Javascript

voor Software Developer

ICT Lyceum, Deltion College, Zwolle

31 augustus 2022

Auteurs: Docenten ICT-Lyceum, Deltion College.

Deze opdracht is geschreven met de volgende tools: LaTeX, Sublime text, Visual Studio Code, TeXstudio, IntelliJ IDEA Ultimate, sourcetree, gitkraken, vi en Git. Allemaal op Fedora Linux, Apple MacOS en Microsoft Windows.

Inhoudsopgave

Hoofdstuk 1

Introductie

In vorige projecten heb je al kennis gemaakt met HTML en CSS. Met HTML kun je de structuur van een webpagina bepalen. CSS kun je vervolgens gebruiken om de opmaak van de pagina te bepalen. Hiermee bepaal je dus de kleuren, het lettertype en de lettergrootte.

Wanneer je kennis hebt van HTML en CSS kun je een webpagina bouwen. In dit project gaan we nog een extra functionaliteit toevoegen aan de webpagina en dat doen we met JavaScript.

Hoofdstuk 2

Backlog

- Als webdeveloper wil ik weten wat Javascript is en waarvoor ik dit kan gebruiken.
- Als webdeveloper wil ik de basis van JavaScript kennen, zodat ik dit kan toepassen in door mij gemaakte websites.
- Als webdeveloper wil ik weten hoe ik JavaScript code kan debuggen, zodat ik fouten in mijn code of in code van andere programmeurs kan vinden en kan oplossen.
- Als webdeveloper wil ik JavaScript kunnen lezen zodat ik code van anderen kan controleren op fouten.
- Als webdeveloper wil ik javascript kunnen lezen zodat ik code van anderen kan beoordelen op bruikbaarheid in eigen projecten.
- Als student wil ik opdrachten maken met JavaScript, zodat ik kan leren hoe ik JavaScript in de praktijk kan toepassen.

Hoofdstuk 3

Javascript

Waarvoor wordt JavaScript gebruikt?

Naast HTML en CSS kunnen we JavaScript toevoegen aan een webpagina hierdoor wordt de pagina interactief. We kunnen met JavaScript afbeeldingen laten bewegen of menu's laten uitschuiven. Ook kunnen we met JavaScript bepalen wat er moet gebeuren wanneer een bezoeker op een knop klikt op de webpagina. JavaScript wordt voornamelijk gebruikt voor de onderstaande toepassingen in de browser.

- **Formuliervalidatie**
  JavaScript is erg geschikt om ingevulde gegevens in een webformulier op de pagina te controleren voordat het formulier wordt verzonden.
- **Dynamische menu's en afbeeldingen**
  Met JavaScript kunnen menu's, kleuren en afbeeldingen tijdens het gebruik van de pagina worden vervangen. Denk bijvoorbeeld aan uitklapmenu's of foto carrousels.
- **Aanpassingen van stijlen en animaties**
  Wanneer een webpagina geladen is in de browser, dan kan met JavaScript de aanwezigheid, positie en inhoud van elk element op de pagina opgehaald en aangepast worden. Denk hierbij aan het wijzigen van de pagina wanneer de bezoeker naar beneden scrolt.

JavaScript wordt als scripttaal vaak onderschat maar het is een krachtige taal. Zo kunnen met JavaScript bijvoorbeeld spelletjes of 3d animaties gemaakt worden. Om te zien hoe krachtig JavaScript is kun je de voorbeelden op de volgende webpagina bekijken: https://threejs.org/examples/#webgl_animation_skinning_blending (**threejs**)

Er zijn online natuurlijk nog veel meer voorbeelden van JavaScript te vinden. Naast HTML en CSS is JavaScript kennis belangrijk voor een webdeveloper.

