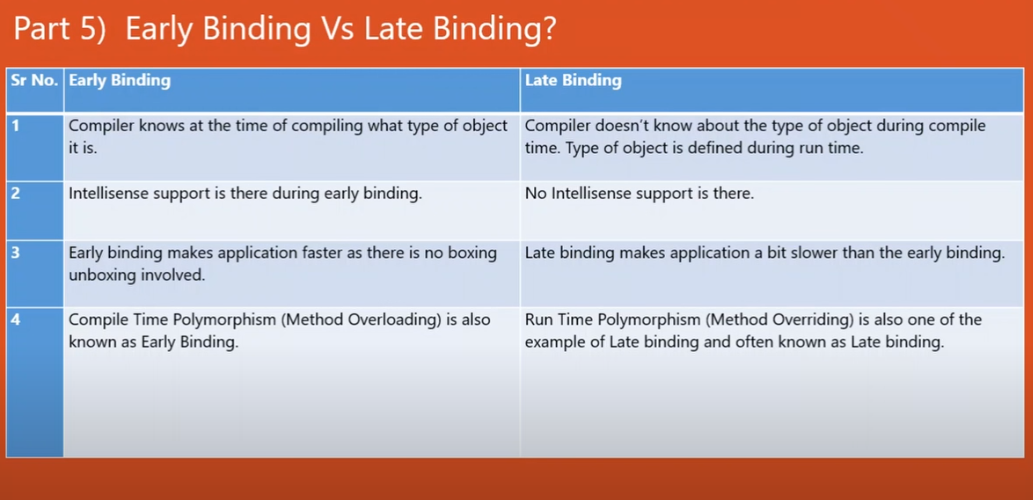
**===EXFO**

# [How late binding in c# is different from compile time?](https://stackoverflow.com/questions/7136961/how-to-keep-a-default-page-with-forms-authentication)

Late Binding is just an opp. of early binding. In Late binding functions, methods, variables and properties are detected and checked during the run-time.

The most popular example for late binding is the use of dynamic keyword and use of reflection.

Dynamic was introduced in C# 4.0 and introduced to achieve dynamically static object which pass the compile-time detection but on runtime it detects the object actually type depending on right-hand object-type and convert it to strong-type.



## Referential Integrity

A **REFERENTIAL INTEGRITY** is a database concept that is used to build and maintain logical relationships between tables to avoid logical corruption of data. It is a very useful and important part in **RDBMS**.

Usually, referential integrity is made up of the combination of a **primary key** and a **foreign key**.

The main concept of **REFERENTIAL INTEGRITY** is that it does not allow to add any record in a table that contains the **foreign key** unless the reference table containing a corresponding **primary key**.

If any record in referenced table (i.e. the table who contain primary key) is deleted, all the corresponding records in the referencing table will be deleted for the referential integrity.

**Are Hashtables Serializable?-**Throws NotSupportedException: The type System.Collections.Hashtable is not supported because it implements IDictionary.

**Short example of situation, when you need cherry pick**

Consider following scenario. You have two branches.

a) **release1** - This branch is going to your customer, but there are still some bugs to be fixed.

b) **master** - Classic master branch, where you can for example add functionality for release2.

**NOW**: You fix something in **release1**. Of course you need this fix also in **master**. And that is a typical use-case for cherry picking. So cherry pick in this scenario means that you take a commit from **release1** branch and include it into the **master** branch.

## Merge vs Rebase

* Merge takes all the changes in one branch and merges them into another branch in one commit.
* Rebase says I want the point at which I branched to move to a new starting point

So when do you use either one?

### Merge

* Let's say you have created a branch for the purpose of developing a single feature. When you want to bring those changes back to master, you probably want **merge** (you don't care about maintaining all of the interim commits).

### Rebase

* A second scenario would be if you started doing some development and then another developer made an unrelated change. You probably want to pull and then **rebase** to base your changes from the current version from the repository.

### Inner Exception in c#

The InnerException is a property of an exception. When there are series of exceptions, the most current exception can obtain the prior exception in the InnerException property.

