BCDV 1022

Node Process Object

2023 Fall

week 03 - class 07

Topics

- Process Object
- Scalability & Child Processes

Process Object

Process Object

- The process object is a global object that provides information about and control over, the current running Node.js process.

- A collection of streams:
    - process.stdout - a writable stream to stdout
    - process.stdin - a writable stream to stdin
    - process.stderr - a writable stream to stderr

- Some process actions:
    - process.cwd() - returns the current directory
    - process.kill(pid,[signal]) - sends a signal to kill process
    - process.chdir() - change process directory

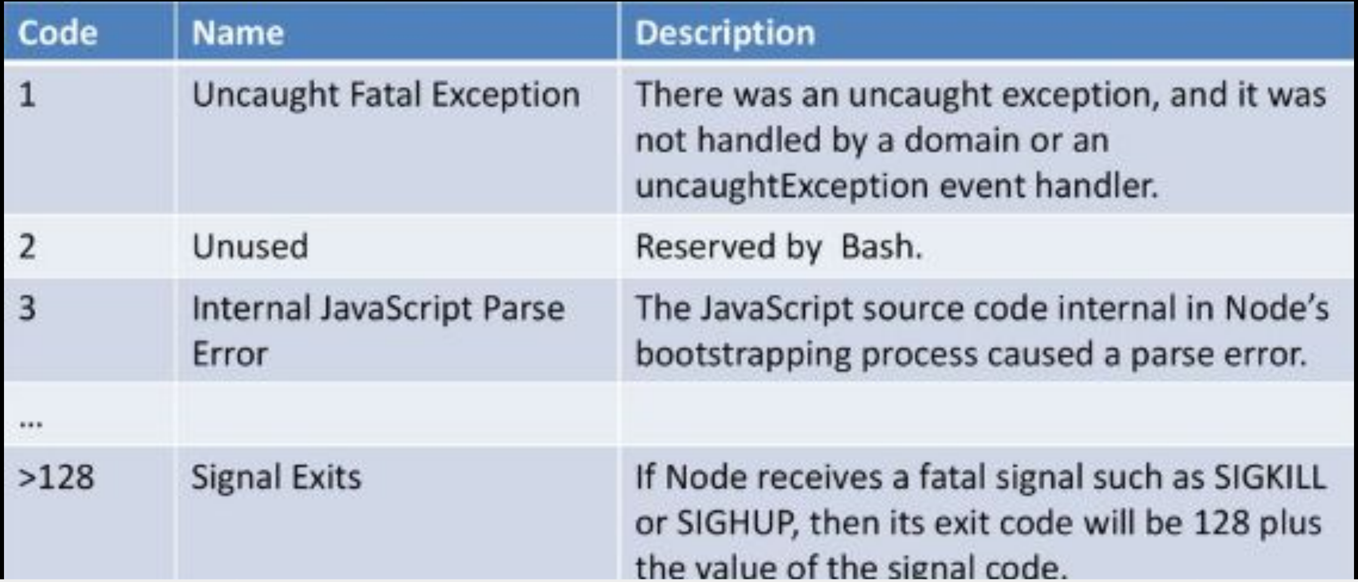
Process Object

- Some process attributes:
  - process.PID
  - process.uptime()
  - process.memoryusage()

- Process is an instance of EventEmitter:
  - Event ‘exit’
  - Event ‘uncaughtException’
  - POSIX signals (SIGINT)

