Documentatie: project JavaScript

Jay Jansen

Jens Hollenberg

Niels Lute

Maritiem College IJmuiden – Nova College Beverwijk

Inhoudsopgave

# **Het doel**

Het doel van dit project is om een browsergame te maken waarbij de gebruiker (ofwel de speler) containers op een boot moet slepen en zetten, zodanig dat alle containers netjes op het schip passen. Je kunt dit dus zien als een soort paspuzzel.

**TODO: Dit project is een onderdeel van een groter geheel.**

# **Middelen**

Zoals aan het begin staat vermeld is het de bedoeling dat dit spel een browserspel wordt, wat betekent dat dit spel te spelen moet zijn op een webpagina. Er zal dus gebruik worden gemaakt van programmeertalen die voornamelijk worden gebruikt bij het in elkaar zetten van een webpagina, deze zijnde HTML, CSS en JavaScript (en een klein beetje PHP).

## **Talen en modules**

### - HTML

HTML (afkorting voor: **H**yper**T**ext **M**arkup **L**anguage) is, zoals de naam aanduidt, een markup-taal en geen werkelijke programmeertaal. “Markup” betekent letterlijk vertaald “opmaak”, dus HTML is als het ware een opmaaktaal.

HTML zal worden gebruikt om het basisontwerp van de webpagina de definiëren. De indeling van welke elementen zich waar bevinden zal dus in deze taal worden geschreven.

### - CSS

CSS (afkorting voor: **C**ascading **S**tyle**S**heets) is een taal die als het ware als een extensie van de HTML kan worden gezien. Over het algemeen wordt CSS gebruikt om extra opmaak en vormgeving aan de webpagina te geven, en er voor te zorgen dat deze er goed en presenteerbaar uitziet.

### - JavaScript

JavaScript is de eerste daadwerkelijke programmeertaal in dit rijtje. Deze taal zal worden gebruikt om het overgrote meerderdeel aan de webpagina toe te brengen, zodat het spel ook daadwerkelijke functioneert.

JavaScript kan worden gebruikt zowel op het vlak van Frontend (wat de gebruiker ziet) als Backend (wat er allemaal achter de schermen gebeurt; serververbindingen, databases, enzovoort). Voor dit project zal JavaScript voor het grootste gedeelte zich focussen op het Frontend-aspect.

Ook zal er gebruik worden gemaakt van jQuery, wat een library (een extensie) is die de functionaliteit van JavaScript sterk versoepelt en extra functionaliteit toevoegt[1]. Ook zal er gebruik worden gemaakt van jQuery-ui, wat staat voor jQuery User Interface. Dit zorgt voor nog meer functionaliteit en veel versoepelingen op het vlak van Frontend-development[2]. jQuery zelf focust meer op de Backend.

### - PHP

PHP (Hypertext Preprocessor) is een programmeertaal die wordt uitgevoerd aan de kant van de server. De voorgaande talen zijn talen die worden uitgevoerd aan de kant van de client (de gebruiker), echter zorgt PHP ervoor dat de server en de client verbonden zijn. Er zal een klein beetje PHP worden gebruikt bij de voorbereiding en in het project zelf, maar het meerderdeel van de geschreven code zal bestaan uit de drie voorgaande beschreven talen.

## **IDE’s**

Om zo efficiënt mogelijk te kunnen programmeren, is er voorkeur aan software die gebruik maakt van markup (automatisch bepaalde functionaliteit binnen een programmeertaal opmaken), automatisch de regels laat inspringen en helpt bij het zo efficiënt mogelijk programmeren. Programma’s zoals Notepad en Notepad++ vallen dan af.

Software die aan de bovenstaande eisen voldoen zijn IDE’s, wat staat voor **I**ntegrated **D**evelopment **E**nvironment (geïntegreerde ontwikkelingsomgeving). Dit zijn programma’s die speciaal ontworpen zijn om softwareontwikkelaars te ondersteunen bij het schrijven van computersoftware.

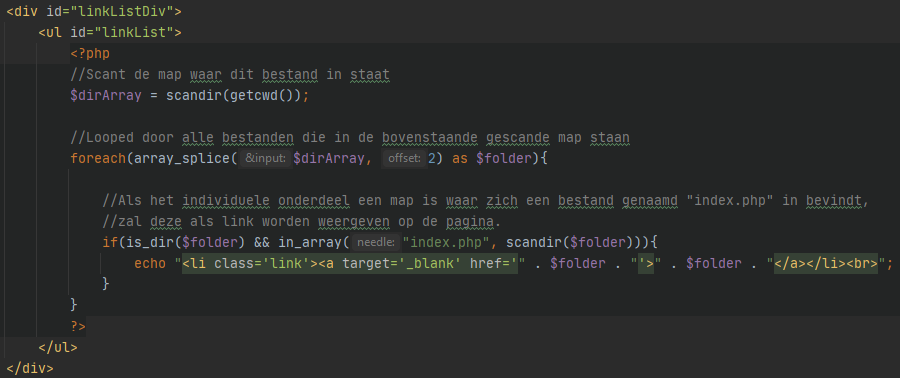
Voor dit project zullen de PhpStorm en de Rider IDE’s worden gebruikt, beide afkomstig van de softwareontwikkelaar “JetBrains”[3]. Tussen deze twee programma’s zit vrij weinig verschil, en welke er wordt gebruikt hangt puur af van onze individuele voorkeur, in plaats van dat de één beter is dan de ander.