**What is cache busting in angular 6**

To prevent visitors from being served with an older, cached version of the file, which might lack some translation, you can employ a cache-busting functionality. This entails attaching a unique version identifier to the file, which ensures that a fresh version is fetched, instead of a stale, cached version.

**===Icertis**

**Role Based authentication in MVC**

<https://www.c-sharpcorner.com/article/role-based-authentication-in-asp-net-mvc/>

**Role Based authentication in Asp.net Core**

<https://www.c-sharpcorner.com/article/policy-based-role-based-authorization-in-asp-net-core/>

**Input validation in API.**

<https://www.c-sharpcorner.com/article/custom-model-validation-in-asp-net-core-3-1/>

<https://www.carlrippon.com/fluentvalidation-in-an-asp-net-core-web-api/>

**Exception handling in API.**

<https://www.talkingdotnet.com/aspnet-core-diagnostics-middleware-error-handling/>

**What basic things we have to do to trace production error.**

Kibana Elastic search & logstash.

# Effective Paging, Sorting And Filtering Using SQL Server Stored Procedure

<https://www.c-sharpcorner.com/article/effective-paging-sorting-and-filtering-using-sql-server-stored-procedure/>

There are a number of articles and blogs where you can find about to do Paging, Sorting and Filtering with Stored Procedure in MS SQL Server. Hence, I started and done some research on this to find out the best solution. I found it in “[Pagination with OFFSET / FETCH : A better way](https://sqlperformance.com/2015/01/t-sql-queries/pagination-with-offset-fetch)”

 Using OFFSET / FETCH into the CTE I have created a stored procedure that was at least faster twice in return time as the alternatives found on the internet.

**Create stored procedure**

1. --GetAllEmployeesWay4 '', '', 1, 25, 'Name', 'Asc'
2. --GetAllEmployeesWay4 'Name', '1', 1, 25, 'Name', 'Asc'
3. --GetAllEmployeesWay4 'City', '1', 1, 25, 'Name', 'Asc'
4. **CREATE** **PROCEDURE** [dbo].[GetAllEmployeesWay4]
5. (
6. @SearchColumn NVARCHAR(50) = NULL,
7. @SearchValue NVARCHAR(50) = NULL,
8. @PageNo **INT** = 1,
9. @PageSize **INT** = 10,
10. @SortColumn NVARCHAR(20) = 'Name',
11. @SortOrder NVARCHAR(20) = 'ASC'
12. )
13. **AS** **BEGIN**
14. **SET** NOCOUNT **ON**;
16. **SET** @SearchColumn = LTRIM(RTRIM(@SearchColumn))
17. **SET** @SearchValue = LTRIM(RTRIM(@SearchValue))
19. ; **WITH** CTE\_Results **AS**
20. (
21. **SELECT** Id, **Name**, City **from** Employee
23. **WHERE** @SearchColumn= '' OR  (
24. CASE @SearchColumn
25. **WHEN** 'Name' **THEN** **Name**
26. **WHEN** 'City' **THEN** City
27. **END**
28. ) LIKE '%' + @SearchValue + '%'
30. **ORDER** **BY**
31. CASE **WHEN** (@SortColumn = 'Name' AND @SortOrder='ASC')
32. **THEN** **Name**
33. **END** **ASC**,
34. CASE **WHEN** (@SortColumn = 'Name' AND @SortOrder='DESC')
35. **THEN** **Name**
36. **END** **DESC**,
37. CASE **WHEN** (@SortColumn = 'City' AND @SortOrder='ASC')
38. **THEN** City
39. **END** **ASC**,
40. CASE **WHEN** (@SortColumn = 'City' AND @SortOrder='DESC')
41. **THEN** City
42. **END** **DESC**
43. OFFSET @PageSize \* (@PageNo - 1) **ROWS**
44. **FETCH** **NEXT** @PageSize **ROWS** **ONLY**
45. ),
46. CTE\_TotalRows **AS**
47. (
48. **select** count(ID) **as** TotalRows **from** Employee
49. **WHERE** @SearchColumn= '' OR  (
50. CASE @SearchColumn
51. **WHEN** 'Name' **THEN** **Name**
52. **WHEN** 'City' **THEN** City
53. **END**
54. ) LIKE '%' + @SearchValue + '%'
55. )
56. **Select** TotalRows, t.Id, t.**Name**, t.City **from** dbo.Employee **as** t, CTE\_TotalRows
57. **WHERE** EXISTS (**SELECT** 1 **FROM** CTE\_Results **WHERE** CTE\_Results.ID = t.ID)
59. **OPTION** (RECOMPILE)
60. **END**

