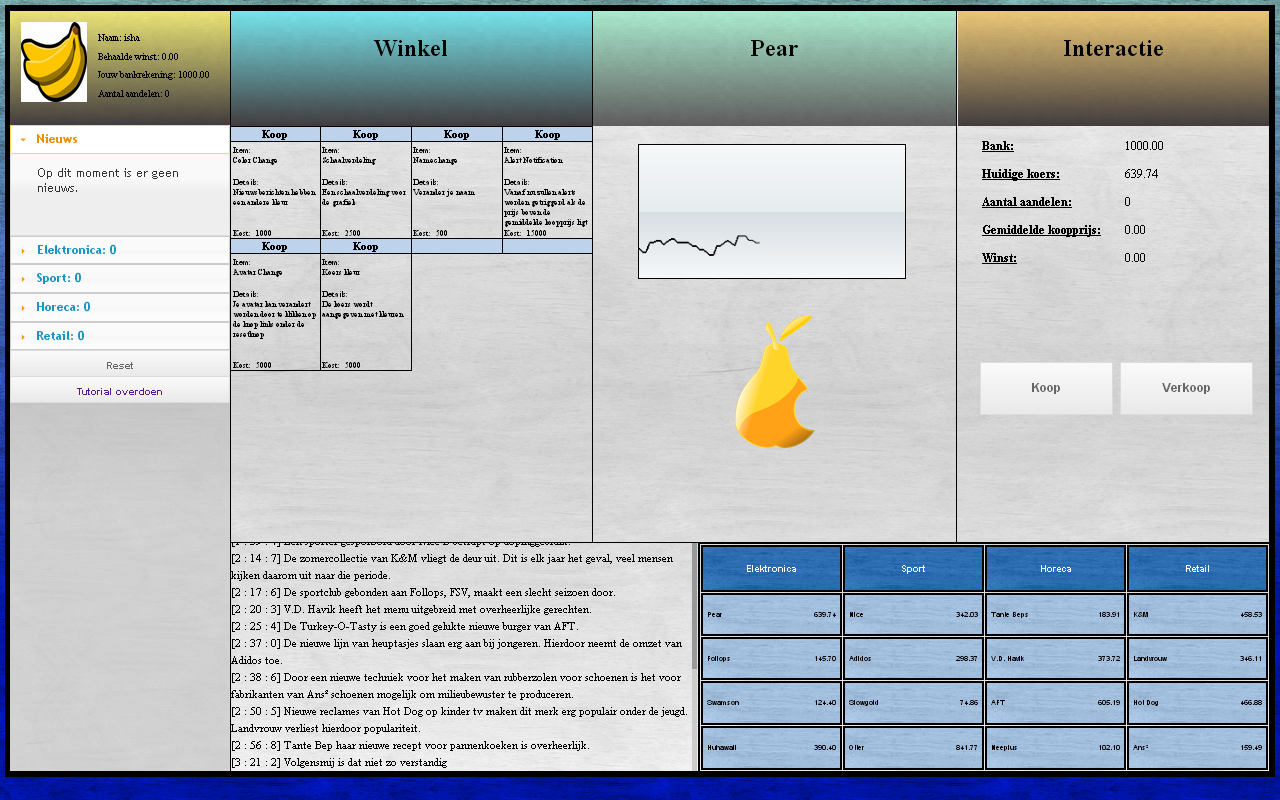
**De beursvloer**

****

**www.tinyurl.com/pwsAF**

Isha Dijcks en Floris List 6V

**Inhoudsopgave**

Inleiding Blz.

Programmeertalen Blz.

Veelgebruikte code Blz.

Uiterlijk Blz.

Eigenschappen van aandelen Blz.

Verschijnen en verbergen van aandelen Blz.

Kopen en verkopen Blz.

Automatisch opslaan Blz.

Schaalverdeling Blz.

Canvas Blz.

Nieuwsberichten Blz.

Aandelenkoers Blz.

Console Blz.

Logboek Isha Blz.

Logboek Floris Blz.

Bronnenlijst Blz.

**Inleiding**

Vanwege onze interesses hadden wij (Floris en Isha) het idee om een site te programmeren. Dit moesten wij echter wel kunnen binden aan een schoolvak. Zo kwamen we bij het maken van een nabootsing van de economische beurs. We vroegen ons af; **Is het mogelijk om een correcte simulatie van de economische beurs te maken?** Deze vraag leek ons geschikt aangezien we iets wilden programmeren.

Er zijn genoeg websites op het internet waar je aandelen kan kopen en verkopen. Dat wij hetzelfde doen als vele anderen hebben gedaan gaat niet om het beter te doen of iets in die richten, maar meer om te proberen of wij zoiets in elkaar kunnen zetten. **Hoeveel kennis van de beurs is er voor nodig om zo’n simulatie mogelijk te maken?** Is een vraag die al gauw kwam bij ons opkwam. Wij zijn ook benieuwd of het resultaat bruikbaar zou kunnen zijn voor het onderwijs. Hierbij dachten we aan de 3de klassen die spelenderwijs in contact kunnen komen met de beurs en zo verbanden tussen koersen en gebeurtenissen leren begrijpen. Dit gaan we onderzoeken door een website te programmeren, waarop je verschillende aandelen kan kopen met “geld”. De prijzen van deze aandelen schommelen onverwachts, zodat het aan de speler over wordt gelaten te bepalen wat een goede investering is. Af en toe komt er een nieuwsbericht dat betrekking heeft op een of meer van de bedrijven waar aandelen van gekocht kunnen worden. Deze mededeling zorgt voor een schommeling in de prijs van bepaalde aandelen. Langzamerhand zal de speler steeds meer geld verdienen, zodat er meer aandelen gekocht kunnen worden. De taken zijn vermengd verdeelt, maar de afspraak die we hebben gemaakt is dat Isha zich meer bezig houdt met het programmeren en Floris meer aan het uiterlijk van de website. Samen zorgen we er uiteindelijk voor dat alles goed werkt en gebalanceerd is.

Wij willen allebei Technische Informatica gaan studeren aan de TU Delft. Vandaar dat we wat willen programmeren. Dit profielwerkstuk kan als een dergelijk portfolio dienen die we bij onze inschrijving voor de studie kunnen laten zien. We zijn beiden erg gedreven om iets zo mooi mogelijks te maken, omdat we het ook leuk vinden om te doen. Dus naast het feit dat dit profielwerkstuk moet vinden we het ook erg leuk om te doen en steken we er zonder tegenstribbelen met veel plezier tijd in.

**Programmeertalen**

Om onze website te maken hebben we gebruik gemaakt van de volgende programmeertalen:

**HTML**

HTML is de standaardtaal voor het schrijven van webpagina's, die door iedereen bekeken kunnen worden op het internet.

HTML staat voor HyperText Markup Language:

* Hypertext is de techniek waarmee een link wordt gelegd naar een ander document, een afbeelding, of een geluidsopname.
* Markup is het gebruik van code om de browser te vertellen hoe de inhoud van het document weergegeven moet worden en naar welke bestemming de hyperlinks moeten leiden.

HTML-code is opgebouwd uit elementen en attributen.

