Function Constructor**
* **TypeScript also supports defining a function with the built-in JavaScript constructor called Function ()**
* 
* Example🡺
  1. **Recursion and TypeScript Functions**
* **Recursion is a technique for iterating over an operation by having a function call to itself repeatedly until it arrives at a result**
* Example

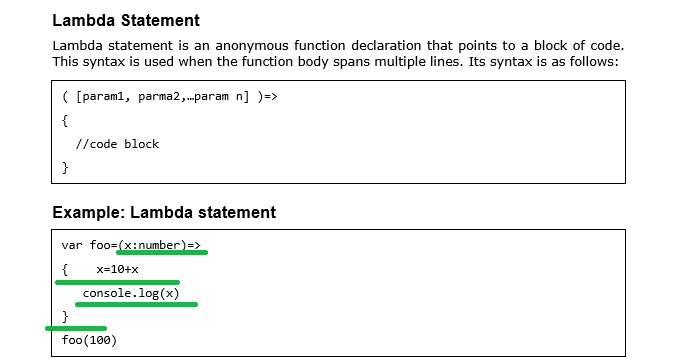


* **Example: Anonymous Recursive Function🡺we Will see this later**
  1. **Lambda Functions**
* **Lambda refers to anonymous functions in programming.**
* These functions are also called as Arrow functions.
* **Lambda Function - Anatomy**
* Parameters: A function may optionally have parameters
* **The fat arrow notation/lambda notation (=>): It is also called as the goes to operator**
* Statements: represent the function’s instruction set
* Syntax



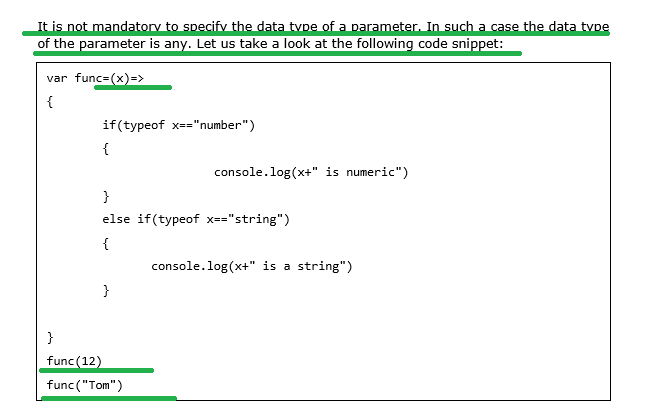
* **Example: Lambda Expression**





**Syntactic Variations**

1. Parameter type Inference



Output🡺

Its output is as follows:

12 is numeric

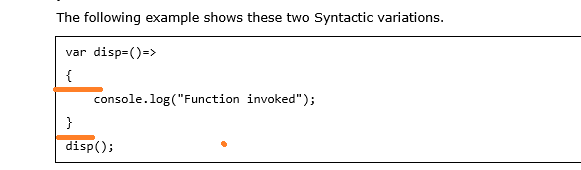
Tom is a string

1. Optional parentheses for a single parameter

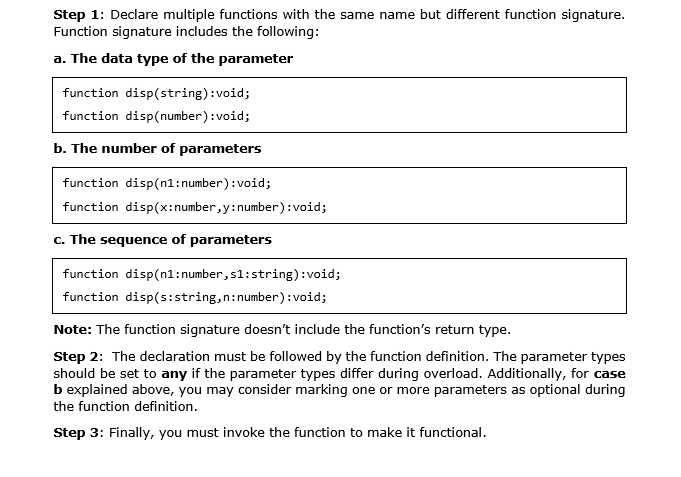


Output🡺

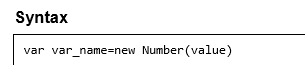
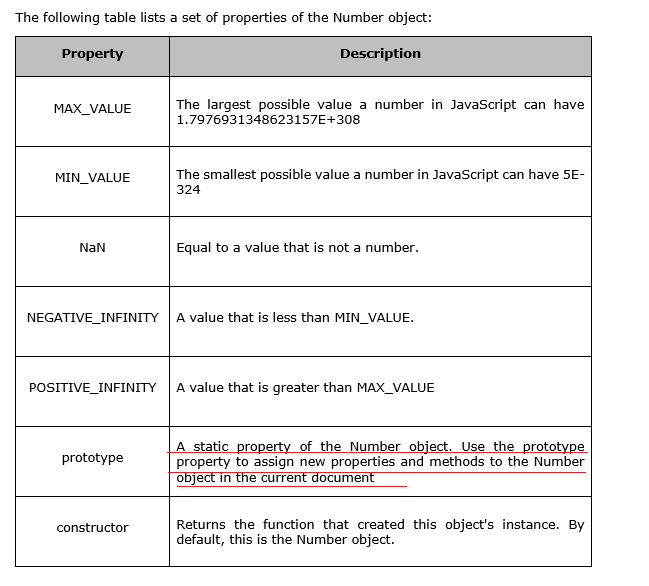
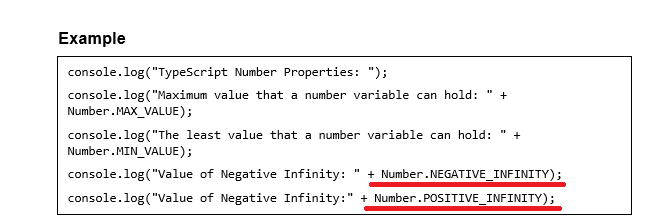
1. Optional braces for a single statement, Empty parentheses for no parameter



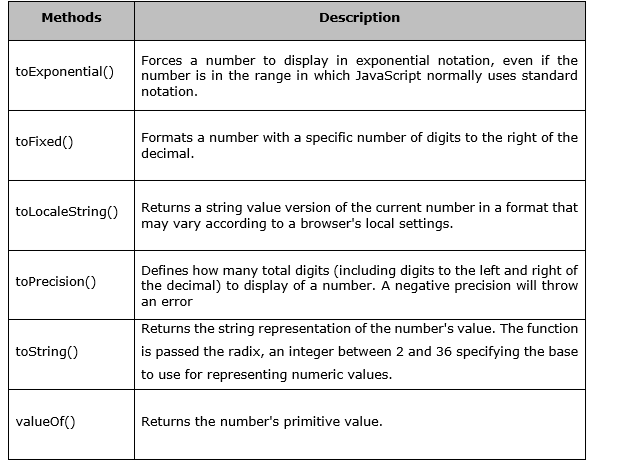
1. Function Overloads



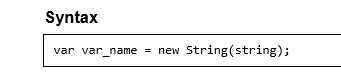
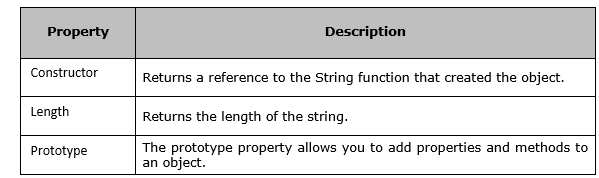
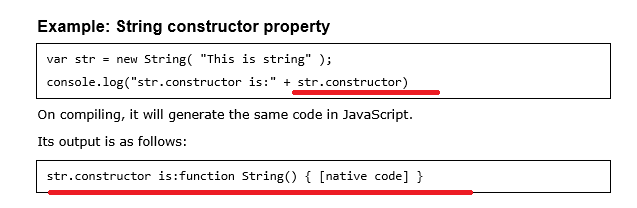
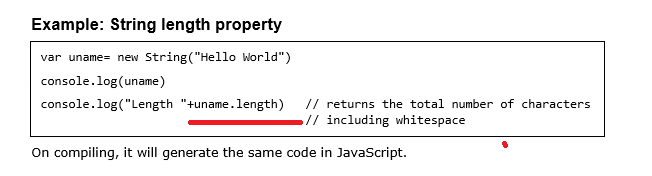
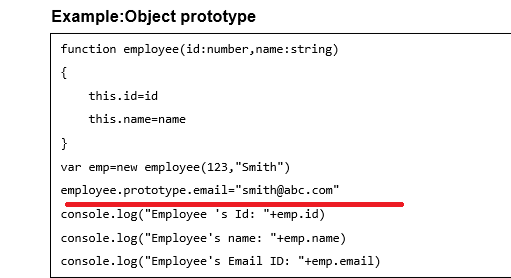
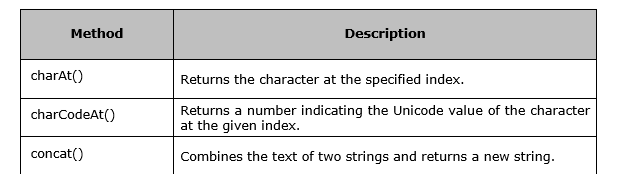
**TypeScript ─ Numbers**

1. A number object converts numeric literal to an instance of the number class. The Number class acts as a wrapper and enables manipulation of numeric literals as they were objects.
2. 
3. **In case a non-numeric argument is passed as an argument to the Number’s constructor, it returns NaN (Not–a–Number)**
4. 
5. 
6. NEEDED A CLEAR EXAMPLE and explanation ON NAN and prototype🡺 we will see this later

