

Front End

Onderdeel van Front End 4

- [Link binnen een pagina](#)
- [Tabellen en ChatGPT](#)
- [FE 2 - 05 Blokken](#)
- [DOM Manipulation](#)
- [DOM manipulation opdrachten](#)

Link binnen een pagina

Vraag

Hoe maak je een hyper link die verwijst naar de bovenkant van dezelfde pagina?

Antwoord

het is relatief eenvoudig om een link te maken die naar de bovenkant van dezelfde HTML-pagina verwijst. Dit kan worden gedaan door het `href` attribuut van de `<a>` tag in te stellen op `"#top"`. Meestal wordt hiervoor een HTML-element met `id="top"` aan het begin van de pagina geplaatst, zodat de link weet waar naartoe te verwijzen.

Hier is een eenvoudig voorbeeld:

```
<!DOCTYPE html>
<html>
<head>
  <title>Voorbeeldpagina</title>
</head>
<body>

<!-- Element aan het begin van de pagina -->
<div id="top"></div>

<!-- De inhoud van de pagina -->
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris.</p>

<!-- Meer inhoud ... -->

<!-- Link aan de onderkant van de pagina -->
<a href="#top">Terug naar boven</a>

</body>
</html>
```

In dit voorbeeld zal de link "Terug naar boven" de gebruiker naar het element met `id="top"` sturen, wat in dit geval aan het begin van de pagina staat. Dit is een veelgebruikte manier om een "terug naar boven" functionaliteit te implementeren op webpagina's.

Tabellen en ChatGPT

Een tabel begint en eindigt met `<table>` en `</table>`.

Regels

Elke tabel heeft een aantal regels of rijen.

Regels staan tussen `<tr>` en `</tr>`

Kolommen

Elke tabel heeft ook een aantal kolommen. Belangrijk is dat het aantal kolommen in de tabel altijd hetzelfde zijn.

Dus heb je in de eerste rij 7 kolommen, dan **moet** je in alle rijen van de tabel ook 7 kolommen maken!

Kolommen staan tussen `<td>` en `</td>`

Basisstructuur

Voor de basis gebruik je dus drie HTML tags.

```
<table border="1">
  <tr>
    <td>a1</td>
    <td>a2</td>
    <td>a3</td>
  </tr>
  <tr>
    <td>b1</td>
    <td>b2</td>
    <td>b3</td>
  </tr>
</table>
```

Door bij de `table border=1` mee te geven, kunnen we beter zien of de tabel wel klopt. Deze tabel ziet er in de browser als volgt uit.

[image1693931409964.png](#)

Je ziet twee rijen (2x `<tr></tr>`) en drie kolommen.

Stel we maken in de eerste rij een kolom extra, dan zijn de kolommen niet meer in alle rijen gelijk. De tabel ziet er dan zo uit.

image-1693931488394.png

Aan de ontbrekend lijntjes kan je zien dat de tabel niet meer correct is.

Wat maakt het uit denk je misschien? Als de browser te veel fouten ziet dan raakt de hele tabel verward en dan wordt het lastig zoeken. Maak re dus een gewoonte van om je tabel te testen!

Header

Een header in tabel is gewoon een regel, maar wel eentje die net iets anders wordt afgedrukt.

Je kunt een header regel als volgt toevoegen:

```
<tr>
  <th>Header 1</th>
  <th>Header 2</th>
  <th>Header 3</th>
</tr>
```

image-1693931844682.png

Colspan

Stel je wilt op de derde regel b2 en b3 samenvoegen tot één kolom. Dat kan met colspan.

De code van de derde regel wordt dan:

```
<tr>
  <td>b1</td>
  <td colspan="2">b2</td>
</tr>
```

image-1693932084087.png

HTML 5

Als je HTML5 gebruikt, kun je ook overwegen om semantische tags zoals `<thead>` en `<tbody>` toe te voegen voor betere leesbaarheid en toegankelijkheid, maar dat is optioneel.

ChatGPT (3.5)

Als je aan ChatGPT de volgende vraag stelt krijg je een blue print van een tabel:

Maak een html tabel met een header en 4 rijen. De tabel bevat 7 kolommen.

Laat ChatGPT jou nog wat verder helpen met de volgende vraag.

Voeg bij elke regel commentaar toe om aan te geven welke regel het is.

En stel we willen kolommen samenvoegen, dan vragen we.

Kun je in de derde rij kolom 2 en 3 samenvoegen?