Elementen geven aan wat er op de pagina geplaatst moet worden, en staan tussen <> haakjes. De meeste elementen moet geopend en gesloten worden, de correcte manier om bijvoorbeeld een paragraaf te plaatsen ziet er als volgt uit:

<p> Dit is een paragraaf </p>

Waarbij <p> het element begint, en </p> het element afsluit.

Attributen geven specificaties aan elementen, zoals kleur, grootte, lettertype, plaats etc.

Een roodgekleurde paragraaf plaats je als volgt:

<p style="color:red"> Dit is een rode paragraaf </p>

Een HTML bestand begint altijd met <!DOCTYPE HTML> om aan te geven dat het een html bestand is. Een HTML bestand heeft altijd een head en een body, tussen de <head> tags plaats je locaties van andere bestanden waar naar wordt refereert, zoals een extern .css bestand om je website mooier te maken:

<link rel="stylesheet" type="text/css" href="cssbestand.css"/>

Met classes en ID’s kan je elementen benoemen waardoor ze makkelijker terug te vinden zijn. Elementen kunnen dezelfde of meer dan 1 class hebben, maar niet hetzelfde ID

<p class="text rodetext" id="text1"> Deze tekst gaan we rood maken in CSS</p>

<p class="text zwartetext" id="text2"> Deze tekst laten we zwart</p>

Er zijn waanzinning veel elementen gedefinieerd in HTML. Zo kan je je website opbouwen uit al deze elementen, die samen uiteen komen tot een compleet plaatje.

De extensie voor HTML bestanden is .html

**CSS**

De programmeertaal waarin websites worden opgemaakt heet CSS. De manier waarop de paragraaf gekleurd is in de HTML wordt heel onoverzichtelijk als je veel elementen in een document hebt staan. Daarom is het handiger een nieuw bestand aan te maken met de extensie .css

In dit bestand kan gemakkelijk de elementen attributen worden geven. De indeling van een .css bestand ziet er als volgt uit:

element{

attributen: waarde

}

Om een paragraaf rood te maken in een extern .css bestand gebruik je de volgende code:

p{

color:red;

}

Dit is veel overzichtelijker, zeker als er meer attributen worden toegevoegd. Met deze bewerking verander je alle paragrafen in het document. Stel dat het de bedoeling is om alleen de paragrafen met de class “rodetext” rood te kleuren, dan kan je die elementen pakken met het “.” teken, deze geeft aan dat het om een class gaat, “#” is voor een ID.

.rodetext{

color:red;

}

Het document gaat opzoek naar elementen met de class “rodetext” en kleurt deze rood. Op deze manier kan je je hele HTML document vorm geven, zo kan je de grootte van elementen aangeven, de hoogte, de plaats, de zichtbaarheid enzovoorts.

**Javascript**

Javascript is de taal die gebruikt wordt voor het interactief maken van websites, het is ook erg handig om variabelen op te slaan en aan te passen. Op onze site maken we gebruik van meer dan 150 variabelen, om de aandelenkoers bij te houden, hoeveel geld de speler heeft, wanneer nieuwsberichten komen etc. Het maken van een variabele is erg eenvoudig:

var getal= 10;

Dit slaat de waarde 10 op in getal, als we voortaan het getal 10 willen gebruiken, kunnen we simpelweg de variabelen oproepen. Je kan ook rekenen met variabelen

var rekensom = getal\*getal;

Ook is het mogelijk om tekst op te slaan in variabelen, dit heet een string:

var text= “Deze tekst gaat in een variabele”;

Een ander soort variabele is een functie, een functie kan je op elk moment aanroepen, waarna deze een voorbepaalde handeling uitvoert, return geeft de uitkomst van deze handeling:

var rekenen= function(){

var uitkomst = 10\*10;

return uitkomst;

};

Deze functie doet niets totdat hij wordt opgeroepen, dit gebeurd door de handeling rekenen();

Waarna de uitkomst wordt gegeven.

Een andere soort variabelen is een array, een array laat je variabelen op een rijtje opslaan:

var array= [“variabele1”,”variabele2”,”variabele3”];

LET OP: EEN ARRAY BEGINT BIJ 0 MET TELLEN, DE EERSTE VARIABELE IN EEN ARRAY IS DUS array[0];

Dit geeft array[0] = “variabele1” en array[2] = variabele 3

De laatste manier is met objects, objects zijn een soort arrays, alleen in plaats van een cijfer hebben de variabele een naam:

var object= {

grootte: 1,

lengte: 2,

prijs: 3;

}

Deze variabelen zijn beschikbaar via object.grootte en object.prijs.

**jQuery**

jQuery is een vorm van javascript, het idee erachter is dat het bepaalde handelingen die je in javascript uitvoert makkelijker en overzichtelijker op kan schrijven. Het laten verdwijnen van een paragraaf als er op een knop gedrukt wordt in javascript ziet er als volgt uit:

function toggleElement(text){

var e = document.getElementById(text);

e.style.display = e.style.display != 'block' ? 'block' : 'none';

};

In jQuery ziet het er veel overzichtelijker uit:

$(“#text”).hide();

jQuery zit vol met trucjes om javascript codering makkelijker te maken, het enige wat er voor nodig is, is het downloaden van de laatste versie en er naar te verwijzen in het HTML bestand

<script src="jquery-1.10.2.js"></script>

jQuery is hier te downloaden:

<http://jquery.com/>

**Veelgebruikte code**

In dit hoofdstuk leggen we bepaalde code uit die vaak terug komt op onze site, waar het bij hoort en wat het doet.

**HTML**

<div><div>

Een <div> is een blok van de pagina die je zelf een lengte en een hoogte kan geven.

Het voordeel van een div is dat deze makkelijk verplaatsbaar en veranderbaar is.

<span></span>

Een <span> is ook een deel van de pagina, alleen in tegenstelling tot een <div> is een

<span> geen blok, maar een enkele regel. Voor tekst is een <span> het handigst.

<p></p>

Een <p> start een paragraaf, de standaard manier om tekst in weer te geven, toch

gebruiken we liever <span>, omdat een <p> witregels creëert tussen paragrafen, wat onhanding kan zijn bij het plaatsen van tekst onder elkaar.

<table></table>

Maakt een tabel, die is opgedeeld uit rijen en cellen, <tr> en <td>

<table>

<tr>

<td>Dit</td>

<td>Is</td>

<td>De</td>

<td>Code</td>

</tr>

<tr>

<td>Voor </td>

<td>Een</td>

<td>Mooie</td>

<td>Tabel</td>

</tr>

</table>

Geeft:

|  |  |  |  |
| --- | --- | --- | --- |
| Dit | Is | De | Code |
| Voor | Een | Mooie | Tabel |

<button></button>

Maakt een standaard knop waar op geklikt kan worden.

<img src=”locatie” />

Toont een plaatje dat gevonden kan worden op de aangegeven locatie.

<h3></h3>

Plaatst dikgedrukte tekst in plaats van normale, varieert van <h1> tot <h6>.

<canvas></canvas>

Plaatst een canvas waarop getekend kan worden met javascript. De beste manier om

grafieken af te beelden in HTML.

<strong></strong>

Wat dit element doet is het dikgedrukt maken van tekst. Je krijgt hetzelfde effect als je

bijvoorbeeld in een word-document op de **B** knop drukt.

class=”” en id=””

Deze tags kan je in elk element zetten tussen de elementen (<’elementnaam’>). Een

toegpast voorbeeld: <h3 class=”header3”> of <h3 id=”header3”> of

<h3 class=”header3” id=”header3”>.