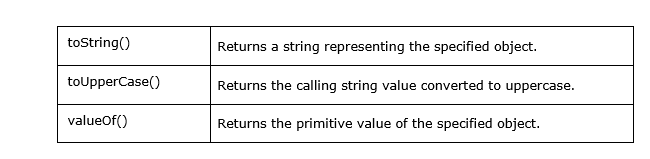
Number Methods



**TypeScript ─ Strings**

1. The String object lets you work with a series of characters.
2. 
3. 
4. 
5. 
6. 
7. String Methods
8. 



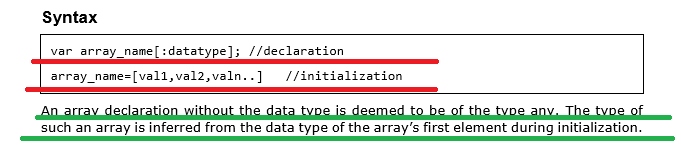


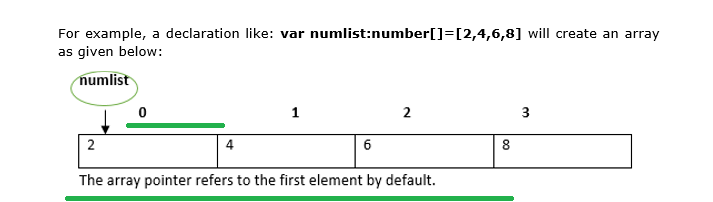
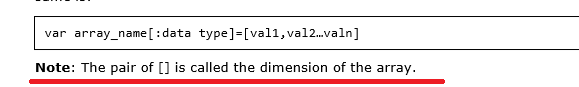
**TypeScript – Arrays**

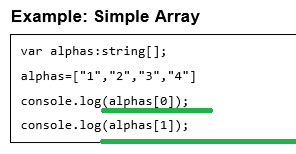
1. Features of an Array

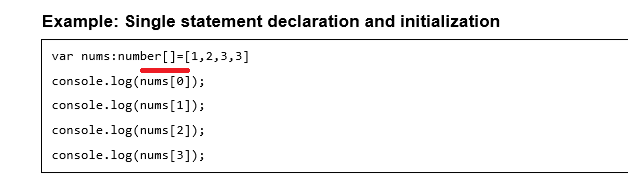
* An array declaration allocates sequential memory blocks.
* Arrays are static. This means that an array once initialized cannot be resized.
* Like variables, arrays too, should be declared before they are used. Use the var keyword to declare an array.
* Array element values can be updated or modified but cannot be deleted.

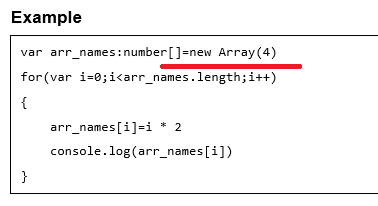
1. Declaring and Initializing Arrays

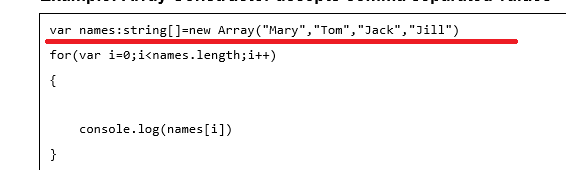


1. Example:
2. Arrays may be declared and initialized in a single statement. The syntax for the same is:
3. Accessing Array Elements example🡺

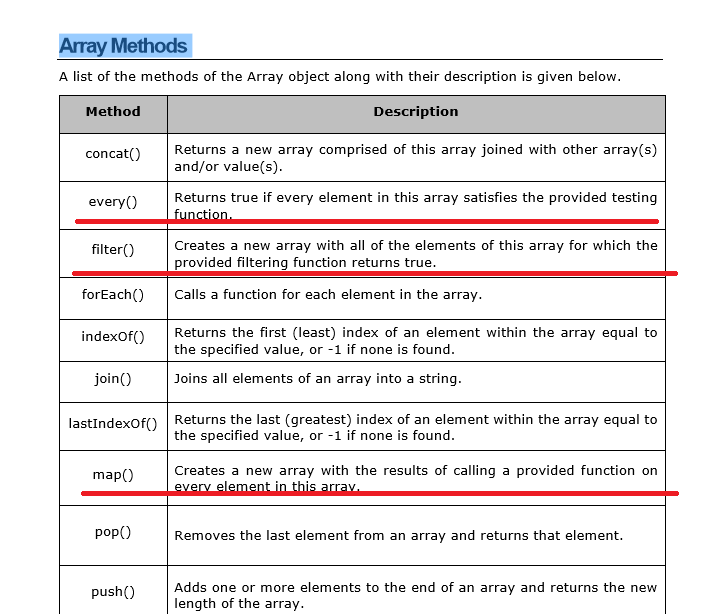


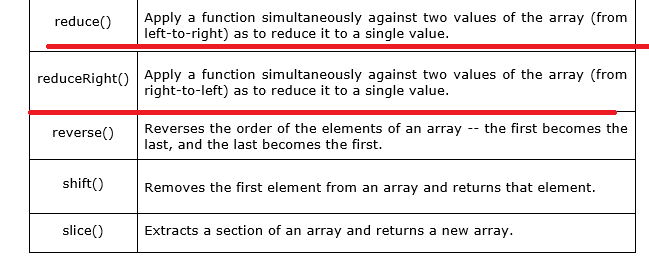
1. 
2. **Array Object**
   1. An array can also be created using the Array object. The Array constructor can be passed

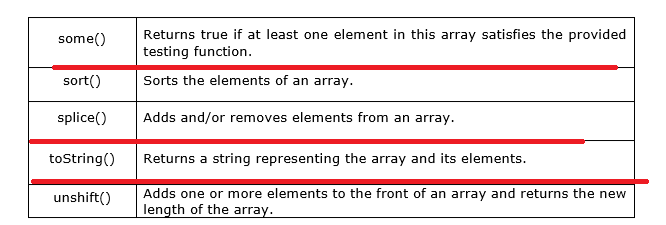
* A numeric value that represents the size of the array or
* A list of comma separated values.
  1. 
  2. Now let’s see the example for list of comma separated values



1. Array Methods



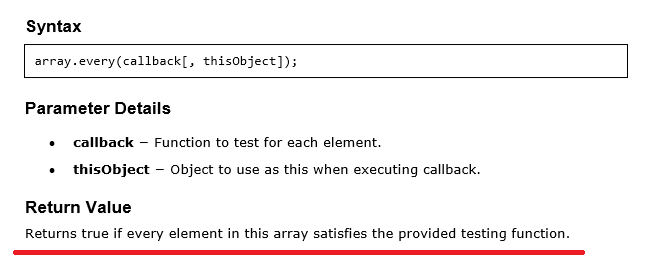


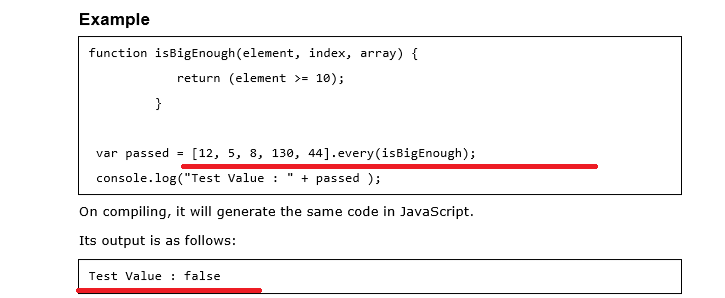


Now let’s see the example for the underlined functions

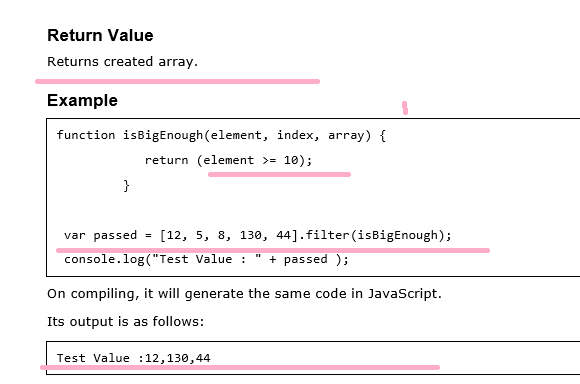
1. Every🡺 every() method tests whether all the elements in an array passes the test implemented by the provided function.

Example🡺Ms🡺Here all the elements of an array is greater than 10 or not is tested

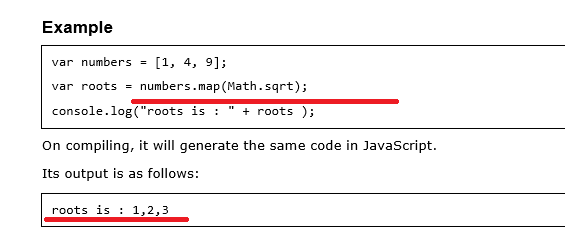




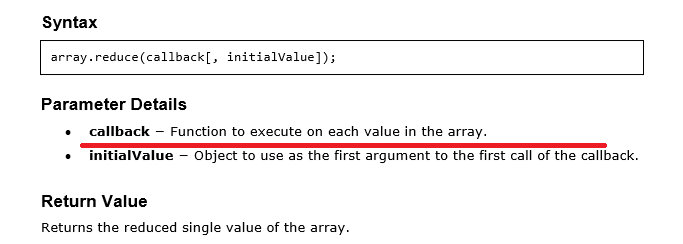
1. Filter🡺 filter() method creates a new array with all elements that pass the test implemented by the provided function.

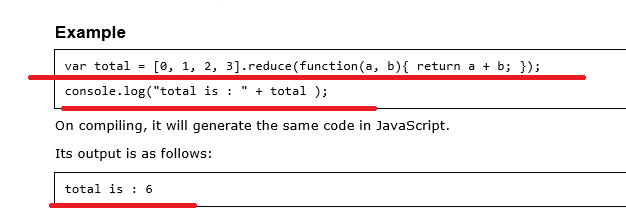


1. Foreach🡺
2. Map🡺 map() method creates a new array with the results of calling a provided function on every element in this array.