JavaScript is geen Java

De programmeertalen JavaScript en Java worden nogal eens met elkaar verward omdat de namen sterk op elkaar lijken. Toch kan je deze vergelijking niet maken. JavaScript en Java zijn echt verschillende talen. Een aantal verschillen tussen Java en Javascript:

- Java is een programmeertaal, Javascript is een scripttaal.
- Java is server-side, JavaScript is client-side.
- Java-code moet, voordat je ze kan gebruiken eerst gecompileerd worden, Javascript is gewoon platte tekst.
- Java-applicaties kunnen zowel in een virtuele machine als in een browser draaien. Javascript code draait alleen in een browser.
- Een gecompileerd Java-programma krijgt de extensie .class. Een Javascript heeft de .js extensie.
- Java is ontwikkeld door Sun Microsystems, JavaScript door Netscape.

Hoofdstuk 4

Basiselementen

In dit hoofdstuk maken we kennis met de volgende JavaScript elementen:

- Alert
- JavaScript debuggen
- Write
- Confirm
- Prompt
- Functies
- Variabelen
- Operatoren

Alert

JavaScript wordt direct uitgevoerd in de browser van de bezoeker (client-side) De browser herkent JavaScript aan de <script>-tag. Om JavaScript in je webpagina te gebruiken dien je dit als volgt te plaatsen:

Listing 4.1: Script

```
1 <script> ........ </script>
```

Taak : Alert
Voorbereiding

- Maak in je `htdocs` folder een nieuwe folder aan voor deze opdracht.
- Zorg voor versiecontrole op deze folder.
- Plaats een `index.html` in deze folder en gebruik je favoriete editor om de code van listing 4.2 in dit bestand te plaatsen.
- Add en Commit in Git.
- Start de webpagina op en kijk wat er gebeurd.

Listing 4.2: alert

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport"
6      content="width=device-width, initial-scale=1.0">
7      <title>Interactieve HTML</title>
8  </head>
9  <body>
10     <h1>Interactieve HTML</h1>
11     <h2>Mijn eerste JavaScript</h2>
12     <script>
13         alert('Hello world...');
14     </script>
15 </body>
16 </html>
```

Debuggen van JavaScript code

Wanneer we tijdens het programmeren een type- of een syntaxfout maken dan kan het gebeuren dat we een lege webpagina te zien krijgen of dat het script niet werkt. We moeten dan teruggaan naar de code van het script zodat we deze

kunnen aanpassen. Dat opsporen en oplossen van fouten in de code noemen we debuggen.

Taak : Debugging

Ga terug naar de code van index.html (listing **??**) en maak een typefout, wijzig alert in alert.

Listing 4.3: aler

```
1 <script>
2     aler('hello world...');
3 </script>
```

Sla vervolgens het script op en roep deze opnieuw op in de browser. Het script werkt nu niet meer en de alertbox wordt niet getoond. Alle browsers hebben hulpmiddelen die ons helpen om de gemaakte fouten op te sporen. We noemen dit Debuggen. Dit debuggen kan via de console. In Google Chrome kunnen we deze oproepen met F12 (of rechtermuisknop ->"Inspecteren"). We krijgen dan de JavaScript console te zien. De console geeft aan welke fouten het script bevat.

Debugger Commando

JavaScript heeft nog een extra commando om fouten in de code op te sporen. Dit is het debugger commando. Met het debugger commando is het mogelijk om tijdens het uitvoeren van de code deze tijdelijk te onderbreken met een breakpoint om deze daarna weer door te laten lopen. Meer informatie over het debuggen van JavaScript code en het debugger commando kun je vinden op: https://www.w3schools.com/js/js_debugging.asp (**w3schoolsjsdebugger**

Bezoek deze pagina en bestudeer hoe het debuggen van JavaScript code werkt.

Ingebouwde methodes

Alle programmeertalen hebben een ingebouwde library met methodes waarmee we de meest voorkomende taken kunnen uitvoeren. Een voorbeeld hiervan is de alert methode van de vorige opdracht. JavaScript heeft nog meer methodes een aantal van deze methodes gaan we hieronder behandelen.

document.write()

Deze methode kunnen we gebruiken om teksten of HTML-elementen vanuit JavaScript op het scherm te tonen. De `write`-methode kan als volgt gebruikt worden:

Listing 4.4: Write