Deze twee tags doen bijna hetzelfde. Het is handig om een element een class=”” te

geven als je CSS (stylering) regels wilt geven en dit toe moet passen op meerdere elementen. Als je een element dus class=”” geeft worden uit het CSS bestand de regels gehaald en vervolgens toegepast op het specifieke element. Dus het is te stellen dat class=”” heel algemeen gebruikt moet worden en aspecifiek is.

id=”” daarentegen is juist heel erg specifiek. Als je specifieke stylering (CSS) wilt toepassen op één element geef je het gewenste element een id=”” en daarmee uitzonderlijke CSS stylering. Het gebruiken van een id=”” maakt het voor de programmeur ook makkelijker om dat specifieke element terug te vinden.

<a></a>

De a staat voor attribute. Dit element kan je gebruiken als je iets linkend wilt maken. Ook

kan het gebruikt worden als er op wordt geklikt een download wordt gestart. Zo heeft dit

element nog meer mogelijkheden, maar deze mogelijkheden hebben we niet veel

gebruikt.

<form></form> en <input></input>

<form> haalt data van de <input>’s die ingesloten in het <form> element staan. <input>

kan verschillende types hebben. De input types die wij hebben gebruikt zijn: file en text.

File haalt een gekozen file uit jouw bestanden binnen en text maakt een wit tekstgebied

waar je tekst in kan voeren.

**CSS**

height

Verandert de hoogte van het element naar de aangegeven waarde.

width

Verandert de breedte van het element, 100% vult het hele scherm.

text-decoration

Als deze op None wordt ingesteld, haal je alle decoratie weg van een tekst, nuttig voor links die standaard onderstreept zijn.

position

Position heeft verschillende mogelijkheden

Relative: De plaats van het element is gebaseerd op wat de oorspronkelijke plaats zou zijn als deze niet was aangepast.

Absolute: De plaats van het element is absoluut bepaald ten opzichte van de linkerbovenhoek

Fixed: Hetzelfde als absolute, alleen blijft het element ook op deze plaats als je naar beneden scrollt.

float

Left of Right, bepaalt of het element aan de linker- of rechterkant van de pagina zit.

margin

De waardes hiervan bepalen hoeveel ruimte er mag zitten tussen dit element en een element dat hier aan grenst, de waardes worden afgelezen in deze volgorde: Boven, rechts, onder, links. Margin: 1px, 0, 0, 1px betekent een marge van 1 pixel aan de boven- en linkerkant van het element.

text-align

Bepaalt waar in het element de tekst wordt geplaatst. Links, rechts of in het midden.

border

Heeft 3 attributen; grootte, soort en kleur. Plaatst een rand om een element heen. border: 1px dotted black geeft een gestipte rand van 1 pixel breed om het element heen.

top, left, right, bottom

Verplaatst het element in de aangegeven richting met de ingestelde waarde.

background (-image, -repeat, -size, -color)

Dit plaats een background in het gespecificeerde element.

-image, plaatst een afbeelding,

-repeat, bepaald in welke richtingen de achtergrond wordt herhaald,

-size, bepaald de grootte van de achtergrond,

-color, bepaald de kleur van de achtergrond.

font (-family, -size, -weight)

Dit is de CSS code voor het lettertype. Hiermee kan je bepaalde eigenschappen geven

aan de geschreven tekst.

-family, bepaald het lettertype,

-size, bepaald de grootte van het lettertype,

-weight, bepaald de dikte van de letters.

display

Hiermee kan je een specifiek element bepaalde weergave eigenschappen geven. Wij

hebben voornamelijk de specificatie ‘hidden’ gebruikt, die zorgt ervoor dat het

aangewezen element niet zichtbaar is.

color

Bepaald een kleur van een element.

vertical-align

Deze code bepaald de plaatsing ten opzichte van de y-as (verticale as)

top

Bepaald de afstand die het aangewezen element heeft van de top van het element waar

het aangewezen element in zit, als deze ergens in zit anders wordt de top van het

document genomen. De input kan in pixels en procenten.

bottom

Doet hetzelfde als ‘top’ maar dan voor de onderkant in plaats van de bovenkant.

overflow (-x, -y)

Deze code geeft aan dat er binnen in het element gescrold kan worden als de inhoud

groter is dan het element zelf.

-x, bepaald eigenschappen over de x-as,

-y, bepaald eigenschappen over de y-as.

**Javascript / jQuery**

if/else statement

Vergelijkt een variabele met een bepaalde waarde, als het klopt wordt een bepaalde code uitgevoerd, anders een andere code.

if ( 3 <4) {

return “klopt”;

}

else {

return “klopt niet”;

}

Deze correcte if/else statement returned “klopt”, want 3 is kleiner dan 4.

Math.random()

Kiest een willekeurig getal uit tussen de 0 en de 1. Math.random()\*100 geeft een willekeurig getal tussen de 0 en de 100.

Math.floor()

Rondt het getal wat tussen de haken staat af naar beneden tot een heel getal.

switch(variabele)

Een ander soort if/else statement, voor als er meer dan 2 mogelijkheden zijn, vaak overzichtelijker.

variabele.toFixed(2)

Rond de variabele af op 2 cijfers.

$(“#id”).remove()

Een jQuery functie die een id of een class van de pagina kan laten verdwijnen.

$(“#id”).hide()

Een jQuery functie die een id of een class van de pagina verbergt totdat deze weer moet worden getoond.

$(“#id”).show()

Een jQuery functie die een id of een class weer kan laten verschijnen.

$(“<p>text</p>”).appendTo(“#id”)

Plaatst de paragraaf in de gewenste ID, erg nuttig om tekst op de juiste plaats te laten verschijnen.

.css()

Hiermee kan je alle css attributen die een element heeft laten aanpassen via jQuery.

localstorage

Het wordt gebruikt om variabelen op te slaan, hier vertellen we later meer over.

ctx

Verwijst naar het canvas waarmee op dat moment bewerkingen worden uitgevoerd.

$(“#id”).click(){}

Voert een functie uit als op het ID of class wordt geklikt.

setInterval()

Voert een functie uit om een bepaald aantal milliseconde.

for(){}

Een for loop voert een functie uit zolang een bepaalde conditie juist is. Wij hebben het meest gebruikt gemaakt van deze loop

for ( var i = 1; i<17; i++){}

Deze loop zorgt ervoor dat een bepaalde handeling 16 keer wordt doorlopen, namelijk totdat i groter is dan 17. Omdat we 16 aandelen hebben kunnen we deze oproepen m.b.v price2[i], omdat i het aandeel is waarmee op dat moment mee wordt gerekend. Door de for loop te gebruiken worden dezelfde handelingen vaak uitgevoerd met verschillende variabelen, dit scheelt veel code en maakt het proces sneller.

//

Alle tekst met // ervoor is een opmerking, dit wordt genegeerd door het systeem. Dit maakt het voor ons makkelijker om code terug te vinden

**Structuur**

De opbouw van de site is heel simpel. Elk aandeel heeft zijn eigen elementen in het document. Om Pear even als voorbeeld te nemen.