Process Exit Codes

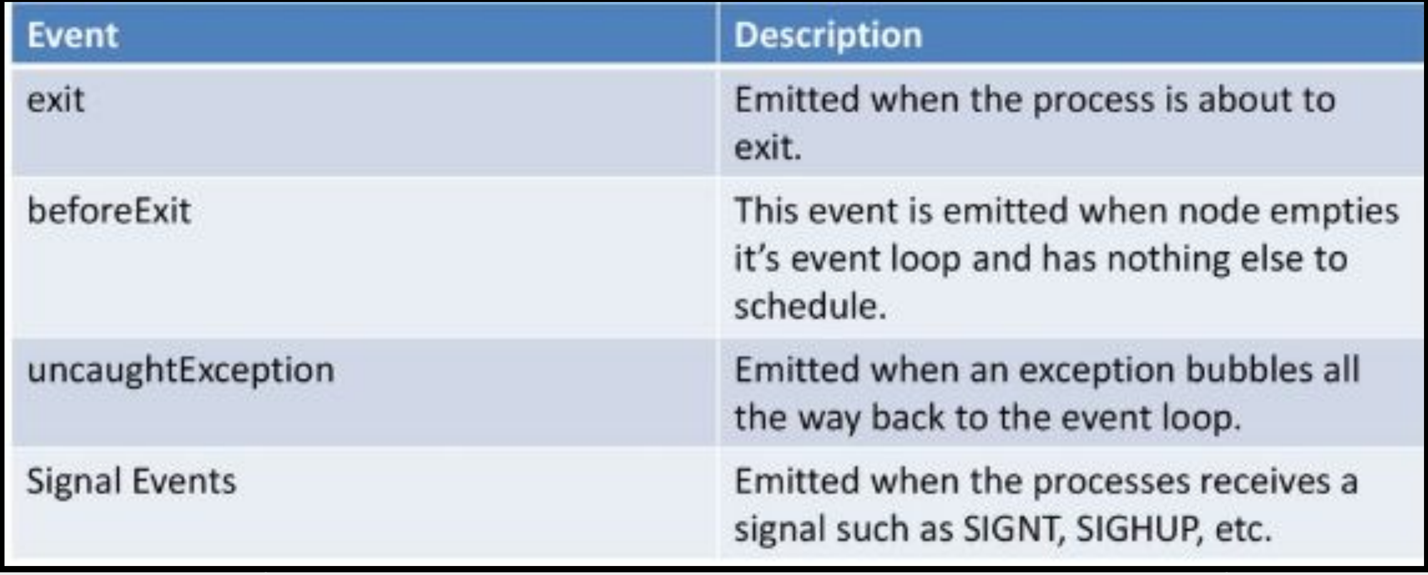
- Node normally exits with a 0 status code when no more async operations are pending. There are other exit codes:



Code	Name	Description
1	Uncaught Fatal Exception	There was an uncaught exception, and it was not handled by a domain or an uncaughtException event handler.
2	Unused	Reserved by Bash.
3	Internal JavaScript Parse Error	The JavaScript source code internal in Node's bootstrapping process caused a parse error.
...		
>128	Signal Exits	If Node receives a fatal signal such as SIGKILL or SIGHUP, then its exit code will be 128 plus the value of the signal code.

Process Events

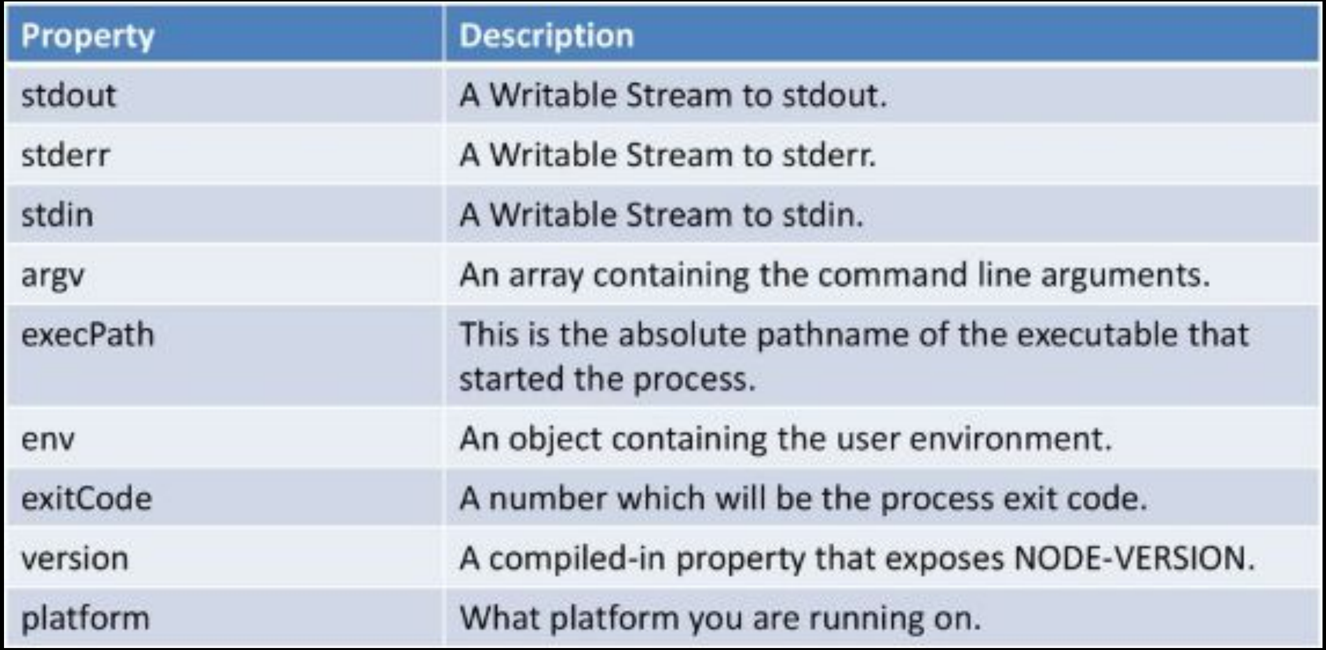
- Process is an eventEmitter and emits the following events:



Event	Description
exit	Emitted when the process is about to exit.
beforeExit	This event is emitted when node empties it's event loop and has nothing else to schedule.
uncaughtException	Emitted when an exception bubbles all the way back to the event loop.
Signal Events	Emitted when the processes receives a signal such as SIGNT, SIGHUP, etc.

