# Hash#slice fails to copy default block

## Description

An intense weekend debugging session discovered the following root cause of a bug: Hash#slice returns a new Hash, which has no default block set, even if the source Hash did have a default block set.

## Simplified code to reproduce:

```ruby
# Default to an empty hash for all accessed keys
hash_with_default = Hash.new { |h, k| h[k] = {} } # => {}
hash_with_default[:a] # => {}
hash_with_default[:b] # => {}
hash_with_default # => {:a=>{}, :b=>{}}

# Later, use Hash#slice
hash_sliced = hash_with_default.slice(:a, :b) # => {:a=>{}, :b=>{}}

# Finally, access a new key
hash_sliced[:c] # => nil

# Error -- that was expected to call the default block
raise "No default value" unless hash_sliced[:c] == {}
```

## History

### #1 - 07/30/2022 10:38 PM - jeremyevans0 (Jeremy Evans)

- Backport deleted (2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN)
- ruby -v deleted (2.7.6)
- Tracker changed from Bug to Feature

Hash#slice documentation states it returns a new hash object, not a modified copy of the receiver. A new hash object by default does not have a default value or block. Hash#slice doesn't copy the default value either: Hash.new(0).slice(:a)[1] # => nil.

I don't think the current behavior is a bug, as there should be no expectation of the default value and block being copied. Hash#except, #select, #transform_keys, #transform_values, and maybe other methods that return new hashes and not modified copies do not copy over the default value or block.

That being said, this seems like a reasonable feature request.

### #2 - 08/05/2022 05:57 PM - Eregon (Benoit Daloze)

This could lead to surprising behavior, e.g.:

```ruby
cache = Hash.new { |h,k| cache[k] = {} }
cache[:a]
sliced = cache.slice(:a)
sliced[:foo] # => actually mutates cache with this proposal
```

Yes this would not be a problem if h[k] = {} is used as the block, but that's not always the case.

Also if we start copying the default value/block, it seems a slippery slope e.g. to copy instance variables, class, and other state too, which all seem not desirable (and even were decided against recently).

OTOH I guess copying compare_by_identity does make sense.