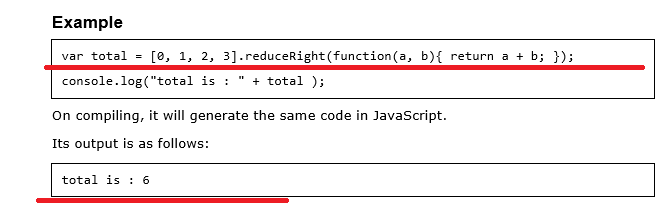


1. Reduce🡺 reduce() method applies a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.

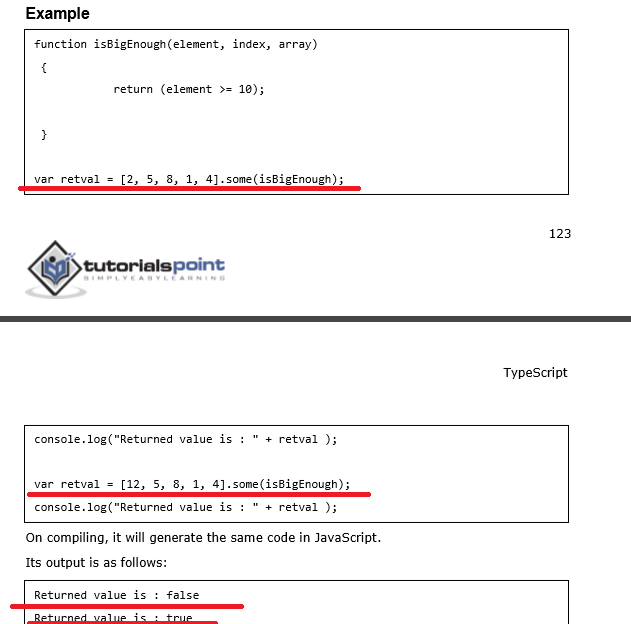




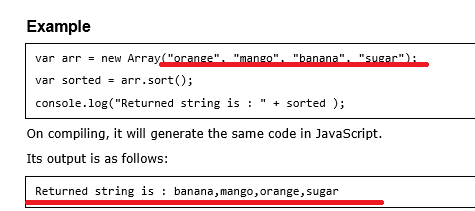
1. reduceRight🡺 reduceRight() method applies a function simultaneously against two values of the array (from right-to-left) as to reduce it to a single value



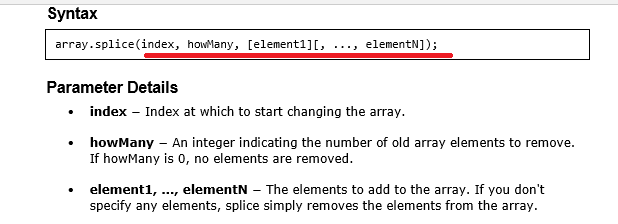
1. some 🡺 some() method tests whether some element in the array passes the test implemented by the provided function.

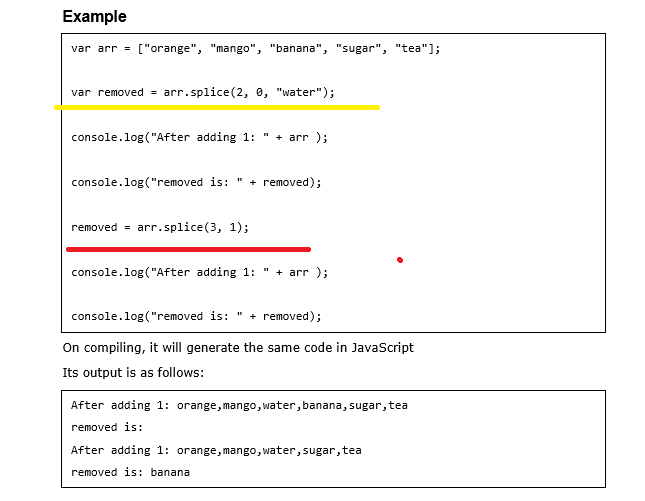


1. sort🡺

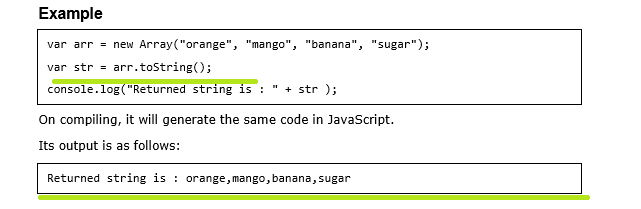


1. splice🡺 splice() method changes the content of an array, adding new elements while removing old elements

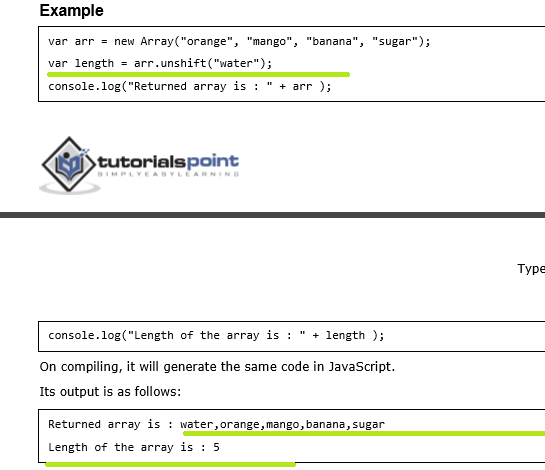




1. toString🡺

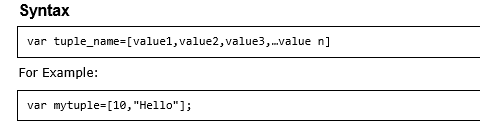


1. unshift🡺 unshift() method adds one or more elements to the beginning of an array and returns the new length of the array.



**Multidimensional Arrays🡺LATER**

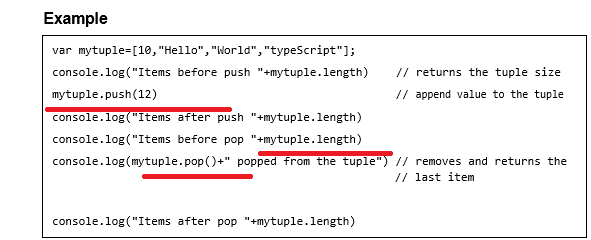
**TypeScript ─ Tuples**

1. **At times, there might be a need to store a collection of values of varied types. Arrays will not serve this purpose. TypeScript gives us a data type called tuple that helps to achieve such a purpose.**
2. It represents a heterogeneous collection of values. In other words, tuples enable storing multiple fields of different types. Tuples can also be passed as parameters to functions.
3. 
4. You can also declare an empty tuple in Typescript and choose to initialize it later.

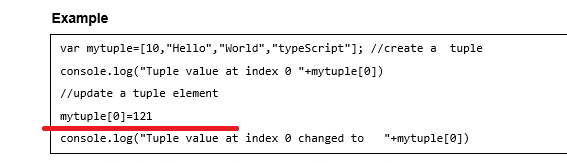


Tuple Operations

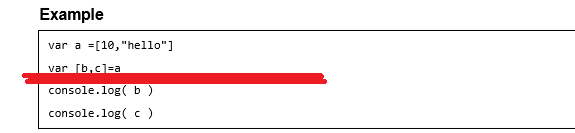
1. Tuples in TypeScript supports various operations like pushing a new item, removing an item from the tuple, etc.

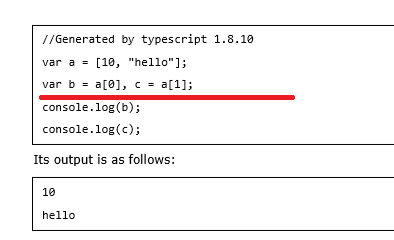


1. Updating Tuples 🡺 Tuples are mutable which means you can update or change the values of tuple elements.



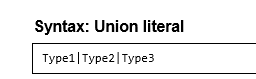
1. Destructuring a Tuple🡺 Destructuring refers to breaking up the structure of an entity. TypeScript supports destructuring when used in the context of a tuple.



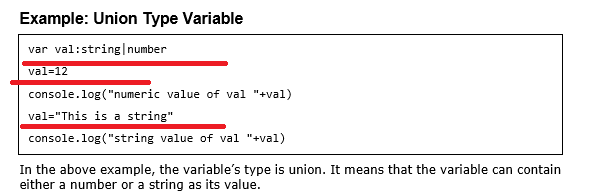
On compiling the following equivalent JS code is generated,

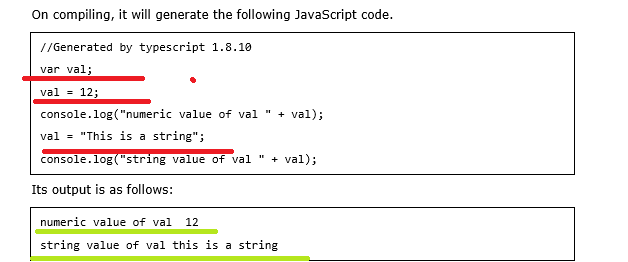
TypeScript – Union

1. TypeScript 1.4 gives programs the ability to combine one or two types. Union types are a powerful way to express a value that can be one of the several types
2. Two or more data types are combined using the pipe symbol (|) to denote a Union Type. In other words, a union type is written as a sequence of types separated by vertical bars.
3. Syntax🡺



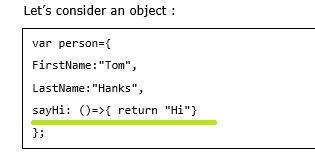
1. Example: Union Type Variable

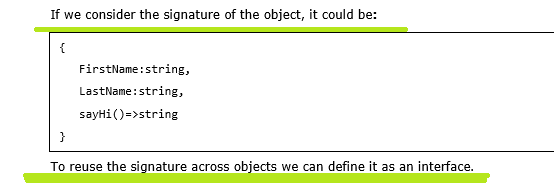


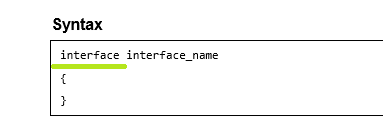


1. **I had covered only basics in this chapter**

TypeScript – Interfaces

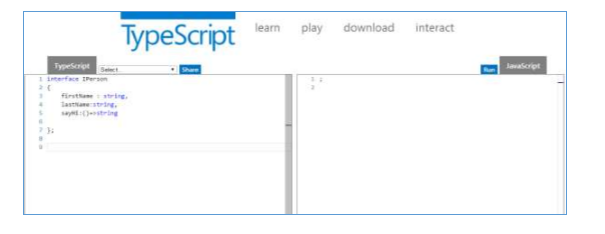
1. **An interface is a syntactical contract that an entity should conform to. In other words, an interface defines the syntax that any entity must adhere to.**
2. Interfaces define properties, methods, and events, which are the members of the interface.
3. 