Process Properties

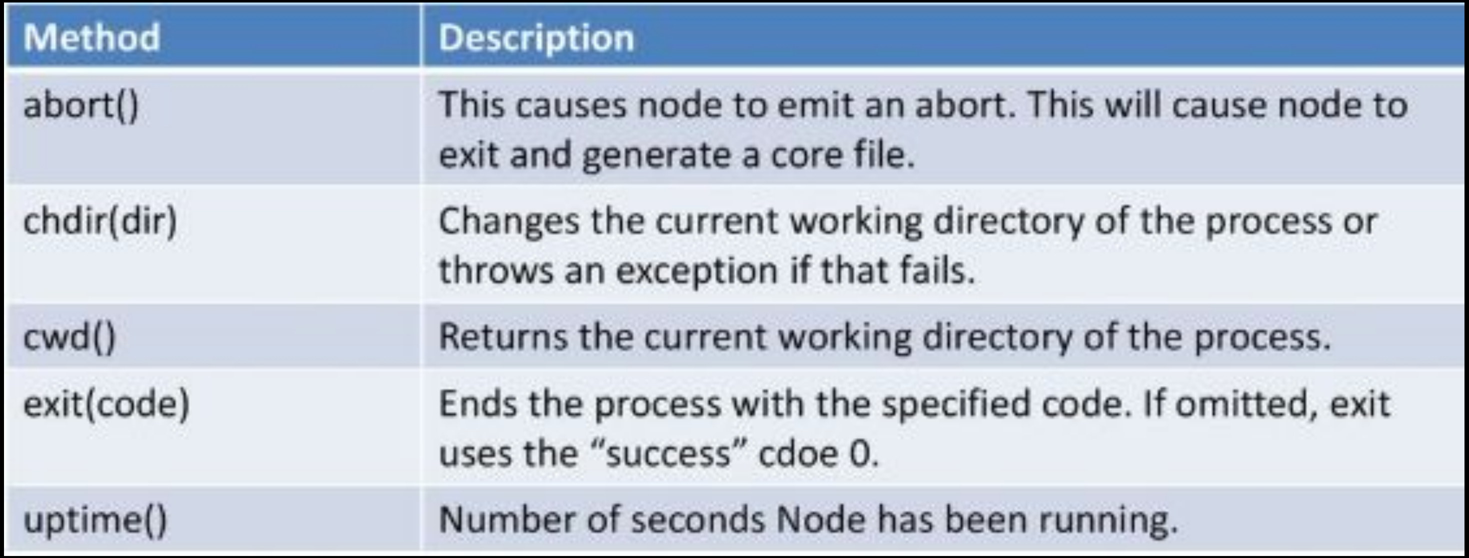
- Process provides many useful properties to get better control over the system interactions



Property	Description
stdout	A Writable Stream to stdout.
stderr	A Writable Stream to stderr.
stdin	A Writable Stream to stdin.
argv	An array containing the command line arguments.
execPath	This is the absolute pathname of the executable that started the process.
env	An object containing the user environment.
exitCode	A number which will be the process exit code.
version	A compiled-in property that exposes NODE-VERSION.
platform	What platform you are running on.

