**Diff B/W abstract class and interface**

At the top level, the are a few basic difference. Abstract classes allow for default default function definition.

This means that whatever class extends the abstract class will have access to this.

If we have a base class where all the classes will perform the same function, then we can define that in our Abstract class.

An interface is a list of functions or properties that if a class implements it, it will have to have those functions defined within it.

It is a situation of “Is-A” vs “Can-Do-this”. Objects that extends an Abstract class “Is-A” base class. Objects that implement “Can-Do-This”.

**1.** Abstract classes can have implementations for some of its members, but the interface can't have implementation for any of its members.  
  
**2.** Interfaces cannot have fields where as an abstract class can have fields.  
  
**3.** An interface can inherit from another interface only and cannot inherit from an abstract class, where as an abstract class can inherit from another abstract class or another interface.  
  
**4.** A class can inherit from multiple interfaces at the same time, where as a class cannot inherit from multiple classes at the same time.  
  
**5.** Abstract class members can have access modifiers where as interface members cannot have access modifiers.

6 Abstract class methods may OR may not have an implementation, while methods in an Interface only have a definition, no implementation.

**Abstract method**: When a class contains an abstract method, that class must be declared as abstract.

The abstract method has no implementation and thus, classes that derive from that abstract class,

must provide an implementation for this abstract method.

**Virtual method:** A class can have a virtual method. The virtual method has an implementation.

When you inherit from a class that has a virtual method, you can override the virtual method and provide additional logic,

or replace the logic with your own implementation.

When to use what:

**When to prefer an Abstract class**

**Another common C# Interview Question, that is commonly asked is, When do you choose interface over an abstract class or vice versa?**  
A general rule of thumb is, If you have an implementation that will be the same for all the [derived classes](http://venkatcsharpinterview.blogspot.com/2011/05/what-are-difference-between-interfaces.html), then it is better to go for an abstract class instead of an interface. So, when you have an interface, you can move your implementation to any class that implements the interface. Where as, when you have an abstract class, you can share implementation for all derived classes in one central place, and avoid code duplication in derived classes.

Abstract classes allow you to provide default functionality for the subclasses. Common knowledge at this point.

Why is this extremely important though?

If you plan on updating this base class throughout the life of your program,it is best to allow that base class to be an abstract class. Why? Because you can make a change to it and all of the inheriting classes will now have this new functionality. If the base class will be changing often and an interface was used instead of an abstract class,we are going to run into problems. Once an interface is changed, any class that implements that will be broken. Now if its just you working on the project,that’s no big deal. However, once your interface is published to the client, that interface needs to be locked down. At that point, you will be breaking the clients code.

Speaking from personal experiences, frameworks is a good place to show when and where to use both an abstract class and an interface.

Another general rule is if you are creating something that provides common functionality to unrelated classes, use an interface.

If you are creating something for objects that are closely related in a hierarchy, use an abstract class.

In some cases, you know that certain types should have a specific method, but,you don't know what implementation this method should have.

In such cases, you can create an interface which contains a method with this signature. However,

if you have such a case, but you know that implementors of that interface will also have another

common method (for which you can already provide the implementation), you can create an abstract class.

This abstract class then contains the abstract method (which must be overriden), and another method which

contains the 'common' logic.

A virtual method should be used if you have a class which can be used directly, but for which you want inheritors to be able to change certain behaviour, although it is not mandatory.

**What’s an abstract class?**

A class that cannot be instantiated. A concept in C++ known as pure virtual method.

A class that must be inherited and have the methods over-ridden. Essentially,

it’s a blueprint for a class without any implementation.

**What’s an interface class?**

It’s an abstract class with public abstract methods all of which must be implemented in the inherited classes.

What’s the difference between System.String and System.StringBuilder classes?

System.String is immutable, System.StringBuilder was designed with the purpose of having a mutable string

where a variety of operations can be performed.

**Diff B/W abstract class and static class**

An abstract class is intended to be used as a base of a class inheritance hierarchy. A static class cannot be the base of a class inheritance hierarchy.

A static class is intended for singleton state or stateless functionality. An abstract class is not suitable for singleton functionality, because, even though it may contain static methods and fields as a static class does, it cannot forbid inheritance, so the singleton use may be defeated by subclasses.

Abstract classes are intended to be used as base classes; they cannot have direct instances. Instead, you have to derive subclasses, which provide the what was (usually intentionally) left out in the abstract base class

Static classes cannot be instantiated, and any state is at the class level rather than the instance level. They are typically used to define utility methods where there is no state associated with the methods.

**Reversing a string through array**

public static string **ReverseString**(string s)

{ char[] arr = s.ToCharArray();

Array.Reverse(arr);

return new string(arr);

}

**Comparing of two arrays**

if (first.Length <= second.Length)  
{  
for (int x = 0; x < first.Length; x++)  
{  
  if (first[x] != second[x]) return false;  
}  
return true;  
}  
return false;

**Please write a sample program that parses the string into a series of substrings where the delimiter between the substrings is "^\*!%~" and then reassembles the strings and delimiters into a single new string where each of the substrings is in the reverse order from the original string. The method must return the final string.**  
  
**Original String:**Token A^\*!%~Token B^\*!%~Token C^\*!%~Token D^\*!%~Token E  
  
**Output String:**   
Token E^\*!%~Token D^\*!%~Token C^\*!%~Token B^\*!%~Token A  
  
**The code sample below shows how to solve the above question:**using System;  
using [System](http://venkataspinterview.blogspot.com/2008/10/practical-real-time-aspnet-interview.html).Text;  
namespace GenericsSample  
{  
class Program  
{  
static void Main()  
{  
string strOriginalString = "Token A^\*!%~Token B^\*!%~Token C^\*!%~Token D^\*!%~Token E";  
string[] strSeperator = new string[1];  
strSeperator[0] = "^\*!%~";  
  
string[] strArrayIndividualStrings = strOriginalString.Split(strSeperator, StringSplitOptions.RemoveEmptyEntries);  
  
int intLengthOfStringArray = strArrayIndividualStrings.Length;  
  
