Ruby master - Bug #11772

Implicit conversion of Array in String interpolation does not call to_str

12/05/2015 05:22 AM - danielpclark (Daniel P. Clark)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN

Description

While providing an example in https://bugs.ruby-lang.org/issues/10930#note-7 I found that the implicit changing of an Array calls to_s without trying to_str first. Isn't String interpolation "implicitly" converting items to strings?

```
class Array
  def to_str
    "Hello from Array!"
  end
end

"#{[1,2,3,4,5]}"
# => [1, 2, 3, 4, 5]
```

I believe String interpolation should call to_str before to_s

History

#1 - 12/05/2015 05:32 AM - danielpclark (Daniel P. Clark)

My main point here is that anything that happens within String interpolation is "implicitly" going to be a String. So shouldn't it call to_str before to_s on anything given?

```
class B
  def to_str
    "Hello from B"
  end
end


"#{B.new}"
# => "#<B:0x00000001969078>"
```

Should work as this does.

```
class B
  def to_s
    "to_s"
  end
end

"#{B.new}"
# => "to_s"
```

I also realize that having Ruby lookup two methods to_str and then to_s may slow it down and that we don't want to make it slower. Since Ruby is optimized for to_s being called first I think it would be okay to have to_str called second. Even though by definition you would think that an "implicit" method would be the first method called.

#2 - 12/05/2015 05:54 AM - duerst (Martin Dürst)

- Status changed from Open to Closed

As (implicitly) requested at ruby-core:71843.

#3 - 12/06/2015 04:37 AM - danielpclark (Daniel P. Clark)

```
class A
```

```
  undef :to_s
  def to_str
    "hello world"
  end
end

"#{A.new}"
#NoMethodError: undefined method `to_s' for #<A:0x00000000dbea70>

"" + A.new
# => "hello world"
```

Using + to call to_str feels like JavaScript.