En als laatste kunnen we ook nog kolommen over één of meer rijen samenvoegen. We hebben dat niet uitgelegd, maar dat gaat met rowspan. Dit wordt vrij ingewikkeld, maar ook daar kan ChatGPT mee helpen.

Kun je de laatste kolom van rij 1, 2 en 3 samenvoegen?

Check je code!

Check de code wel zelf, jij blijft altijd verantwoordelijk voor juiste code. Kijk dus of er bij het samenvoegen van kolommen rowspan is gebruikt en check of met het samenvoegen van regels/rijen colspan is gebruikt.

Soms 'halucineert' ChatGPT en jij wilt dat jouw klant of opdrachtgever natuurlijk een goede website krijgt!

--

FE 2 - 05 Blokken

Deze uitleg hoort bij opdracht 5 van Front End 2

Check of je bestanden op de juiste plaats staat

[image.1668536345276.png](#)

Stap 1

Op pagina 317 staat hoe je de `` elementen naast elkaar kan krijgen.

Voer dat uit. Het resultaat ziet er dan als volgt uit.

[image.1668536449446.png](#)

Stap 2

Op pagina 310 staat beschreven hoe je een box kan maken. Maar nu voor elk van de vier menu-items een box en maak een kleur die lijkt op de kleur in de opdracht.

Elk menu-item krijgt zijn eigen box, voor elk box is dan ook een aparte `class` gemaakt.

[image.1668538831859.png](#)

In dit voorbeeld zijn alle boxes 5px dik.

Stap 3

Op pagina 310 staat ook het verschil uitgelegd tussen solid, dotted, dashed en double boxes. Lees dat door en pas de boxes aan zodat ze nog wat meer gaan lijken op de boxes in de opdracht.

[image.1668538860055.png](#)

Het lijkt al wat beter zo toch?

Laten we nu de boxes wat hoger maken met de width, bijv 100px. En zet de padding op 20px. Op pagina 313 wordt uitgelegd wat padding is.

[image.1668539385071.png](#)

Stap 4

Op pagina 322 staat beschreven hoe we ronde hoeken maken. De ronde hoeken moeten we telkens op drie manieren coderen. Dit heeft te maken met oude browsers die niet allemaal 100% dezelfde 'CSS-taal' spreken.

Van de linker box passen we alleen de bottom-right aan:

[image.1668539450174.png](#)

Van de tweede box passen we alle hoeken aan.

Top left en bottom right zijn hetzelfde en top left en bottom right zijn hetzelfde.

[image.1668539464039.png](#)

De derde box heeft allemaal dezelfde hoeken en de laatste box hoeft je iets aan te wijzigen.

[image.1668539529713.png](#)

Stap 5

We moeten van de vierde box de onderkant nog aanpassen. De

In plaats van

```
border: 20px solid #000000
```

kunnen we ook alleen de onderkant van de border bepalen met

```
border-bottom: 20px solid #000000
```

De toevoeging- *bottom* zorgt er dus voor dat het nu alleen over de onderkant (=bottom) gaat.

[image.1668539628281.png](#)

De blokjes bovenaan de pagina zijn nu goed.

We gaan nu kijken naar de tekst blokken.

Stap 6

Op pagina 317 staat hoe je een div kan 'verstoppen'. Pas dit toe op de class die bij het laatste vierkantje blokje hoort zodat deze niet meer wordt getoond.

Tip: het gaat om het blokje met de `class="hidden"`.

Stap 7

Allereerst gaan we weer borders maken.

[image 1668544319487.png](#)

Stap 8

Nu gaan we nog iets doen met witruimte. Lees pagina 307.

[image 1668544432423.png](#)

Stap 9

De witruimte hebben we als het goed is met padding gedaan. De laatste twee blokken hebben een top en bottom padding van 0px en een left en right padding van (ong.) 20px.

Je kunt achter padding dus net als bij border -left, -right, -top of -bottom zetten.

[image 1668544639437.png](#)

Stap 10

Met margin (zie pag 307) bepalen we de witruimte buiten de <p> zet die voor de laatste twee <p> op 0px.

Stap 11

Pas gelijk de breedte even aan. Weet je niet meer hoe dat moet? Zoek dan even op

[image 1668545067575.png](#)

Stap 12

Nu de kleur van de <p> aanpassen.

[image 1668545240365.png](#)

Stap 13

En het font. We gebruiken weer een Google font zoals we dat eerder hebben gedaan.

Pas de CSS aan.

[image 1668545365564.png](#)

Tip het gebruikte font is Pacifico

Stap14

Als laatste gaan we de regelafstand aanpassen.