Hier wordt alle informatie over de aandelen gegeven: gemiddelde koopprijs, prijs, hoeveelheid etc.

<p class="1 a aandeeltext" id="aandeeltext1"></p>

De koop en verkoop knoppen.

<button data-type="1" id="1buy" class="a\_button buy 1 a">Koop</button>

<button data-type="1" id="1sell" class="a\_button sell 1 a">Verkoop</button>

Een header boven aan de pagina:

<h3 class="1 a">Pear</h3>

Een canvas:

<canvas class="canvas 1 a" id="canvas1"></canvas>

Een plek voor de scale:

<div class="scale 1 a" id="scalel1"></div>

En een plaatje:

<img id="peer" class="1 a" src="logos/pear.png" width=120px height=200px></img>

De opbouw van de koers rechtsonder is een simpele tabel en de accordion is een list met de jQuery accordion functie.

**Uiterlijk**

Het uiterlijk heeft verschillende stadia gehad. Het eerste stadium was dat het nog een kale pagina was met een primitieve teller die later als aandeel moest gaan functioneren.

Het tweede stadium was het al wat beter geplaatst op de pagina in een container. Na het tweede stadium is Floris voornamelijk met het uiterlijk gaan stoeien.

In het derde stadium was alles omvat in een ‘mainframe’. Dit ‘mainframe’ was opgedeeld in twee stukken, een stuk heette ‘rechts’ en het andere ‘links’. In het stuk ‘links’ werd de navigator geplaatst en de specificaties van de speler boven de navigator. In ‘rechts’ kwamen de interactie, canvas, bedrijf beschrijvingen en de console. De kleuren waren in dit stadium plat, lelijk en saai.

In het vierde stadium vonden we de console veel te groot dus besloten we een tabel toe te voegen rechts onder in ‘rechts’. In deze tabel kwamen alle bedrijven en huidige koersen te staan. Hierdoor werd de console aanzienlijk kleiner en bleek dat die tabel verdomd handig was. We hadden ook besloten dat we gradients als achtergronden gingen gebruiken. Dus hadden we gradients gemaakt.

In het vijfde stadium besloten we dat we de bedrijf beschrijvingen niet gingen maken, omdat we te weinig tijd hadden. In plaats daarvan hadden we dat vlak veranderd in een winkel waar spelers upgrades en toevoegingen kunnen kopen. De inhoud qua tekst en knoppen van ‘rechts’ werden dit stadium ook aanzienlijk vergroot om het ‘mainframe’ op te vullen en het er zo beter uit te laten zien.We zijn van de gradient achtergronden overgestapt naar hoe het er nu uitziet. Deze grafische beslissing was de beste beslissing. Na het vijfde stadium is het geworden hoe het er nu uitziet.

**Eigenschappen van aandelen**

Alle aandelen hebben 10 variabelen: start, price2, price1, startprijs, daalkans, hoogte, quantity, cost,average en profit. Deze variabelen zitten in netjes in een object geplaatst.

Start is een variabele die 1 is als de site wordt geopend, zodat de schaalverdeling van de grafiek goed wordt geplaatst. Zodra dit is gebeurd voor alle aandelen veranderen de start variabelen naar 0.

Price2 is de huidige prijs van het aandeel, deze wordt via allerlei formules beïnvloed, hier leggen we later meer over uit.

Price1 is de prijs van 2 seconden geleden, dit wordt gebruikt om de grafiek mee te tekenen.

Startprijs geeft de beginwaarde van aandelen aan, deze kan verhoogt of verlaagt worden door nieuwsberichten. Als de huidige prijs te veel afwijkt van de startprijs worden er aanpassingen gedaan zodat de prijs altijd om de startprijs blijft schommelen

Daalkans is de variabele die we gebruiken om te bepalen of de prijs stijgt of daalt, daalkans wordt beïnvloed door nieuwsberichten.

Hoogte is de prijs die het aandeel heeft afgerond op 150 naar boven, dit is omdat het canvas 150 pixels hoog is, zo kan de grafiek altijd op de juiste plaats worden afgebeeld.

hoogte[i] = Math.floor(1+price2[i]/150)\*150;

Een prijs van 210 geeft Math.floor(1+1.4)\*150, een hoogte van 300.

De variabelen quantity, average, cost en profit zijn standaard nul.

Quantity is het aantal aandelen dat je van die soort hebt. quantity wordt bij het kopen en verkopen aangepast met

quantity[pick]+=amount;

Waarbij pick het aandeel is waar het om gaat, en amount de hoeveelheid is die wordt gekocht of verkocht. Bij het verkopen gaat

quantity[pick]-=amount;

Cost is hoeveel je hebt uitgegeven aan de aandelen die je op dat moment in bezit hebt, en wordt berekend met de volgende formule bij het kopen, zodat er later de gemiddelde inkoopprijs mee kan worden berekend.

cost[pick]+=price2[pick]\*amount;

Bij het verkopen van alle aandelen wordt de cost op 0 gezet, omdat je geen aandelen meer hebt.

cost[pick]= 0;

Bij het verkopen van één aandeel

cost[pick]-=average[pick];

Gaat de gemiddelde inkoopprijs van de totale kosten af.

Average is de gemiddelde inkoopprijs van deze aandelen, deze berekening wordt uitgevoerd als het bestand geupdatet wordt, en vindt dus plaats in een for loop.

if(quantity[i] === 0) { quantity[i]++; average[i] = cost[i]/quantity[i];

quantity[i]--;

}

else { average[i] = cost[i]/quantity[i]; }

Omdat deze vergelijking in de update functie zit moet deze op alle tijden kunnen worden uitgevoerd, ook als de speler geen aandelen heeft zou het programma door 0 moeten delen, daarom maakt het programme de quantity voor heel even groter, zodat het programma door 1 deelt, wat wel mag. Als de speler wel aandelen heeft dan berekent het programma het gemiddelde door de kosten te delen door het aantal aandelen.

Profit houdt bij hoeveel de speler heeft verdiend aan dat aandeel, een simpele berekening die wordt uitgevoerd bij het kopen en verkopen

Kopen: profit[pick]-=price2[pick]\*amount;

Verkopen: profit[pick]+=price2[pick]\*amount;

**Verschijnen en verbergen van aandelen**

In onze html hebben we alles wat ooit moet verdwijnen of verschijnen de class “a” gegeven, en de class van het aandeel waar het bij hoort (1-16). De canvas van Pear ziet er als volgt uit:

<canvas class="canvas 1 a" id="canvas1"></canvas>

Dit element heeft 3 classes; canvas, 1 en a.

Zodra er op een element wordt geklikt dat linkt naar andere aandelen(“.aandeelLink”) zoals deze

<a data-type="1" class="aandeelLink" id="pear" href="#"></a>

Worden alle elementen met class a verborgen, alle aandeeltexten, alle canvassen, alle buttons etc. Dan worden meteen al deze elementen met de class van het aandeel waar op wordt werd geklikt getoont.

$(".aandeelLink").click(function(){

var deze = $(this).data("type");

$(".a").hide();

$("."+deze).show();

});

**Kopen en verkopen**

Onze HTML bevat 16 koop- en 16 verkoop buttons, 1 voor elk aandeel.

button data-type="1" id="1buy" class="a\_button buy 1 a">Koop</button>

button data-type="1" id="1sell" class="a\_button sell 1 a">Verkoop</button>

Data-type is een handig attribuut wat het makkelijker maakt om informatie door te geven.