StringBuilder sbOutputString = new StringBuilder();  
for (int i = (intLengthOfStringArray - 1); i >= 0; i--)  
{  
sbOutputString.Append(strArrayIndividualStrings[i] + strSeperator[0]);  
}  
Console.WriteLine("Original String : " + strOriginalString);  
Console.WriteLine("Output String : " + sbOutputString.ToString());  
Console.ReadLine();  
}  
}  
}

**Counting words,characters and lines from a textfile.**

StreamReader sr = new StreamReader(Environment.CurrentDirectory + "\\Bhagawat.txt");

string line = null;

do

{

line = sr.ReadLine();

if ((line != null))

{

MessageBox.show(line); //Read line

foreach (char cnt <strong class="highlight">in</strong> line)

{

MessageBox.Show(cnt.ToString()); //Read characters

}

}

} while (!(line == null));

sr.Close();

**More C# interview questions on strings**

**Will the following code compile and run?**  
string str = null;  
Console.WriteLine(str.Length);  
The above code will compile, but at runtime System.NullReferenceException will be thrown.  
  
**How do you create empty strings in C#?**  
Using string.empty as shown in the example below.  
string EmptyString = string.empty;  
  
**What is the difference between System.Text.StringBuilder and System.String?  
1.** Objects of type StringBuilder are mutable where as objects of type System.String are immutable.   
**2.** As StringBuilder objects are mutable, they offer better performance than string objects of type System.String.  
**3.** StringBuilder class is present in System.Text namespace where String class is present in System namespace.

**How do you determine whether a String represents a numeric value?**  
To determine whether a String represents a numeric value use TryParse method as shown in the example below. If the string contains nonnumeric characters or the numeric value is too large or too small for the particular type you have specified, TryParse returns false and sets the out parameter to zero. Otherwise, it returns true and sets the out parameter to the numeric value of the string.  
  
string str = "One";  
int i = 0;  
if(int.TryParse(str,out i))  
{  
     Console.WriteLine("Yes string contains Integer and it is " + i);  
}  
else  
{  
     Console.WriteLine("string does not contain Integer");  
}  
 **What is the difference between int.Parse and int.TryParse methods?**  
Parse method throws an exception if the string you are trying to parse is not a valid number where as TryParse returns false and does not throw an exception if parsing fails. Hence TryParse is more efficient than Parse.

### C# Interview Questions on partial classes, structs and methods.

**What is a partial class. Give an example?**  
A **partial class** is a class whose definition is present in 2 or more files. Each source file contains a section of the class, and all parts are combined when the [application](http://venkatcsharpinterview.blogspot.com/2009/07/c-interview-questions-on-partial.html) is compiled. To split a class definition, use the partial keyword as shown in the example below. Student class is split into 2 parts. The first part defines the study() method and the second part defines the Play() method. When we compile this program both the parts will be combined and compiled. Note that both the parts uses partial keyword and public access modifier.  
  
using System;  
namespace Partial Class  
{  
  public partial class Student  
  {  
    public void Study()  
    {  
      Console.WriteLine("I am studying");  
    }  
  }  
  public partial class Student  
  {  
    public void Play()  
    {  
      Console.WriteLine("I am Playing");  
    }  
  }  
  public class Demo  
  {  
    public static void Main()  
    {  
      Student StudentObject = new Student();  
      StudentObject.Study();  
      StudentObject.Play();  
    }  
  }  
}  
  
**It is very important to keep the following points in mind when creating partial classes.**  
**1.** All the parts must use the partial keyword.  
**2.** All the parts must be available at compile time to form the final class.  
**3.** All the parts must have the same access modifiers - public, private, protected etc.  
**4.** Any class members declared in a partial definition are available to all the other parts.   
**5.** The final class is the combination of all the parts at compile time.  
  
**What are the advantages of using partial classes?**  
**1.** When working on large projects, spreading a class over separate files enables multiple programmers to[work](http://venkatcsharpinterview.blogspot.com/2009/07/c-interview-questions-on-partial.html) on it at the same time.  
  
**2.** When working with automatically generated source, code can be added to the class without having to recreate the source file. Visual Studio uses this approach when it creates Windows Forms, Web service wrapper code, and so on. You can create code that uses these classes without having to modify the file created by Visual Studio.  
  
**Is it possible to create partial structs, interfaces and methods?**  
Yes, it is possible to create partial structs, interfaces and methods. We can create partial structs, interfaces and methods the same way as we create partial classes.

**Can you create partial delegates and enumerations?**  
No, you cannot create partial delegates and enumerations.  
  
**Can different parts of a partial class inherit from different interfaces?**  
Yes, different parts of a partial class can inherit from different interfaces.   
  
**Can you specify nested**[**classes as**](http://venkatcsharpinterview.blogspot.com/2009/07/c-interview-questions-on-partial.html)**partial classes?**  
Yes, nested classes can be specified as partial classes even if the containing class is not partial. An example is shown below.  
  
class ContainerClass  
{  
  public partial class Nested  
  {  
    void Test1() { }  
  }  
  public partial class Nested  
  {  
    void Test2() { }  
  }  
}

**How do you create partial methods?**  
To create a **partial method** we create the declaration of the method in one part of the partial class and implementation in the other part of the partial class. The implementation is optional. If the implementation is not provided, then the method and all the calls to the method are removed at compile time. Therefore, any code in the partial class can freely use a partial method, even if the implementation is not supplied. No compile-time or run-time errors will result if the method is called but not implemented. In summary a partial method declaration consists of two parts. The definition, and the implementation. These may be in separate parts of a partial class, or in the same part. If there is no implementation declaration, then the compiler optimizes away both the defining declaration and all calls to the method.  
  
**The following are the points to keep in mind when creating partial methods.**  
**1.** Partial method declarations must begin partial keyword.  
**2.** The return type of a partial method must be void.  
**3.** Partial methods can have ref but not out parameters.  
**4.** Partial methods are implicitly private, and therefore they cannot be virtual.  
**5.** Partial methods cannot be extern, because the presence of the body determines whether they are defining or implementing.  
  
**What is the use of partial methods?**  
**Partial methods** can be used to customize generated code. They allow for a method name and signature to be reserved, so that generated code can call the method but the developer can decide whether to implement the method. Much like partial classes, partial methods enable code created by a code generator and code created by a human developer to work together without run-time costs.