Hoe? Zie: https://www.w3schools.com/cssref/pr_dim_line-height.php

[image-1668545554130.png](#)

--

DOM Manipulation

Inleiding

Stel je voor dat een webpagina een grote tekening is met allerlei onderdelen, zoals koppen, tekst en plaatjes. De **DOM** (Document Object Model) is als een soort "instructieboek" dat uitlegt hoe die tekening in stukjes is verdeeld, zodat we precies weten waar elk onderdeel zit.

Waarom is dit handig met JavaScript?

- **Veranderen van de tekening:** JavaScript is als een magische potlood dat de tekening kan aanpassen. Met de DOM kan JavaScript precies aanwijzen welk onderdeel van de tekening hij wil veranderen, toevoegen of verplaatsen, zonder dat je de hele tekening opnieuw moet maken.
- **Reageren op wat jij doet:** Stel je voor dat je op een knop klikt. JavaScript kan dat opvangen via de DOM en dan iets leuks doen, zoals een verborgen bericht laten zien of een plaatje veranderen.

Kortom, de DOM zorgt ervoor dat JavaScript precies weet hoe de pagina in elkaar zit, zodat je webpagina interactief en dynamisch kan worden.

Theorie

(English below)

DOM is het Document Object Model, dat is zeg maar de hele HTML pagina. Je kunt met JavaScript de DOM aanpassen. Je kunt bijvoorbeeld een kleur van een element aanpassen, of de inhoud van een element aanpassen.

Dit kan natuurlijk ook met PHP, maar daarvoor moet je (terug) naar de server. Met JS kan je lokaal (front End) dingen aanpassen. Dat is vaak sneller en ziet er vaak beter uit.

Om de DOM aan te passen, heb je een paar specifieke commando's. De belangrijkste worden hier besproken.

1. Toegang krijgen tot Elementen

Om de inhoud te manipuleren, moet je eerst toegang krijgen tot de elementen. Veelgebruikte methoden om toegang te krijgen tot DOM-elementen zijn:

- `document.getElementById(id)`: Selecteert een enkel element op basis van zijn ID.
- `document.getElementsByTagName(name)`: Geeft een verzameling van elementen terug met de gegeven tag-naam.
- `document.getElementsByClassName(name)`: Geeft een verzameling van elementen terug die de gegeven klassenaam hebben.
- `document.querySelector(selector)`: Geeft het eerste element terug dat overeenkomt met een gespecificeerde CSS-selector.
- `document.querySelectorAll(selector)`: Geeft een lijst terug van alle elementen die overeenkomen met een gespecificeerde CSS-selector.

2. Wijzigen van Elementeigenschappen

Zodra je een referentie naar een element hebt, kun je de eigenschappen ervan wijzigen. Dit omvat het wijzigen van de innerlijke HTML, tekstinhoud, attributen en stijl van het element. Bijvoorbeeld:

- `element.innerHTML`: Haalt op of stelt de HTML- of XML-markering in die binnen het element is opgenomen.
- `element.textContent`: Stelt de tekstuele inhoud van een element en zijn nakomelingen in of retourneert deze.
- `element.setAttribute(name, value)`: Voegt een gespecificeerd attribuut toe aan een element, of werkt de waarde ervan bij als het al bestaat.
- `element.style.property`: Wijzigt de stijl van een element, waarbij property een CSS-eigenschap is (in camelCase, bijvoorbeeld `backgroundColor`).

3. Creëren en Verwijderen van Elementen

JavaScript stelt je in staat om nieuwe elementen te creëren en deze in het DOM in te voegen of bestaande elementen te verwijderen:

- `document.createElement(tagName)`: Creëert een nieuw element met de gegeven tag-naam.

- `parentNode.appendChild(childNode)`: Voegt een knooppunt toe als het laatste kind van een ouder knooppunt.
- `parentNode.insertBefore(newNode, referenceNode)`: Voegt een nieuw knooppunt in vóór het referentieknooppunt als een kind van een gespecificeerd ouderknooppunt.
- `parentNode.removeChild(child)`: Verwijdert een kindknooppunt uit het DOM.
- `element.remove()`: Verwijdert het element uit het DOM.

4. Event Handling

Het toevoegen van event listeners aan elementen stelt je in staat om code uit te voeren als reactie op gebruikersacties, zoals klikken, toetsenbordinput of muisbewegingen:

- `element.addEventListener(event, function, useCapture)`: Voegt een event handler toe aan een element. Veelvoorkomende events zijn `click`, `mouseover`, `mouseout`, `keydown`, `keyup`, enz.

