Debug NodeJS con VSCode

Gabriel Rodríguez Flores

October 26, 2021

ADA ITS
INSTITUTO TECNOLÓGICO SUPERIOR

- Set vscode for debug

Contents

1 Teoría

1.1 Debug con Chrome (Nativo NodeJS)

1. Insertar la instrucción `debugger` en el código

```
const chalk = require('chalk');
debugger;
console.log(chalk.blue('I\'m blue'));
```

2. Ejecutar node en modo debug con la instrucción `inspect`:

```
node --inspect index.js
```

> Nota: Dependiendo de la versión de NodeJS usada, en Windows puede dar un error con el puerto de escucha. Si esto ocurre, se podrá ejecutar el comando con la instrucción `inspect-brk`:
>
> ```
> node --inspect-brk index.js
> ```

3. Abrir en el navegador Chrome, la dirección `chrome://inspect` y poner algún punto de ruputa
4. El código se detenrá y podremos avanzar paso a paso mirando y pudiendo alterar el estado del programa.

1.2 Debug con VSCode

De la misma manera, VSCode integra un sistema de debug prácticamente idéntico, pero más automatizado.

- Referencia

1.2.1 Primera configuración (Básica)

1. Accedemos al panel de Debug, menu lateral o `Ctrl+Shift+D`
2. Si no hay debug configurado, permitirá ejecutar directamente y depurar el fichero actual. Si queremos configurarlo, crearemos un archivo dentro de la carpeta oculta vscode `.vscode/launch.json`
3. Seleccionando NodeJS cargará una configuración base para la ejecución del fichero que esté abierto en el editor.

```
{
  "version": "0.2.0",
```

```
  "configurations": [
    {
      "type": "pwa-node",
      "request": "launch",
      "name": "Launch Program",
      "skipFiles": [
        "<node_internals>/**"
      ],
      "program": "${file}"
    }
  ]
}
```

4. Como nos indican en los comentarios, podemos obtener más información en el siguiente enlace

1.2.2 Launch.json

- Variables

Ejemplo extendido:

1.2.3 Breakpoints y estados de variables

- Panel lateral de variables y distintos scopes
- Gestionar los puntos de interrupcion activando y desactivandolos individual o global

1.2.4 Panel de Inspección

1.2.5 Consola de depuración

2 Ejemplos

2.1 Configuración básica para un fichero fijo

```
{
  "version": "0.2.0",
  "configurations": [
    { // Configuración para arrancar la aplicación sin importar donde
       se está
      "type": "node",
      "request": "launch",
      "name": "Launch API",
      "outputCapture": "std",
```

```
      "skipFiles": [
        "<node_internals>/**"
      ],
      "program": "${workspaceFolder}/src/index.js",
      "cwd": "${workspaceFolder}"
    }
  ]
}
```

2.2 Configuración para ejercutar el fichero actual

```
{
  "version": "0.2.0",
  "configurations": [
    { // Configuración base para ejecutar el archivo abierto en el
        editor
      "type": "node",
      "request": "launch",
      "name": "Launch Current Opened File",
      "outputCapture": "std",
      "program": "${file}"
    }
  ]
}
```

2.3 Configuración con argumentos para los test

```
{
  "version": "0.2.0",
  "configurations": [
    { // Configuración para ejecutar el test (con AVA) abierto en el
        editor
      "type": "node",
      "request": "launch",
      "name": "Launch API Test",
      "cwd": "${workspaceFolder}",
      "runtimeExecutable": "${workspaceFolder}/node_modules/.bin/ava",
      "runtimeArgs": [
        // "debug",
        // "--break",
        "${file}"
      ],
      "port": 9229,
      "outputCapture": "std",
      "skipFiles": [
        "<node_internals>/**/*.js"
      ]
```

```
    }
  ]
}
```

2.4 Configuración extendida

```
{
  "version": "0.2.0",
  "configurations": [
    { // Configuración para arrancar la aplicación sin importar donde
        se está
      "type": "node",
      "request": "launch",
      "name": "Launch API",
      "outputCapture": "std",
      "skipFiles": [
        "<node_internals>/**"
      ],
      "program": "${workspaceFolder}/src/index.js",
      "cwd": "${workspaceFolder}"
    },
    { // Configuración para ejecutar el test (con AVA) abierto en el
        editor
      "type": "node",
      "request": "launch",
      "name": "Launch API Test",
      "cwd": "${workspaceFolder}",
      "runtimeExecutable": "${workspaceFolder}/node_modules/.bin/ava",
      "runtimeArgs": [
        // "debug",
        // "--break",
        "${file}"
      ],
      "port": 9229,
      "outputCapture": "std",
      "skipFiles": [
        "<node_internals>/**/*.js"
      ]
    },
    { // Configuración base para ejecutar el archivo abierto en el
        editor
      "type": "node",
      "request": "launch",
      "name": "Launch Current Opened File",
      "outputCapture": "std",
      "program": "${file}"
    }
  ]
}
```

3 Ejercicios

1. Usar la consola en medio de una ejecucíon
2. Añadir variables y expresiones en el panel de debug
3. Alterar el valor de una variable y continuar la ejecución
4. Deshabilitar los puntos de ruptura
5. Realizar una petición con node-fetch e inspeccionar la respuesta en el inspector
6. Extraer los datos de un objeto de una librería

- Propiedades
- Métodos
- Eventos

4 Entregables

4.1 En clase

Ejercicios del 1 al 6

4.2 Tarea

Crear un proyecto con dos archivos javascript (`index.js`, `app.js`) los cuales el primero importará el segundo. El proyecto contará con el fichero de configuración para depurar en vscode. Deberá tener al menos dos configuraciones:

1. Ejecutar siempre el `index.js`.
2. Ejercutar el archivo actual del editor.