**===Principle**

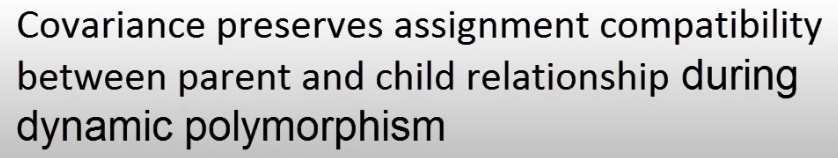
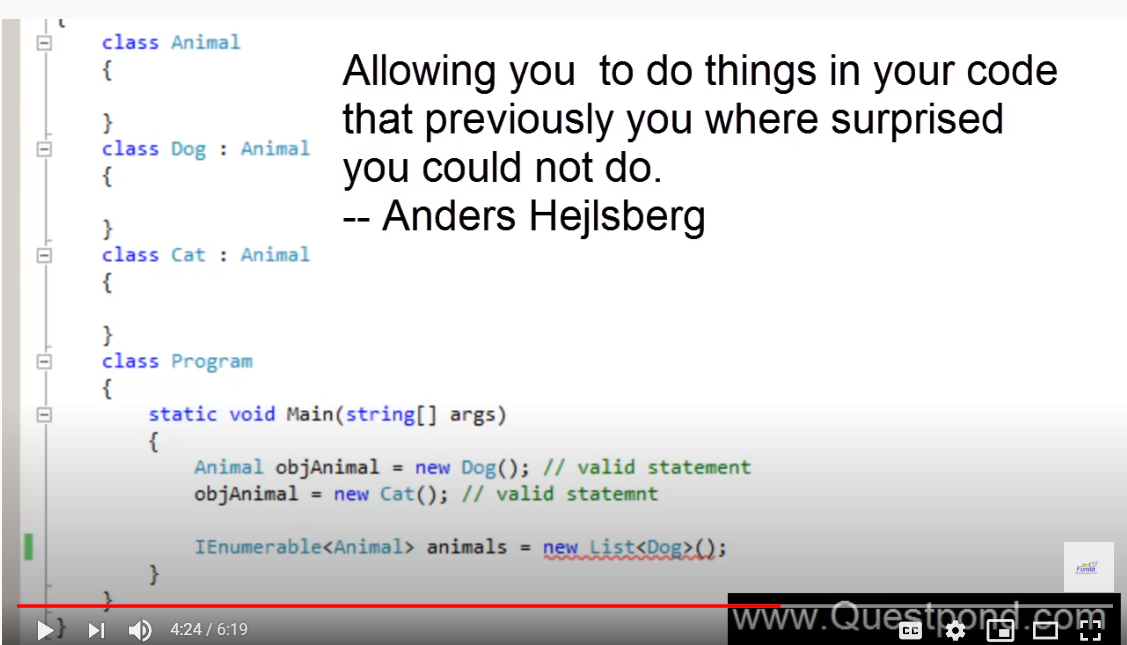
How to persist data in angular 6

You should be using localstorage/sessionstorage or shared service among components.

You can store the information(object/arrays) using JSON.stringify. If you are using shared service which becomes singleton so you can access across any components.

Make a singleton shared service in which you can save your data.