```
1 <script>
2     document.write('hier komt jouw tekst');
3 </script>
```

Op de website https://www.w3schools.com/jsref/met_doc_write.asp (**w3schools-write**) kan je hierover meer informatie vinden. Zoek dit uit.

confirm()

Soms is het nodig om feedback van een gebruiker op te vragen. Een manier omdat te doen is de `confirm`-methode. Deze methode kan als volgt gebruikt worden:

Listing 4.5: Confirm

```
1 <script>
2     confirm("Wilt u deze weizigingen doorvoeren?");
3 </script>
```

Op de website: https://www.w3schools.com/jsref/met_win_confirm.asp (**w3schools-configm**) kan je hierover meer informatie vinden. Zoek dit uit.

prompt()

Met de promptbox kun je de gebruiker om input vragen. De `prompt`-methode kan als volgt gebruikt worden:

Listing 4.6: Prompt

```
1 <script>
2     prompt("Wat is uw leeftijd?");
```

```
3 </script>
```

Op de website: https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_prompt (**w3schools-promtp**) kan je hierover meer informatie vonden. Zoek dit uit.

Variabelen

Net als in PHP is het ook in JavaScript mogelijk om variabelen te gebruiken. Door variabelen te gebruiken is het mogelijk om bijvoorbeeld te gaan rekenen met getallen.

Op de website: https://www.w3schools.com/js/js_variables.asp (**w3schools-variabele**) kan je hierover meer informatie vinden. Zoek dit uit.

Functies

Soms heb je stukjes code nodig die op meerdere plekken op je website of in je programma gebruikt moeten worden. Denk bijvoorbeeld aan het berekenen van een Btw-bedrag in een webshop. Je kan deze code dan op verschillende plekken in je programma plaatsen, maar wanneer je dan wat wilt aanpassen in dit stukje code, dan dien je dat op verschillende plekken in je programma te doen. De kans dat je een fout maakt is dan erg groot.

Om dit te voorkomen kan je gebruik maken van functies. Functies zijn stukjes code die buiten de rest van het script geschreven worden. De code in een functie wordt alleen aangeroepen wanneer jij daar expliciet om vraagt. Met een functie kan je bijvoorbeeld bepalen welk script er uitgevoerd moet worden of wanneer een bezoeker op een bepaalde button op je website klikt.

De onderstaande code is een voorbeeld hoe je met JavaScript een functie op een knop kan plaatsen.

Listing 4.7: Button

```
1 <!DOCTYPE html>
2 <html lang="nl">
3 <head>
4     <meta charset="UTF-8">
```

```
5     <title>Interactieve HTML</title>
6     <script>
7         function buttonClick()
8         {
9             alert('Er is op de knop geklikt...');
10        }
11    </script>
12 </head>
13 <body>
14    <input type="button" value="Klik op deze knop"
15    onclick="buttonClick()" />
16 </body>
17 </html>
```

Een functie om het btw-bedrag (uitgaande van een btw-percentage van 21%) te berekenen kan er als volgt uitzien:

Listing 4.8: BTW bereken function

```
1 <script>
2     function berekenBtw(bedragExBtw)
3     {
4         return bedragExBtw * 0.21;
5     }
6 </script>
```

Als we deze functie bestuderen dan zien we dat functie wordt aangeroepen met een bedragExBtw. De functie geeft vervolgens het btw-bedrag terug dmv. het `return`-commando. Wanneer we met deze functie het btw-bedrag van €100,- willen uitrekenen dan dienen we deze functie als volgt aan te roepen:

Listing 4.9: Function