Process Methods

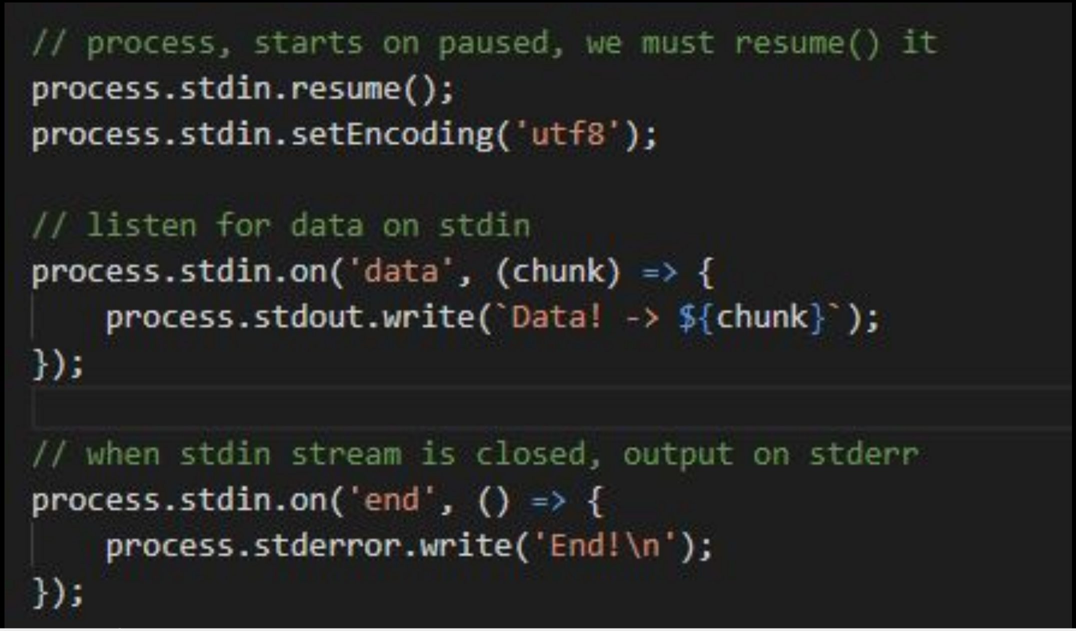
- Process provides many useful methods to get better control over the system interactions



Method	Description
abort()	This causes node to emit an abort. This will cause node to exit and generate a core file.
chdir(dir)	Changes the current working directory of the process or throws an exception if that fails.
cwd()	Returns the current working directory of the process.
exit(code)	Ends the process with the specified code. If omitted, exit uses the “success” cdoe 0.
uptime()	Number of seconds Node has been running.

Process Streams

- Process does not need to be required, it is available by default to Node



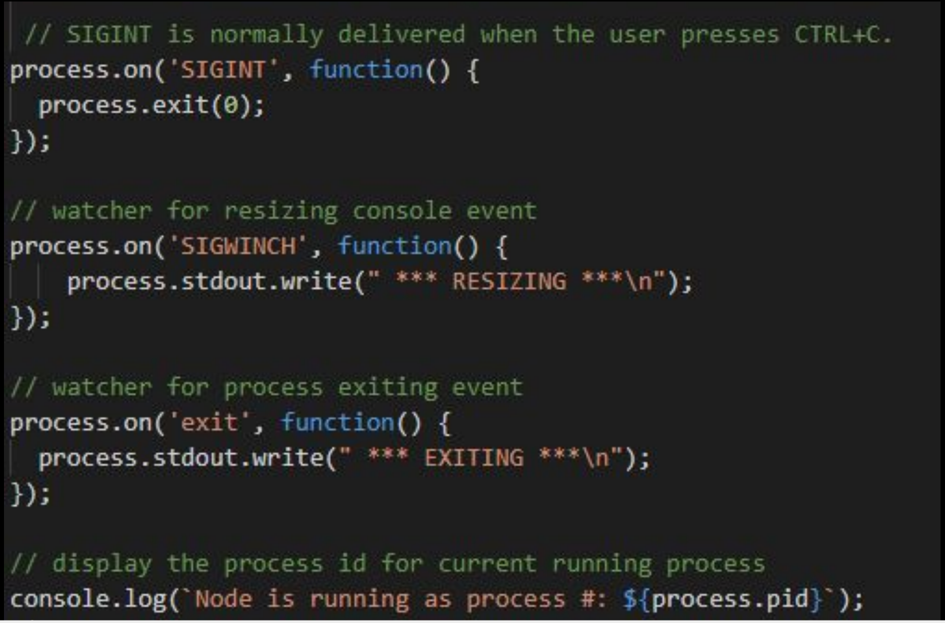
```
// process, starts on paused, we must resume() it
process.stdin.resume();
process.stdin.setEncoding('utf8');

// listen for data on stdin
process.stdin.on('data', (chunk) => {
    process.stdout.write(`Data! -> ${chunk}`);
});

// when stdin stream is closed, output on stderr
process.stdin.on('end', () => {
    process.stderror.write('End!\n');
});
```

Process Events

- Watchers for other signals can be created. These signals are: SIGFPE, SIGTERM and SIGKILL etc..



```
// SIGINT is normally delivered when the user presses CTRL+C.
process.on('SIGINT', function() {
  process.exit(0);
});

// watcher for resizing console event
process.on('SIGWINCH', function() {
    process.stdout.write(" *** RESIZING ***\n");
});

// watcher for process exiting event
process.on('exit', function() {
  process.stdout.write(" *** EXITING ***\n");
});

// display the process id for current running process
console.log(`Node is running as process #: ${process.pid}`);
```