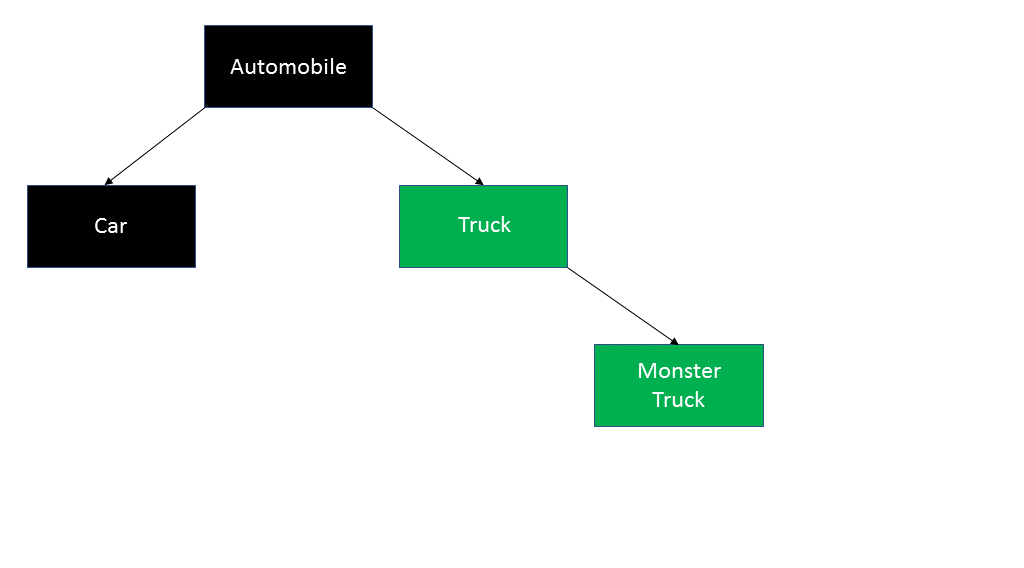
**covariance and contra-variance in c#**



Co-variance can be used on array, interface, delegates, etc in C#

Covariance and Contravariance are polymorphism extension to the arrays, delegates and generics. It provides implicit reference conversion for Arrays, Delegates and Generic parameter types. Covariance preserves the assignment compatibility and Contravariance opposite the covariance functionality.

* Covariance enables you to use a more derived type than originally specified.
* Contravariance enables you to use a less derived type than originally specified.



<https://codepureandsimple.com/covariance-and-contravariance-with-c-410fc4102a02>

Contra variance is for parameters. A method with the parameter of a base class is allowed to be assigned to a delegate that expects the parameter of a derived class with Contravariance.

Covariance and contravariance are features added with C# 4.0 that provides polymorphic extension to delegates, arrays and even generics.

**Map vs foreach javascript performance**

They are not one and the same. Let me explain the difference.

forEach: This iterates over a list and applies some operation with side effects to each list member (example: saving every list item to the database)

map: This iterates over a list, transforms each member of that list, and returns another list of the same size with the transformed members (example: transforming list of strings to uppercase)

[**forEach()**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach) — executes a provided function once for each array element.

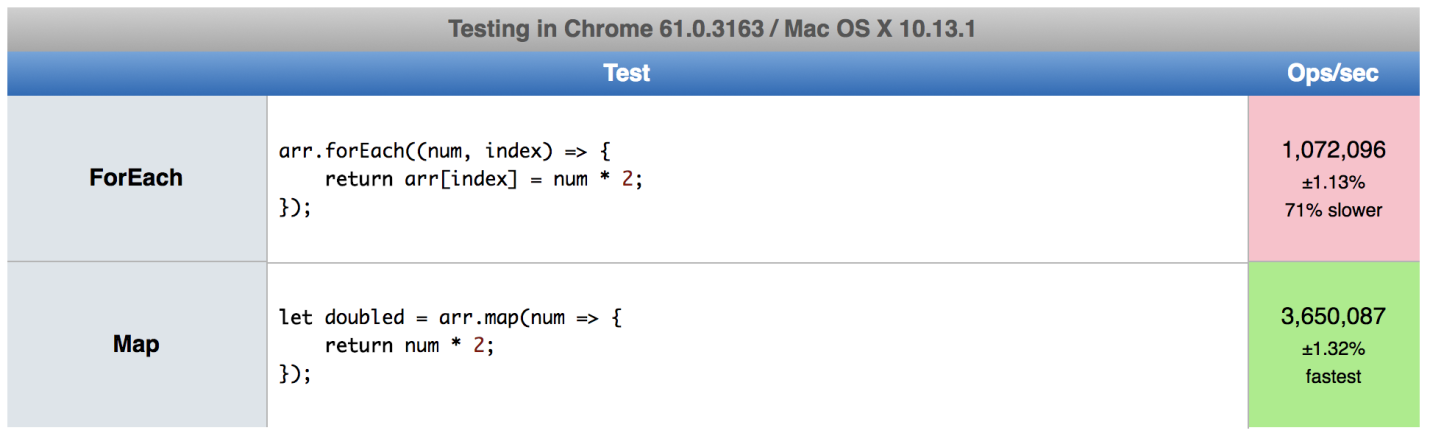
[**map()**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map) — creates a new array with the results of calling a provided function on every element in the calling array.