```
1 <script>
2     berekenBtw(100);
3 </script>
```

De functie geeft vervolgens de uitkomst terug en zal in dit geval €21,- returnen.

Op de website: https://www.w3schools.com/js/js_functions.asp (**w3schools-jsfunctions**) kan je meer informatie vinden over functies. Zoek dit uit.

Hoofdstuk 5

Meer verdieping in JavaScript

Maak de cursus “Learn JavaScript ”op Codecademy. https://www.codecademy.com/learn/introduction-to-javascript (**codecademy-javascript**) om je kennis over JavaScript te verdiepen.

Opdracht 1 : Getallen

- Gebruik je favoriete editor om de code in listing 5.1 in het bestand `getallen.html` te plaatsen.
- Commit `getallen.html` in Git.
- Start de webpagina op en kijk wat er gebeurt.
- Pas het script aan zodat je twee getallen kunt invoeren.
- Wanneer op de knop gelikt wordt, moeten de twee getallen opgeteld worden, de uitvoer moet zichtbaar gemaakt worden met een alert box.
- Commit

Listing 5.1: Opdracht1

```
1 <!DOCTYPE html>
2 <html lang="nl">
3 <head>
4   <meta charset="UTF-8">
```

```
5   <title>Interactieve HTML</title>
6   <script>
7     function checkForm(){
8       let getal = document.getElementById('getal')
9       .value;
10      if(getal < 0 || getal >=10){
11        alert('Het ingevulde getal is onjuist...');
12      }else{
13        alert('Het ingevulde getal is juist...');
14      }
15    }
16  </script>
17  </head>
18
19  <body>
20     <h1>Oefenen met JavaScript</h1>
21
22     Vul een getal tussen 0 en 10 in:
23     <br /><br />
24     <label for="getal">Getal:</label><br>
25     <input type="text" id="getal" name="getal"><br>
26     <br />
27     <input type="button" value="Controleer getal"
28     onclick="checkForm()" />
29
30  </body>
31  </html>
```

Interactieve teksten aanpassen

Met JavaScript is het ook mogelijk om zonder de pagina te vernieuwen teksten en afbeeldingen te vervangen. Dit kan op de volgende manier gebruikt worden:

Listing 5.2: Interactieve tekst

```
1  <!--JavaScript-->
2  <script>
3     document.getElementById("textDiv").innerHTML =
4  "Uw ingevoerde postcode is onjuist."
```