Zodra er op een knop met de class “buy” of “sell” wordt geklikt, wordt via data type achterhaald met welk aandeel berekeningen uitgevoerd moeten worden. De klik functie heeft als input e, deze controleert of er op shift wordt geklikt. Als dat het geval is wordt er zo veel mogelijk gekocht of verkocht. De variabelen die bij de aandelen horen worden aangepast, en een confirmatie van de ver- of aankoop wordt gegeven in de console. Als de speler niet genoeg geld heeft wordt dit aangegeven.

$(".buy").click(function(e){

var pick = $(this).data("type");

if (e.shiftKey){ var amount = Math.floor(money/price2[pick]); cost[pick]+=price2[pick]\*amount; profit[pick]-=price2[pick]\*amount; quantity[pick]+=amount; money-=price2[pick]\*amount; consoleUpdate("Je hebt "+amount+ " aandelen gekocht voor "+price2[pick].toFixed(2)\*amount+" euro." );

update(); }

else {

if (money>=price2[pick]) { quantity[pick]++; cost[pick]+=price2[pick];

profit[pick]-=price2[pick]; money-=price2[pick]; consoleUpdate("Je hebt 1 aandeel gekocht voor "+price2[pick].toFixed(2) + " euro");

update();

}

else {consoleUpdate("Je hebt niet genoeg geld")};

}

});

$(".sell").click(function(e){ var pick = $(this).data("type");

if (e.shiftKey) { money+=quantity[pick]\*price2[pick]; profit[pick]+=quantity[pick]\*price2[pick]; cost[pick]= 0;

consoleUpdate("Je hebt "+quantity[pick]+ " aandelen verkocht voor "+price2[pick].toFixed(2)\*quantity[pick]+" euro") quantity[pick]=0; update(); }

else { if (quantity[pick]>0) { quantity[pick]--; cost[pick]-=average[pick]; money+=price2[pick]; profit[pick]+=price2[pick]; consoleUpdate("Je hebt 1 aandeel verkocht voor "+price2[pick].toFixed(2)+ " euro");

update();

}

else {consoleUpdate("Je hebt geen aandelen);} } });

**Automatisch opslaan**

Het automatisch opslaan gebeurt via localStorage. Met localstorage kan je variabelen lokaal opslaan op de computer van de gebruiker. Bij elke update worden alle variabelen opgeslagen. En bij het laden van het bestand worden alle variabelen geladen via de volgende formules.

var save = function(){ localStorage.setItem("money",money);

localStorage.setItem("nieuwsUpgrade",nieuwsUpgrade);

localStorage.setItem("schaalUpgrade",schaalUpgrade);

localStorage.setItem("naamUpgrade",naamUpgrade);

localStorage.setItem("koersUpgrade",koersUpgrade);

localStorage.setItem("avatarUpgrade",avatarUpgrade);

localStorage.setItem("alertUpgrade",alertUpgrade);

for (var i = 1; i<17; i++){

localStorage.setItem("price2["+i+"]",price2[i]); //Deze commands zorgen ervoor dat er op de computer een

localStorage.setItem("quantity["+i+"]",quantity[i]); localStorage.setItem("cost["+i+"]", cost[i]); localStorage.setItem("profit["+i+"]", profit[i]);

}

};

var load = function(){

money = Number(localStorage.getItem("money"));

nieuwsUpgrade = Number(localStorage.getItem("nieuwsUpgrade"));

schaalUpgrade = Number(localStorage.getItem("schaalUpgrade"));

naamUpgrade = Number(localStorage.getItem("naamUpgrade"));

koersUpgrade = Number(localStorage.getItem("koersUpgrade"));

avatarUpgrade = Number(localStorage.getItem("avatarUpgrade"));

alertUpgrade = Number(localStorage.getItem("alertUpgrade"));

for (var i = 1; i<17; i++) { //Deze functie heeft als doel de variabelen die zijn opgeslagen

price2[i] = Number(localStorage.getItem("price2["+i+"]"));

quantity[i] = Number(localStorage.getItem("quantity["+i+"]")); //Localstorage slaat alle variabelen op als een string

cost[i] = Number(localStorage.getItem("cost["+i+"]")); //Een string is text, geen nummer

profit[i] = Number(localStorage.getItem("profit["+i+"]")); //De Number() vergelijking zorgt ervoor dat tekst wordt omgezet

}

};

**Schaalverdeling**

De schaalverdeling is een handige upgrade die de speler later in het spel kan vrijspelen. Het is een plaatje aan de linkerkant van de grafiek die de schaal aangeeft. Het maakt het makkelijker voor de speler om de koers te bekijken. Als de prijs buiten het gebied van de canvas komt, dan wordt het canvas leeggemaakt. Als de speler de upgrade heeft gekocht wordt de schaalverdeling toegevoegd. Dit is een voorbeeld van de code voor als de prijs tussen 150 en 300 zit. In de code hebben we deze functies voor alle trajecten tussen de 0 en de 1500

if (schaalUpgrade === 1) {

if (price2[i] > 150 && price2[i] < 300){

if (start[i] === 1) {

$("#canvas"+i).css("margin-left", "-84px");

$("<img class=feestbeest id=scale src=scales/scale150-300.png height=202px />").appendTo("#scalel"+i);

start[i] = 0;

}

}

else {

if (price2[i] > 150 && price2[i] < 300){

if (price1[i] < 150 && price2[i] >150 || price1[i] >300 && price2[i] <300){

ctx.beginPath();

ctx.clearRect(0,0,800,150);

}}

}

}

**Canvas**

Het canvas is zonder twijfel het meest ingewikkelde onderdeel van de code. Canvas is een stuk HTML waar je op kan tekenen. Je kan lijnen zetten, vierkanten tekenen etc. Hier is de code voor de canvas stapje voor stapje uitgelegd.

for (var i = 1; i<17; i++) {

//Deze loop zorgt er voor dat de bewerkingen voor elk aandeel worden uitgevoerd

hoogte[i] = Math.floor(1+price2[i]/150)\*150;

//Een nieuwe hoogte wordt berekend

var c=document.getElementById("canvas"+[i]);

//Deze 2 vergelijkingen zorgen ervoor dat je het canvas object pakt

var ctx=c.getContext("2d"); //Dit wordt uitgevoerd voor alle canvassen die in de loop zitten

price1[i]=price2[i]; //De oude prijs van het aandeel wordt gelijkgezet aan de huidige prijs

ctx.moveTo(time,hoogte[i]-price2[i]); //Het canvas beweegt naar dit punt in de grafiek

time+=4;

ctx.lineTo(time,hoogte[i]-price2[i]);

//Nu wordt er op het canvas een lijn getekend van het vorige gekozen punt (x = time, y = oude prijs)

ctx.stroke();

//Naar het nieuwe punt (x = time+4, y = nieuwe prijs)

update();

if ( i < 16) {

time-=4; //Dit zorgt er voor dat als de loop niet voor de laatste keer doorlopen wordt, de tijd een stapje terug gaat zodat de grafiek ook op het volgende canvas goed wordt getekend