5. Doorlopen en Wijzigen

Het DOM biedt methoden en eigenschappen om knooppunten te doorlopen en te wijzigen:

- `element.childNodes`: Geeft een live NodeList van kindknooppunten van een element terug.
- `element.parentElement`: Geeft het ouder-element van het gespecificeerde element terug.
- `element.nextSibling`: Geeft de volgende sibling in de boom terug.
- `element.previousSibling`: Geeft de vorige sibling in de boom terug.

Theory in English

(Dutch above)

The DOM is the Document Object Model, which is essentially the entire HTML page. You can modify the DOM using JavaScript. For example, you can change the color of an element, or modify the content of an element.

This can also be done with PHP, but that requires a (return) trip to the server. With JS, you can make local (front-end) adjustments. This is often faster and usually looks better.

To modify the DOM, you need a few specific commands. The most important ones are discussed here.

1. Accessing Elements

To manipulate the content, you first need to access the elements. Common methods to access DOM elements include:

- `document.getElementById(id)`: Selects a single element by its ID.
- `document.getElementsByTagName(name)`: Returns a collection of elements with the given tag name.
- `document.getElementsByClassName(name)`: Returns a collection of elements that have the given class name.
- `document.querySelector(selector)`: Returns the first element that matches a specified CSS selector.
- `document.querySelectorAll(selector)`: Returns a list of all elements that match a specified CSS selector.

2. Changing Element Properties

Once you have a reference to an element, you can modify its properties. This includes changing the element's inner HTML, text content, attributes, and style. For example:

- `element.innerHTML`: Gets or sets the HTML or XML markup contained within the element.
- `element.textContent`: Sets or returns the textual content of an element and its descendants.
- `element.setAttribute(name, value)`: Adds a specified attribute to an element, or updates its value if it already exists.
- `element.style.property`: Changes the style of an element, where `property` is a CSS property (in camelCase, e.g., `backgroundColor`).

3. Creating and Removing Elements

JavaScript allows you to create new elements and insert them into the DOM or remove existing elements:

- `document.createElement(tagName)`: Creates a new element with the given tag name.

- `parentNode.appendChild(childNode)`: Appends a node as the last child of a parent node.
- `parentNode.insertBefore(newNode, referenceNode)`: Inserts a new node before the reference node as a child of a specified parent node.
- `parentNode.removeChild(child)`: Removes a child node from the DOM.
- `element.remove()`: Removes the element from the DOM.

4. Event Handling

Adding event listeners to elements enables you to execute code in response to user actions, such as clicks, keyboard input, or mouse movements:

- `element.addEventListener(event, function, useCapture)`: Attaches an event handler to an element. Common events include `click`, `mouseover`, `mouseout`, `keydown`, `keyup`, etc.

5. Traversal and Modification

The DOM provides methods and properties to traverse and modify nodes:

- `element.childNodes`: Returns a live `NodeList` of child nodes of an element.
- `element.parentElement`: Returns the parent element of the specified element.
- `element.nextSibling`: Returns the next sibling in the tree.
- `element.previousSibling`: Returns the previous sibling in the tree.

DOM manipulation opdrachten

Inleiding

Neem eerste de theorie door op de [vorige](#) pagina!

De opdrachten van dit blok sluiten op elkaar aan. Je moet ze dus in de juiste volgorde maken.

Waarschuwing: als je de opdrachten met ChatGPT maakt en je snapt niet wat je doet dan loop je met de laatste opdrachten vast. Deze opdrachten zijn getest en kunnen niet met behulp van ChatGPT 4.0 integraal worden opgelost.

Kom je ergens niet verder, ga dan **niet** verder met de volgende opdracht maar vraag hulp!

Opdracht 1, kleur aanpassen (via id)

Je hebt de volgende HTML/JS-code.

