



INTERACTIEVE WEBSITES

JavaScript LES 6

Classes

Graduaat Informatica Programmeren

OPDRACHTEN

Opdracht: CoronaTest

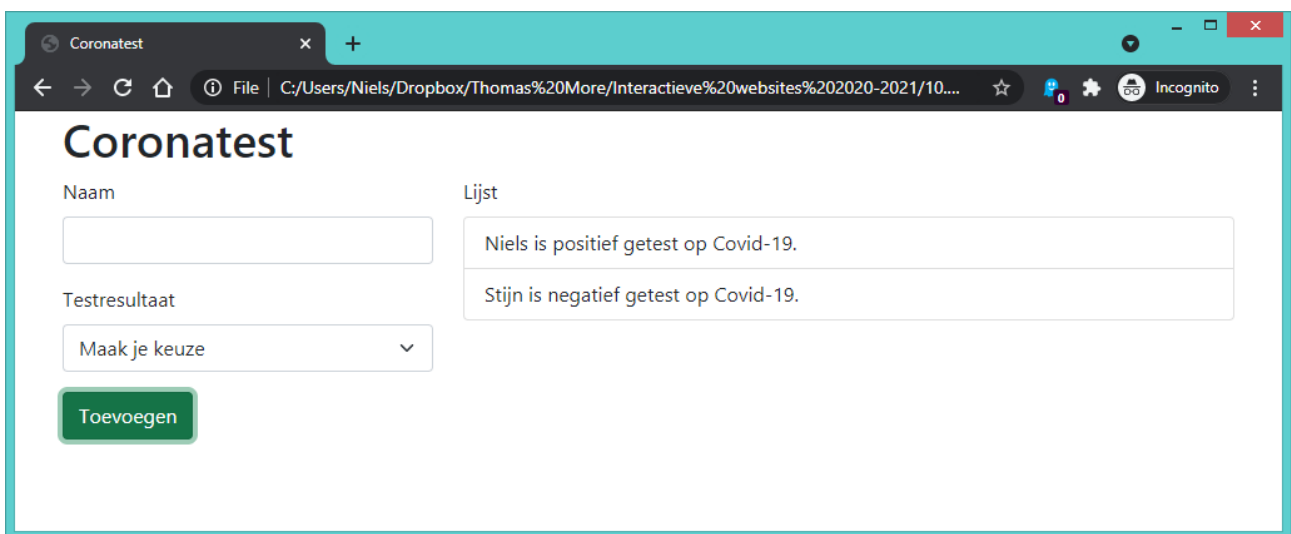
Maak een klasse **CoronaTest** met twee properties **name** en **result**. Wanneer de gebruiker een naam en een resultaat invult en op de knop **Toevoegen** klikt zal het **resultaat** automatisch te zien zijn in de lijst.

De klasse heeft nog een **methode** genaamd **resultaat** die onderstaande **tekst samenstelt** afhankelijk van het testresultaat.

Het resultaat bij de keuze positief is: \$name is positief getest op Covid-19.

Het resultaat bij de keuze negatief is: \$name is negatief getest op Covid-19.

Je kan geen resultaat toevoegen wanneer de **naam leeg** is **en/of** wanneer er **geen keuze is gemaakt** bij **Testresultaat**.



The screenshot shows a web browser window with the title 'Coronatest'. The address bar shows the file path: 'C:/Users/Niels/Dropbox/Thomas%20More/Interactieve%20websites%202020-2021/10...'. The page has a teal header. Below the header, there is a form with two columns. The left column has a 'Naam' label above an empty text input field, a 'Testresultaat' label above a dropdown menu with the text 'Maak je keuze' and a downward arrow, and a green 'Toevoegen' button below. The right column has a 'Lijst' label above a list container. The list contains two items: 'Niels is positief getest op Covid-19.' and 'Stijn is negatief getest op Covid-19.'.

Na het toevoegen van een testresultaat aan de lijst, maak je het inputveld naam **terug leeg** en zet je de dropdown van testresultaat terug op '**Maak een keuze**'.

