### 1. The main advantage using setImmediate() over setTimeout()

Using `setImmediate()` over `setTimeout()` is `setImmediate()` will always be executed before any timers if scheduled within an I/O cycle, independently of how many timers are present.

If you run a script from the main module it depends on the performance of the process which also can be impacted by the other applications running on the machine, the following example demonstrates this

```
setTimeout(function() {
    console.log('setTimeout')
  }, 0)

setImmediate(function() {
    console.log('setImmediate')
})
res.end();
```

But if we execute the above code inside I/O callback the setImmediate() code will always runs before the setTimeout code

```
const fs = require('fs');

fs.readFile(__filename, () => {
    setTimeout(function() {
        console.log('setTimeout')
      }, 0)

    setImmediate(function() {
        console.log('setImmediate')
    })
    res.end();
})
```

2. **The difference between process.nextTick() and setImmediate()**

- process.nextTick() fires immediately on the same phase
- setImmediate() fires on the following iteration or 'tick' of the event loop

In essence, the names should be swapped. process.nextTick() fires more immediately than setImmediate().

3. **Node.js global modules**
    1. __dirname
    2. __filename
    3. clearImmidate(immidateObject)
    4. clearInterval(intervalObject)
    5. clearTimeout(timeoutObject)
    6. console
    7. exports
    8. global
    9. module
    10 process