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Tekstkleur Veranderen</title>
</head>
<body>
  <p id="tekst">Dit is een voorbeeldtekst.</p>
  <button onclick="veranderKleur()">Verander Kleur</button>

  <script>
    function veranderKleur() {
      // maak de code af en plaats hieronder de code die de kleur van de tekst verandert
```



```
...
}
</script>
</body>
</html>
```

De JavaScript-functie wordt aangeroepen vanuit de `onclick` attribuut van de HTML-knop.

Wanneer de knop wordt ingedrukt, moet de kleur van de tekst in het paragraafelement veranderen. Maak hiervoor het script af.

Zorg dat je in het script zoekt naar het element met het id "tekst". Als je het element hebt gevonden dan kan je de daarna de kleur aanpassen.

Inleveren

Zet jouw naam als commentaar in de JS-code en lever het geteste en werkende HTML-pagina in.

Opdracht 2, kleur aanpassen (via class)

Je hebt de volgende HTML/JS-code:

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Achtergrondkleur Veranderen</title>
</head>
<body>
  <p class="kleurVerander">Dit is een voorbeeldtekst 1.</p>
  <p class="kleurVerander">Dit is een voorbeeldtekst 2.</p>
  <p class="kleurVeranderNiet">Dit is een voorbeeldtekst 3.</p>
  <p class="kleurVerander">Dit is een voorbeeldtekst 4.</p>

  <button onclick="veranderAchtergrondKleur()">Verander Achtergrondkleur</button>

  <script>
```

```
function veranderAchtergrondKleur() {  
    // maak de code af en plaats hieronder de code die de kleur van de tekst met de class kleurVerander  
    verandert  
    ...  
}  
</script>  
</body>  
</html>
```

De JavaScript-functie wordt aangeroepen vanuit de `onclick` attribuut van de HTML-knop.

Wanneer de knop wordt ingedrukt, moet de kleur van de tekst in alle paragraafelement met de class *kleurVerander* veranderen. Maak hiervoor het script af.

Maak gebruik van `document.getElementsByClassName` en gebruik een for-loop.

Denk erom dat `document.getElementsByClassName` een array met elementen terug geeft en dat je met de loop door het array moet heenlopen.

Bedek zelf een mooie kleur.

Inleveren

Zet jouw naam als commentaar in de JS-code en lever het geteste en werkende HTML-pagina in.

Opdracht 3, stijl toggle

Je hebt de volgende HTML/JS-code:

```
<!DOCTYPE html>  
<html lang="nl">  
<head>  
    <meta charset="UTF-8">  
    <title>Tekststijl Veranderen</title>  
</head>  
<body>  
    <h1>Dit is een koptekst</h1>
```

```

<p>Dit is een gewone paragraaf.</p>
<p class="speciaal">Dit is een speciale paragraaf.</p>
<div>Dit is een div element.</div>
<p class="speciaal">Nog een speciale paragraaf.</p>
<p id="speciaal">Nog een speciale paragraaf.</p>

<button onclick="veranderTekstStijl()">Verander Tekststijl</button>

<script>
  function veranderTekstStijl() {
    var elementen = document.querySelectorAll(.....);
    for (.....) {
      // Maak de elementen green, bold, and italic (schuingedrukt)
    }
  }
</script>
</body>
</html>

```

Vul de code aan op de puntje en zorg ervoor dat zodra op de knop wordt gedrukt:

- alle elementen met de class *speciaal* **groen, vet en schuingedrukt** worden.

Als er weer op de knop wordt gedrukt dan worden:

- alle elementen met de class *speciaal* worden weer zwarte, niet vet en niet schuingedrukt.

De knop wisselt dus telkens de stijl alle tekst met de class *speciaal*.

Inleveren

Zet jouw naam als commentaar in de JS-code en lever het geteste en werkende script in.

Zet jouw naam als commentaar in de JS-code en lever het geteste en werkende HTML-pagina in.

Opdracht 4, counter

Bekijk het volgende voorbeeld.

```
<div id="nummerDiv">123</div>
<button onclick="toonGetal()">Toon Getal</button>

<script>
  function toonGetal() {
    var divElement = document.getElementById('nummerDiv');
    var getal = divElement.innerText; // Lees de tekst (in dit geval een getal) uit de div
    alert("Het getal in de div is: " + getal);
  }
</script>
```

Probeer te begrijpen wat er gebeurt en test de code om te kijken of je snapt wat er gebeurt!

Opdracht

1. maak een extra button
2. als je op de button druk dan roep je een javascript funcite aan (bedenk zelf een naam voor de functie).
3. met een console.log druk je de waarde af die in de div staat; in dit voorbeeld dus 123.
4. tel 1 op bij het getal en druk dit getal ook af met een console.log. In dit voorbeeld is dat dus 124.

Als je in JS een getal bij een getal optelt met + dan 'denkt' JS standaard dat je een string hebt. Dan wordt 1+1 dus 11 en niet 2! Om dit te voorkomen moet je aangeven dat je geen string maar een integer hebt.

Dit doe je met parseInt(string).

Dus parseInt(getal) + 1 wordt dan 2.