1. **Declaring Interfaces**
   1. The interface keyword is used to declare an interface. Here is the syntax to declare an interface
   2. 

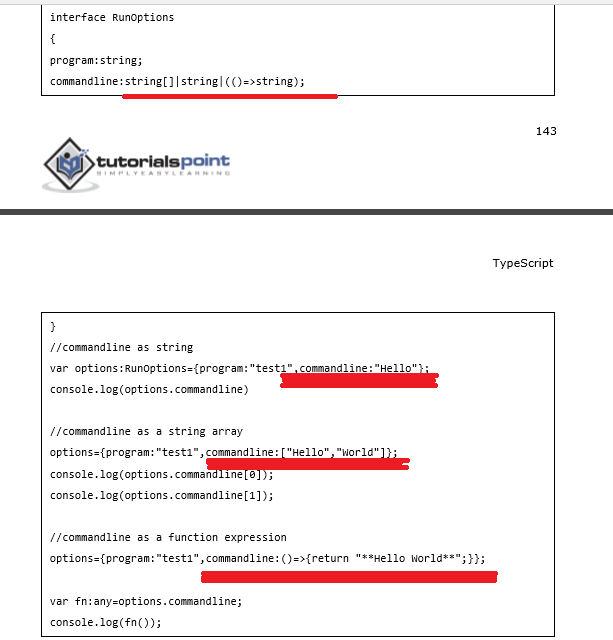


1. **Interfaces are not to be converted to JavaScript. It’s just part of TypeScript. If you see the screen shot of TS Playground tool there is no java script emitted when you declare an interface unlike a class. So interfaces have zero runtime JavaScript impact.**

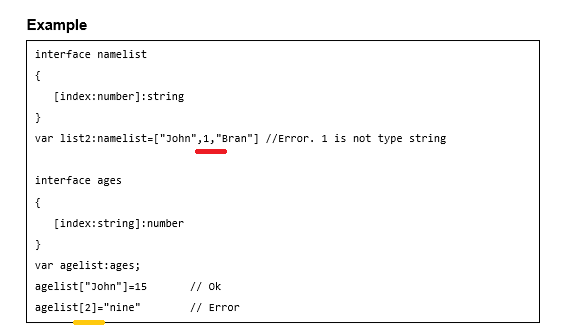


**Union Type and Interface**

1. The following example shows the use of Union Type and Interface:
2. Example🡺



Interfaces and Arrays

1. **Interface can define both the kind of key an array uses and the type of entry it contains. An array Index can be of type string or type number.**
2. Example🡺

**Interfaces and Inheritance**

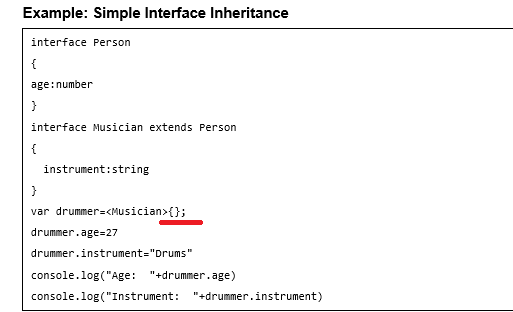
1. **An interface can be extended by other interfaces. In other words, an interface can inherit from other interface. Typescript allows an interface to inherit from multiple interfaces.**
2. Syntax: Single Interface Inheritance



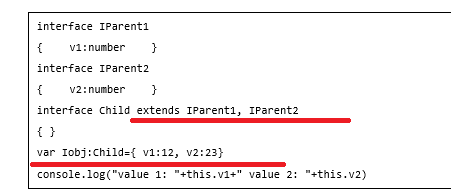
1. Syntax: Multiple Interface Inheritance



1. Example: Simple Interface Inheritance



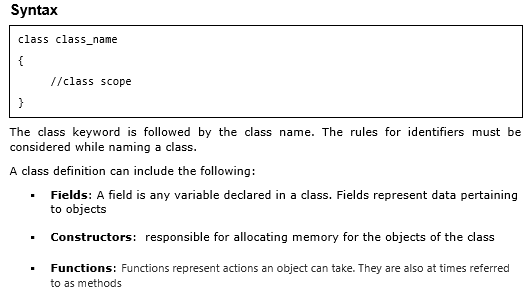
1. Example: Multiple Interface Inheritance



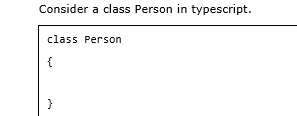
**TypeScript – Classes**

1. **TypeScript is object oriented JavaScript.**
2. **TypeScript supports object-oriented programming features like classes, interfaces, etc. A class in terms of OOP is a blueprint for creating objects. A class encapsulates data for the object. Typescript gives built in support for this concept called class. JavaScript ES5 or earlier didn’t support classes. Typescript gets this feature from ES6.**

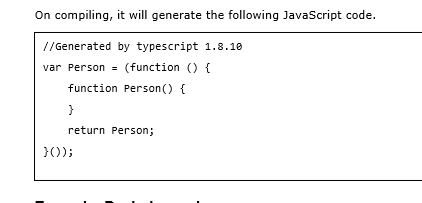
Creating classes



1. Example🡺

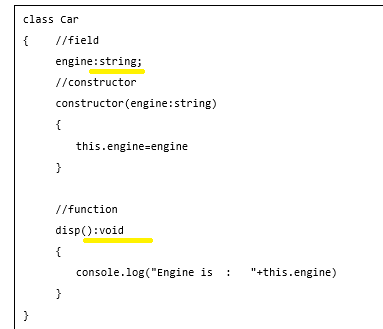


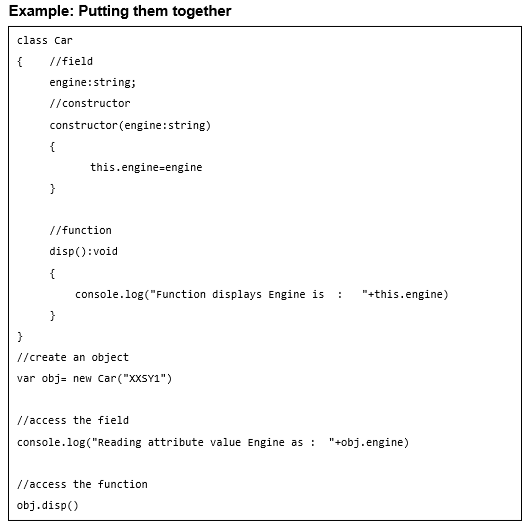
The equivalent JS code for the above code is



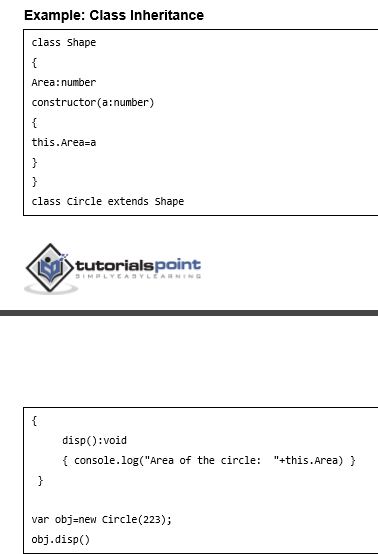
NOTE JS CREATES EQUIVALENT CODE FOR CLSS BUT IN CASE OF INTERFACE THERE IS NO EQUIVALENT CODE IN JS

1. Example: Declaring a class

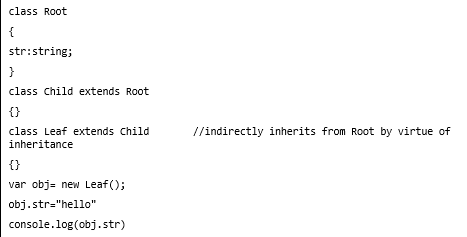


1. 

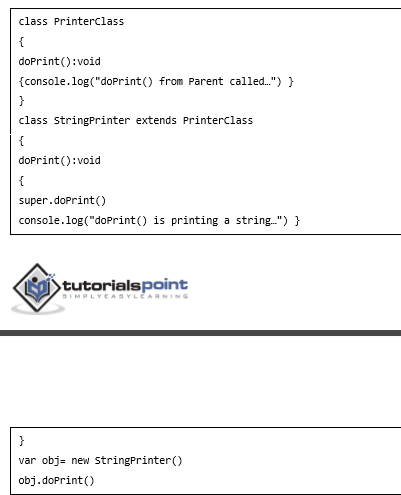
Class Inheritance

1. **TypeScript supports the concept of Inheritance. Inheritance is the ability of a program to create new classes from an existing class**
2. 
3. Inheritance can be classified as:

* Single: Every class can at the most extend from one parent class
* Multiple: A class can inherit from multiple classes. TypeScript doesn’t support multiple inheritance.
* Multi-level: The following example shows how multi-level inheritance works:

1. 

**TypeScript ─ Class inheritance and Method Overriding**

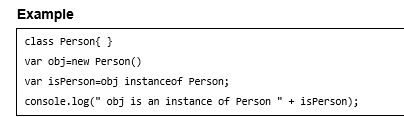
1. 

