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
```

2 Debugging and Diagnostics

Start node in debugging mode:

```
node --inspect file.js # runs immediately
node --inspect-brk file.js # breakpoint at start of program
```

3 Core JavaScript Concepts

3.1 Types

Everything besides the following primitive types is an object - functions and array, too,s are objects, for example.

```
1  // The null primitive is typically used to describe
2  the absence of an object...
3  // Null
4  null
5
6  // Undefined
7  // ... whereas undefined is the absence
8  of a defined value.
9  // Any variable initialized without a value will be undefined.
10 // Any expression that attempts access of a non-existent
11 property on an object will result in undefined.
12 // A function without a return statement will return undefined.
13 undefined
14
15 // Number
```

```
        // The Number type is double-precision floating-point format.
        // It allows both integers and decimals but
        // has an integer range of $-253^-1$ to $253^-1$.
        1, 1.5, -1e4, NaN

        // BigInt
        // The BigInt type has no upper/lower limit on integers.
        1n, 9007199254740993n

        // String
        'str', "str", `str ${var}`

        // Boolean
        true, false

        // Symbol
        // Symbols can be used as unique identifier keys in objects.
        The Symbol.for method creates/gets a global symbol.
        Symbol('description'), Symbol.for('namespace')...


```

An object is a set of key value pairs, where values can be any primitive type or an object (including functions, since functions are objects). Object keys are called properties.

All JavaScript objects have prototypes. A prototype is an implicit reference to another object that is queried in property lookups. If an object doesn't have a particular property, the object's prototype is checked for that property. and so on. This is how inheritance in JavaScript works.