}

if (time === 304){

for (var i = 1; i<17; i++){

var c=document.getElementById("canvas"+i);

//Als de tijd 304 is, en het einde van het canvas is bereikt

var ctx=c.getContext("2d");

//Wist deze statement de grafieken op alle canvassen, en begint opnieuw met tekenen

ctx.beginPath();

ctx.clearRect(0,0,800,200);}

time = -4;

//Dit zet de tijd terug op -4, zodat de eerste lijn op t=0 wordt getekend

} //Eindigt de functie

} //Eindigt de for-loop

**Nieuwsberichten**

Nieuwsberichten worden willekeurig gekozen. Alle nieuwsberichten zijn voorgeschreven in de code gezet. Dit hebben we allemaal in een ander bestand gezet genaamd ‘nieuwsberichten.js’. Nieuwsberichten worden geprint in de console en in de navigator onder het kopje nieuws. Ook hebben ze invloed op de aandelenkoersen. Als het nieuwsbericht positief is dan heeft een aandeel meer kans om te stijgen en als het negatief is meer kans om te dalen. Sommige nieuwsberichten kunnen ook een hogere of lagere startprijs geven aan een aandeel waardoor het evenwicht hoger of lager gaat liggen. Het kijken of een nieuwsbericht geplaatst moet worden gaat via de volgende vergelijking:

var nieuwsCheck = function(){

var roll = Math.floor(Math.random()\*100)

if (roll <= odds) {

oudnieuws = 1;

nieuwsChoice();

}

else {

tries++;

odds+=0.1;

}

}

Als het willekeurige getal tussen de 0 en de 100 kleiner is dan de kans, komt er een nieuwsbericht. De kans wordt elke 2 seconden met 0.1 verhoogt

**Aandelenkoers**

De aandelenkoers wordt door willekeurige factoren bepaald als er geen invloed is van een nieuwsbericht. We hebben het stijgen of dalen van aandelen willekeurig gemaakt, omdat er ellendig veel werk bij komt kijken om cijfers te bepalen op heel veel factoren. Als we dit wel wilden doen waren we een aantal maanden langer bezig geweest. Wat ook een optie is om een voorspelbare aandelenkoers te bepalen is het integreren van jaarcijfers van grote bedrijven die wij in naam hebben geparodieerd. Deze cijfers zouden dan in een database moeten worden gestopt met bijbehorende invloeden en die op hetzelfde moment presenteren aan de speler. Zo kan de speler in betere mate een aandeelkoers voorspellen en zo beter beslissen wanneer hij of zij aandelen moet kopen en verkopen. Dit is allemaal niet gelukt en we hebben het dus bij willekeurigheid gehouden (behalve bij nieuwsberichten natuurlijk).

**var chance = Math.floor(Math.random()\*100);**

Hier wordt een variabele aangemaakt met de naam ‘chance’. ‘chance’ wordt bepaald door een willekeurig number dat wordt gekozen door (Math.random()\*100) dit nummer ligt tussen de 0 en 100. Vervolgens wordt dit nummer door Math.floor afgerond naar een geheel getal.

if(chance > daalkans[i]+20 && chance < daalkans[i]+45) {

// Minimale stijging

price2[i]+=chance/10-5

$("<span style=color:green class=AText id=getal> "+price2[i].toFixed(2)+" </span>").appendTo("#aandeel"+i); //De nieuwe prijs wordt veranderd door het willekeurige getal te vergelijken met de huidige prijs

}

Vervolgens worden dit soort vergelijkingen uitgevoerd. Dit is de vergelijking van de minimale stijging. Er wordt gekeken of de ‘chance’ voldoet aan de voorwaarden in het if() statement. Als dit het geval is dan wordt price2[i] met een waarde van de ‘chance’/10-5 tot een nieuwe price2[i]. Vervolgens wordt dit nieuwe getal geplaatst op de gewenste plek zodat de speler te zien krijgt dat de prijs van het aandeel (in dit geval) is verlaagd. Voor een maximale stijging, minimale daling en maximale daling wordt op een soortgelijke manier alles bepaald. Alleen dan wordt de ‘price2[i]’ stijging of daling via een andere formule geregeld en de voorwaarden waaraan ‘chance’ moet voldoen in het if() statement zijn anders. Er is ook een mogelijkheid dat de prijs niet daalt en dat is in het gebied van het geval dat ‘chance’ zich tussen de 70 en 30 bevindt.

else { // Geen stijging

$("<span stye=color:black class=AText id=getal> "+price2[i].toFixed(2)+" </span>").appendTo("#aandeel"+i);

}

Als ‘chance’ niet voldoet aan de voorwaarden van een van de stijgingen of dalingen wordt alleen de onveranderde huidige prijs geprint.

Het was mogelijk voor prijzen van de aandelen om negatief te worden. Dit is natuurlijk niet de bedoeling een daar hebben we dit stukje code op bedacht dat ook zorgt dat de prijzen om een evenwicht balanceren.

// BALANCEREN VAN DE PRIJZEN

if (price2[i] <= 20) {

daalkans[i] -= 20;

}

else {

daalkans[i] = 50;

}

Wat we hebben gedaan om het negatief worden van de prijzen is om elke twee seconden te checken of de prijs kleiner of gelijk is aan twintig. Als dat geval is dan wordt elke twee seconden de daalkans met twintig, wat dit doet is de daalkans die standaard op vijftig staat verlagen met twintig elke 2 seconden. Dit heeft als gevolg dat in de vergelijking die bepaald of een aandeel daalt of stijgt, ‘chance’ vrijwel niet meer voldoet aan de voorwaarden voor de minimale daling en maximale daling.

else if (chance < daalkans[i]-20 && chance > daalkans[i]-44) {

// Minimale daling

price2[i]-=chance/10+2;

$("<span style=color:red class=AText id=getal> "+price2[i].toFixed(2)+" </span>").appendTo("#aandeel"+i);

}

else if (chance < daalkans[i]-45) {

// Grote daling

if (chance <= 5) {

price2[i]-=chance\*3;

}

else {

price2[i]-=chance/3;

}

$("<span style=color:red class=AText id=getal> "+price2[i].toFixed(2)+" </span>").appendTo("#aandeel"+i);

}

Zie hier dat ‘chance’ kleiner moet zijn dan een daalkans van 30 (daalkans[i] is standaard 50) om überhaupt te dalen. Als daalkans elke 2 seconden met 20 daalt heb je al na 4 seconden een negatieve daalkans. Negatieve getallen worden automatisch positief gemaakt waardoor uiteindelijk een ontzettend hoge stijgkans ontstaat en zo worden aandelen meteen omhoog gebracht in prijs.

if (price2[i] >= startprijs[i] \* 1.1 && price2[i] < startprijs[i] \* 1.2) {

daalkans[i] += 5;

}

else if (price2[i] >= startprijs[i] \* 1.2 && price2[i] < startprijs[i] \* 1.3) {

daalkans[i] += 7.5;

}

else if (price2[i] >= startprijs[i] \* 1.3) {

daalkans[i] += 10;

}

else if (price2[i] <= startprijs[i] \* 0.9 && price2[i] > startprijs[i] \* 0.8) {

daalkans[i] -= 5;

}

else if (price2[i] <= startprijs[i] \* 0.8 && price2[i] > startprijs[i] \* 0.7) {

daalkans[i] -= 7.5;

}

else if (price2[i] <= startprijs[i] \* 0.7) {

daalkans[i] -= 10;

}

Dit is de balancering van de prijzen. Hier worden de startprijzen x getallen gedaan om percentages van de startprijzen te krijgen (\* 1.1 geeft een startprijs die 10% hoger is, \* 1.2 wordt 20%, \* 1.3 wordt 30%, \* 0.9 wordt 90%, \* 0.8 wordt 80%, \* 0.7 wordt 70%). Als price2[i] voldoet aan een van deze voorwaarden wordt de daalkans[i] aangepast zodat de daalkans groter wordt of kleiner wordt (een kleinere daalkans geeft een grotere kans om te dalen en vice versa). Door dit zo toe te passen blijven aandelen schommelen om een evenwicht. Dit evenwicht wordt bepaald door de startprijs[i]. Elk aandeel heeft aan het begin van het programma een startprijs gekregen die door nieuwsberichten beïnvloed kunnen worden.