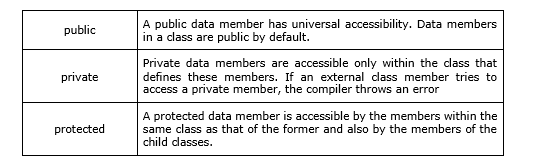
Output🡺

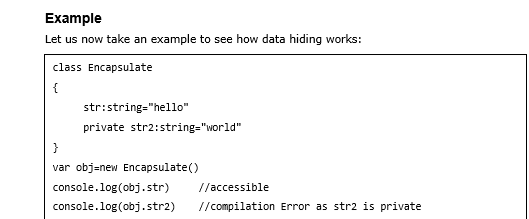
1. The static Keyword 🡺 The static keyword can be applied to the data members of a class. A static variable retains its values till the program finishes execution. Static members are referenced by the class name. 🡺Ms🡺 static can be used for variable, methods and etc…

**instanceof operator**

1. The instanceof operator returns true if the object belongs to the specified type

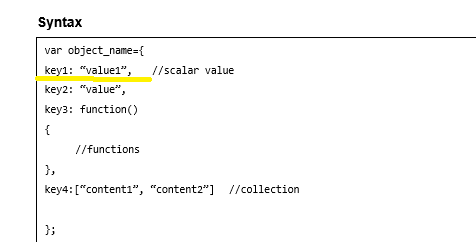


1. Data Hiding **🡺 A class can control the visibility of its data members to members of other classes. This capability is termed as Data Hiding or Encapsulation.**
2. 

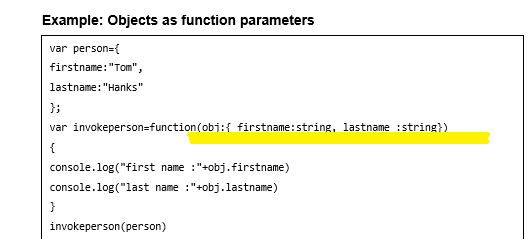


**TypeScript ─ Objects**

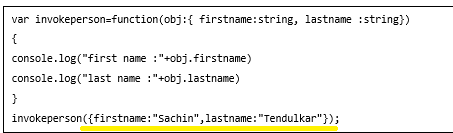
1. **An object is an instance which contains set of key value pairs. The values can be scalar values or functions or even array of other objects. The syntax is given below:**



1. **Objects as function parameters**



1. **Anonymous Object**



1. Duck-typing🡺 we will see this later

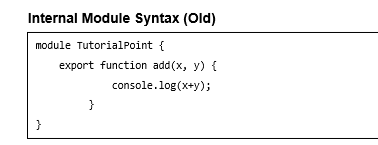
TypeScript ─ Namespaces🡺 later we will see this

**TypeScript – Modules**

1. Module is used to organize code written in TypeScript
2. Modules are broadly divided into

* Internal Modules
* External Modules

1. Internal modules came in earlier version of Typescript. This was used to logically group classes, interfaces, functions into one unit and can be exported in another module. This logical grouping is named namespace in latest version of TypeScript. So internal modules are obsolete instead we can use namespace. Internal modules are still supported, but its recommended to use namespace over internal modules.



NOTE 🡺 WE WILL SEE THE NAMESPACE AND MODULES LATER ONCE DONE WITH OUR BASIC CONCEPT

Material Design

1. Although [NgModel](https://angular.io/api/forms/NgModel) is a valid Angular directive, it isn't available by default. It belongs to the optional [FormsModule](https://angular.io/api/forms/FormsModule). You must opt-in to using that module.
2. Immediately I save the TS file corresponding JS and js map file will be created.
3. I can’t use “const” keyword inside a class i.e.

Export class ClassName{

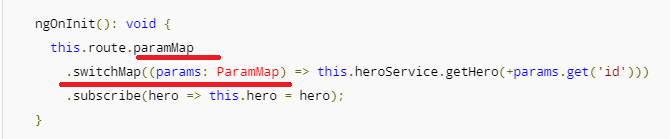
}

1. An simple example on how to compare in Angular 2is 🡺\*ngIf = ”obj1===obj2”

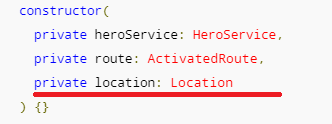
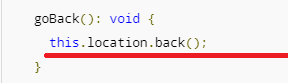
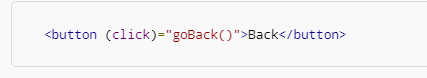
|  |  |
| --- | --- |
|  | By default, an arrow function () => {...} has the return type void unless you return something. |

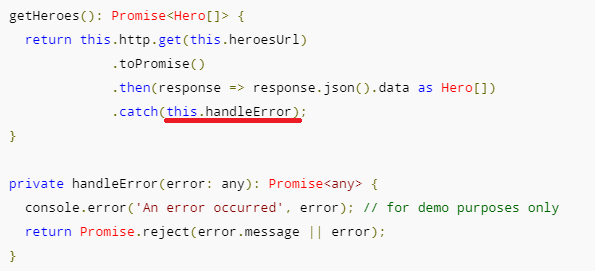
1. import { [Component](https://angular.io/api/core/Component), [Input](https://angular.io/api/core/Input) } from '@angular/core';
2. **Examples on @Input we will see this once after seeing Session notes**
3. While Using the Promise I got the Following error in the console🡺

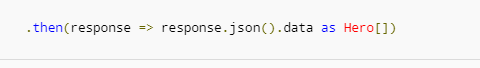
**“Cannot find a differ supporting object ‘[object Promise]’ of type ‘object’ . NgFor only supports binding to iterables such as arrays”.**

1. Multi Component Chapter has 🡺 @Input
2. Service Chapter has🡺 life cycle hooks
3. The Angular router is an external, optional Angular NgModule called [RouterModule](https://angular.io/api/router/RouterModule).
4. ([RouterOutlet](https://angular.io/api/router/RouterOutlet), [RouterLink](https://angular.io/api/router/RouterLink), [RouterLinkActive](https://angular.io/api/router/RouterLinkActive))🡺 all are directive it seems
5. Needed more information on the [ParamMap](https://angular.io/api/router/ParamMap) 🡺 this is present in the Routing chapter
6. import 'rxjs/add/operator/**switchMap'**🡺 this is present in the Routing chapter
7. example on SwitchMap and ParamMap is🡺
8. Do you need to unsubscribe?

The subscriptions are cleaned up when the component is destroyed, protecting against memory leaks, so you don't need to unsubscribe from the route paramMap Observable.

1. **Routing was not working properly for me.**
2. Now add an option, a goBack() method that navigates backward one step in the browser's history stack using the [Location](https://angular.io/api/common/Location) service.
3. import { [Location](https://angular.io/api/common/Location) } from '@angular/common';
4. 
5. 
6. 
7. The [HttpModule](https://angular.io/api/http/HttpModule) is not a core NgModule. [HttpModule](https://angular.io/api/http/HttpModule) is Angular's optional approach to web access. It exists as a separate add-on module called @angular/http
8. **Needed more explanation on InMemoryWebApiModule and InMemoryDbService**
9. Example on catch operator



1. The Angular http.get returns an RxJS Observable. Observables are a powerful way to manage asynchronous data flows
2. We can converted the Observable to a Promise using the toPromise operator. 🡺 import 'rxjs/add/operator/toPromise';
3. 
4. Each [Http](https://angular.io/api/http/Http) service method returns an Observable of HTTP [Response](https://angular.io/api/http/Response) objects.
5. toPromise operator to the Observable result of http.get()converted the Observable into a Promise and you passed that promise back to the caller.
6. Observables are to good for Asynchronous response
7. A request-cancel-new-request sequence is difficult to implement with Promises, but easy with Observables.
8. export class InMemoryDataService\_MyCUSTOMCLass implements InMemoryDbService 🡺 suppose if I change the method bame to createDB1 I get the following error in the editor

“[ts] class InMemoryDataService\_ MyCUSTOMCLass incompletely implements interface InMemoryDbService property cretaeDB is missing in type InMemoryDataService\_MyCUSTOMCLass”

1. **Angular apps are modular** and Angular has its own modularity system called NgModules.
2. Every Angular app has at least one NgModule class, [the root module](https://angular.io/guide/bootstrapping), conventionally named AppModule
3. While the root module may be the only module in a small application, most apps have many more feature modules,
4. **Angular has many decorators that attach metadata to classes so that it knows what those classes mean and how they should work**
5. The NgModule — a class decorated with @[NgModule](https://angular.io/api/core/NgModule)
6. **Data binding is also important for communication between parent and child components.**
7. **Directives** 🡺Angular templates are dynamic. When Angular renders them, it transforms the DOM according to the instructions given by **directives**.