## **Zelfgemaakte cataloguspagina**

Dit project zal worden opgebouwd in verschillende stappen, bestaande uit verschillende onderdelen. Om een goed overzicht te krijgen en te behouden over al deze individuele onderdelen en stappen, hebben wij een kleine catalogus/verzamelingspagina geschreven door middel van HTML en PHP (en een klein beetje CSS).

De PHP-code checkt de map waar alle projectbestanden (en het bestand zelf) instaat, en controleert of er zich een “index.php”-bestand in elke map bevindt. Zo ja, wordt de map weergeven op een webpagina in een menu.

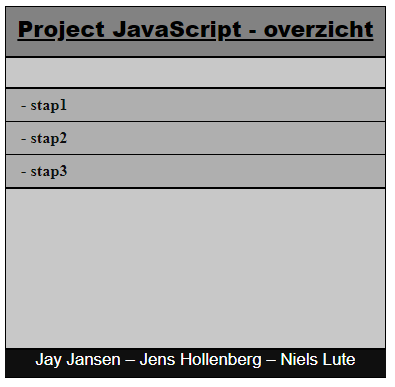
De code ziet er als volgt uit:



Er staat nog wat HTML omheen en er wordt, zoals eerder genoemd, wat CSS gebruikt om de pagina wat op te maken. Dit is niet relevant en zal dus buiten wege worden gelaten.

***Het resultaat is te zien op de volgende pagina.***

De cataloguspagina ziet er als volgt uit (uiteraard zijn op het moment van het schrijven van dit onderdeel nog niet alle onderdelen aanwezig):



## **Downloaden en importeren van benodigde (JavaScript) modules**

Onder het kopje “JavaScript” onder het onderdeel “Talen en modules” is aangegeven dat er gebruik zal worden gemaakt van twee JavaScript libraries, deze zijnde jQuery en jQuery-ui.

Deze worden respectievelijk van de volgende twee websites gedownload:

- jQuery: <https://jquery.com/>

- jQuery-ui: <https://jqueryui.com/>

jQuery is één JavaScript bestand. Dit is eenvoudig in te laden in HTML, in het “head”-gedeelte:

<script src="jQuery/jquery-3.3.1.min.js"></script> <!--jQuery-->

In dit geval staat het jQuery-JavaScript bestand (vanuit de map waar het bestand in staat waar de bovenstaande regel code in staat) in de map jQuery.

Hetzelfde geldt voor het jQuery-ui bestand. Een klein verschil met het bovenstaande hiermee is dat jQuery-ui uit meerdere bestanden bestaat; er staan namelijk ook een reeks CSS-functionaliteit en thema’s in. Echter is voor ons het JavaScript-bestand het enige wat van belang is:

<script src="jquery\_ui/jquery-ui.js"></script> <!--jQuery JS-->

Vanuit de map waar het bestand waar de bovenstaande regel code in staat, staat het jQuery-ui bestand in een map genaamd “jquery\_ui”.

Wanneer we nu ons eigen JavaScript-bestand oproepen vanuit het HTML-gedeelte, kan er gebruik worden gemaakt van de functionaliteit die in de gelinkte libraries staan.

# **Onderdelen webpagina**

Het uiteindelijke spel is opgebouwd uit verschillende elementen. In totaal zijn er drie kernelementen, en een aantal optionele elementen:

* **Kernelement 1:**

Het speelgebied (het geheel waar de beweegbare elementen in kunnen worden bewogen);

* **Kernelement 2:**

De beweegbare elementen (ofwel de containers);

* **Kernelement 3:**

Het gebied waar alle containers uiteindelijk in moeten (ofwel het vrachtschip);

* **Optioneel element 1:**

*Een knop om het spel opnieuw te kunnen spelen;*

* **Optioneel element 2:**

*Een timer die laat zien hoe snel je de puzzel hebt opgelost, en/of en timer die een tijdslimiet bijhoudt. Hiermee kan je het spel bijvoorbeeld zodanig maken dat de speler de puzzel binnen een bepaalde tijd moet oplossen;*

* **Optioneel element 3:**

*Aan de hand van de bovenstaande timer kan er een database worden gebruikt die de highscores van gebruikers bijhoudt.*

# **Het stappenplan**

Net als bij elk ander project, is de beste aanpak om het eindresultaat stap-voor-stap op te bouwen. In dit geval zullen de stappen de elementen zijn die als geheel een functioneel spel zijn. Daarom zullen wij een stappenplan opstellen per onderdeel, om zo geleidelijk naar een werkend eindproduct te werken.

