**Node.js Core Objects – Process Object**

**Process Object**

* **Need to import in process objects from node module ‘process’**

**process.env**

* stores and controls information about the environment
* Often used to store whether it’s a production or dev environment
  + E.g. would look like: process.env.NODE\_ENV === “production”

{

TERM: 'xterm-256color',

SHELL: '/usr/local/bin/bash',

USER: 'maciej',

PATH: '~/.bin/:/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin',

PWD: '/Users/maciej',

EDITOR: 'vim',

SHLVL: '1',

HOME: '/Users/maciej',

LOGNAME: 'maciej',

\_: '/usr/local/bin/node'

}

* Can modify the object:
  + E.g.

process.env.foo = 'bar';

console.log(process.env.foo);

**process.memoryUsage()**

* Returns information on the CPU demands of the current process.
* Returns a property that looks like this:
* {rss: 1341345245,

heapTotal: 123435,

heapUsed: 536544,

external: 3842 }

* process.memoryUsage().heapUsed will return a number which represents how many bytes of memory the current process is using

**process.argv**

* Returns an array of command line values
* First element is the absolute path to Node
* The Second is the path to the file that it’s running
* The following elements will be command line arguments provided when the process initiated.
* E.g.

node myProgram.js testing several features (run this in command line)

console.log(process.argv[3]); // Prints 'several' (run in .js file)

0: /usr/local/bin/node

1: /Users/imperativethane/developer-projects/node/process-args.js

2: testing

3: several

4: features

**process.arch**

* Will return a value which represents the operating system CPU architecture for which the Node.js binary was compiled.
* Possible examples are:
  + 'arm', 'arm64', 'ia32', 'mips','mipsel', 'ppc', 'ppc64', 's390', 's390x', 'x32', and 'x64'
* console.log(`This processor architecture is ${process.arch}`);

**process.chdir(directory)**

* This method is used to change the working directory of the node.js process or throw an exception if it fails to do so.

console.log(`Starting directory: ${process.cwd()}`);

try {

process.chdir('/tmp');

console.log(`New directory: ${process.cwd()}`);

} catch (err) {

console.error(`chdir: ${err}`);

}

**process.config**

* Returns an object containing the JS representation of the configuration options used to compile Node.
* E.g.

{

target\_defaults:

{ cflags: [],

default\_configuration: 'Release',

defines: [],

include\_dirs: [],

libraries: [] },

variables:

{

host\_arch: 'x64',

napi\_build\_version: 5,

node\_install\_npm: 'true',

node\_prefix: '',

node\_shared\_cares: 'false',

node\_shared\_http\_parser: 'false',

node\_shared\_libuv: 'false',

node\_shared\_zlib: 'false',

node\_use\_dtrace: 'false',

node\_use\_openssl: 'true',

node\_shared\_openssl: 'false',

strict\_aliasing: 'true',

target\_arch: 'x64',

v8\_use\_snapshot: 1

}

}

process.stdout.write()

* stdout stands for standard output
* console.log is a process.stdout.write() in a wrapper.
* E.g.

process.stdout.write("I'm thinking of a number from 1 through 10. What do you think it is?");

process.stdin.on()

* Can receive input from a user through the terminal using stdin.on().
* Can use .on() as process.stdin is an event emitter under the hood.
* E.g.

process.stdin.on('data', (userInput) => {

let input = userInput.toString()

console.log(input)

});

* When the user enters text into the terminal and hits enter, a ‘data’ event will be fired and the anonymous listener callback will be invoked