Documentation

API Collection

AsyncSequence Implementations

HummingbirdWebSocket | WSCore | WSClient

Topics

Instance Methods

```
func adjacentPairs() -> AsyncAdjacentPairsSequence<Self>
```

```
func allSatisfy((Self.Element) async throws -> Bool) async rethrows -> Bool
```

```
func buffer(policy: AsyncBufferSequencePolicy) -> AsyncBufferSequence<Self>
```

```
func cancelOnGracefulShutdown() -> AsyncCancelOnGracefulShutdownSequence<Self>
```

```
func chunked(by: (Self.Element, Self.Element) -> Bool) -> AsyncChunkedByGroupSequence<Self, [Self.Element]>
```

```
func chunked<Signal, Collected>(by: Signal, into: Collected.Type) -> AsyncChunksOfCountOrSignalSequence<Self, Collected, Signal>
```

```
func chunked<C, Collected>(by: AsyncTimerSequence<C>, into: Collected.Type) -> AsyncChunksOfCountOrSignalSequence<Self, Collected, AsyncTimerSequence<C>>
```

```
func chunked<Collected>(into: Collected.Type, by: (Self.Element, Self.Element) -> Bool) -> AsyncChunkedByGroupSequence<Self,
```

```
Collected>

func chunked<Subject, Collected>(into: Collected.Type, on: (Self.
Element) -> Subject) -> AsyncChunkedOnProjectionSequence<Self,
Subject, Collected>

func chunked<Subject>(on: (Self.Element) -> Subject) -> AsyncChunked
OnProjectionSequence<Self, Subject, [Self.Element]>

func chunks(ofCount: Int) -> AsyncChunksOfCountSequence<Self, [Self.
Element]>

func chunks<Collected>(ofCount: Int, into: Collected.Type) -> Async
ChunksOfCountSequence<Self, Collected>

func chunks<C>(ofCount: Int, or: AsyncTimerSequence<C>) -> Async
ChunksOfCountOrSignalSequence<Self, [Self.Element], AsyncTimer
Sequence<C>>

func chunks<Signal>(ofCount: Int, or: Signal) -> AsyncChunksOfCount
OrSignalSequence<Self, [Self.Element], Signal>

func chunks<Signal, Collected>(ofCount: Int, or: Signal, into:
Collected.Type) -> AsyncChunksOfCountOrSignalSequence<Self,
Collected, Signal>

func chunks<C, Collected>(ofCount: Int, or: AsyncTimerSequence<C>,
into: Collected.Type) -> AsyncChunksOfCountOrSignalSequence<Self,
Collected, AsyncTimerSequence<C>>

func compactMap<ElementOfResult>((Self.Element) async -> ElementOf
Result?) -> AsyncCompactMapSequence<Self, ElementOfResult>

func compactMap<ElementOfResult>((Self.Element) async throws ->
ElementOfResult?) -> AsyncThrowingCompactMapSequence<Self, ElementOf
Result>

func compacted<Unwrapped>() -> AsyncCompactedSequence<Self,
Unwrapped>

func contains(Self.Element) async rethrows -> Bool

func contains(where: (Self.Element) async throws -> Bool) async
rethrows -> Bool
```

```
func debounce(for: Duration, tolerance: Duration?) -> AsyncDebounce
Sequence<Self, ContinuousClock>

func debounce<C>(for: C.Instant.Duration, tolerance: C.Instant.
Duration?, clock: C) -> AsyncDebounceSequence<Self, C>

func drop(while: (Self.Element) async -> Bool) -> AsyncDropWhile
Sequence<Self>

func dropFirst(Int) -> AsyncDropFirstSequence<Self>

func filter((Self.Element) async -> Bool) -> AsyncFilterSequence<
Self>

func first(where: (Self.Element) async throws -> Bool) async
rethrows -> Self.Element?

func flatMap<SegmentOfResult>((Self.Element) async -> SegmentOf
Result) -> AsyncFlatMapSequence<Self, SegmentOfResult>

func flatMap<SegmentOfResult>((Self.Element) async throws -> Segment
OfResult) -> AsyncThrowingFlatMapSequence<Self, SegmentOfResult>

func flatMap<SegmentOfResult>((Self.Element) async -> SegmentOf
Result) -> AsyncFlatMapSequence<Self, SegmentOfResult>

func flatMap<SegmentOfResult>((Self.Element) async -> SegmentOf
Result) -> AsyncFlatMapSequence<Self, SegmentOfResult>

func interspersed(every: Int, with: () -> Self.Element) -> Async
InterspersedSequence<Self>

func interspersed(every: Int, with: Self.Element) -> Async
InterspersedSequence<Self>

func interspersed(every: Int, with: () throws -> Self.Element) ->
AsyncThrowingInterspersedSequence<Self>

func interspersed(every: Int, with: () async -> Self.Element) ->
AsyncInterspersedSequence<Self>

func interspersed(every: Int, with: () async throws -> Self.Element)
-> AsyncThrowingInterspersedSequence<Self>
```

```
func map<Transformed>((Self.Element) async -> Transformed) -> Async
MapSequence<Self, Transformed>

func map<Transformed>((Self.Element) async throws -> Transformed) ->
AsyncThrowingMapSequence<Self, Transformed>

func max(by: (Self.Element, Self.Element) async throws -> Bool)
async rethrows -> Self.Element?

func min(by: (Self.Element, Self.Element) async throws -> Bool)
async rethrows -> Self.Element?

func prefix(Int) -> AsyncPrefixSequence<Self>

func prefix(while: (Self.Element) async -> Bool) rethrows -> Async
PrefixWhileSequence<Self>

func reduce<Result>(Result, (Result, Self.Element) async throws ->
Result) async rethrows -> Result

func reduce<Result>(into: Result, (inout Result, Self.Element) async
 throws -> Void) async rethrows -> Result

func reductions((Self.Element, Self.Element) async -> Self.Element)
-> AsyncInclusiveReductionsSequence<Self>

func reductions<Result>(Result, (Result, Self.Element) async ->
Result) -> AsyncExclusiveReductionsSequence<Self, Result>

func reductions<Result>(Result, (Result, Self.Element) async throws
-> Result) -> AsyncThrowingExclusiveReductionsSequence<Self, Result>

func reductions<Result>(into: Result, (inout Result, Self.Element)
async throws -> Void) -> AsyncThrowingExclusiveReductionsSequence<
Self, Result>

func reductions<Result>(into: Result, (inout Result, Self.Element)
async -> Void) -> AsyncExclusiveReductionsSequence<Self, Result>

func removeDuplicates() -> AsyncRemoveDuplicatesSequence<Self>

func removeDuplicates(by: (Self.Element, Self.Element) async -> Bool
) -> AsyncRemoveDuplicatesSequence<Self>
```