**Console**

De console is een simpel stukje code wat de laatste 7 variabelen van een array plaats in een daarvoor bestemde div via de volgende code.

var consoleText = ["","","","",""]; //Deze array zorgt ervoor dat de console weet wat hij moet neerzetten

var consoleUpdate = function(text) { //Deze functie update de console, en neemt text als input

var number = consoleText.length //De lengte van de console wordt berekend

consoleText.push(text); //De text die je gedisplayed wilt hebben wordt toegevoegd aan de array `

//Alle oude console text wordt weggehaald

if (nieuwscolor === "green" && nieuwsdisplay === 1){

$("<span class=cText style=color:green >["+derp+"] "+consoleText[number]+"<br></span>").appendTo("#console\_log");

nieuwsdisplay = 0;

}

else {

$("<span class=cText style=color:black >["+derp+"] "+consoleText[number]+"<br></span>").appendTo("#console\_log");

}

if (number === 40){

$(".cText").remove();

consoleText.splice(0,20);

}

$("#console\_log").animate({ scrollTop: $(document).height() }, "fast");

return false;

}

Als de array 40 variabelen heeft worden de eerste 20 verwijderd waardoor de console soepeler werkt. Als de speler de upgrade heeft gekocht en er nieuws wordt afgebeeld, wordt de tekst in het groen geplaatst. Anders gewoon in het zwart. De .animate() functie zorgt ervoor de je kan scrollen in de console. De console wordt opgeroepen met

consoleUpdate(“Text die in de console moet komen”);

**De winkel**

In de winkel kunnen speler upgrades kopen die het spel makkelijker maken, zoals gekleurde tekst, of een notificatie bij een hoge prijs. Elke upgrade heeft een variabele die 0 is als de upgrade niet is gekocht, en 1 als de speler het heeft gekocht. Deze variabele wordt lokaal opgeslagen. Als de speler de upgrade heeft gekocht verdwijnt de koop button.

<div class="voorwerp border" id="nieuwsUpgrade"><span id="upgrade1" class="voorwerp\_text">Koop</span></div>

Als de speler op de klop van de upgrade klikt, en de speler meer dan 300 geld heeft dan de upgrade kost, en de upgrade nog niet heeft wordt de upgrade gekocht. De variabelen wordt dan op 1 gesteld.

De javascript ziet er als volgt uit:

$("#nieuwsUpgrade").click(function(){

if ( money <= 1300 ){

consoleUpdate("Volgensmij is dat niet zo verstandig");

}

else{

if ( money>=1000 && nieuwsUpgrade === 0){

money-=1000;

nieuwsUpgrade = 1;

consoleUpdate("Nieuwsberichten komen voortaan in het groen");

nieuwscolor="green";

$("#nieuwsUpgrade").addClass( "gekocht" );

$('#upgrade1').remove();

$('<span class="voorwerp\_text">Gekocht</span>').appendTo('#nieuwsUpgrade');

}

else if (nieuwsUpgrade === 1){

consoleUpdate("Deze upgrade heb je al")

}

else {consoleUpdate("Je hebt niet genoeg geld");}

}

});

if(nieuwsUpgrade === 1){

$("#nieuwsUpgrade").addClass( "gekocht" );

nieuwscolor = "green";

$('#upgrade1').remove();

$('<span class="voorwerp\_text">Gekocht</span>').appendTo('#nieuwsUpgrade');

}

**Logboek Isha**

|  |  |  |
| --- | --- | --- |
| Datum | Activiteit | Uur |
| 2-11-2013 | Bedenken van een onderwerp | 0,5 |
| 3-11-2013 | Een voorbeeld maken van een aandeel | 2 |
| 4-11-2013 | Bespreken van het onderwerp met Rudolf, Rudolf had al te veel leerlingen om te begeleiden, waardoor we opzoek moesten naar een nieuwe begeleider | 1 |
| 8-11-2013 | We hebben een gesprek gehad met Leon, die was erg enthousiast en wou graag onze begeleider zijn | 1 |
| 9-11-2013 | Een begin maken met het programmeren, een aandeel wat je kan kopen en verkopen, alles kopen en verkopen door op shift te klikken | 1,5 |
| 11-11-2013 | Examenweek, het werk is tijdelijk gestaakt | 0 |
| 17-11-2013 | Uitzoeken hoe de canvas werkt zodat we grafieken kunnen tekenen van de aandelen | 2 |
| 17-11-2013 | Het schrijven van de console | 3 |
| 20-11-2013 | Oefenen met localstorage | 4 |
| 21-11-2013 | Ervoor zorgen dat de canvas zichzelf wist als deze vol is | 1 |
| 23-11-2013 | De variabelen average en profit toevoegen, die de gemiddelde inkoopprijs en winst bijhouden | 3 |
| 24-11-2013 | Leren hoe localstorage werkt zodat alle variabelen worden opgeslagen | 5 |
| 24-11-2013 | Save() en Load() functies schrijven | 1 |
| 26-11-2013 | Meerdere aandelen toevoegen | 2 |
| 30-11-2013 | Een formule schrijven die betrekking heeft op alle aandelen | 7 |
| 1-12-2013 | Een formule schrijven die betrekking heeft op alle canvassen | 4 |
| 6-12-2013 | Leren over data-type inplaats van ID's | 2 |
| 14-12-2013 | Optimalisering van de code | 1 |
| 16-12-2013 | Ervoor zorgen dat alle canvassen beschikbaar zijn via hetzelfde menu, het veranderen van variabelen waardoor de code veel soepeler verloopt | 6 |
| 17-12-2013 | Het inprogrammeren van nieuwsberichten, een functie schrijven die willekeurig wordt uitgevoerd en een nieuwsbericht start | 6 |
| 18-12-2013 | Het schrijven van 5 nieuwsberichten voor alle 16 aandelen | 5 |
| 19-12-2013 | Het optimaliseren van de code door copy paste werk te vervangen door functies | 3 |
| 2-1-2014 | Het uitleggen van alle programmeertalen in het werkstuk | 2 |
| 2-1-2014 | Het uitleggen van alle veelgebruikte stukjes code | 3 |
| 3-1-2014 | Werken aan de beschrijvingen van de code | 3 |
| 4-1-2014 | Het maken van een winkel, uitleggen hoe de code werkt | 3 |
| 5-1-2014 | Beschrijven van de code | 3 |
| 6-1-2014 | Het aanpassen aan de website zodat het past op schoolcomputers | 4 |
| 6-1-2014 | Printen van het eindverslag | 1 |
| Totaal |  | 80 |

