Debugging made easier

Per Enström

Do you recognize this way of debugging?

```javascript
console.log("BEFORE");
const array = [1,2,3,4,5];

for (let index = 0; index < array.length; index++) {
  console.log("ARE WE HERE?");
  console.log(index);
  const element = array[index];
  console.log(element);
}

console.log("ASDFASDF");
```

Use the browser or IDE debugger instead!

Trigger the debug

```javascript
const array = [1,2,3,4,5];

for (let index = 0; index < array.length; index++) {
  debugger;
  const element = array[index];
}
```

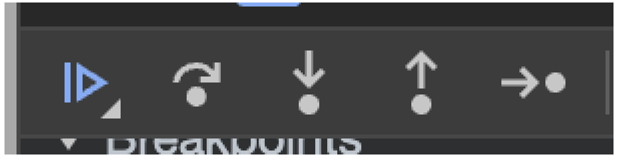
This will open the debugger in your browser if the code is running in a browser, or in your IDE if you've started the code from there.

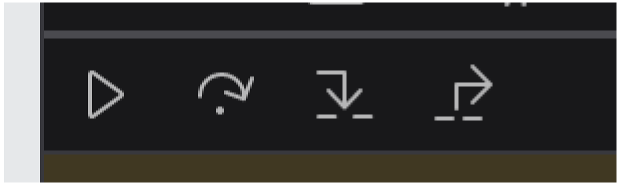
The debugging interface

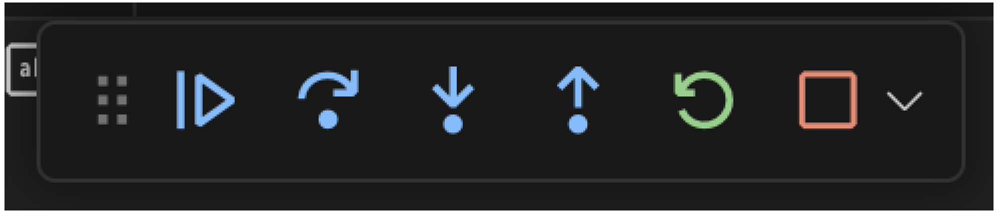
Live demo time!

demo-1.js

Debugging controls

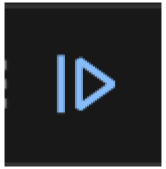






Debugging controls

Play



- Continues the normal running of the code, until the next breakpoint, where it will stop again.
- Demo!

demo-1.js

Debugging controls

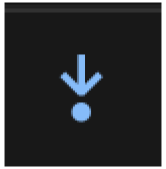
Step over

- Executes and steps to the next expression in the current scope (i.e. same function)
- This is the most common way to step through your code
- Demo!

demo-2.js

Debugging controls

Step into



- Steps one level deeper into the code, i.e. into functions being called
- Demo!

demo-2.js

Debugging controls

Step out of

- Executes the full current scopes and pauses when it’s returned to the calling scope.
- A way to get out of functions you’re not interested in looking at
- Demo!

demo-3.js

Breakpoints

- We’ve already seen the use of `` `debugger` ``
- Breakpoints can be set directly in the debugger as well
- Demo!

demo-4.js

Conditional breakpoints

- Some cases you don’t want your code to stop every time a certain line is evaluated
- You can set conditions on the breakpoints to tell the debugger when to stop
- Demo!

demo-4.js

Watchers

- Instead of inspecting variables
- Evaluates continuously
- Good for checking calculated properties
- Demo!

demo-4.js

Log points

- Instead of writing `` `console.log` `` to print stuff to the console, or checking the variables in scope in the debugger

- Lets you log whatever you please, without changing your source code

- Demo!

demo-5.js

Debugging client side code

- Starting single js scripts from VS Code is rarely what we do
- Either in the browser or in IDE
- A ``` `launch.json` ``` is needed for VS Code to know what to do
- Most frameworks have documentation on how these should look
- Demo!

App.vue

Adding breakpoints in the browser

- Instead of adding a `` `debugger` `` statement
- Code can be found in the "Sources" tab of the dev tools (or under the "Debugging" tab in Firefox)
- A bit of a mess to navigate, if everything is set up correctly it should map to your actual source files
- Demo!

Thank you for listening

Resources

- VS Code Debugging Docs – code.visualstudio.com/docs/editor/debugging
- These slides (made with slidev) – per.fyi/talks