What exactly does this mean?

Well, the forEach() method doesn’t actually return anything (undefined). It simply calls a provided function on each element in your array. This callback is allowed to mutate the calling array.

Meanwhile, the map() method will also call a provided function on every element in the array. The difference is that map() utilizes return values and actually returns a new Array of the same size.

## Speed Considerations

As you can see, on my machine forEach() was more than 70% slower than map().

This is because forEach() affects and changes our original Array, whereas map() returns an entirely new Array — thus leaving the original array unchanged.

## Which is better?

That depends on what you’re trying to accomplish.

forEach() may be preferable when you’re not trying to change the data in your array, but instead want to just**do something** with it — like saving it to a database or logging it out:

And map() might be preferable when changing or altering data. Not only is it faster but it returns a new Array. This means we can do cool things like chaining on other methods ( map(), filter(), reduce(), etc.)

# PathLocationStrategy vs HashLocationStrategy

The main difference is whether Server is easy to implement the redirect mechanism

useHash: true is easy to implement than useHash: false

# Principle

when you set useHash: false, it's using **PathLocationStrategy**, It's using HTML5 pushstate that requires server support

When you enter the Url to Browser

localhost:4200/my-base/welcome/

The server needs to redirect **localhost:4200/my-base/welcome/** to your index.html

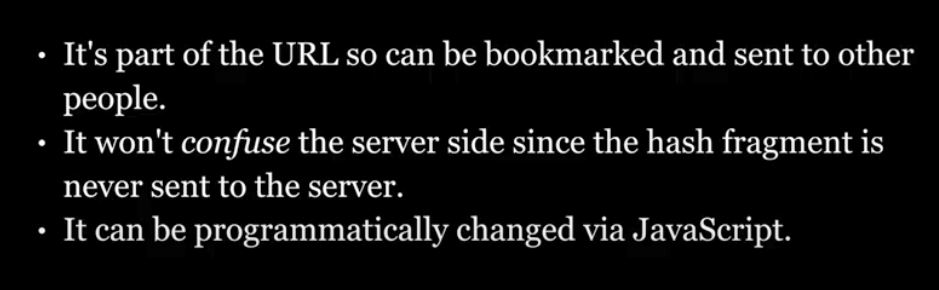
useHash: true, it's using **HashLocationStrategy**

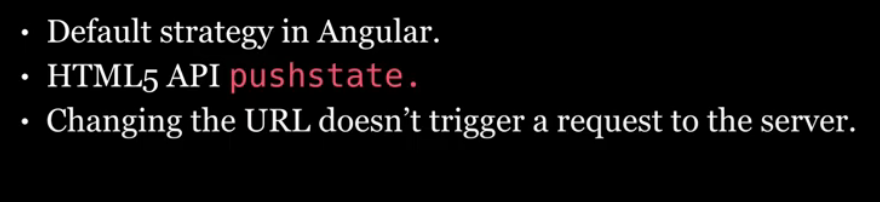
you need to add # after your base-href('my-base'), the URL is

