Debugging in JS

Duvvuri Naga Sai Aparna
ENG18CS0096
VII sem CSE B

Debugging is not easy. But fortunately, all modern browsers have a built-in JavaScript debugger. Built-in debuggers can be turned on and off, forcing errors to be reported to the user.

There are mainly 3 methods in debugging JS:
1. The console.log() method
2. Setting breakpoints
3. The debugger keyword

1.Console.log() method:
The console.log() is a function in JavaScript which is used to print any kind of variables defined
before in it or to just print any message that needs to be displayed to the user.

Syntax:

```
console.log(A);
```

2.Setting Breakpoints
In the debugger window, you can set breakpoints in the JavaScript code.
At each breakpoint, JavaScript will stop executing, and let you examine JavaScript values.
After examining values, you can resume the execution of code (typically with a play button).

3.The debugger Keyword
The debugger keyword stops the execution of JavaScript, and calls (if available) the debugging
function.
This has the same function as setting a breakpoint in the debugger.
If no debugging is available, the debugger statement has no effect.
With the debugger turned on, this code will stop executing before it executes the third line.

Syntax:

```
Debugger;
```

MVC

The model-view-controller pattern of designing software (also known as MVC) was one of the first architectural design-patterns based on the responsibilities of its component software constructs. Trygve Reenskaug introduced the concept of MVC while visiting Xerox Parc (the same research facility that pioneered everything from GUIs to laser printers) in the 1970s. MVC was revolutionary then and is still a key concept when discussing how to create new software. When using an MVC pattern, you can lay out the solution to any problem you expect to solve with the software you intend to build. The three components included by name in the pattern are:

The model, which describes the behavior of the application regarding data, logic, and rules. The view, which includes any information that the application may output (from tabular data to representations)and the controller, which accepts and translates input.

Note that the model, which is the core component on which you will wind up focusing, is independent of any user interface you are planning to create eventually.

The Continuing Evolution of MVC and Isomorphic JavaScript:
While the concept of MVC first evolved for desktop applications, the pattern has been adapted as a key approach for web-based applications. Even today, many common MVC frameworks, such as Ruby on Rails, put most of the burden of the model, view, and controller logic on the server.
However, modern web applications have evolved to take advantage of maturing client technologies, allowing new frameworks to handle more of the MVC logic on the client. The the result is isomorphic JavaScript, an approach that enables you to use MVC patterns across both the client and the server. Not all Node.js frameworks rely on MVC as a design pattern, nor is doing so necessary to implement isomorphic JavaScript. However, using a familiar design pattern makes for an easier entry into these concepts.