```
5 </script>
6
7 <!--In HTML-->
8 <div id="textDiv"></div>
```

Op de website https://www.w3schools.com/html/tryit.asp?filename=tryhtml_script" (**w3schools-tekstaanpassen**) kan je hierover meer informatie vinden. Zoek dit uit.

Taak : Interactieve Tekst

Open het bestand `getallen.html` uit de vorige opdracht en pas de code aan zodat de uitkomst op het scherm getoond wordt en niet in een alert box. De tekst moet dus in een div zijn genest. Test je programma en commit.

Formulieren valideren met JavaScript

Een veel gebruikte toepassing van JavaScript is het valideren van gegevens in formulieren. Wanneer een bezoeker bijvoorbeeld een postcode of een telefoonnummer moet invullen dan is het mogelijk om voordat het formulier verzonden wordt, met JavaScript te controleren of de ingevulde gegevens correct zijn. Een voorbeeld van formulier validatie met JavaScript kun je vinden op https://www.w3schools.com/js/js_validation.asp (**w3schools-validation**). Bestudeer deze code en oefen ermee.

Hoofdstuk 6

Opdrachten

Door middel van een aantal opdrachten ga je nu proberen je kennis over JavaScript in de praktijk te brengen. Maak deze opdrachten, test je programma's en commit je code in git. Zorg voor nette leesbare code. Plaats CSS en JavaScript in aparte bestanden.

- Opdracht 2 Een toets.
- Opdracht 3 Memory.
- Opdracht 4 Steen, Papier, Schaar.

Opdracht 2: Een toets

Een docent vraagt of je hem wilt helpen met de controle van een toets. Hij wil dat studenten een antwoord kunnen invoeren in een webpagina. De webpagina moet na het invoeren van de antwoorden meteen controleren of het antwoord correct is.

Vragen met antwoorden:

De toets moet de volgende vragen bevatten:

1. Vraag: Wat is de hoofdstad van Spanje? Antwoord: Madrid.
2. Vraag: In welke zee ligt het eiland Mallorca? Antwoord: Middellandse zee.
3. vraag Hoeveel paar poten heeft een duizendpoot? Antwoord: 15 paar.

Opdracht:

1. Maak een pagina waar een student de vragen kan beantwoorden.
2. Maak een knop met de tekst ”controleer antwoorden”. Zodra je op de knop klikt worden de invoer velden gecontroleerd op correctheid en word het resultaat weergegeven aan de gebruiker.

Opdracht 3 : Memory

In deze opdracht gaan we het spel ”memory”programmeren met HTML/CSS en JavaScript. Om je op weg te helpen krijg je een basis script die je kan gebruiken om het spel uit te bereiden en verder af te maken.

1. Gebruikt je favoriete IDE of editor om de onderstaande code in het bestand ./javascript/memory/index.html te plaatsen. (Listing: 6.1)
2. Commit dit in git.
3. Start de webpagina en kijk wat er gebeurd.
4. Vervang de alert boxen in het script door interactieve teksten ( ).
5. Zoek afbeeldingen die je wilt gebruiken voor het memory spel. Denk aan het auteursrecht.
6. Pas het script aan, zodat het spel werkt met de door jouw gekozen afbeeldingen.
7. Pas het script aan, zodat het spel werkt met minimaal 8 memory kaarten.

8. Alle kaarten liggen nu altijd op dezelfde volgorde. Pas het script aan zodat bij elk nieuw spel dat gespeeld wordt de kaarten op een willekeurige plek in het spel liggen.

   Je mag hiervoor onderstaande functie shuffleCards gebruiken (Listing 6.2), maar je mag het natuurlijk ook zelf programmeren.

9. Pas het script aan, zodat er bij elke klik op een kaart een geluidje afgespeeld wordt. Je kan hiervoor zelf een geluidje downloaden. (Bijvoorbeeld op de volgende website: https://blog.felgo.com/game-resources/16-sites-featuring-free-game-sounds Een voorbeeld hoe je met JavaScript geluiden kan afspelen kan je vinden op: https://www.w3schools.com/JSREF/met_audio_play.asp (**felgo-sounds**)

10. Voeg een variabele toe aan het script waarmee geteld wordt hoe vaak er geklikt is, toon deze variabele na elke klik op het scherm.

Listing 6.1: JavaScript Index

```
<html>
<head>
    <title>Simple Memory</title>
    <style>
        h1 {
            text-align: center;
            font-family: verdana;
            font-size: 25px;
            color: white;
        }
        .memoryContainer {
            width: 240px;
            height: 325px;
            border: 1px solid #E9E9E9;
            margin: auto;
            background-color: #666;
        }
        .memoryCard{
            width:100px;
            height: 100px;
            background-color: #0099FF;
            margin:10px;
            border:1px solid #999999;
            font-size:25px;
            font-weight:bold;
            color:white;
            float:left;
        }
    </style>
    <script>
        var userClick1 = 0;
        var userClick2 = 0;
        var userSelectedCard1 = "";
        var userSelectedCard2 = "";
        function checkClickedCard(cardNum, cardId){
            //Geklikte kaart disablen, want je mag maar een
                ↪ keer op dezelfde kaart klikken
            document.getElementById(cardId).disabled = true;
            //juiste waarde op de kaart zetten
```

```
39              document.getElementById(cardId).value = cardNum;
40              //document.getElementById(cardId).style = "
                    ↪ background: url(myimage.png)";
41
42              //Lees en onthoudt klik1 en klik2
43              if(userClick1==0){
44                  userClick1 = cardNum;
45                  userSelectedCard1 = cardId;
46              }else{
47                  userClick2 = cardNum;
48                  userSelectedCard2 = cardId;
49              }
50
51              if(userClick2!=0){
52                  if(userClick1==userClick2){
53                      alert('Goed zo...');
54                  }else{
55                      alert('Fout...');
56                      document.getElementById(userSelectedCard1).
                            ↪ disabled = false;
57                      document.getElementById(userSelectedCard2).
                            ↪ disabled = false;
58                      document.getElementById(userSelectedCard1).
                            ↪ value = "";
59                      document.getElementById(userSelectedCard2).
                            ↪ value = "";
60                  }
61
62                  userClick1 = 0;
63                  userClick2 = 0;
64              }
65          }
66
67      </script>
68  </head>
69  <body>
70
71      <div class="memoryContainer">
72          <h1>Memory</h1>
73          <input type="button" value="" id="kaart1" class="
                ↪ memoryCard" onclick="checkClickedCard(1, 'kaart1
```