**Logboek Floris**

|  |  |  |
| --- | --- | --- |
| Datum | Logboek Floris |  |
| 2-11-2013 | Bedenken van een onderwerp | 0,5 |
| 3-11-2013 | Een voorbeeld maken van een aandeel | 2 |
| 4-11-2013 | Bespreken van het onderwerp met Rudolf, Rudolf had al te veel leerlingen om te begeleiden, waardoor we opzoek moesten naar een nieuwe begeleider | 1 |
| 8-11-2013 | We hebben een gesprek gehad met Leon, die was erg enthousiast en wou graag onze begeleider zijn | 1 |
| 9-11-2013 | Een begin maken met het progammeren, een aandeel wat je kan kopen en verkopen, alles kopen en verkopen door op shift te klikken | 1,5 |
| 11-11-2013 | Examenweek, het werk is tijdelijk gestaakt | 0 |
| 17-11-2013 | Uitzoeken hoe de canvas werkt zodat we grafieken kunnen tekenen van de aandelen | 2 |
| 18-11-2013 | HTML en CSS herhaalt om weer een beetje in de flow te raken. | 2 |
| 19-11-2013 | jQuery doorgenomen zodat ik ook de code kon volgen die al geschreven was, aanpassingen en toevoegingen kon maken. | 2,5 |
| 26-11-2013 | Met Isha do code doorgenomen zodat ik op dezelfde manier te werk kan gaan. | 1 |
| 4-12-2013 | Gewerkt aan de layout. Verschillende bronnen opgezocht om te helpen in het waarmaken van de site. | 2,5 |
| 5-12-2013 | Aan de lay-out gewerkt en ook geprobeerd accordion te kunnen maken. Dat is gelukt alleen moet het er nog grafisch mooi uitzien. | 2 |
| 6-12-2013 | Gewerkt aan accordion en kleuren aangepast. Ook de namen van aandelen en bedrijven verzonnen. Ook met Leon gepraat over hoe of wat ons plan is. | 2 |
| 9-12-2013 | Gewerkt aan layout. Nieuw bestand gemaakt voor een soort template voor het aandeel scherm. Dit template moet nog afgemaakt worden. Moet hier nog over in gesprek met isha over hoe en wat. | 2,5 |
| 10-12-2013 | Gewerkt aan layout van de website met een beetje script om het speler scherm volledig te laten kloppen. | 3,5 |
| 11-12-2013 | Verder gewerkt aan de layout van de aandeel pagina en achtergronden uitgekozen om het er grafisch beter uit te laten zien. Ook minimale aanpassingen gemaakt aan het script om het laden van de speler info goed te laten lopen. | 4 |
| 12-12-2013 | Gewerkt aan de layout van de index en alle kleuren van de aandeelpagina overgeplaatst. | 2 |
| 16-12-2013 | Gewerkt aan de lay-out. De aanpassingen in het html en css bestand kloppend gemaatk met het script. Enkele bugs die in het script zaten opgelost. OVerlegt hoe we het willen aanpakken qua html bestanden. We hebben uiteindelijk besloten om alles in een html bestand te doen. Mede omdat het met canvas niet mogelijk was om dat over verschillende bestanden te verspreiden. We hebben alle canvassen en variabelen in een html bestand verwerkt en alles werkt. Er moet nog een save functie aan bepaalde dingen worden toegevoegd zodat die waarden worden opgeslagen. Ook nagedacht over dingen die we extra kunnen toevoegen mochten we voortijdig klaar zijn (wat waarschijnlijk het geval is). | 5,5 |
| 18-12-2013 | Een tabel toegevoegd die de actuele beursstand van alle bedrijven tegelijk bijhoudt en de site geoptimaliseerd zodat de site minder langzaam is. De opstelling van verschillende dingen in volgorde veranderen zodat de site er wat beter uit ziet. Logo's gemaakt voor de bedrijven zodat die bij de bedrijven kan worden weergegeven. Nieuwsberichten bedacht die de beurskoers van bedrijven beïnvloedt. | 7,5 |
| 25-12-2013 | Nieuwsberichten van follops toegevoegd aan de al bestaande nieuwsberichten. Ook erg lang gestoeid met de schaal van de canvas. Een overzicht gemaakt van wat nog gedaan moet worden. | 3,5 |
| 2-1-2014 | Gewerkt aan het script. Een admin functie toegevoegd waarmee we de daalkansen konden testen. De daalkansen in evenwicht gesteld, zodat voorkomen wordt dat aandelen negatieve bedragen kunnen worden. Alle nieuwsberichten toegevoegd. Groot stuk van de documentering afgemaakt (werkstuk). Toegevoegd dat aandelen 10-20-30% kunnen verschillen van de variabele startprijs. Startprijs kan met de nieuwsberichten worden beïnvloed als een bepaald bedrijf een hoger aanzien krijgt in de branche. De nieuwsberichten zijn in een apart .js bestand gezet voor een overzichtelijkere code. Gefixt dat de scale niet om de 2 seconden herlaad (wat een knipper effect gaf). Dit hebben we gedaan door een start variabele te maken die standaard op 1 staat. Zo wordt bij elke canvas standaard een keer de goede scale geprint, dit was voorheen niet zo als we van het knipperen af wilden komen. Er ontstond een probleem en dat was dat als een canvas een nieuwe scale nodig had die alle scales die geprint waren door de variabele start verwijderde. Dit hebben we gefixt door start een inhoud te geven van 16 variabele waardoor er op specifieke plekken een scale wordt toegevoegd en verwijdert. | 10,25 |
| 3-1-2014 | Gewerkt aan nieuwe lay-out. Scrollbar gemaakt. Groot probleem met de switch ivan de nieuwsberichten wat na anderhalf uur was gefixt. De fix was break; toevoegen na elke case wat dus niet overal stond. De consolelog gedetailleerder maken. Bij de navigator nieuwsberichten toegevoegd onder nieuws met notificatie afbeelding erbij. | 7 |
| 4-1-2014 | Gewerkt aan de shop. De tutorial gemaakt. Puntjes op de i gezet. Dingen voor in de shop gemaakt en toegepast. De shop gestylleerd. Een favicon gemaakt. | 14,5 |
| 5-1-2014 | CSS, HTML en wat jquery uitgelegd in de documentering. | 4 |
|  |  | 80,25 |

**Bronnenlijst**

**Accordion:**

http://stackoverflow.com/questions/14460231/how-do-you-import-multiple-javascript-files-in-html-without-the-bloat

http://www.youtube.com/watch?v=hvOqlCha1SQ

http://jqueryui.com/accordion/#default

http://www.youtube.com/watch?v=uHixbmuVfak

**Vormgeving en kleuren:**

http://www.colorzilla.com/gradient-editor/

http://www.colorpicker.com/

<http://www.handleidinghtml.nl/html/karakters/karakters03.html>

**Canvas:**

<http://www.w3schools.com/html/html5_canvas.asp>