### C# Interview Questions related to Interfaces.

**Explain what is an Interface in C#?**  
An Interface in C# is created using the interface keyword. An example is shown below.  
  
using System;  
namespace Interfaces  
{  
interface IBankCustomer  
{  
void DepositMoney();  
void WithdrawMoney();  
}  
public class Demo : IBankCustomer  
{  
public void DepositMoney()  
{  
Console.WriteLine("Deposit Money");  
}  
  
public void WithdrawMoney()  
{  
Console.WriteLine("Withdraw Money");  
}  
  
public static void Main()  
{  
Demo DemoObject = new Demo();  
DemoObject.DepositMoney();  
DemoObject.WithdrawMoney();  
}  
}  
}

**In our example we created IBankCustomer interface. The interface declares 2 methods.**  
**1.** void DepositMoney();  
**2.** void WithdrawMoney();  
  
Notice that method declarations does not have access modifiers like public, private, etc. By default all interface members are public. It is a compile time [error](http://venkatcsharpinterview.blogspot.com/2009/07/c-interview-questions-related-to.html) to use access modifiers on interface member declarations.Also notice that the interface methods have only declarations and not implementation. It is a compile time error to provide implementation for any interface member. In our example as the Demo class is inherited from the IBankCustomer interface, the Demo class has to provide the implementation for both the methods (WithdrawMoney() and DepositMoney()) that is inherited from the interface. If the class fails to provide implementation for any of the inherited interface member, a compile time error will be generated. Interfaces can consist of methods, properties, events, indexers, or any combination of those four member types. When a class or a struct inherits an interface, the class or struct must provide implementation for all of the members declared in the interface. The interface itself provides no functionality that a class or struct can inherit in the way that base class functionality can be inherited. However, if a base class implements an interface, the derived class inherits that implementation.  
  
**Can an Interface contain fields?**  
No, an Interface cannot contain fields.

**What is the difference between class inheritance and interface inheritance?**  
Classes and structs can inherit from interfaces just like how classes can inherit a base class or struct. However there are 2 differences.  
**1.** A class or a struct can inherit from more than one interface at the same time where as A class or a struct cannot inherit from more than one class at the same time. An example depicting the same is shown below.  
  
using System;  
namespace Interfaces  
{  
interface Interface1  
{  
void Interface1Method();  
}  
interface Interface2  
{  
void Interface2Method();  
}  
class BaseClass1  
{  
public void BaseClass1Method()  
{  
Console.WriteLine("BaseClass1 Method");  
}  
}  
class BaseClass2  
{  
public void BaseClass2Method()  
{  
Console.WriteLine("BaseClass2 Method");  
}  
}  
  
//Error : A class cannot inherit from more than one class at the same time  
//class DerivedClass : BaseClass1, BaseClass2  
//{  
//}  
  
//A class can inherit from more than one interface at the same time  
public class Demo : Interface1, Interface2  
{  
public void Interface1Method()  
{  
Console.WriteLine("Interface1 Method");  
}  
  
public void Interface2Method()  
{  
Console.WriteLine("Interface2 Method");  
}  
  
public static void Main()  
{  
Demo DemoObject = new Demo();  
DemoObject.Interface1Method();  
DemoObject.Interface2Method();  
}  
}  
}  
  
**2.** When a class or struct inherits an interface, it inherits only the method names and signatures, because the interface itself contains no implementations.  
  
**Can an interface inherit from another interface?**  
Yes, an interface can inherit from another interface. It is possible for a class to inherit an interface multiple times, through base classes or interfaces it inherits. In this case, the class can only implement the interface one time, if it is declared as part of the new class. If the inherited interface is not declared as part of the new class, its implementation is provided by the base class that declared it. It is possible for a base class to implement interface members using virtual members; in that case, the class inheriting the interface can change the interface behavior by overriding the virtual members.  
  
**Can you create an instance of an interface?**  
No, you cannot create an instance of an interface.  
  
**If a class inherits an interface, what are the 2 options available for that class?**  
**Option 1:** Provide Implementation for all the members inheirted from the interface.  
  
namespace Interfaces  
{  
interface Interface1  
{  
void Interface1Method();  
}  
  
class BaseClass1 : Interface1  
{  
public void Interface1Method()  
{  
Console.WriteLine("Interface1 Method");  
}  
public void BaseClass1Method()  
{  
Console.WriteLine("BaseClass1 Method");  
}  
}  
}  
  
**Option 2:** If the class does not wish to provide Implementation for all the members inheirted from the interface, then the class has to be marked as abstract.  
  
namespace Interfaces  
{  
interface Interface1  
{  
void Interface1Method();  
}  
  
abstract class BaseClass1 : Interface1  
{  
abstract public void Interface1Method();  
public void BaseClass1Method()  
{  
Console.WriteLine("BaseClass1 Method");  
}  
}  
}  
  
**A class inherits from 2 interfaces and both the interfaces have the same method name as shown below. How should the class implement the drive method for both Car and Bus interface?**  
namespace Interfaces  
{  
interface Car  
{  
void Drive();  
}  
interface Bus  
{  
void Drive();  
}  
  
class Demo : Car,Bus  
{  
//How to implement the Drive() Method inherited from Bus and Car  
}  
}  
  
To implement the Drive() method use the fully qualified name as shown in the example below. To call the respective interface drive method type cast the demo object to the respective interface and then call the drive method.  
  
using System;  
namespace Interfaces  
{  
interface Car  
{  
void Drive();  
}  
interface Bus  
{  
void Drive();  
}  
  
class Demo : Car,Bus  
{  
void Car.Drive()  
{  
Console.WriteLine("Drive Car");  
}  
void Bus.Drive()  
{  
Console.WriteLine("Drive Bus");  
}  
  
static void Main()  
{  
Demo DemoObject = new Demo();  
((Car)DemoObject).Drive();  
((Bus)DemoObject).Drive();  
}  
}  
}  
  