# **Angular Material Design**

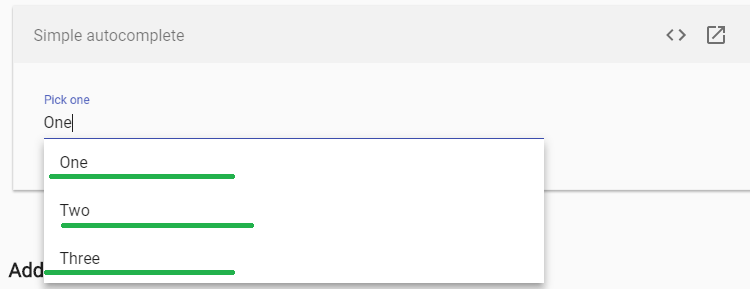
1. **Autocomplete**
2. The autocomplete is a normal input text box enhanced by a panel of suggested options
3. First lets see how to create an angular based input field



1. Now let’s see the syntax for Autocomplete
2. Now let’s see how to add this autocomplete panel to the Input text field
3. Output: 🡺

Part 1:initially🡺

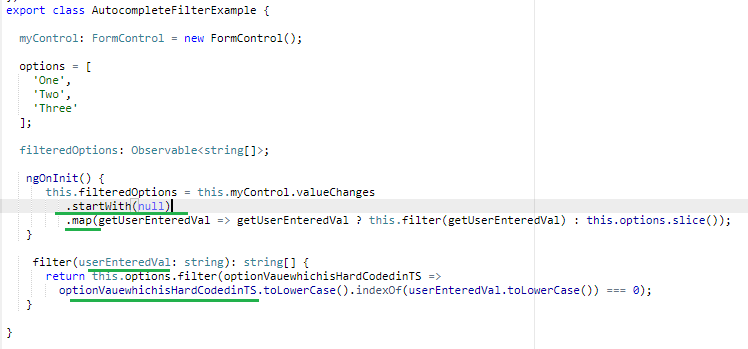
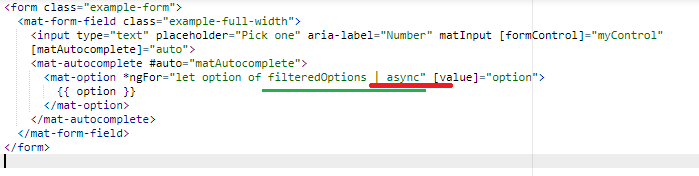
Part2:While placing the Cursor in the text box

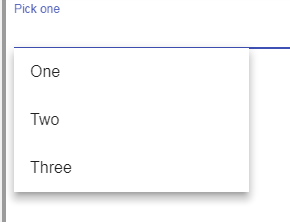


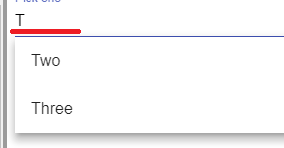
### Part2 of AutoComplete 🡺 Adding a custom filter

1. if we want our options to filter when we type, we need to add a custom filter.
2. Here we will perform a simple string test on the option value to see if it matches the input value, starting from the option's first letter
3. Needed more information on valueChanges observable ( in our example we had added it on the FormControl)
4. The resulting observable (filteredOptions) can be added to the template in place of the options property using the async pipe.
5. If you want the option's control value (what is saved in the form) to be different than the option's display value (what is displayed in the actual text field), you'll need to set the **displayWith** property on your autocomplete element.
6. For Practical explanation🡺 I will provide this later once I get the full concept

**My Understanding on Custom Filter**

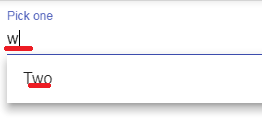
1. 
2. 
3. Explanation 🡺the flow goes as below
4. Initially filteredOptions which is an Observable of type String array is Empty
5. On Component loading/ page loading ngOnInit() method is called in which filteredOptions is set
6. Slice is used to remove the array that is passed inside it as an argument from the source array
7. Where filter() is used to filter and return a new array which is passed inside it from the source array
8. indexOf 🡺 returns position of the passed element and its start from 0 for the First position of the element
9. here Custom Pipe is used but needed more information on the second input paramenter of the Pipe “async” (See the red color underlined line in html part code
10. The output for the above code is 🡺

Initially🡺

On Entering First character 🡺

Now let’s see small change in filter method and see how output behaves🡺 🡺

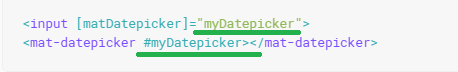
initially indexOf was compared with 0 now let’s see the output

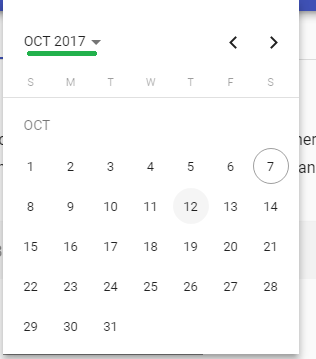
** 🡺**

**this is because of IndexOf Value**

1. **Checkbox**
2. **<mat-checkbox>** provides the same functionality as a native <input type="checkbox"> enhanced with Material Design styling and animations.
3. 

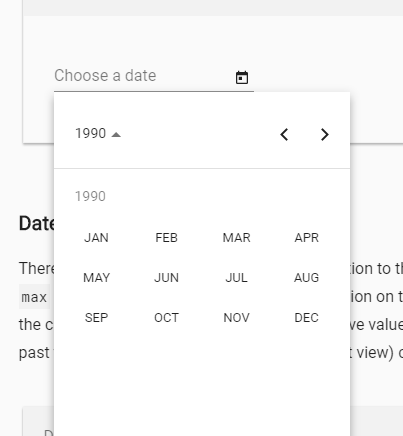
### Checkbox label

1. The checkbox label is provided as the content to the <mat-checkbox> element. The label can be positioned before or after the checkbox by setting the labelPosition property to 'before' or 'after'.
2. If you don't want the label to appear next to the checkbox, you can use [aria-label](https://www.w3.org/TR/wai-aria/states_and_properties#aria-label) or [aria-labelledby](https://www.w3.org/TR/wai-aria/states_and_properties#aria-labelledby)to specify an appropriate label.
3. The color of a <mat-checkbox> can be changed by using the color property. By default, checkboxes use the theme's accent color. This can be changed to 'primary' or 'warn'.
4. **Datepicker**
5. A datepicker is composed of a text input and a calendar pop-up, connected via the matDatepicker property on the text input.
6. 
7. <mat-datepicker-toggle> 🡺 we will see this later with an example
8. By default the calendar will open in month view, as shown below

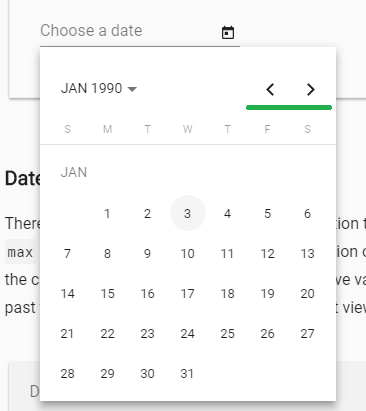


1. this can be changed by setting the startView property of mat-datepicker to "year". In year view the user will see all months of the year and then proceed to month view after choosing a month. As shown below

part 1🡺



Part2 🡺 on clicking Jan



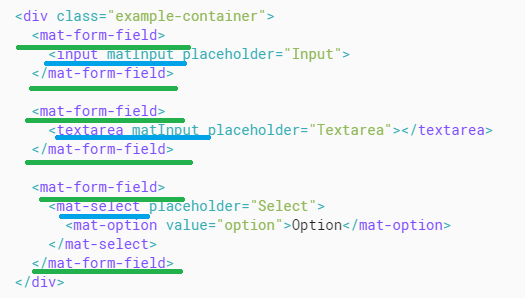




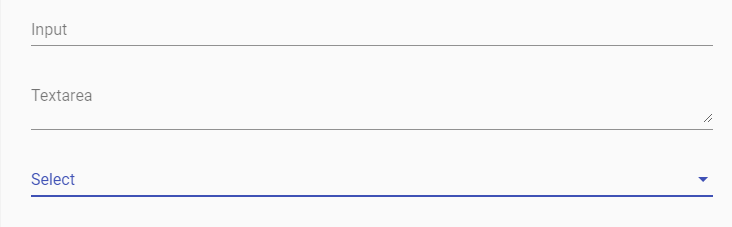
### Date validation 🡺 later

### Form field

1. <mat-form-field> 🡺 refers to the wrapper component
2. The following Angular Material components are designed to work inside a <mat-form-field>:
3. [<input matInput> & <textarea matInput>](https://material.angular.io/components/input/overview)
4. <mat-select>
5. <mat-chip-list>
6. Example🡺



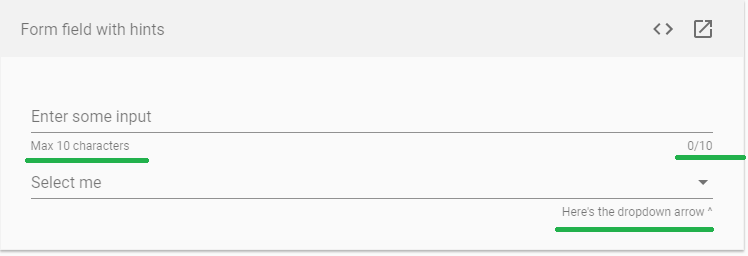
Output:🡺



1. **Floating placeholder** 🡺
2. The floating placeholder is a text label displayed on top of the form field control when the control does not contain any text
3. ***By default, when text is present the floating placeholder floats above the form field control.***
4. Placeholder text can be specified using the placeholder property on the form field control, or by adding a <mat-placeholder> element inside the form field. Only one of these options should be used, specifying both will raise an error.
5. If the form field control is marked with a required attribute, an asterisk will be appended to the placeholder to indicate the fact that it is a required field. If unwanted, this can be disabled by setting the **hideRequiredMarker** property on <mat-form-field>
6. The**floatPlaceholder** property of <mat-form-field> can be used to change this default floating behavior. It can set to **never** to hide the placeholder instead of float it when text is present in the form field control. It can be set to **always** to float the placeholder even when no text is present in the form field control. It can also be set to **auto** to restore the default behavior.
7. I was not able to see the code in doc some error is displaying

### Hint labels

1. Hint labels are additional descriptive text that appears below the form field's underline
2. A <mat-form-field> can have up to two hint labels; one start-aligned (left in an LTR language, right in RTL), and one end-aligned.
3. Hint labels are specified in one of two ways: either by using the **hintLabel** property of <mat-form-field>, or by adding a <**mat**-**hint**> element inside the form field
4. When adding a hint via the hintLabel property, it will be treated as the start hint
5. Hints added via the <mat-hint> hint element can be added to either side by setting the align property on <mat-hint> to either start or end. Attempting to add multiple hints to the same side will raise an error.



### Error messages

### Error messages can be shown under the form field underline by adding mat-error elements inside the form field

### Errors are hidden initially and will be displayed on invalid form fields after the user has interacted with the element or the parent form has been submitted. Since the errors occupy the same space as the hints, the hints are hidden when the errors are shown.

### If a form field can have more than one error state, it is up to the consumer to toggle which messages should be displayed. This can be done with CSS, ngIf or ngSwitch.