```
            ↪ ');" />
74          <input type="button" value="" id="kaart2" class="
               ↪ memoryCard" onclick="checkClickedCard(2, 'kaart2
               ↪ ');"/>
75          <input type="button" value="" id="kaart3" class="
               ↪ memoryCard" onclick="checkClickedCard(1, 'kaart3
               ↪ ');"/>
76          <input type="button" value="" id="kaart4" class="
               ↪ memoryCard" onclick="checkClickedCard(2, 'kaart4
               ↪ ');"/>
77      </div>
78
79  </body>
80  </html>
```

Function shuffleCards()

Onderstaande functie kun je gebruiken om de kaarten willekeurig in het spel te tonen. Je mag natuurlijk ook je eigen functie hiervoor programmeren.

Listing 6.2: function shuffleCards()

```
1   function shuffleCards(){
2       document.getElementById("memoryCards").innerHTML = '';
3       cardsArr = new Array();
4       for(i=0;i<numberOfCards / 2; i++){
5           cardsArr.push(i);
6           cardsArr.push(i);
7       }
8       while(cardsArr.length>0){
9           var randomNumber = Math.floor(Math.random() * (cardsArr
               ↪ .length));
10          document.getElementById("memoryCards").innerHTML += '<
               ↪ input type="button" value="' + cardsArr[
               ↪ randomNumber] + '" id="kaart' + cardsArr.length +
               ↪  '" class="memoryCard" onclick="checkClickedCard(
               ↪ ' + cardsArr[randomNumber] + ', \'kaart' +
               ↪ cardsArr.length + '\');" />';
11          cardsArr.splice(randomNumber,1);
12      }
```