**What do you mean by "Explicitly Implemeting an Interface". Give an example?**  
If a class is [implementing](http://venkatcsharpinterview.blogspot.com/2009/07/c-interview-questions-related-to.html) the inherited interface member by prefixing the name of the interface, then the class is "Explicitly Implemeting an Interface member". The disadvantage of Explicitly Implemeting an Interface member is that, the class object has to be type casted to the interface type to invoke the interface member. An example is shown below.  
  
using System;  
namespace Interfaces  
{  
interface Car  
{  
void Drive();  
}  
  
class Demo : Car  
{  
// Explicit implementation of an interface member  
void Car.Drive()  
{  
Console.WriteLine("Drive Car");  
}  
  
static void Main()  
{  
Demo DemoObject = new Demo();  
  
//DemoObject.Drive();  
// Error: Cannot call explicitly implemented interface method  
// using the class object.  
// Type cast the demo object to interface type Car  
((Car)DemoObject).Drive();  
}  
}  
}

### C# Interview Questions on constructors

**What is a constructor in C#?**  
Constructor is a class method that is executed when an object of a class is created. Constructor has the same name as the class, and usually used to initialize the data members of the new object.   
  
**In C#, What will happen if you do not explicitly provide a constructor for a class?**  
If you do not provide a constructor explicitly for your class, C# will create one by default that instantiates the object and sets all the member variables to their [default values](http://venkatcsharpinterview.blogspot.com/2009/07/c-interview-questions-on-constructors.html).  
  
**Structs are not reference types. Can structs have constructors?**  
Yes, even though Structs are not reference types, structs can have constructors.  
  
**We cannot create instances of static classes. Can we have constructors for static classes?**  
Yes, static classes can also have constructors.

**Can you prevent a class from being instantiated?**  
Yes, a class can be prevented from being instantiated by using a private constructor as shown in the example below.  
  
using System;  
namespace TestConsole  
{  
  class Program  
  {  
    public static void Main()  
    {  
      //Error cannot create instance of a class with private constructor  
      SampleClass SC = new SampleClass();  
    }  
  }  
  class SampleClass  
  {  
    double PI = 3.141;  
    private SampleClass()  
    {  
    }  
  }  
}

**Can a class or a struct have multiple constructors?**  
Yes, a class or a struct can have multiple constructors. Constructors in csharp can be overloaded.

**If a child class instance is created, which class constructor is called first - base class or child class?**  
When an instance of a child class is created, the base class constructor is called before the child class constructor. An example is shown below.  
  
using System;  
namespace TestConsole  
{  
  class BaseClass  
  {  
    public BaseClass()  
    {  
      Console.WriteLine("I am a base class constructor");  
    }  
  }  
  class ChildClass : BaseClass  
  {  
    public ChildClass()  
    {  
      Console.WriteLine("I am a child class constructor");  
    }  
    public static void Main()  
    {  
      ChildClass CC = new ChildClass();  
    }  
  }  
}

**Can you mark static constructor with access modifiers?**  
No, we cannot use access modifiers on static constructor.  
  
**Can you have parameters for static constructors?**  
No, static constructors cannot have parameters.  
  
**What happens if a static constructor throws an exception?**  
If a static constructor throws an exception, the runtime will not invoke it a second time, and the type will remain uninitialized for the lifetime of the [application](http://venkatcsharpinterview.blogspot.com/2009/07/c-interview-questions-on-constructors.html) domain in which your program is running.  
  
**Give 2 scenarios where static constructors can be used?**  
**1.** **A typical use of static constructors is when the class is using a log file and the constructor is used to write entries to this file.**  
**2.** **Static constructors are also useful when creating wrapper classes for unmanaged code, when the constructor can call the LoadLibrary method.**  
  
**Does C# provide copy constructor?**  
No, C# does not provide copy constructor.

### C# Interview Questions on value types and reference types

**What are the 2 types of data types available in C#?  
1.** Value Types  
**2.** Reference Types  
  
**If you define a user defined data type by using the struct keyword, Is it a value type or reference type?**  
Value Type  
  
**If you define a user defined data type by using the class keyword, Is it a value type or reference type?**  
Reference type  
  
**Are Value types sealed?**Yes, Value types are sealed.  
  
**What is the base class from which all value types are derived?**  
System.ValueType

**Give examples for value types?**Enum  
Struct  
  
**Give examples for reference types?**  
Class  
Delegate  
Array  
Interface  
  
**What are the differences between value types and reference types?  
1.** Value types are stored on the stack where as reference types are stored on the managed heap.  
**2.** Value type variables directly contain their values where as reference variables holds only a reference to the location of the object that is created on the managed heap.  
**3.** There is no heap allocation or garbage collection overhead for value-type variables. As reference types are stored on the managed heap, they have the over head of object allocation and garbage collection.  
**4.**Value Types cannot inherit from another class or struct. Value types can only inherit from interfaces. Reference types can inherit from another class or interface.

### Basic C# Interview Questions on classes and structs

**What do you mean by saying a "class is a reference type"?**  
A class is a reference type means when an object of the class is created, the variable to which the object is assigned holds only a reference to that memory. When the object reference is assigned to a new variable, the new variable refers to the original object. Changes made through one variable are reflected in the other variable because they both refer to the same data.  
  
**What do you mean by saying a "struct is a value type"?**A struct is a value type mean when a struct is created, the variable to which the struct is assigned holds the struct's actual data. When the struct is assigned to a new variable, it is copied. The new variable and the original variable therefore contain two separate copies of the same data. Changes made to one copy do not affect the other copy.  
  
**When do you generally use a class over a struct?**  
A class is used to model more complex behavior, or data that is intended to be modified after a class object is created. A struct is best suited for small data structures that contain primarily data that is not intended to be modified after the struct is created.

**List the 5 different access modifiers in C#?  
1.** public  
**2.** protected  
**3.** internal  
**4.** protected internal  
**5.** private  
  
**If you donot specify an access modifier for a method, what is the default access modifier?**private  
  
**Classes and structs support inheritance. Is this statement true or false?**  
False, Only classes support inheritance. structs do not support inheritance.  
  
**If a class derives from another class, will the derived class automatically contain all the public, protected, and internal members of the base class?**  
Yes, the derived class will automatically contain all the public, protected, and internal members of the base class except its constructors and destructors.  
  