### 

### Code🡺 error msg displayed

### Prefix & suffix

### Custom content can be included before and after the input tag, as a prefix or suffix by using adding the matPrefix and matSuffix directive to an element inside the <mat-form-field>

### 

### 

### 

### Custom form field controls and Theming 🡺 later

### Input

### Supported input types

### The following [input types](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input) can be used with matInput

* date
* datetime-local
* email
* month
* number
* password
* search
* tel
* text
* time
* url
* week

### Radio button

### mat-radio> provides the same functionality as a native <input type="radio"> enhanced with Material Design styling and animations.

### 

### 

### Radio-button label🡺 The label can be positioned before or after the radio-button by setting the labelPosition property to 'before' or 'after'.

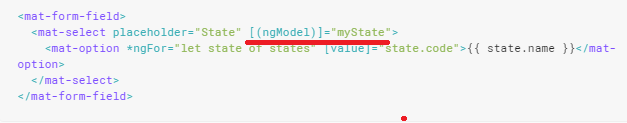
### Radio groups

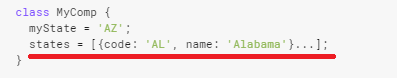
### Radio-buttons should typically be placed inside of an <mat-radio-group>

### The radio-group has a valueproperty that reflects the currently selected radio-button inside of the group

# **Select**

* + 1. <mat-select> is a form control for selecting a value from a set of options, similar to the native<select> element
    2. Note that you can disable items by adding the **disabled** Boolean attribute or binding to it.
    3. 
    4. 2nd example by using 2 way data binding



* + 1. 

### Resetting the select value

* + 1. Example🡺
    2. Output 🡺
       1. Initially 🡺
       2. On Placing the cursor in the select field🡺
       3. On selecting “None” in the option🡺

### Customizing the trigger label

* + 1. **If you want to display a custom trigger label inside a select, you can use the mat-select-trigger element 🡺 Example later**

# **Slider**

1. **<mat-slider>** allows for the selection of a value from a range via mouse, touch, or keyboard, similar **to <input type="range">.**
2. 
3. By default the minimum value of the slider is 0, the maximum value is 100, and the thumb moves in increments of 1. **These values can be changed by setting the min, max, and step attributes respectively**. The initial value is set to the minimum value unless otherwise specified.
4. 

### Orientation

1. By default sliders are horizontal with the minimum value on the left and the maximum value on the right. The vertical attribute can be added to a slider to make it vertical with the minimum value on bottom and the maximum value on top.
2.  🡺 output🡺 

### 🡺

### Output🡺

# **Slide toggle**

### <mat-slide-toggle> is an on/off control that can be toggled via clicking or dragging.

### 

### 

# **Menu**

### <mat-menu> is a floating panel containing list of options.

### Example 🡺

### 

### On Clicking Menu button (see point b)🡺

### Nested menu 🡺 now lets see the syntax for nested menu

### 

### On clicking the above Animal Index button 🡺

### Code🡺

### 

1. **Needed More information on 🡺@ViewChild**

### Now lets see an example with ICON and text in the menu

### 

### Code🡺and Final Output for the above code

### 

### Customizing menu position

### By default, the menu will display below (y-axis), after (x-axis),

### The position can be changed using the xPosition (before | after) and yPosition (above | below) attributes.

### 

### Output 🡺

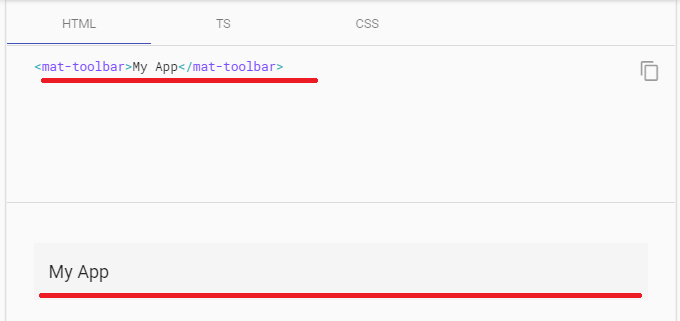
### on clicking the vert icon 🡺

### Nested menu example for above scenario🡺 code we will see later now lets see just the output

### 

# **Sidenav 🡺 Later**

# **Toolbar**

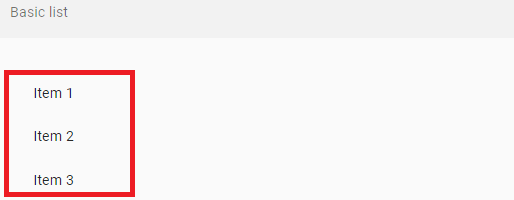
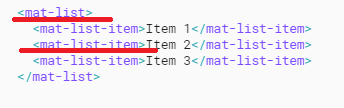
1. <mat-toolbar> is a container for headers, titles, or actions.

**Layouts**

# **List**

### Type of list🡺 simple list, Navigation list, Selection lists, Multi-line lists, Lists with icons, Lists with avatars(means images), Dense lists, Lists with multiple sections

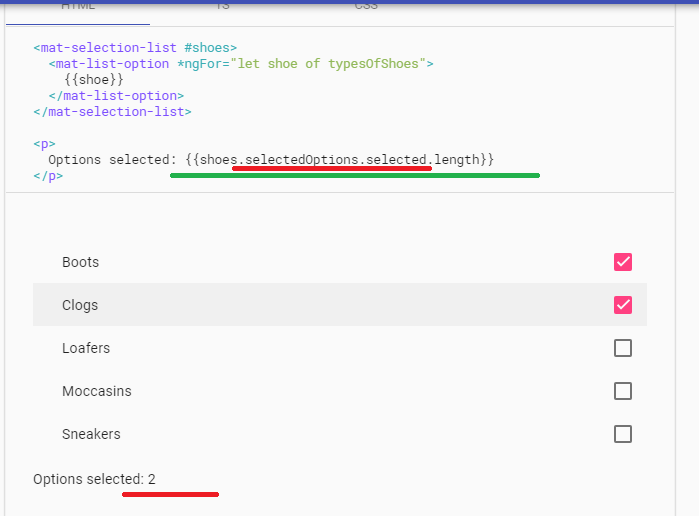
,

1. <mat-list> is a container component that wraps and formats a series of line items.
2. 
3. Code for Simple list🡺

### Navigation lists

1. Use mat-nav-list tags for navigation lists (i.e. lists that have anchor tags).
2. 
3. 

### Selection lists

1. 



### Dense lists

1. Lists are also available in "dense layout" mode, which shrinks the font size and height of the list to suit UIs that may need to display more information. To enable this mode, add a dense attribute to the main mat-list tag.

# **Grid list**

### mat-grid-list is a two-dimensional list view that arranges cells into grid-based layout.

### 

### Output🡺

### Setting the number of columns🡺 ms for a grid list

### An mat-grid-list must specify a cols attribute which sets the number of columns in the grid. The number of rows will be automatically determined based on the number of columns and the number of items.

### Setting the row height 🡺 ms for a grid list

### 

### mat-grid-tile 🡺 A header and footer can be added to an mat-grid-tile using the mat-grid-tile-header and mat-grid-tile-footer elements respectively.

# **Card**

### <mat-card> is a content container for text, photos, and actions in the context of a single subject.

### 

### 

### Basic card sections🡺some of the attributes and elements that can be added in the card is🡺

# **Stepper**

### Angular Material's stepper provides a wizard-like workflow by dividing content into logical steps.

### Example 🡺

### Step1🡺

### 

### Step2🡺

### Step3🡺

### 

“Enable linear mode” 🡺 this button is used to enable the form validation that is provided by material on clicking on it

### 

### 

### Stepper variants 🡺There are two stepper components: mat-horizontal-stepper and mat-vertical-stepper. They can be used the same way. The only difference is the orientation of stepper.

### Labels 🡺 if a step's label is only text, then the label attribute can be used🡺

### For more complex labels, add a template with the matStepLabel directive inside the mat-step.

### Stepper buttons🡺 There are two button directives to support navigation between different steps: matStepperPrevious and matStepperNext.

### 

### Linear stepper🡺 The linear attribute can be set on mat-horizontal-stepper and mat-vertical-stepper to create a linear stepper that requires the user to complete previous steps before proceeding to following steps.

# **Tabs**

### 

### dynamicHeight  🡺 Needed more information on this attribute of tab section

### Tabs and navigation

### 

# **Expansion Panel**

### <mat-expansion-panel> provides an expandable details-summary view.

### Output 🡺

### Now on clicking Down ward arrow

### 

### Code🡺

#### Action bar🡺 Actions may optionally be included at the bottom of the panel, visible only when the expansion is in its expanded state.

#### **Disabling a panel** 🡺

### Accordion🡺 Multiple expansion-panels can be combined into an accordion. *The multi="true" input allows the expansions state to be set independently of each other. When multi="false" (default) just one panel can be expanded at a given time:*

### Example 🡺

### Now on clicking the down ward arrow we get the following output

### 

### <mat-accordion multi="true"> 🡺 allows us to open all the expandable sections independent of each other

**BUTTONS & INDICATORS**

# **Button**

### Angular Material buttons are native <button> or <a> elements enhanced with Material Design styling and ink ripples.

### 🡺

### 

# **Button toggle**

### a. <mat-button-toggle> are on/off toggles with the appearance of a button.

<mat-button-toggle>Toggle me!</mat-button-toggle> 🡺

Output:



Now on clicking the above button

### 

# **Chips**

### <mat-chip-list> displays a list of values as individual, keyboard accessible, chips.

### 

### 

### Unstyled chips 🡺 By default, <mat-chip> has Material Design styles applied.

1. Selection🡺 Chips can be selected via the selected property. Selection can be disabled by setting selectable to false on the <mat-chip-list>.Whenever the selection state changes, a **ChipSelectionChange** event will be emitted via(**selectionChange**).