Dus, stel getal = 1

getal + 1 wordt 11 en
parseInt(getal) + 1 wordt 2

5. dan zet je het getal (in dit voorbeeld dus 124) terug in de div.

Als je goed is, heb je nu een div met een getal en telkens als je op de knop druk wordt het getal opgehoogd.

Inleveren

Zet jouw naam als commentaar in de JS-code en lever het geteste en werkende HTML-pagina in.

Opdracht 5, Ronde teller

Maak een interactieve webpagina met een tabel die schaatsters en hun rondeteller weergeeft.

image 1712404637053.png

De gebruiker kan het aantal ronden voor elke schaatser verhogen door op de teller in de tabel te klikken.

Begin met deze HTML/CSS-code:

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Schaatser Ronde Teller</title>
  <style>
    table {
      width: 50%;
      border-collapse: collapse;
      margin: 20px auto;
    }
    th, td {
      border: 1px solid #dddddd;
      text-align: left;
      padding: 8px;
    }
    th {
      background-color: #f2f2f2;
    }
    td.rondes {
      cursor: pointer;
      background-color: #e6ffe6;
```

```
}
td.rondes:hover {
    background-color: #ccffcc;
}
</style>
</head>
<body>

<table>
  <tr>
    <th>Schaatser</th>
    <th>Aantal Rondes</th>
  </tr>
  <tr>
    <td>Mo Mot</td>
    <td class="rondes" onclick="verhoogRondes(this)">0</td>
  </tr>
  <tr>
    <td>Ayoub Allemaal</td>
    <td class="rondes" onclick="verhoogRondes(this)">0</td>
  </tr>
  <tr>
    <td>Klaas Kaas</td>
    <td class="rondes" onclick="verhoogRondes(this)">0</td>
  </tr>
  <tr>
    <td>Goedele Goedgeluk</td>
    <td class="rondes" onclick="verhoogRondes(this)">0</td>
  </tr>
  <tr>
    <td>Roos Rommelhuis</td>
    <td class="rondes" onclick="verhoogRondes(this)">0</td>
  </tr>
  <tr>
    <td>Lars Leeghoofd</td>
    <td class="rondes" onclick="verhoogRondes(this)">0</td>
  </tr>
</table>

</body>
```

</html>

Maak de code af en maak een script waarbij als je op het ronde nummer drukt, de ronde met een wordt verhoogd. Denk aan het gebruik van `parseInt` (vorige opdracht).

Inleveren

Zet jouw naam als commentaar in de JS-code en lever het geteste en werkende HTML-pagina in.

Opdracht 6, eventListener

In plaats van een functie aan heel veel rijen toe te voegen, kan je ook een zogenaamde **eventListener** maken.

Een event is een gebeurtenis en een **eventListener** luistert dus of er een bepaalde gebeurtenis plaatsvindt.

We gaan onze code uit de vorige opdracht (5) aanpassen.

1. haal alle onclick attributen uit de `<td>`'s weg.
2. geef de tabel een id `'rondeTellerTabel'`

Bekijk de volgende code:

```
document.addEventListener('DOMContentLoaded', function() {  
    var rondesCellen = document.querySelectorAll('#rondetellerTabel .rondes');  
    rondesCellen.forEach(function(cel) {  
        cel.addEventListener('click', verhoogRondes);  
    });  
});
```

Uitleg

Regel	Uitleg
1	Zodra de DOM (HTML-document) is geladen voer de volgende code uit.
2	Laad alle cellen uit de tabel met het id <code>rondeTellerTabel</code> en de class <code>rondes</code> .

3	Itereer door alle gevonden cellen (<i>iteren</i> betekent 'doorlopen').
4	Laat alle gevonden cellen luisteren naar een click en als er een click plaatsvindt, roep dan de functie <i>verhoogRondes</i> aan.

De functie *verhoogRondes* moeten we iets aanpassen.

