Soorten errors en debuggen

Soorten fouten

Er zijn drie categorieën fouten die we in de code kunnen genereren: syntax errors, runtime errors en logical errors.

Syntax errors

Fouten in de opbouw van de taal (syntax errors). Deze komen er direct bij het laden van de pagina uit en zijn direct zichtbaar in de console. Een goed voorbeeld is:

```
alert("Hi); // Uncaught SyntaxError: Invalid or unexpected token
```

Runtime errors

Fouten in de benaming van objecten (runtime errors). Deze komen er pas uit als het script uitgevoerd wordt en er naar een niet bestaand object wordt verwezen. Een voorbeeld is een ReferenceError:

```
let myNumber = 42;

mynumber ++; // Uncaught ReferenceError: mynumber is not defined
```

Logical errors

Logische fouten zijn het moeilijkst te debuggen omdat er feitelijk technisch gezien niets mis is, de taal is correct toegepast en de verwijzingen en benamingen kloppen allemaal. Een goed voorbeeld van een logische fout is concatenation in plaats van optellen. Maar er zijn veel complexere voorbeelden, vooral als we data van een web API ophalen en er moet iets met die data gebeuren. Als we geen rekening houden met het asynchrone gedrag van data die als promise binnenkomt (via de fetch() API, waarover later meer), kan het maar zo zijn dat we wel data verwachten te zien maar die toch nooit te zien krijgen omdat de functie die iets met die data moet doen al uitgevoerd is voordat de data daadwerkelijk binnengekomen is. Door `alert()` en `console.log()` statements op verschillende plekken in de code te plaatsen en breekpunten met `debugger;` kunnen we de logische fouten steeds beter isoleren en bepalen wáár het nu precies misgaat.

De volgende tekst is in het Engels omdat dit mogelijk essentiële informatie bevat voor de examenvragen:

Steps for debugging JavaScript code

The debugging process begins after you have written your script code, placed it in (or attached it to) an HTML file, run the file, and discovered that it has some errors. However, many developers forget that testing their code is really the first step in debugging it. You should take the following steps to debug your code:

- Test the code: You should run every script to see if it works as intended. It is necessary to test each script in multiple browsers, including various brands of browsers and various versions of each brand. It is also helpful to test on a variety of hardware and software platforms because all these factors can affect the way your script renders to the end-user.
- Identify the problem: What exactly is wrong with the script? Does the script run but render an unexpected result? Does the script not run at all? Does the script work as expected in one browser but not another? Which browsers work with the script and which do not? Sometimes the problem gives you a clue to the type of error you have.
- Locate the error: Some debugging tools can help with this by specifying a line number in the code for you to review. However, a debugger might not catch every error in a script. For example, some error types (such as logic errors) are not always detected by debuggers. Or the debugger might stop at the first error it finds, instead of running the script to the end and locating further errors.
- Review the code: It is always wise to review your code manually, but manual reviews do not always catch errors and with long programs, they can be time-consuming. Debugging tools, such as the Debugger in the Firefox Developer Tools and the Sources panel in the Chrome Developer Tools, can help but you should still review your code line-by-line after using the debugger. Even if you find no other errors, you are reinforcing your knowledge of the language by reading and understanding the code.
- Determine the error type: A debugger might help you find the location of an error but does not necessarily specify the problem. The type of error dictates the required repair. A syntax error might require only a quick addition or deletion of a code snippet, a command, some punctuation or a single letter. A logic error may require you to reorder the execution of a large portion of the program.
- Determine how to fix the problem: You may know how to correct the code. Or you may need to do some research to learn how to achieve the effect you want if your first attempt did not work.
- Correct the code: Implement the corrections you deem necessary, and any other changes you decide to make. Sometimes an error leads you to reconsider the effect that you want to produce on the page. Further modification might be in order. It is best to change only one or two things at a time before testing your changes. Otherwise, you might create new problems in your code while attempting to fix old ones.
- Review the code again: Even though you have corrected the known error, it is worthwhile to carefully review all your code manually at this point. Making a change in one location sometimes necessitates a related change or changes elsewhere. While you are reviewing, watch for other errors.
- Test your code again, and again: After every round of corrections, test your code again, using various browsers. Sometimes you will find an error that you had not noticed because the previous error stopped the program prematurely. You might even introduce a new error while correcting another.
- Repeat as needed: It is not unusual for programs to require several rounds of debugging, especially longer programs. Be prepared to spend as much time debugging as it takes to clean up your script. The debugging process is not finished until your code runs as expected when tested repeatedly in a variety of browsers and platforms.