**Can you create an instance for an abstract class?**  
No, you cannot create an instance for an abstract class.  
  
**How do you prevent a class from being inherited by another class?**Use the sealed keyword to prevent a class from being inherited by another class.  
  
**Classes and structs can be declared as static, Is this statement true or false?**  
False, only classes can be declared as static and not structs.  
  
**Can you create an instance of a static class?**No, you cannot create an instance of a static class.  
  
**Can a static class contain non static members?**  
No, a static class can contain only static members.

### C# Interview Questions on structs

**Can a struct have a default constructor (a constructor without parameters) or a destructor in C#?**  
No  
  
**Can you instantiate a struct without using a new operator in C#?**Yes, you can instantiate a struct without using a new operator

**Can a struct inherit from another struct or class in C#?**  
No, a struct cannot inherit from another struct or class, and it cannot be the base of a class.  
  
**Can a struct inherit from an interface in C#?**Yes  
  
**Are structs value types or reference types?**  
Structs are value types.  
  
**What is the base type from which all structs inherit directly?**All structs inherit directly from System.ValueType, which inherits from System. Object.

### C# Interview Questions on Inheritance

**What are the 4 pillars of any object oriented programming language?  
1.** Abstraction  
**2.** Inheritance  
**3.** Encapsulation  
**4.** Polymorphism  
  
**Do structs support inheritance?**No, structs do not support inheritance, but they can implement interfaces.  
  
**What is the main advantage of using inheritance?**  
Code reuse

**Is the following code legal?**class ChildClass : ParentClassA, ParentClassB{}  
No, a child class can have only one base class. You cannot specify 2 base classes at the same time. C# supports single class inheritance only. Therefore, you can specify only one base class to inherit from. However, it does allow multiple interface inheritance.

### C# Interview Questions on Abstract and Sealed Class Members

**What is an abstract class?**  
An abstract class is an incomplete class and must be implemented in a derived class.  
  
**Can you create an instance of an abstract class?**No, abstract classes are incomplete and you cannot create an instance of an abstract class.  
  
**What is a sealed class?**  
A sealed class is a class that cannot be inherited from. This means, If you have a class called Customer that is marked as sealed. No other class can inherit from Customer class. For example, the below code generates a compile time [error](http://venkatcsharpinterview.blogspot.com/2009/02/c-interview-questions-on-abstract-and.html) "MainClass cannot derive from sealed type Customer.  
using System;public sealed class Customer{}public class MainClass : Customer{public static void Main(){}}  
**What are abstract methods?**  
Abstract methods are methods that only the declaration of the method and no implementation.

**Can a sealed class be used as a base class?**  
No, sealed class cannot be used as a base class. A compile time error will be generated.

### C# Interview Questions on polymorphism

**When can a derived class override a base class member?**  
A derived class can override a base class member only if the base class member is declared as virtual or abstract.  
  
**What is the difference between a virtual method and an abstract method?**  
A virtual method must have a body where as an abstract method should not have a body.

**Can fields inside a class be virtual ?**No, Fields inside a class cannot be virtual. Only methods, properties, events and indexers can be virtual.

### C# Interview Questions on data type casting

### **What do you mean by casting a data type?**  Converting a variable of one data type to another data type is called casting. This is also called as data type conversion. **What are the 2 kinds of data type conversions in C#?** **Implicit conversions:** No special syntax is required because the conversion is type safe and no data will be lost. Examples include conversions from smaller to larger integral types, and conversions from [derived classes](http://venkatcsharpinterview.blogspot.com/2009/01/c-interview-questions-on-data-type.html)to base classes. **Explicit conversions:** Explicit conversions require a cast operator. The source and destination variables are compatible, but there is a risk of data loss because the type of the destination variable is a smaller size than (or is a base class of) the source variable. **What is the difference between an implicit conversion and an explicit conversion?**  **1.** Explicit conversions require a cast operator where as an implicit converstion is done automatically. **2.** Explicit conversion can lead to data loss where as with implicit conversions there is no data loss. **What type of data type conversion happens when the compiler encounters the following code?** ChildClass CC = new ChildClass(); ParentClass [PC](http://venkatcsharpinterview.blogspot.com/2009/01/c-interview-questions-on-data-type.html) = new ParentClass(); Implicit Conversion. For reference types, an implicit conversion always exists from a class to any one of its direct or indirect base classes or interfaces. No special syntax is necessary because a derived class always contains all the members of a base class.

### **Will the following code compile?**  double d = 9999.11; int i = d; No, the above code will not compile. Double is a larger data type than integer. An implicit conversion is not done automatically bcos there is a data loss. Hence we have to use explicit conversion as shown below. double d = 9999.11; int i = (int)d; //Cast double to int. **If you want to convert a base type to a derived type, what type of conversion do you use?** Explicit conversion as shown below. //Create a new derived type. Car C1 = new Car(); // Implicit conversion to base type is safe. Vehicle V = C1; // Explicit conversion is required to cast back to derived type. The code below will compile but throw an exception at run time if the right-side object is not a Car object. Car C2 = (Car) V; **What operators can be used to cast from one reference type to another without the risk of throwing an exception?**  The is and as operators can be used to cast from one reference type to another without the risk of throwing an exception. **If casting fails what type of exception is thrown?** InvalidCastException

### C# Interview questions on Boxing and Unboxing

### **What is Boxing and Unboxing?**  [Boxing](http://venkatcsharpinterview.blogspot.com/2009/01/c-interview-questions-on-boxing-and.html) - Converting a value type to reference type is called boxing. An example is shown below. int i = 101; object obj = (object)i; // Boxing **Unboxing** - Converting a reference type to a value type is called unboxing. An example is shown below. obj = 101; i = (int)obj; // Unboxing

### **Is boxing an implicit conversion?** Yes, boxing happens implicitly. **Is unboxing an implicit conversion?**  No, unboxing is an explicit conversion. **What happens during the process of boxing?** Boxing is used to store value types in the garbage-collected heap. Boxing is an implicit conversion of a value type to the type object or to any interface type implemented by this value type. Boxing a value type allocates an object instance on the heap and copies the value into the new object. Due to this boxing and unboxing can have performance impact.

