Arquitectura de Software

Clase 2 - Javascript en ejecución

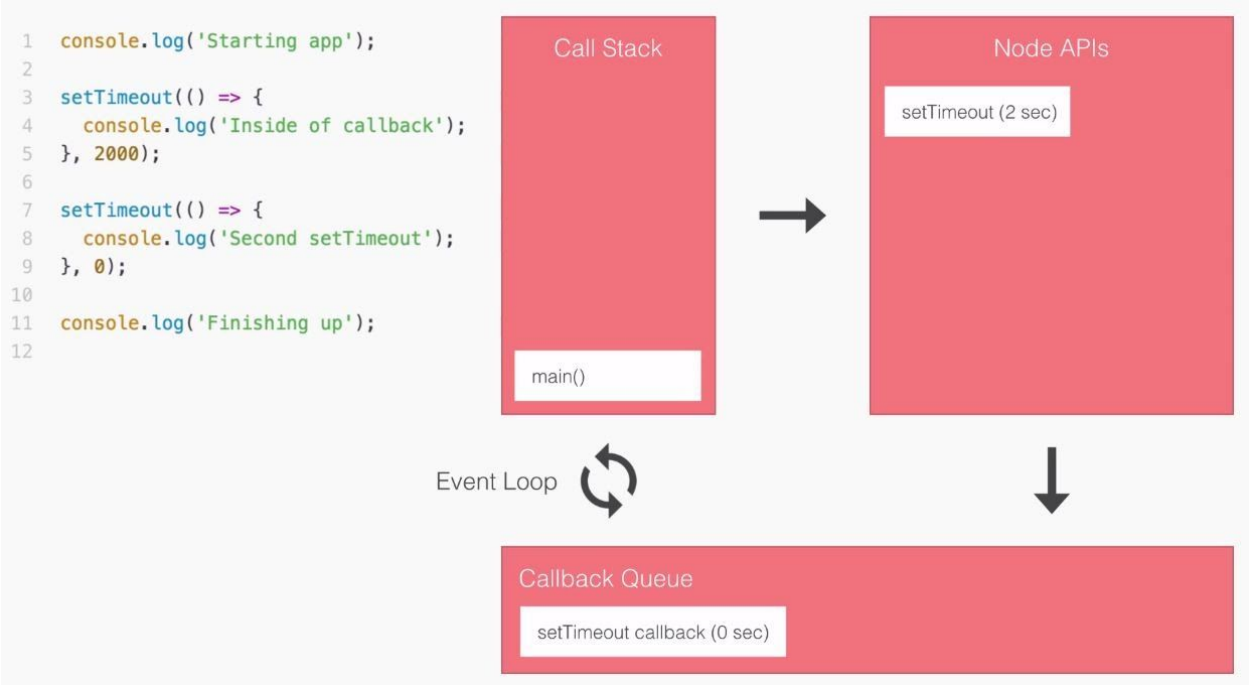
OBJETIVOS

- Javascript en runtime, call stack, event loop, blocking vs non-blocking
- Introducción a Node.js

INTRODUCCIÓN

Si bien Javascript es un lenguaje diseñado para ser ejecutado en los navegadores web hoy también es comúnmente utilizado del lado del servidor.

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google[1].



[2] Ejecución de código JS en runtime Node.js

1. https://es.wikipedia.org/wiki/Node.js
2. https://www.oreilly.com/library/view/learning-nodejs-development/9781788395540/dfdddbf7-1521-4c8e-af8e-e47900ae5002.xhtml

Una aplicación Node.js se ejecuta en un solo proceso, sin crear nuevos threads por cada request.

Node.js cuenta con un conjunto de de operaciones de I/O en su librería estándar que evita tener código JS bloqueante, de hecho las librerías escritas para Node.js tienen como norma no utilizar código bloqueante, ejecutar código con ese comportamiento es realmente una excepción.

Cuando Node.js tiene que hacer una operación de I/O (lectura desde la red, acceso a filesystem, etc.) en vez de bloquear el thread esperando por un resultado el mismo continuará con la ejecución y volverá a la operación cuando tenga un resultado. Este es el secreto que permite a Node.js manejar miles de conexiones concurrentes con un solo servidor de manera eficiente.

El objetivo de este práctico es entender cómo funciona Javascript en runtime. Se realizarán ejercicios en un simulador, en el navegador y también en Node.js

EJERCICIOS

Javascript en runtime

Call Stack

1. Call Stack
2. Stacktrace (Ejecutar también en navegador)
3. Uncaught RangeError: Maximum call stack size exceeded (Ejecutar también en navegador)

Event loop, blocking & non-blocking code

1. Blocking code
2. Non blocking code
3. Node.js: Blocking vs Non-Blocking

Lecturas recomendadas para profundizar en los temas anteriores:

- Overview of Blocking vs Non-Blocking
- Don't Block the Event Loop (or the Worker Pool)

Introducción a Node.js

- Node REPL (Read Evaluate Print Loop)
    - Evalúa en tiempo real expresiones
    - Permite probar rápidamente código
- Módulos
    - Conjunto de funciones que queremos incluir en nuestro código
    - Node cuenta con módulos incorporados
- NPM (Node Package Manager)
    - CLI
    - npm init
    - npm install <package> -s
- Primer servidor en Node.js
- Usando el framework Koa.js
    - Un primer ejemplo
    - Servicio para devolver la hora de un país
- Desafío: Implementar un API REST utilizando Koa.js (o Express.js) para el ABM de un recurso “user” con la siguiente información:

```
{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "sincere@april.biz",
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    “country”: “Neverland”
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  “timezone”: “America/Montevideo”
}
```