Asynchronous Programming

Terminology

Synchronous

Only one process executes at a time.

JavaScript is synchronous.

Blocking

When a given process, such as reading from a file or waiting on user input, prevents the further execution of other code.

Asynchronous

More than one process runs simultaneously.

Node is asynchronous, even though JavaScript and the V8 engine are not.

Non-blocking

Code will continue with execution without waiting for certain processes, such as reading from a file or waiting on user input, to complete.

Thread

Short for a thread of execution.

Threads are a way for a program to divide itself into two or more simultaneously (or pseudo-simultaneously) running tasks.

Synchronous vs Asynchronous

```
var result = database.query('SELECT * FROM largedatabasetable');
console.log('Hello World');
```

```
database.query('SELECT * FROM largedatabasetable', function(rows) {
 var result = rows;
 // results of database fetch are used in code here
});
console.log('Hello World');
```

libuv and the Event Loop

libuv

A multi-platform support library that focuses on asynchronous input and output.

Primarily developed for Node.js, but is used by other technologies.

Diagram of V8 and libuv inside Node

Event Loop

The Event Loop allows us to write synchronous code and still respond to processes that are running asynchronously. Node handles the asynchronous processes and we don't have to worry about how they will affect each other.

The result is that many people can ask the server to do many things, and no one is blocked while the server is executing these requests, and we simply respond as things finish.

```c
int uv_run(uv_loop_t* loop, uv_run_mode mode) {
  int timeout;
  int r;
  int ran_pending;

  r = uv__loop_alive(loop);
  if (!r)
    uv__update_time(loop);

  while (r != 0 && loop->stop_flag == 0) {
    uv__update_time(loop);
    uv__run_timers(loop);
    ran_pending = uv__run_pending(loop);
    uv__run_idle(loop);
    uv__run_prepare(loop);

    timeout = 0;
    if ((mode == UV_RUN_ONCE && !ran_pending) || mode == UV_RUN_DEFAULT)
      timeout = uv_backend_timeout(loop);

    uv__io_poll(loop, timeout);
    uv__run_check(loop);
    uv__run_closing_handles(loop);

    if (mode == UV_RUN_ONCE) {
      uv__update_time(loop);
      uv__run_timers(loop);
    }

    r = uv__loop_alive(loop);
    if (mode == UV_RUN_ONCE || mode == UV_RUN_NOWAIT)
      break;
  }

  if (loop->stop_flag != 0)
    loop->stop_flag = 0;

  return r;
}
```

Callbacks

Callback

A function that is passed as an argument to another function, which will be invoked at a later time, usually after the called function has finished executing.

Code Demo

Callback Pattern

Allows for different uses of data that's a result of commonly needed heavy operations, such as file reading or database querying.

The result data needs to be passed to the callback function from the calling function as an argument.

Asynchronous File Read Demo

Problems with Callbacks

Callback Pyramid of Doom

You can call other methods that require callbacks inside of callbacks, that themselves call functions that require callbacks.