### Disabled chips🡺 Individual chips may be disabled by applying the disabled attribute to the chip. When disabled, chips are neither selectable nor focusable

### Orientation 🡺 If you want the chips in the list to be stacked vertically, instead of horizontally, you can apply the mat-chip-list-stacked class, as well as the aria-orientation="vertical" attribute:

### 

# **Icon**

### mat-icon makes it easier to use *vector-based* icons in your app. This directive supports both icon fonts and SVG icons,

### 🡺

### Remaining topics under this section we will see it later

# **Progress spinner**

### <mat-progress-spinner> and <mat-spinner> are a circular indicators of progress and activity.

### 

### 

### Note 🡺 Progress Snipper is circular loading symbol and Progress bar is horizontal line to show the progress

### 

### Or

### 

# **Progress bar**

### <mat-progress-bar> is a horizontal progress-bar for indicating progress and activity.

### 

### 

### Now lets see the example for all the above modes🡺

### Determinate🡺

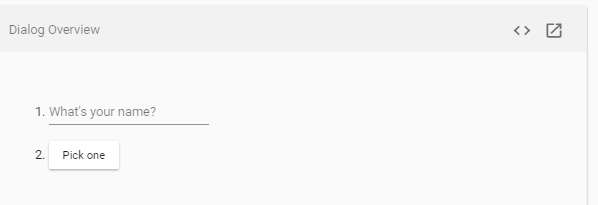
### Indeterminate🡺

### Buffer🡺

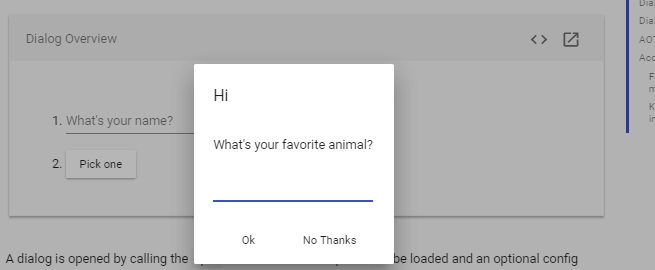
### Query🡺 is just moves in the opposite direction of Indeterminate

### POPUPS & MODALS

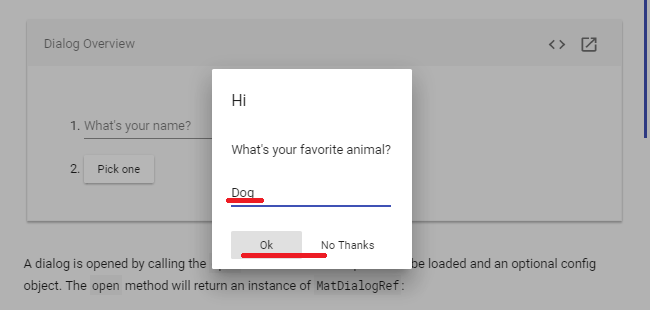
# **Dialog**

1. The **MatDialog** service can be used to open modal dialogs with Material Design styling and animations.
2. 

Now on click ok “Pick One” button, we get the following dialog box

 🡺 see here apart from the modal all other contents are disabled

Now Type “Dog” and click on Ok button as shown below



Now On click of ok we get the following output

### 

### Code 🡺

### @Inject 🡺 needed to know the purpose of This Annotation

### Step 1🡺

### Step 2🡺 Here let’s see the content of the DialogOverviewExample Component🡺 needed more explanation on why Subscribe() method is used here

### And for what is the “data” object used for here

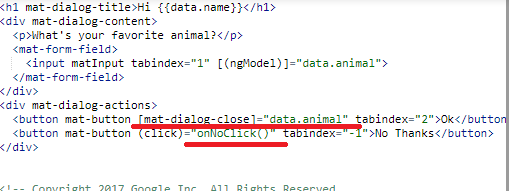
### 

**Step 3🡺 Here let’s see the content of the** DialogOverviewExampleDialog

Doubts🡺 here constructor part and onNoClick() method is not clear so I need more info on this

### 

**Step 4🡺now the content of the “**dialog-overview-example-dialog.html”



Here needed more information on the purpose of “mat-dialog-close” 🡺 note on closing the dialop box what is the further action needed to be taken is explained by

“dialogRef.afterClosed().subscribe(-----)”

### Note 🡺 for detailed explanation we can come back to this documentation again

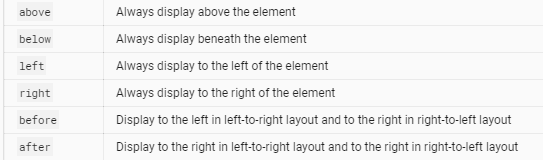
# **Tooltip**

### The Angular Material tooltip provides a text label that is displayed when the user hovers over or long presses an element.

### Code🡺

### Output🡺

1. **Positioning🡺**The tooltip will be displayed below the element but this can be configured using the **matTooltipPosition** input. The tooltip can be displayed above, below, left, or right of the element. By default the position will be below.

****

### Showing and hiding🡺 The tooltip is immediately shown when the user's mouse hovers over the element and immediately hides when the user's mouse leaves. A delay in showing or hiding the tooltip can be added through the inputs matTooltipShowDelay and matTooltipHideDelay.

# **Snackbar**

### MatSnackBar is a service for displaying snack-bar notifications.

### Lets see an example on Snackbar 🡺

### 

### Now let’s click on “Show snack-bar” and we get the snack bar as shown below🡺

### 

### Code🡺first let’s see the html content

### 

### Now let’s see the component file contents

### 

### Needed more information on the Syntax of the open()

# **Table**

### The mat-table provides a Material Design styled data-table that can be used to display rows of data.

### This table builds on the foundation of the CDK data-table and uses a similar interface for its data source input and template, except that its element and attribute selectors will be prefixed with mat- instead of cdk-.

### RXJS Library

# [**ReactiveX**](https://github.com/ReactiveX)**/**[rxjs](https://github.com/ReactiveX/rxjs)**🡺**A **reactive programming library** for JavaScript

#### Operators🡺[RxJS 5 operators](https://www.learnrxjs.io/operators/)

### A complete list of RxJS 5 operators,

### [Combination](https://www.learnrxjs.io/operators/combination/)

### [Conditional](https://www.learnrxjs.io/operators/conditional/)

### [Creation](https://www.learnrxjs.io/operators/creation/)

### [Error Handling](https://www.learnrxjs.io/operators/error_handling/)

### [Filtering](https://www.learnrxjs.io/operators/filtering/)

### [Multicasting](https://www.learnrxjs.io/operators/multicasting/)

### [Transformation](https://www.learnrxjs.io/operators/transformation/)

### [Utility](https://www.learnrxjs.io/operators/utility/)

### Now let’s see the operators in detail

### [Combination](https://www.learnrxjs.io/operators/combination/)

### The combination operators allow the joining of information from multiple observables.

### Order, time, and structure of emitted values is the primary variation among these operators.

### The List of Combination Operators are

* [combineAll](https://www.learnrxjs.io/operators/combination/combineall.html)
* [**combineLatest**](https://www.learnrxjs.io/operators/combination/combinelatest.html)
* [**concat**](https://www.learnrxjs.io/operators/combination/concat.html)
* [concatAll](https://www.learnrxjs.io/operators/combination/concatall.html)
* [forkJoin](https://www.learnrxjs.io/operators/combination/forkjoin.html)
* [**merge**](https://www.learnrxjs.io/operators/combination/merge.html)
* [mergeAll](https://www.learnrxjs.io/operators/combination/mergeall.html)
* [pairwise](https://www.learnrxjs.io/operators/combination/pairwise.html)
* [race](https://www.learnrxjs.io/operators/combination/race.html)
* [**startWith**](https://www.learnrxjs.io/operators/combination/startwith.html)
* [**withLatestFrom**](https://www.learnrxjs.io/operators/combination/withlatestfrom.html)
* [zip](https://www.learnrxjs.io/operators/combination/zip.html)

### combineAll 🡺

### Takes an Observable of Observables, and collects all Observables from it. Once the outer Observable completes, it subscribes to all collected Observables and combines their values using the [combineLatest](http://reactivex.io/rxjs/class/es6/Observable.js~Observable.html#static-method-combineLatest) strategy, such that:

### Syntax🡺public combineAll(project: [function](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function)): [Observable](http://reactivex.io/rxjs/class/es6/Observable.js~Observable.html)

### 

### Example 🡺 this observable is making use of other observables so Later we will see the example

### combineLatesr 🡺

### Combines multiple Observables to create an Observable whose values are calculated from the latest values of each of its input Observables.

### 

### Syntax Explanation🡺

### Example🡺 here “of” operator has used so needed to know this purpose first

### Concat🡺Creates an output Observable which sequentially emits all values from every given input Observable after the current Observable.

### 

### Syntax🡺

### startWith🡺Returns an Observable that emits the items you specify as arguments before it begins to emit items emitted by the source Observable.

### debounceTime🡺Emits a value from the source Observable only after a particular time span has passed without another source emission.

### Debounce🡺

### distinctUntilChanged🡺Only emit when the current value is different than the last.

### Filter🡺Emit values that pass the provided condition.

### Throw🡺Creates an Observable that emits no items to the Observer and immediately emits an error notification.

### Catch🡺Gracefully handle errors in an observable sequence.

### Retry🡺Returns an Observable that mirrors the source Observable with the exception of an error. If the source Observable calls error, this method will resubscribe to the source Observable for a maximum of count resubscriptions (given as a number parameter) rather than propagating the error call. 🡺

### retryWhen🡺Retry an observable sequence on error based on custom criteria 🡺

### fromPromise🡺Create observable from promise, emitting result.

### toPromise 🡺 Convert observable to promise🡺

### 

### Map🡺You use map to transform a collection of items into a collection of different items

### mapTo🡺

### switch🡺 MS🡺Observe following diagram carefully this is what Switch Operator of RXJS does🡺  ms that as soon as the next trigger emits, the current async operation is canceled and retriggered.

### SwitchMap🡺

### concatMap🡺

### flatMap🡺