```
13 }
```

Opdracht 4 : Steen-Papier-Schaar

We kennen allemaal hoe steen-papier-schaar werkt. In deze opdracht gaan je dit spel nabouwen in JavaScript.

Werking van het spel:

Steen-papier-schaar kent 2 spelers. Beide spelers steken tegelijkertijd de hand naar voren in een bepaalde vorm. Het doel van het spel is om de vorm te kiezen die je tegenstander verslaat.

Beide spelers tellen tegelijk af en steken tegelijk zonder aarzelen de hand uit in een van de volgende vormen:

1. Een vuist (= steen), wint van de schaar door deze bot te maken.
2. Een vlakke hand (= papier), wint van steen door deze te bedekken.
3. Twee gespreide vingers (= schaar), wint van papier door het te verknippen.

Het spel wordt doorgaans een afgesproken oneven aantal keren gespeeld (meestal 3 keer), zodat er geen gelijkspel mogelijk is. De winnaar is degene die de meeste winnende combinaties maakte.

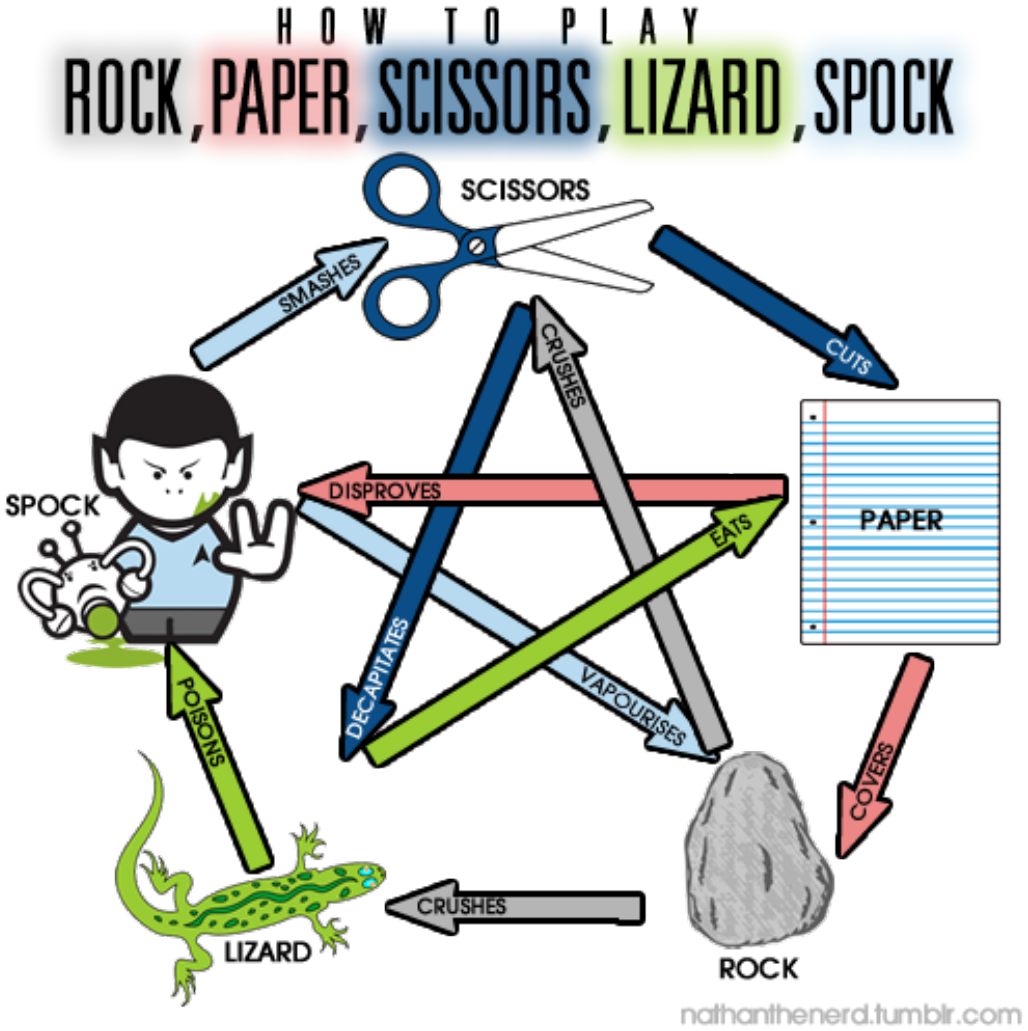
Opgave:

Programmeer met JavaScript het spel steen-papier-schaar, zodat je dit kan spelen tegen de computer. Je kan het spel drie keer spelen, daarna wordt de winnaar op het scherm getoond. Gebruik hiervoor HTML5, CSS3 en JavaScript.

Extra Uitdaging:

Wanneer je dit erg makkelijk vond is dit een extra uitdaging om je JavaScript skills te testen. In de show ”The Big Bang Theory”spelen de hoofdpersonen een andere versie van steen-papier-schaar. Deze versie noemen ze ”rock, paper, scissors, lizard, spock”. Het toevoegen van 2 extra opties biedt veel meer mogelijkheden waardoor een geleikspel minder vaak voor zal komen.

Deze versie heeft wel een hoop meer mogelijkheden, vormen en regels. Deze regels staan uitgelegd in figuur 6.1 op pagina 24.



Figuur 6.1: Rock-Paper-Scissors-Lizard-Spock (**nathanthenerd**)

Hoofdstuk 7

Afronding

- Gebruik de rubric om te bepalen of je een voldoende of goed hebt voor deze opdracht.
- Deel de code van deze opdracht via Github met de docenten.
- Bespreek een reviewmoment bij een docent.
- Bereidt je voor op het reviewmoment.
    1. Ken jouw code.
    2. Weet waar de functionaliteit van jouw programma's gecodeerd is.
    3. Weet hoe jouw programma's werken.
    4. Verwacht vragen om de code aan te passen.
    5. Verwacht vragen om de code uit te leggen.

Hoofdstuk 8

Planning Javascript

Je kunt onderstaande sjabloon gebruiken voor jouw eigen planning. Neem onderstaande sjabloon daarvoor over in jouw favoriete planningtool en voeg waar gewenst eigen regels toe. Zet de verschillende acties daarna op de juiste data in jouw agenda.

PS: Je kunt taken niet altijd na elkaar te plannen. Je zult taken ook moeten laten overlappen. Dit geldt ook voor taken in de verschillende opdrachten.

