< Basis1B>

Module 5: Javascript error handling (voor de student)

Inhoudsopgave

Introductie

In deze module ga je dieper in op het onderwerp foutafhandeling in Javascript. Je leert hoe foutmeldingen kunnen worden gegenereerd en hoe ze moeten worden gelezen. Tevens wordt er aandacht besteed aan het oplossen van fouten en aan de ‘strict mode’.

De leerdoelen voor deze module zijn;
De student kan:

- foutmeldingen in de Javascript console lezen en interpreteren
- systematisch fouten opsporen in code
- zelf foutmeldingen genereren
- gebruik maken van try/catch
- de strict mode gebruiken

Tijdsplanning

Week 1			
BASIS1B, les 1	2 uur	Console / zelfstandig werken	Oefening + opdracht
BASIS1B, les 2	2 uur	Try/catch, use strict en stack trace / zelfstandig werken	Oefening + opdracht
Week 2			
BASIS1B, les 3	2 uur	Debuggen van bestaande code	Oefening + opdracht
BASIS1B, les 4	2 uur	Debuggen van bestaande code	Eindopdracht

Inhoud

Les 1 - Console

//theorie toevoegen + tutorials (van-tot) als oefening

Console

Hoe lees je de console? Waar kan je deze vinden? Hoe genereer je je eigen consoleberichten en waar moet je naar kijken als de console fouten aangeeft die je niet op kan lossen?

Veel internetbrowsers hebben tegenwoordig een eigen console. In sommige gevallen moet je hier een plug-in voor downloaden en installeren. Voor deze module gaan we uit van het gebruik van Google Chrome. Indien jij graag met een andere webbrowser wilt werken is dat prima, je moet dan echter wel zelf uitzoeken hoe je de console daarmee kan bereiken.

In Chrome kan je de console vinden door op F12 te drukken. Een andere manier is door de rechtermuisknop te klikken in het venster en dan naar “inspecteren” te gaan.

Met de console zijn veel verschillende acties uit te voeren en kun je veel informatie ophalen. Voor het gebruik van Javascript is de console heel belangrijk. Als de developer een fout heeft gemaakt, geeft de programmeertaal de fouten aan in deze console. De developer kan ook zelf meldingen en checks in de console aanmaken om te kijken of de code het gewenste resultaat geeft. Het is zelfs mogelijk, om in de console zelf, stukken code te schrijven en te praten tegen de javascript die bij de website hoort.

OEFENING
Ga naar de onderstaande links en volg de stappen die daar beschreven zijn.
// chrome console
https://developers.google.com/web/tools/chrome-devtools/console/log

//console javascript
https://developers.google.com/web/tools/chrome-devtools/console/javascript

//debugging in dev-tools
https://developers.google.com/web/tools/chrome-devtools/javascript

Opdracht 1:

Je gaat aan de slag om in de console een klein spelletje te maken. D.m.v. een aantal knoppen op de webpagina met onclick events kunnen er in de console schadepunten uitgedeeld worden aan een “eindbaas”. Iedere actie maakt een nieuwe regel aan in de console waarin wordt vermeld wat er is gebeurd. Zodra de eindbaas 0 levenspunten heeft, zal er een nieuw event aangeroepen worden dat vertelt dat de baas verslagen is.

Les 2 – Try/catch, use strict en stacktrace

// Try/catch

Try en Catch is wat er met het woord al wordt gezegd: een zogenoemde "Exception handler". Als je ergens uit de code loopt door een mogelijke 'division by zero' o.i.d. kun je met catch de exception verder afhandelen met een sprong of een bericht of iets anders. Doe je dit niet dan krijg je buiten je routine een error die je niet kan afvangen. Het is netjes om met catch te werken als je de zwakke plekken in je code wilt afdekken zonder buggy te worden. Hieronder kan je de syntax vinden en een uitgebreid voorbeeld.

Syntax

```
try {
  //  Block of code to try
}
catch(Exception e) {
  //  Block of code to handle errors
}
```

Voorbeeld van een try/catch methode:

```
<!DOCTYPE html>
<html>
<body>

<p>Voer een nummer in tussen 5 en 10</p>

<input id="demo" type="text">
<button type="button" onclick="myFunction()">Probeer nummer</button>
<p id="p01"></p>

<script>
function myFunction() {
  var message, x;
  message = document.getElementById("p01");
  message.innerHTML = "";
  x = document.getElementById("demo").value;
  try {
    if(x == "")  throw "empty";
    if(isNaN(x)) throw "not a number";
    x = Number(x);
    if(x < 5)  throw "too low";
    if(x > 10)   throw "too high";
  }
  catch(err) {
    message.innerHTML = "Input is " + err;
  }
}
</script>

</body>
</html>
```

Extra uitleg over try/catch en error handling in het algemeen.
https://www.w3schools.com/js/js_errors.asp

// Use-strict

Use strict, ook wel strict mode genoemd, is een manier om ervoor te zorgen dat Javascript harder optreedt tegen slordigheden in de code. Waar Javascript een loosely-typed taal is, waar je variabelen kan aanmaken zonder te definiëren wat voor soort datatype de variabele heeft, zal dat met strict mode een belangrijk iets zijn om mee rekening te houden. Strict mode is te gebruiken door bovenaan de javascript de regel "use strict"; te plaatsen. Hierdoor zal alle javascript sneller foutmeldingen geven, waardoor het in sommige gevallen makkelijker zal zijn om fouten in de code op te sporen en op te lossen.