```
function verhoogRondes() {
    var huidigeRondes = parseInt(this.innerText);
    this.innerText = huidigeRondes + 1;
}
```

Hier gebruiken we *this*, *this* is het huidige element waarop is gelicked.

Zorg ervoor dat de ronde teller weer werkt maar nu met de eventListener zoals hierboven uitgelegd.

Lees goed en aandachtig en als het moet lees het nog eens stap voor stap. Als je alle stappen volgt en begrijpt dan zul je de opdracht kunnen maken.

Inleveren

De aangepaste code waarin je ronde kan tellen van de vorige opdracht, maar nu maak je gebruik van de eventlistener zoals uitgelegd.

Opdracht 7, shift key

We nemen de code van onze vorige opdracht en gaan functionaliteit uitbreiden.

Soms wordt er namelijk per ongeluk op de verkeerde kolom geclicked. Je wilt dan het rondenummer verlagen.

Als de shif-key wordt ingedrukt terwijl we clicken moet de het rondenummer worden verlaagd.

Je kun in JS testen of de shift-key wordt ingedrukt:

```
if (event.shiftKey) {
```

Gebruik dit en bereid de JS code uit zodat:

- als je clicked, wordt het rondenummer één hoger (dat hadden we al).
- als je shift-clicked, wordt het rondenummer één lager (nieuw).

We hebben nog één extra eis:

- het rondenummer mag mag niet negatief worden.

Inleveren

De aangepaste code (volledig werkende HTML/CSS/JS) van de vorige opdracht aangevuld met de functionaliteit zoals hierboven beschreven.

Plaats commentaar in de code met uitleg én plaats commentaar met jouw naam.

Opdracht 8, shift key en kleur

We maken nog een laatste aanpassing aan de ronde teller. Zorg dat je de vorige opdracht helemaal af hebt.

Als je op de rondeteller klikt wordt de rondeteller verhoogd, doe je dat met shift dan wordt de ronde teller verlaagd.

De achtergrondkleur van de ronde teller is groen. Om aan te geven dat de rondeteller is verlaagd wordt de achtergrondkleur rood. Als daarna de teller weer wordt verhoogd, wordt de achtergrond kleur weer groen.

image1712404975032.png

De eisen:

- standaard staan alle ronde tellers op 0 en zijn ze groen.
- als je op een ronde teller clicked dan, wordt die verhoogd en is/wordt de achtergrondkleur groen.
- als je op een rondeteller shift-clicked, dan wordt die verlaagd en/is wordt de achtergrondkleur rood.
- de achtergrondkleur mag neit negatief worden (lager dan 0), dus als je op een 0 shift-clicked dan gebeurt er niets; de achtergrondkleur blijft hetzelfde en de rondeteller wordt

niet verlaagd.

Inleveren

De aangepaste code (volledig werkende HTML/CSS/JS) van de vorige opdracht aangevuld met de functionaliteit zoals hierboven beschreven.

Plaats commentaar in de code met uitleg én plaats commentaar met jouw naam.

Opdracht 9, uitklappen

Bekijk en test de volgende code.

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Elektronica Producten</title>
  <style>
    .detailRow {
      display: none;
      background-color: #f9f9f9;
    }
    .clickableRow:hover {
      background-color: #f1f1f1;
      cursor: pointer;
    }
    table {
      width: 100%;
      border-collapse: collapse;
    }
    th, td {
      border: 1px solid #dddddd;
      text-align: left;
      padding: 8px;
    }
    th {
      background-color: #f2f2f2;
```

```
    }
</style>
</head>
<body>

<table id="infoTabel">
  <tr>
    <th>Product</th>
    <th>Categorie</th>
  </tr>
  <tr class="clickableRow">
    <td>iPhone 13</td>
    <td>Smartphone</td>
  </tr>
  <tr class="detailRow">
    <td colspan="2">Merk: Apple | Opslag: 128GB | Kleur: Zwart</td>
  </tr>
  <tr class="clickableRow">
    <td>Samsung Galaxy S21</td>
    <td>Smartphone</td>
  </tr>
  <tr class="detailRow">
    <td colspan="2">Merk: Samsung | Opslag: 256GB | Kleur: Phantom Gray</td>
  </tr>
  <tr class="clickableRow">
    <td>Dell XPS 13</td>
    <td>Laptop</td>
  </tr>
  <tr class="detailRow">
    <td colspan="2">Merk: Dell | Scherm: 13.3 inch | CPU: Intel i7</td>
  </tr>
  <tr class="clickableRow">
    <td>MacBook Air</td>
    <td>Laptop</td>
  </tr>
  <tr class="detailRow">
    <td colspan="2">Merk: Apple | Scherm: 13.3 inch | CPU: M1</td>
  </tr>
  <tr class="clickableRow">
    <td>Sony WH-1000XM4</td>
```

```

        <td>Hoofdtelefoon</td>
    </tr>
    <tr class="detailRow">
        <td colspan="2">Merk: Sony | Type: Over-ear | Noise Cancelling: Ja</td>
    </tr>
    <tr class="clickableRow">
        <td>Google Nest Hub</td>
        <td>Smart Home</td>
    </tr>
    <tr class="detailRow">
        <td colspan="2">Merk: Google | Scherm: 7 inch | Assistent: Google Assistent</td>
    </tr>
</table>

<script>
    document.querySelectorAll('.clickableRow').forEach(row => {
        row.addEventListener('click', function() {
            var detailRow = this.nextElementSibling;
            detailRow.style.display = 'table-row';
        });
    });
</script>

</body>

