Step Debugging JavaScript

with Visual Studio Code

https://where.matsinet.codes/presentations/step-debugging-js

How we all get started debugging

```
console.log('Hello World');
```

This is ok simple application logic
Advanced logic === more console.log statements
console.log output statements then become overwhelming

As complexity grows console becomes too verbose:

```
console.log('function begin', variable);
console.log('function end', variable);
console.group();
   console.log('another function begin', variable);
   console.log('another function end', variable);
   // Numerous other console.log calls
console.groupEnd();
```

Advantages & Disadvantages of console.log

- Easy (NO) setup
- Have to add code wherever you need output
- The output is only after execution has completed
- There is no way to update the variable values interactively
- There is no way to pause execution interactively
- Requires clean up code once debugging completed

What is Step Debugging

...is an interactive connection between the IDE and browser allowing the IDE to control the JavaScript execution within the browser.

Advantages & Disadvantages of step debugging

- Harder to setup initially
- Can evaluate variable values during execution

- Can update the variable values interactively
- Can pause execution interactively, using breakpoints
- Can debug compiled code referencing the source code such as TypeScript or Elm
- No clean up required

Getting VS Code ...ready to debug

Prerequisites

- Install Debugger for Chrome extension
- Restart VS Code
- Install http-server or live-server node application globally

Configuration

- Click Debug -> Add Configuration -> Chrome: Launch
- Alternately: Click Debug -> Open Configuration -> Chrome: Launch

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "chrome",
      "request": "launch",
      "name": "Launch Chrome against localhost",
      "url": "http://localhost:8080",
      "webRoot": "${workspaceFolder}"
    }
  ]
}
```

- Restart VS Code

Where is the configuration stored?

By default, the launch configuration is stored in project
in the .vscode folder

The launch configuration can be stored across projects (globally) in the settings.json under the "launch" key.

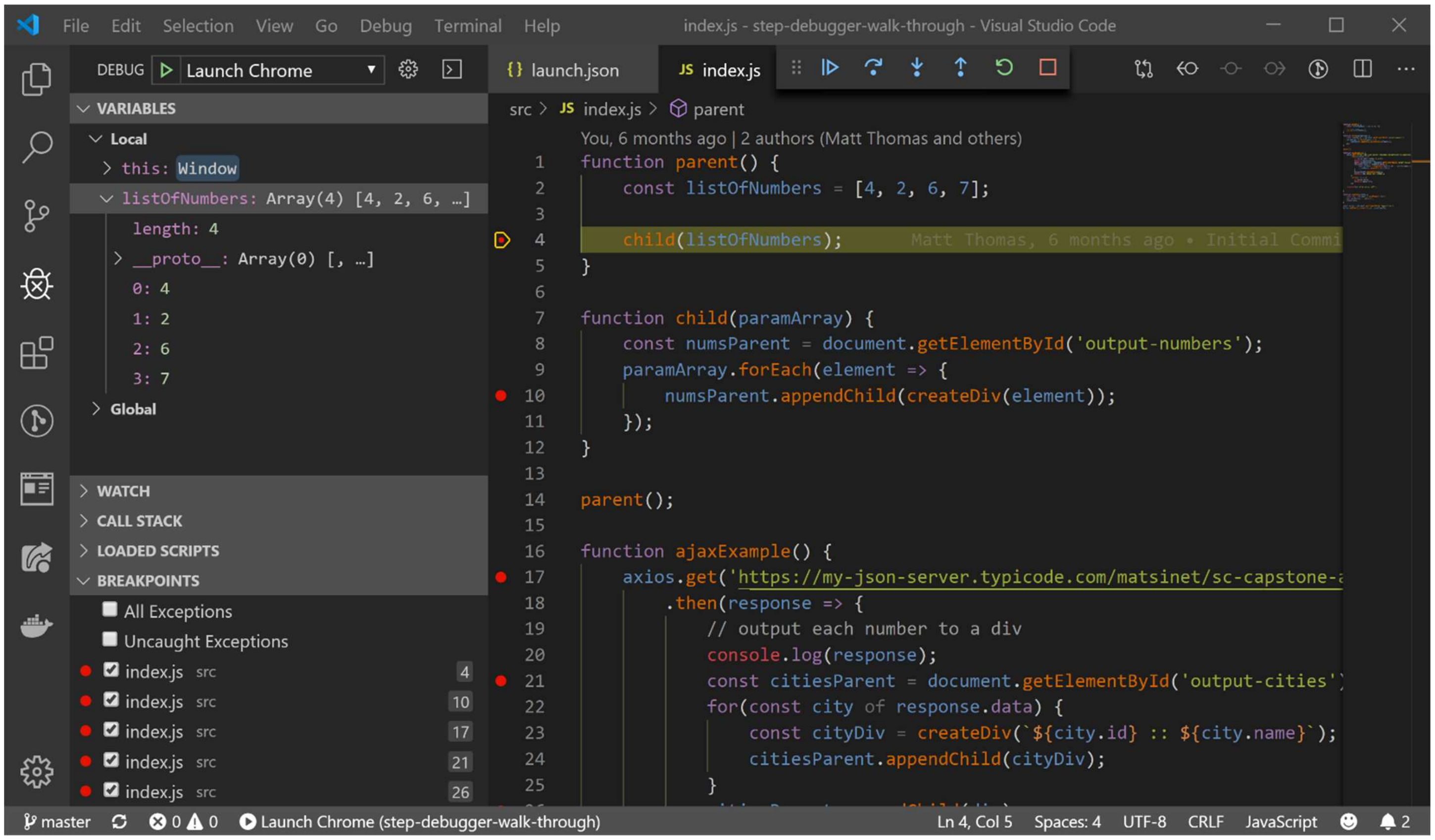
Sample Parcel Configuration

