1 The Node Binary

1.1 Common Command Line Arguments

```
# only check syntax
node --check app.js
node -c app.js

# evaluate (but don't print)
node --eval "1+1"
node -e "console.log(1+1)"
2
node -e "console.log(1+1);0"
2

# evaluate and print
node -e "console.log(1+1)"
2
undefined

node -p "console.log(1+1);0"
2
0
```

1.2 Module availability

All Node core modules can be accessed by their namespaces within the code evaluation context - no require required:

```
node -p "fs.readdirSync('.').filter((f)=>/.js$/.test(f))"
[]
```

1.3 Preloading files

```
// preload.js
console.log('preload.js: this is preloaded')

// app.js
console.log('app.js: this is the main file')

```

```
// CommonJS
node -r ./preload.js app.js
node --require ./preload.js app.js

// ES modules
node --loader ./preload.js app.js
```

1.4 Stack trace limit

By default, only the first 10 stack frames are shown, which can lead to the root cause of the error not being shown.

In this case, modify the V8 option –stack-trace-limit:

```
node --stack-trace-limit=20 file.js
```