```

Deze code laat een tabel zien en als je op de regel clicked dan worden details getoond.

clickableRow

De hoofdregels waar je op kan clicken hebben de class *clickableRow*.

detailRow

De detail regels die zichtbaar/onzichtbaar worden door op de hoofdregels te clicken hebben de class *detailRow*.

Opdracht

Stap 1

Pas de code aan zodat als de detailregel wordt getoond en er weer op de *clickableRow* wordt gedrukt de detailregel weer wordt 'weggehaald' (`style.display = 'none'`).

Het klikken op de *clickableRow* wordt dus een toggle: het laat de details zien en als er weer wordt geclicked worden de details weer weggehaald.

Stap 2a

We willen een andere vormgeving; de detail-regels moeten inspringen. Jou lead developer gevraagd om de code op te bouwen en om geen table te gebruiken maar div. Je krijgt dus een div met een class *clickableDiv* en daaronder een div met een class *detailRow*

Zorge ervoor dat de twee kolommen in de *clickableDiv* apart in de div staan. Je hebt telkens de naam van het product (bijvoorbeeld iPhone 13) en de categorie (bijvoorbeeld Smartphone), Maak hier twee divs van die naast elkaar staan (tip: gebruik flexbox). Je hebt dus twee divs die in een div staan:

```
<div class="clickableRow">
  <div class="product">iPhone 13</td>
  <div class="category">Smartphone</td>
</div>
<div class="detailRow">
  .....
```

Stap 2b

Als de table is omgebouwd in divs, zorg er dan voor dat de *clickableRows* en *detailRow* beter te onderscheiden zijn:

- De grootte van het lettertype van de *clickableRows* is meer dan die van de *detailsRows*.
- De *detailsRows* krijgen een margin waardoor het lijkt dat deze inspringen (iets naar rechts gaan)

Stap 3

Pas de kleuren van de tabel aan zodat de tabel mooi is en goed leesbaar is

Stap 4

Plaats commentaar in de code waarin je uitlegt hoe de code werkt. Zet ook je naam in ergens in het commentaar.

Voorbeeld

Als je klaar bent dan zou het er zo kunnen uitzien (kleuren zelf kiezen).

image-1712409645862.png

Inleveren

De aangepaste code (volledig werkende HTML/CSS/JS) van de vorige opdracht aangevuld met de functionaliteit zoals hierboven beschreven.

Opdracht 10, Winkelmandje

Je hebt nu een soort productpagina gemaakt bij opdracht 9.

We gaan een begin maken met een winkelmandje. Hiervoor gebruiken we de code van de vorige opdracht en breiden deze uit.

1. Op elke detailRow komt een knopje *Bestellen* - maak deze knop en geef deze knop een mooi CSS stijl.

Voorbeeld:

image-1712409789855.png

2. Bovenin de pagina maak je een nieuwe div, dit is het winkelmandje.
3. Als de knop *Bestel* wordt ingedrukt dan wordt het product toegevoegd als een regel in het winkelmandje. Als je bijvoorbeeld op iPhone 13 clicked dan verschijnt er een regel in het winkelmandje met iPhone 13.
4. Als je dezelfde knop weer indrukt dan komt er een tweede regel iPhone13 in het winkelmandje.
5. Maak vervolgens een knop in de het winkelmandje waarmee je het winkelmandje leeg kunt maken.

Voorbeeld

image-1712418445270.png

Inleveren

De aangepaste code (volledig werkende HTML/CSS/JS) van de vorige opdracht aangevuld met de functionaliteit zoals hierboven beschreven.

Optie, Teller in winkelmandje

Zet in het winkelmandje achter elk product een teller waarmee je klan aangeven hoeveel je wilt bestellen. Gebruik een (deel van de) code van de rondeteller.

- Zorg ervoor dat het aantal niet lager dan 0 en niet hoger dan 10 kan worden.
- (optioneel) zorg ervoor dat er geen dubbele regels in het winkelmandje kunnen ontstaan.

Inleveren

De aangepaste code (volledig werkende HTML/CSS/JS) van de vorige opdracht aangevuld met de functionaliteit zoals hierboven beschreven.