### Basic C# Interview Questions on arrays

### **What is an array?**  An array is a data structure that contains several variables of the same type. **What are the 3 different types of arrays?** **1.** Single-Dimensional **2.** Multidimensional **3.** Jagged

### **What is Jagged Array?**  A jagged array is an array of arrays. **Are arrays value types or reference types?** Arrays are reference types. **What is the base class for Array types?**  System.Array **Can you use foreach iteration on arrays in C#?** Yes,Since array type implements **IEnumerable**, you can use **foreach** iteration on all arrays in C#.

**C# Interview Questions on Fields**

**What are the 2 broad classifications of fields in C#?**  
**1.** Instance fields  
**2.** Static fields  
  
**What are instance fields in C#?**Instance fields are specific to an instance of a type. If you have a class T, with an instance field F, you can create two objects of type T, and modify the value of F in each object without affecting the value in the other object.  
  
**What is a static field?**   
A static field belongs to the class itself, and is shared among all instances of that class. Changes made from instance A will be visible immediately to instances B and C if they access the field.

**What is the difference between a constant and a static readonly field?**   
A static readonly field is very similar to a constant, except that the C# compiler does not have access to the value of a static read-only field at compile time, only at run time.

**C# Interview Questions on Access Modifiers**

**What are Access Modifiers used for?**Access Modifiers are used to control the accessibility of types and members with in the types.

**Can you use all access modifiers for all types?**   
No, Not all access modifiers can be used by all types or members in all contexts, and in some cases the accessibility of a type member is constrained by the accessibility of its containing type.  
  
**Can derived classes have greater accessibility than their base types?**No, Derived classes cannot have greater accessibility than their base types. For example the following code is illegal.

**Is the following code legal?**   
  
using System;  
private class Test  
{  
   public static void Main()  
   {  
   }  
}  
  
No, a compile time error will be generated stating **"Namespace elements cannot be explicitly declared as private, protected, or protected internal"**  
  
**Can you declare struct members as protected?**   
No, struct members cannot be declared protected. This is because structs do not support inheritance.  
  
**Can the accessibility of a type member be greater than the accessibility of its containing type?**No, the accessibility of a type member can never be greater than the accessibility of its containing type. For example, a public method declared in an internal class has only internal accessibility.  
  
**Can destructors have access modifiers?**   
No, destructors cannot have access modifiers.  
  
**What does protected internal access modifier mean?**The protected internal access means protected OR internal, not protected AND internal. In simple terms, a protected internal member is accessible from any class in the same assembly, including derived classes. To limit accessibility to only derived classes in the same assembly, declare the class itself internal, and declare its members as protected.  
  
**What is the default access modifier for a class, struct and an interface declared directly with a namespace?**   
internal  
  
**Will the following code compile?**  
using System;  
interface IExampleInterface  
{  
   public void Save();  
}  
  
No, you cannot specify access modifier for an interface member. Interface members are always public.  
  
**Can you specify an access modifier for an enumeration?**   
Enumeration members are always public, and no access modifiers can be specified.

**ASP.NET Interview Questions on Globalization**

**What is Globalization?**  
Globalization is the process of creating an application that meets the needs of users from multiple cultures. This process involves translating the user interface elements of an application into multiple languages, using the correct currency, date and time format, calendar, writing direction, sorting rules, and other issues. Accommodating these cultural differences in an application is called localization.  
  
The Microsoft .NET Framework simplifies localization tasks substantially by making its formatting, date/time, sorting, and other classes culturally aware. Using classes from the System.Globalization namespace, you can set the application’s current culture, and much of the work is done automatically!  
  
**What are the 3 different ways to globalize web applications?  
  
Detect and redirect approach :** In this approach we create a separate Web application for each supported culture, and then detect the user’s culture and redirect the request to the appropriate application. This approach is best for applications with lots of text content that requires translation and few executable components.  
  
**Run-time adjustment approach :** In this approach we create a single Web application that detects the user’s culture and adjusts output at run time using format specifiers and other tools. This approach is best for simple applications that present limited amounts of content.  
  
**Satellite assemblies approach :** In this approach we create a single Web application that stores culture-dependent strings in resource files that are compiled into satellite assemblies. At run time, detect the user’s culture and load strings from the appropriate assembly. This approach is best for applications that generate content at run time or that have large executable components.  
  
**In ASP.NET, how do you detect the user's language preference on his/her computer?**   
Use the Request object’s UserLanguages property to return a list of the user’s language preferences. The first element of the array returned by UserLanguages is the user’s current language on his/her computer.

**What are the steps to follow to get user's culture at run time?**  
To get the user’s culture at run time, follow these steps:  
**1.** Get the Request object’s UserLanguages property.  
**2.** Use the returned value with the CultureInfo class to create an object representing the user’s current culture.  
  
For example, the following code gets the user’s culture and displays the English name and the abbreviated name of the culture in a label the first time the page is displayed:  
private void Page\_Load(object sender, System.EventArgs e)  
{  
// Run the first time the page is displayed  
if (!IsPostBack)  
{  
// Get the user's preferred language.  
string sLang = Request.UserLanguages[0];  
// Create a CultureInfo object from it.  
CultureInfo CurrentCulture = new CultureInfo(sLang);  
lblCulture.Text = CurrentCulture.EnglishName + ": " +  
CurrentCulture.Name;  
}  
}  
  
**What are the advantages of using detect and redirect approach to globalizing web applications?**   
**1.** Content is maintained separately, so this approach allows the different applications to present very different information, if needed.  
**2.** Users can be automatically directed to sites that are likely to be geographically close, and so can better meet their needs.  
**3.** Content files (Web forms and HTML pages, for example) can be authored in the appropriate natural language without the complexity of including resource strings.  
  
**What are the disadvantages of using detect and redirect approach to globalizing web applications?**   
**1.** Using this approach requires that the executable portion of the Web application be compiled and deployed separately to each culture-specific Web site.  
**2.** This approach requires more effort to maintain consistency and to debug problems across Web sites.  
  