## **Stap 1: indeling van het speelgebied**

Het handigste is om eerst de basisindeling van de webpagina te regelen. Dit betekent dus dat het algehele speelgebied eerst moet worden gedefiniëerd, en daarna het gebied waar de startpositie van de containers staan opgenomen, en natuurlijk het “vrachtschip” waar deze allemaal naartoe moeten.

## **Stap 2: beweegbare element + deze beperken tot het algehele speelgebied**

De volgende stap is het aanmaken van een beweegbaar element, en ervoor zorgen dat deze niet buiten het speelgebied kan worden bewogen. Dit element is uiteraard de zeecontainer die uiteindelijk op het schip moet.

## **Stap 3: meerdere beweegbare elementen + collision**

Nu we één beweegbaar element hebben kunnen aanmaken en hebben kunnen beperken tot het speelgebied, is het nu de stap om meerder beweegbare elementen te maken (2 of meer), en ervoor te zorgen dat deze niet over elkaar heen kunnen bewegen. Er moet dus collision (botsingen) tussen deze elementen zijn.

## **Stap 4: check of element zich binnen een bepaald gebied bevindt**

In deze stap wordt er gekeken naar hoe er door de code kan worden gecheckt wanneer een (beweegbaar) element zich binnen een bepaald gebied bevindt.

Hierbij is het belangrijk om te letten op de manier waarop dit wordt gedaan: dit moet gecheckt worden wanneer het beweegbare element wordt losgelaten; andere momenten zullen niet werken (bijvoorbeeld tijdens het bewegen). Er moet gewacht worden tot de gebruiker het element ook daadwerkelijk neerzet.

## **Stap 5: het generen van willekeurige setjes containers**

Als de gebruiker elke keer precies dezelfde puzzel moet oplossen, wordt het spel snel wel heel erg saai. Om dit tegen te gaan, zullen wij verschillende setjes containers laten genereren. Dit houdt in dat wanneer de gebruiker het spel start, er verschillende reeksen containers zullen worden gegeneerd. Dit zal worden gedaan door middel van het willekeurig kiezen tussen een aantal voorbereide, voorgeprogrammeerde setjes.

Ook zal de startpositie willekeurig zijn voor elke container; dit zal niet afhangen van voorgeprogrammeerde setjes, en zal daadwerkelijk willekeurig zijn.

## **Stap 6: continu checken of alle elementen zich in een bepaald gebied bevinden, en wanneer dit zo is de webpagina zodanig aanpassen dat dit wordt weergeven (het winscherm)**

De meest logische, volgende stap gezien de vorige stap is om te checken of *alle* containers zich binnen een bepaald gebied bevinden (ofwel het containerschip). Als dit zo is, moet de website zodanig gestyled worden dat de gebruiker in één oogopslag kan zien dat hij of zij heeft gewonnen.

## **Stap 7: bepaalde gebieden “verboden” maken om elementen in te plaatsen, maar niet elementen overheen te bewegen**

Deze stap is erg vanzelfsprekend – een container mag zich niet op de rand van een schip bevinden. Dit zullen wij proberen mee te nemen in het spel. Wanneer een container op de rand van het schip wordt geprobeerd te plaatsen, moet de container bij voorkeur teruggezet worden op de startpositie. Anders moet de container blijven “plakken” aan de muiscursor, totdat deze op een geldige plek wordt neergezet.

## **Stap 8: eventueel: een knop om het spel opnieuw te spelen**

Indien de tijd en middelen dit toestaan, is het natuurlijk altijd een handig idee om een restart-knop toe te voegen waarbij de gebruiker het spel opnieuw kan spelen.

## **Stap 9: eventueel: het maken van een timer, die telt hoeveel seconden(+milliseconden) er zijn verlopen tijdens de huidige spelsessie**

Mits er tijd en middelen voor zijn, kan er een timer worden gemaakt voor het spel die bijhoudt hoe lang de speler (gebruiker) erover doet om alle containers op het schip te laten passen. Deze timer stopt uiteraard zodra de gebruiker heeft gewonnen.

Aan de hand hiervan kan het spel ook zodanig worden gemaakt dat de speler het spel binnen een bepaald tijdslimiet moet oplossen.

Stap 10: eventueel: de gebruiker in staat stellen om een naam op te geven, en de highscore (de tijd) op te slaan in een database en deze lijst te weergeven

Mits er tijd en middelen voor zijn, kan op het winscherm de mogelijkheid aan de speler worden gegeven om zijn of haar naam in te voeren. Als de speler dit doet, kan er een database-record worden aangemaakt met de tijd en de spelersnaam. Op deze manier kan er een highscore tussen spelers worden bijgehouden.

# **Bronnen**

1. “What is jQuery?”, jQuery.

<https://jquery.com/>

2. “jQuery user interface”, jQueryui.

<https://jqueryui.com/>

3. “Essential tools for software developers and teams”, JetBrains.

<https://www.jetbrains.com/>