Class

MTLHeapDescriptor

A configuration that customizes the behavior for a Metal memory heap.

iOS 10.0+ | iPadOS 10.0+ | Mac Catalyst 13.1+ | macOS 10.13+ | tvOS 10.0+ | visionOS 1.0+

```
class MTLHeapDescriptor
```

Mentioned in

- Creating sparse heaps and sparse textures
- Understanding the Metal 4 core API

Overview

Create an `MTLHeap` by configuring an `MTLHeapDescriptor` instance's properties and passing it to the `makeHeap(descriptor:)` method of an `MTLDevice`.

Each new heap inherits the descriptor's configuration as you create it, which means you can modify and reuse a descriptor to create other heaps.

Topics

Configuring a heap

```
var type: MTLHeapType
```

The memory placement strategy for any resources you allocate from the heaps you create with this descriptor.

`var storageMode: MTLStorageMode`

The storage mode for the heaps you create with this descriptor.

`var cpuCacheMode: MTLCPUCacheMode`

The CPU cache behavior for any resources you allocate from the heaps you create with this descriptor.

`var hazardTrackingMode: MTLHazardTrackingMode`

The hazard tracking behavior for any resources you allocate from the heaps you create with this descriptor.

`var resourceOptions: MTLResourceOptions`

The combined behavior for any resources you allocate from the heaps you create with this descriptor.

`var size: Int`

The total amount of memory, in bytes, for the heaps you create with this descriptor.

`var sparsePageSize: MTLSparsePageSize`

The page size for any resources you allocate from the heaps you create with this descriptor.

Instance Properties

`var maxCompatiblePlacementSparsePageSize: MTLSparsePageSize`

Specifies the largest sparse page size that the Metal heap supports.

Relationships

Inherits From

`NSObject`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`

CustomStringConvertible
Equatable
Hashable
NSCopying
NSObjectProtocol

See Also

Resource memory allocation and management

{} Using argument buffers with resource heaps

Reduce CPU overhead by using arrays inside argument buffers and combining them with resource heaps.

{} Implementing a multistage image filter using heaps and events

Use events to synchronize access to resources allocated on a heap.

{} Implementing a multistage image filter using heaps and fences

Use fences to synchronize access to resources allocated on a heap.

`protocol` MTLHeap

A memory pool from which you can suballocate resources.

`enum` MTLHeapType

The options you use to choose the heap type.

`struct` MTLSizeAndAlign

The size and alignment of a resource, in bytes.