**What is the use of culture attribute of the globalization element in web.config?**  
The Web.config file’s globalization element is used to create a culture-specific Web application. The culture attribute of the globalization element specifies how the Web application deals with various culture-dependent issues, such as dates, currency, and number formatting.  
  
Web.config globalization settings in subordinate folders override the globalization settings in the application’s root Web.config file. You can store content for various cultures in subfolders within your application, add Web.config files with the globalization settings for each culture, then direct users to the appropriate folder based on the user’s CurrentCulture.  
  
**The text on the webform is usually written from left to right. How do you change the writing direction to "right to left"?**   
The wrting direction of a webform can be changed using the HTML dir attribute as shown below.  
**<body dir="rtl">**   
  
You can use the dir attribute individually in panels, text boxes, or other controls as well. Setting the dir attribute on the body element applies right-to-left formatting to the entire page.  
  
**What do you mean by neutral cultures?**  
Neutral cultures represent general languages, such as English or Spanish, rather than a specific language and region. When you set the culture attribute for a Web application in Web.config, ASP.NET assigns that culture to all the threads running for that Web application. Threads are the basic unit to which the server allocates processor time. ASP.NET maintains multiple threads for a Web application within the aspnet\_wp.exe worker process.  
  
**What are advantages of setting the culture dynamically at the thread level over creating separate Web applications for each culture?**   
**1.** All cultures share the same application code, so the application doesn’t have to be compiled and deployed for each culture.  
**2.** The application resides at a single Web address, you don’t need to redirect users to other Web applications.  
**3.** The user can choose from a full array of available cultures.  
  
**For what type of web applications setting the culture dynamically is best suited?**  
Setting the culture dynamically is best suited for simple Web applications that don’t contain large amounts of text that must be translated into different languages.

# Cross-site scripting(XSS)

**Cross-site scripting** (**XSS**) is a type of [security](http://en.wikipedia.org/wiki/Computer_insecurity) [vulnerability](http://en.wikipedia.org/wiki/Vulnerability_%28computer_science%29) typically found in [Web applications](http://en.wikipedia.org/wiki/Web_application) that enables attackers to [inject](http://en.wikipedia.org/wiki/Code_injection) [client-side script](http://en.wikipedia.org/wiki/Client-side_script) into [Web pages](http://en.wikipedia.org/wiki/Web_page) viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass [access controls](http://en.wikipedia.org/wiki/Access_control) such as the [same origin policy](http://en.wikipedia.org/wiki/Same_origin_policy). Cross-site scripting carried out on websites accounted for roughly 80.5% of all security vulnerabilities documented by Symantec as of 2007.[[1]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-Symantec-2007-2nd-exec-0) Their effect may range from a petty nuisance to a significant security risk, depending on the sensitivity of the data handled by the vulnerable site and the nature of any security mitigation implemented by the site's owner.

**Cross-site scripting holes are web-application vulnerabilities which allow attackers to bypass client-side security mechanisms normally imposed on web content by modern** [**web browsers**](http://en.wikipedia.org/wiki/Web_browser)**. By finding ways of injecting malicious scripts into web pages, an attacker can gain elevated access-privileges to sensitive page content, session cookies, and a variety of other information maintained by the browser on behalf of the user. Cross-site scripting attacks are therefore a special case of** [**code injection**](http://en.wikipedia.org/wiki/Code_injection)**.**

The expression "cross-site scripting" originally referred to the act of loading the attacked, third-party web application from an unrelated attack site, in a manner that executes a fragment of [JavaScript](http://en.wikipedia.org/wiki/JavaScript) prepared by the attacker in the [security context](http://en.wikipedia.org/wiki/Same_origin_policy) of the targeted domain (a *reflected* or *non-persistent* XSS vulnerability). The definition gradually expanded to encompass other modes of code injection, including persistent and non-JavaScript vectors (including [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29), [ActiveX](http://en.wikipedia.org/wiki/ActiveX), [VBScript](http://en.wikipedia.org/wiki/VBScript), [Flash](http://en.wikipedia.org/wiki/Adobe_Flash), or even pure [HTML](http://en.wikipedia.org/wiki/HTML), and [SQL](http://en.wikipedia.org/wiki/SQL) Queries), causing some confusion to newcomers to the field of [information security](http://en.wikipedia.org/wiki/Information_security).[[2]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-Grossman-1)

XSS vulnerabilities have been reported and exploited since the 1990s. Prominent sites affected in the past include the social-networking sites [Twitter](http://en.wikipedia.org/wiki/Twitter),[[3]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-2) [Facebook](http://en.wikipedia.org/wiki/Facebook),[[4]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-3) [MySpace](http://en.wikipedia.org/wiki/MySpace), and [Orkut](http://en.wikipedia.org/wiki/Orkut).[[5]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-4)[[6]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-5) In recent years, cross-site scripting flaws surpassed [buffer overflows](http://en.wikipedia.org/wiki/Buffer_overflow) to become the most common publicly-reported security vulnerability,[[7]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-6) with some researchers in 2007 viewing as many as 68% of websites as likely open to XSS attacks.[[8]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-7)

Types

The *non-persistent* (or *reflected*) cross-site scripting vulnerability is by far the most common type.[[10]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-HopeWalther-9) These holes show up when the data provided by a web client, most commonly in HTTP query parameters or in HTML form submissions, is used immediately by server-side scripts to generate a page of results for that user, without properly [sanitizing](http://en.wikipedia.org/wiki/HTML_sanitization) the request.[[11]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-WASC-2005-10)

|  |
| --- |
| **Persistent** |

*persistent* (or *stored*) XSS vulnerability is a more devastating variant of a cross-site scripting flaw: it occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing, without proper HTML escaping. A classic example of this is with online message boards where users are allowed to post HTML formatted messages for other users to read

**Traditional versus DOM-based vulnerabilities**

|  |
| --- |
| **Example of DOM-based XSS** |
|  |

Traditionally, cross-site scripting vulnerabilities would occur in server-side code responsible for preparing the HTML response to be served to the user. With the advent of [web 2.0](http://en.wikipedia.org/wiki/Web_2.0) applications a new class of XSS flaws emerged: *DOM-based* vulnerabilities. DOM-based vulnerabilities occur in the content processing stages performed by the client, typically in [client-side JavaScript](http://en.wikipedia.org/wiki/Client-side_JavaScript). The name refers to the standard model for representing HTML or XML contents which is called the [Document Object Model](http://en.wikipedia.org/wiki/Document_Object_Model) (DOM). JavaScript programs manipulate the state of a web page and populate it with dynamically-computed data primarily by acting upon the DOM.