localhost:4200/my-base/#/welcome/

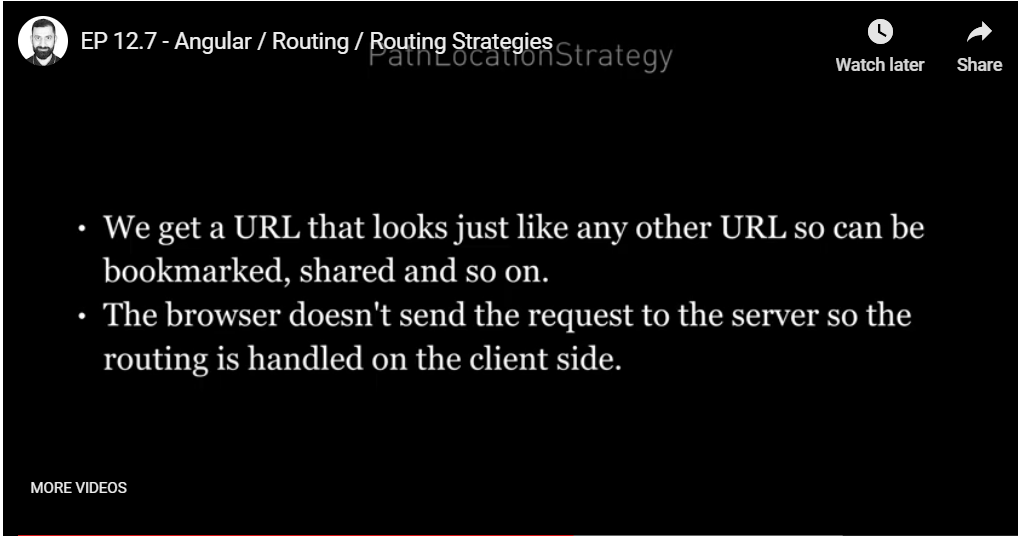
The server directly makes a request to **localhost:4200/my-base/** to your index.html, It's easy to implement in server side.

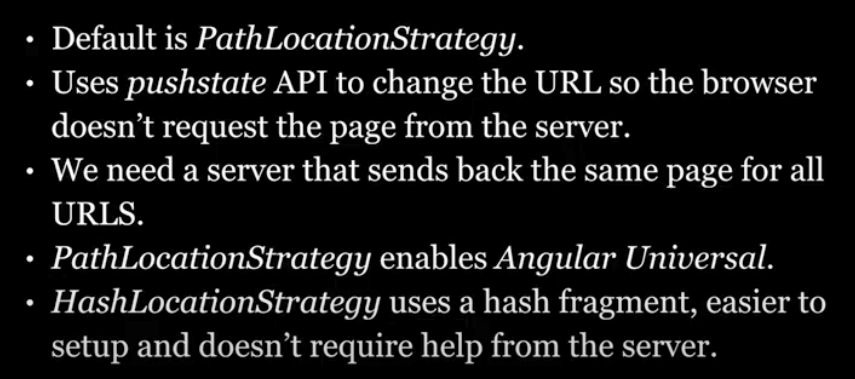
**HashLocation Strategy**





**Path Location Strategy**





**Difference between is and as keyword**

### is

The [is operator](https://msdn.microsoft.com/en-us/library/scekt9xw.aspx) checks if an object can be cast to a specific type.

Example:

if (someObject is StringBuilder) ...

### as

The [as operator](https://msdn.microsoft.com/en-us/library/cscsdfbt.aspx) attempts to cast an object to a specific type, and returns null if it fails.

Example:

StringBuilder b = someObject as StringBuilder;

if (b != null) ...

Also related:

### Casting

The [cast operator](https://msdn.microsoft.com/en-us/library/ms173105.aspx) attempts to cast an object to a specific type, and throws an exeption if it fails.

Example:

StringBuilder b = (StringBuilder)someObject.

### The Difference between IS and As is that..

**IS** - Is Operator is used to Check the Compatibility of an Object with a given Type and it returns the result as a Boolean (True Or False).

The is operator checks whether an object is compatible with a given type, and the result of the evaluation is a Boolean: true or false.

The as operator will never throw an exception.

**AS** - As Operator is used for Casting of Object to a given Type or a Class.

Ex.

Student s = obj as Student;

is equivalent to:

Student s = obj is Student ? (Student)obj : (Student)null;

**Must Go link**

[**https://www.csharpstar.com/is-vs-as-operators-in-csharp/**](https://www.csharpstar.com/is-vs-as-operators-in-csharp/)