Hier nog een extra link voor meer uitleg over strict mode.
https://www.w3schools.com/js/js_strict.asp

Kijk de video hieronder voor een uitgebreide uitleg over use-strict en het gebruik daarvan.
Strict Mode — "use strict" - Beau teaches JavaScript

//stack trace

Wat is een call stack?

In Javascript wordt gebruik gemaakt van een call stack. Wat is een call stack eigenlijk? We gaan dit bekijken door het eerst te hebben over een aparte stack. Een stack, ook wel een stapel, wordt vaker gebruikt binnen programmeertalen. In ons geval gaat het over het afwerken van de code in een bepaalde volgorde. De volgorde die gebruikt wordt met een stack is LIFO – Last in, First out.

Beeld je een stapel boeken in. Als je een boek wilt toevoegen aan de stapel leg je deze er bovenop. We gaan niet moeilijk doen door het boek onderop de stapel te leggen of ergens in het midden te stoppen. Hetzelfde gebeurt als we een boek van de stapel af willen halen. We pakken dan eerst het boek wat het makkelijkst te pakken is; het bovenste boek.

Het tweede woord van call stack is "call", in het Nederlands ook wel "aanroepen". Binnen Javascript, als we een nieuwe functie willen gebruiken, noemen we dat ook wel het

aanroepen van een functie. De call stack is de volgorde waarin Javascript jouw functies afhandelt.

Bekijk de video hieronder voor meer informatie over een call stack
https://www.youtube.com/watch?v=W8AeMrVtFLY

// Gebruik maken van console.trace();
Binnen Javascript kunnen developers veel input zelf in de console plaatsen. Een van deze dingen is het aanroepen van de stacktrace. Door de functie console.trace() aan te roepen binnen een functie, kan de developer in de console zien door welke functies heen is gelopen om bij de console.trace() terecht te komen. Hier kan de developer nagaan of de volgorde van afhandelen wel goed is verlopen binnen zijn/haar code.

Bekijk de video hieronder voor het gebruik van console.trace();
https://www.youtube.com/watch?v=QuO0UDkW2rk

Opdracht 2:

De studenten gaan het spel van les 1 uitbouwen door een try/catch te implementeren en gebruik te maken van “strict mode”. Hierbij worden ook de benodigde aanpassingen binnen de code gemaakt om het spel weer werkend te krijgen. In de try/catch wordt ook de stacktrace meegenomen om te laten zien wat de error is die aangemaakt wordt door Javascript.
(Indien je moeite hebt met de try/catch, kan de docent een voorbeeld geven. Wanneer zal een try/catch doorgaan en wat gebeurt er als het niet lukt?)

Les 3 – Debuggen van een bestaande code

Eindopdracht

Om de module af te sluiten, ga je aan de slag met een eindopdracht. Op digitale lesstof staat een zip met een stuk Javascriptcode. Deze code is gebaseerd op de game cookie clicker. Als je deze code test, zie je dat er niks werkt. Het is aan jou de taak om deze code op te lossen, zodat je een werkende cookie clicker game kan spelen in de browser. Per werkende oplossing zijn er punten te verdienen. Al deze bug-oplossingen zijn een 7 waard. Wil je een hoger cijfer? Dan moet je de game uitbreiden met extra functionaliteiten.

Ga aan de slag met de geleerde statements zoals de try/catch, strict-mode (verplicht), throw en error-logging. Indien een fout is opgelost door enkel de code aan te passen, zal dit **niet** als een juiste oplossing meetellen.

Les 4 – Follow up les 3 > Debuggen van bestaande code

Verder met eindopdracht

THEORIE

https://developers.google.com/web/tools/chrome-devtools/console/javascript

https://www.w3schools.com/java/java_try_catch.asp#:~:text=Java%20try%20and%20catch,occurs%20in%20the%20try%20block.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch

https://www.w3schools.com/js/js_errors.asp#:~:text=JavaScript%20Throws%20Errors,two%20properties%3A%20name%20and%20message.

Voorbeeld 1

EXTERNE BRONNEN

Google “Error handling Javascript”
Google “Try & Catch Javascript”

Beoordeling

De leerdoelen voor deze module zijn;

De student kan:

- foutmeldingen in de Javascript console lezen en interpreteren
- systematisch fouten opsporen in code
- zelf foutmeldingen genereren
- gebruik maken van try/catch
- de strict mode gebruiken

De werkprocessen zijn:

- < B1-K1-W3 >
- < B1-K1-W4 >

De werkprocessen worden individueel beoordeeld.

Op de volgende pagina staat het gebruikte beoordelingsformulier afgedrukt

Module	THEMA <nummer
Naam beoordelaar	
Naam student	
Datum	
Leerdoelen:	O/V
Je kan een foutmelding herkennen (opdr1)	
Je kan een foutmelding genereren (opdr1)	
Je kan een foutmelding oplossen (opdr2/eindopdr)	
Je kan gebruik maken van de console (opdr1)	
Je kan programmeerstructuren toepassen die foutcodes afhandelen (opdr2)	
Je kan een try/catch en throw implementeren en de strictmode toepassen (opdr2/eindopdr)	
Opmerkingen:	

Normering:	**Score:**
Leerdoelen voldoende	7
Extra onderdelen	1,5 (maximaal)
Vormgeving van het ingeleverde document	1 (maximaal)
Bij uitzonderlijk goed werk	0,5 (maximaal)
Bij te laat inleveren	*Maximum cijfer: 6*
Niets inleveren	*Cijfer: 1*
Cijfer:	

Werkprocessen:	O/V/G
< B1-K1-W3 >	
- indicator 1	
- indicator 2	
< B1-K1-W4 >	
- indicator 1	
- indicator 2	
Feedback:	