Actie	**Datum start**	**Datum klaar**	**Klaar**
Bestudeer Javascript basiselementen			
Verdiep je in Javascript			
Maak Javascript opdacht 1 - Getallen			
Maak Javascript opdracht 2 - Een toets			
Maak Javascript opdracht 3 - Memory			
Maak Javascript opdracht 4 - Steen-Papier-Schaar			
Afronding			

Hoofdstuk 9

Rubric Javascript

Opdracht 1 - Getallen

Werking			
Goed	*Voldoende*	*Matig*	*Onvoldoende*
Het programma werkt foutloos.	Het programma werkt, maar geeft waarschuwingen tijdens het uitvoeren.	Het programma werkt, maar geeft fouten tijdens het uitvoeren	Het programma werkt niet of kan niet gestart worden.
Code review			
Goed	*Voldoende*	*Matig*	*Onvoldoende*
Ik kan de werking van de source code foutloos en zonder hapering uitleggen.	Ik moet soms (niet hindelijk) nadenken over de source code van pagina.	Ik moet veel nadenken of antwoorden opzoeken bij het uitlegen van de source code.	Ik kan de source code niet uitleggen.

Opdracht 2 - Een toets

Werking			
Goed	*Voldoende*	*Matig*	*Onvoldoende*
Het programma werkt foutloos.	Het programma werkt, maar geeft waarschuwingen tijdens het uitvoeren	Het programma werkt, maar geeft fouten tijdens het uitvoeren.	Het programma werkt niet of kan niet gestart worden.
Code review			
Goed	*Voldoende*	*Matig*	*Onvoldoende*
Ik kan de werking van de source code foutloos en zonder hapering uitleggen.	Ik moet soms (niet hindelijk) nadenken over de source code van pagina.	Ik moet veel nadenken of antwoorden opzoeken bij het uitlegen van de source code.	Ik kan de source code niet uitleggen.

Opdracht 3 - Memory

opdracht 3 - Memory			
Goed	*Voldoende*	*Matig*	*Onvoldoende*
Het programma werkt foutloos.	Het programma werkt, maar geeft waarschuwingen tijdens het uitvoeren	Het programma werkt, maar geeft fouten tijdens het uitvoeren.	Het programma werkt niet of kan niet gestart worden.
Ik kan de werking van de source code foutloos en zonder hapering uitleggen.	Ik moet soms (niet hindelijk) nadenken over de source code van pagina.	Ik moet veel nadenken of antwoorden opzoeken bij het uitlegen van de source code.	Ik kan de source code niet uitleggen.

Opdracht 4 - Steen, papier, schaar

Opdracht 4 Steen, Papier, Schaar			
Goed	*Voldoende*	*Matig*	*Onvoldoende*
Het programma werkt foutloos en heeft de bonus opdracht verwerkt.	Het programma werkt foutloos.	Het programma werkt, maar geeft fouten tijdens het uitvoeren.	Het programma werkt niet of kan niet gestart worden.
Ik kan de werking van de source code foutloos en zonder hapering uitleggen.	Ik moet soms (niet hindelijk) nadenken over de source code van pagina.	Ik moet veel nadenken of antwoorden opzoeken bij het uitlegen van de source code.	Ik kan de source code niet uitleggen.

Index