```
{
   "type": "chrome",
   "request": "launch",
   "name": "Launch Chrome w/Parcel",
   "url": "http://localhost:1234",
   "webRoot": "${workspaceFolder}",
   "trace": true,
   "breakOnLoad": true,
   "sourceMapPathOverrides": {
      "*": "${webroot}/src/*"
   }
}
```

Ready the code base ...source code and server

- Open a terminal in the root source code folder
- Serve your code via [http-server (https://www.npmjs.com/package/http-server)](https://www.npmjs.com/package/http-server) or [live-server (https://www.npmjs.com/package/live-server)](https://www.npmjs.com/package/live-server)

Start Debugging

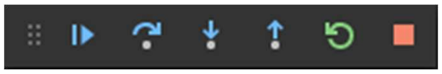

File Edit Selection View Go Debug Terminal Help
index.js - step-debugger-walk-through - Visual Studio Code
DEBUG
Launch Chrome
launch.json
index.js
src > index.js > parent
You, 6 months ago | 2 authors (Matt Thomas and others)
Matt Thomas, 6 months ago • Initial Commi
VARIABLES
WATCH
CALL STACK
LOADED SCRIPTS
BREAKPOINTS
master
0 0
Launch Chrome (step-debugger-walk-through)
Ln 4, Col 5 Spaces: 4 UTF-8 CRLF JavaScript
2

```
Local
  > this: Window
  v listOfNumbers: Array(4) [4, 2, 6, …]
      length: 4
    > __proto__: Array(0) [, …]
      0: 4
      1: 2
      2: 6
      3: 7
> Global
```

```
All Exceptions
Uncaught Exceptions
index.js  src   4
index.js  src  10
index.js  src  17
index.js  src  21
index.js  src  26
```

```
function parent() {
    const listOfNumbers = [4, 2, 6, 7];

    child(listOfNumbers);
}

function child(paramArray) {
    const numsParent = document.getElementById('output-numbers');
    paramArray.forEach(element => {
        numsParent.appendChild(createDiv(element));
    });
}

parent();

function ajaxExample() {
    axios.get('https://my-json-server.typicode.com/matsinet/sc-capstone-a
        .then(response => {
            // output each number to a div
            console.log(response);
            const citiesParent = document.getElementById('output-cities')
            for(const city of response.data) {
                const cityDiv = createDiv(`${city.id} :: ${city.name}`);
                citiesParent.appendChild(cityDiv);
            }
```

Debugger Operations



Pause/Continue -
Pause or continue the execution of the code
will stop at the next breakpoint

Step Over -
Execute the current line without introspection

Step Into -
Execute the current line, "diving" into any scopes (functions)

Step Out Of -
Execute the code "climbing out" of the current scope

Restart -
Reset the debugging session, including refreshing the browser

Stop -
End the debugging session, code execution will continue

Demo

Sample code, instructions and configuration
https://github.com/matsinet/step-debugger-walk-through

Where to go from here...

Documentation: Visual Studio Code Manual : Debugging

Video Tutorial: Debugging JavaScript (Google Chrome and Visual Studio Code) (Thanks Bradley)

Feedback...

- Matt Thomas
- Twitter: @matsinet
- Website: https://where.matsinet.codes
- Slides: https://where.matsinet.codes/presentations/step-debugging-js
- Created with: Reveal.js
- Step Debugging JS with VS Code https://where.matsinet.codes/presentations/step-debugging-js