## Reducing the threat

**Contextual output encoding/escaping of string input**

The primary defense mechanism to stop XSS is contextual output encoding/escaping. There are several different escaping schemes that must be used depending on where the untrusted string needs to be placed within an HTML document including HTML entity encoding, JavaScript escaping, CSS escaping, and [URL (or percent) encoding](http://en.wikipedia.org/wiki/Percent-encoding).[[17]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-OWASP-16) Most web applications that do not need to accept rich data can use escaping to largely eliminate the risk of XSS in a fairly straightforward manner.

It is worth noting that although it is widely recommended, simply performing HTML entity encoding on the [five XML significant characters](http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references#Predefined_entities_in_XML) is not always sufficient to prevent many forms of XSS. Encoding can be tricky, and the use of a security encoding library is highly recommended.[[17]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-OWASP-16)

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=9)**] Safely validating untrusted HTML input**

Many operators of particular web applications (e.g. forums and webmail) wish to allow users to utilize some of the features HTML provides, such as a limited subset of HTML markup. When accepting HTML input from users (say, <b>very</b> large), output encoding (such as &lt;b&gt;very&lt;/b&gt; large) will not suffice since the user input needs to be rendered as HTML by the browser (so it shows as "**very** large", instead of "<b>very</b> large"). Stopping XSS when accepting HTML input from users is much more complex in this situation. Untrusted HTML input must be run through an HTML policy engine to ensure that it does not contain XSS. [HTML sanitization](http://en.wikipedia.org/wiki/HTML_sanitization) tools such as OWASP AntiSamy [[18]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-OWASP2-17) and <http://htmlpurifier.org/> accomplish this task.

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=10)**] Cookie security**

Besides content filtering, other imperfect methods for cross-site scripting mitigation are also commonly used. One example is the use of additional security controls when handling [cookie](http://en.wikipedia.org/wiki/HTTP_cookie)-based user authentication. Many web applications rely on session cookies for authentication between individual HTTP requests, and because client-side scripts generally have access to these cookies, simple XSS exploits can steal these cookies.[[19]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-Sharma-18) To mitigate this particular threat (though not the XSS problem in general), many web applications tie session cookies to the IP address of the user who originally logged in, and only permit that IP to use that cookie.[[20]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-ModSecurity-19) This is effective in most situations (if an attacker is only after the cookie), but obviously breaks down in situations where an attacker [spoofs their IP address](http://en.wikipedia.org/wiki/IP_address_spoofing), is behind the same [NATed](http://en.wikipedia.org/wiki/Network_address_translation) IP address or [web proxy](http://en.wikipedia.org/wiki/Web_proxy)—or simply opts to tamper with the site or steal data through the injected script, instead of attempting to hijack the cookie for future use.[[20]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-ModSecurity-19)

Another mitigation present in IE (since version 6), Firefox (since version 2.0.0.5), Safari (since version 4), Opera (since version 9.5) and Google Chrome, is an *HttpOnly* flag which allows a web server to set a cookie that is unavailable to client-side scripts. While beneficial, the feature does not fully prevent cookie theft nor can it prevent attacks within the browser.[[21]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-20)

**[**[**edit**](http://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=11)**] Disabling scripts**

Finally, while [Web 2.0](http://en.wikipedia.org/wiki/Web_2.0) and [Ajax](http://en.wikipedia.org/wiki/Ajax_%28programming%29) designers favor the use of JavaScript,[[22]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-21) some web applications are written to (sometimes optionally) operate completely without the need for client-side scripts.[[23]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-22) This allows users, if they choose, to disable scripting in their browsers before using the application. In this way, even potentially malicious client-side scripts could be inserted unescaped on a page, and users would not be susceptible to XSS attacks.

Some browsers or browser plugins can be configured to disable client-side scripts on a per-domain basis. If scripting is allowed by default, then this approach is of limited value, since it blocks bad sites only *after* the user knows that they are bad, which is too late. Functionality that blocks all scripting and external inclusions by default and then allows the user to enable it on a per-domain basis is more effective. This has been possible for a long time in IE (since version 4) by setting up its so called "Security Zones",[[24]](http://en.wikipedia.org/wiki/Cross-site_scripting" \l "cite_note-23) and in Opera (since version 9) using its "Site Specific Preferences".[[25]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-24) A solution for Firefox and other [Gecko](http://en.wikipedia.org/wiki/Gecko_%28layout_engine%29)-based browsers is the open source [NoScript](http://en.wikipedia.org/wiki/NoScript) add-on which, in addition to the ability to enable scripts on a per-domain basis, provides some anti-XSS protection even when scripts are enabled

### Emerging defensive technologies

There are three classes of XSS defense that are emerging. These include, Mozilla's [Content Security Policy](http://en.wikipedia.org/wiki/Content_Security_Policy),[[30]](http://en.wikipedia.org/wiki/Cross-site_scripting" \l "cite_note-Mozilla_CSP_Specification-29) Javascript Sandbox tools, and Auto-escaping templates. These mechanisms are still evolving but promise a future of heavily reduced XSS.

## Scanning service

Some companies offer a periodic scan service, essentially simulating an attack from their server to a client's in order to check if the attack is successful. If the attack succeeds, the client receives detailed information on how it was performed and thus has a chance to fix the issues before the same attack is attempted by someone else. A [trust seal](http://en.wikipedia.org/wiki/Trust_seal) can be displayed on the site that passes a recent scan. The scanner may not find all possible vulnerabilities,[[31]](http://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-30) and therefore sites with trust seals may still be vulnerable to new types of attack, but the scan may detect some problems. After the client fixes them, the site is more secure than it was before using the service. For sites that require complete mitigation of XSS, assessment techniques like manual code review are necessary. Additionally, if javascript is executing on the page, the seal can be overwritten with a static copy of the seal.