:c:type:`uv_async_t` --- Async handle

Async handles allow the user to "wakeup" the event loop and get a callback called from another thread.

Data types

```
.. c:type:: uv_async_t

    Async handle type.
```

```
.. c:type:: void (*uv_async_cb)(uv_async_t* handle)

    Type definition for callback passed to :c:func:`uv_async_init`.
```

Public members

N/A

```
.. seealso:: The :c:type:`uv_handle_t` members also apply.
```

API

```
.. c:function:: int uv_async_init(uv_loop_t* loop, uv_async_t* async, uv_async_cb async_cb)

    Initialize the handle. A NULL callback is allowed.

    :returns: 0 on success, or an error code < 0 on failure.

    .. note::
        Unlike other handle initialization  functions, it immediately starts the handle.
```

```
.. c:function:: int uv_async_send(uv_async_t* async)

    Wake up the event loop and call the async handle's callback.

    :returns: 0 on success, or an error code < 0 on failure.

    .. note::
        It's safe to call this function from any thread. The callback will be called on the
        loop thread.

    .. note::
        :c:func:`uv_async_send` is `async-signal-safe <https://man7.org/linux/man-pages/man7/signal-safety.
        It's safe to call this function from a signal handler.

    .. warning::
        libuv will coalesce calls to :c:func:`uv_async_send`, that is, not every call to it will
        yield an execution of the callback. For example: if :c:func:`uv_async_send` is called 5
```

```
times in a row before the callback is called, the callback will only be called once. If
:c:func:`uv_async_send` is called again after the callback was called, it will be called
again.
```

Unknown directive type "seealso".

```
.. seealso::
    The :c:type:`uv_handle_t` API functions also apply.
```