Gebruik in deze oefening twee functies namelijk:

- **addCoronaTest**
 - Hierin controleer je de input naam en de dropdown
 - Maak je een instantie aan van de klasse CoronaTest en voegt deze toe aan een array
 - Maak het veld leeg en zet de dropdown goed
- **addToList** => ontvangt als argument de methode resultaat van de klasse
 - Creëer een nieuw list-item element
 - Vergeet niet de Bootstrap class "list-group-item" mee te geven!
 - Voeg dit toe aan de bestaande lijst

Bewaar deze oefening als **coronatest.html**

Opdracht: Politiekers

Maak een klasse **Politieker** met drie eigenschappen **naam**, **partij** en **loon**.

Maak ook nog een subklasse van **Politieker** namelijk **BlauwePolitieker** met vier eigenschappen **naam**, **partij** en **loon** en **inEigenZak**.

Beide klassen hebben een **methode** namelijk **overzicht** die volgende **tekst samenstelt** afhankelijk van de input:

- **Politieker**
 - *\$naam* is van de partij *\$partij* en heeft als loon € *\$loon*.
- **BlauwePolitieker**
 - *\$naam* is van de partij *\$partij* en heeft als loon € *\$loon*.
Hiervan steken ze € \$inEigenZak in hun eigen zak.

Het eerste deel "*\$naam ... \$loon*" laat je bij **BlauwePolitieker** retourneren uit de superklasse **Politieker**!

Gebruik 3 functies in deze opdracht:

- **selectPartij**
 - Enkel wanneer de gebruiker "blauw" selecteerd in de dropdown, verschijnt de div #pocketmoney.
- **politiekerToevoegen**
 - Enkel een controle of er een partij geselecteerd is, is hier voldoende.
 - Op basis van welke partij er gekozen is maak je instanties aan.
 - Spreek hier de functie **addToList** en geeft als **argument** het resultaat van de **methode overzicht** mee.
- **addToList**
 - Creëer een nieuw list-item element voor de UL #list.
 - Vergeet niet de Bootstrap class "list-group-item" mee te geven!
 - Voeg dit toe aan de bestaande lijst

Wanneer de gebruiker op de knop **Toevoegen** klikt starten we bij de functie **politiekerToevoegen**.

Politiekers

Maak een lijst van al je favoriete politiekers.

Naam

Partij

Purper

Loon

Toevoegen

Lijst

test rood is van de partij rood en heeft als loon €2500.

test blauw is van de partij blauw en heeft als loon €5500.
Hiervan steken ze €2000 in hun eigen zak.

test geel is van de partij geel en heeft als loon €3500.

test purper is van de partij purper en heeft als loon €3800.

Bewaar deze oefening als **politiekers.html**

Opdracht: Pokémon

Versie 1

Maak een webapplicatie waarmee je twee Pokémons tegen elkaar kan laten vechten. Elke Pokémon heeft een **naam**, **type**, **attack (min)**, **attack (max)**, **image**, **currentHitpoints** en **hitpoints**.

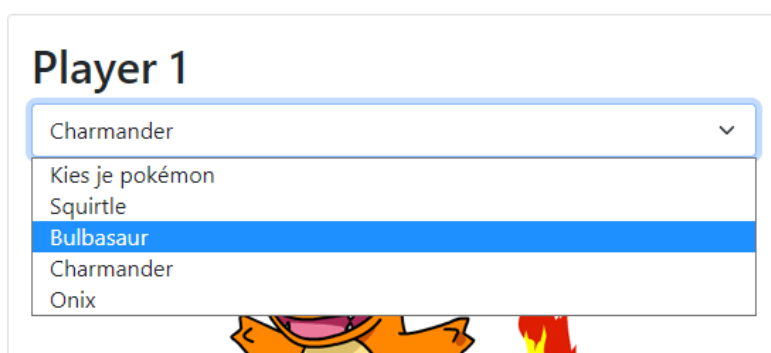
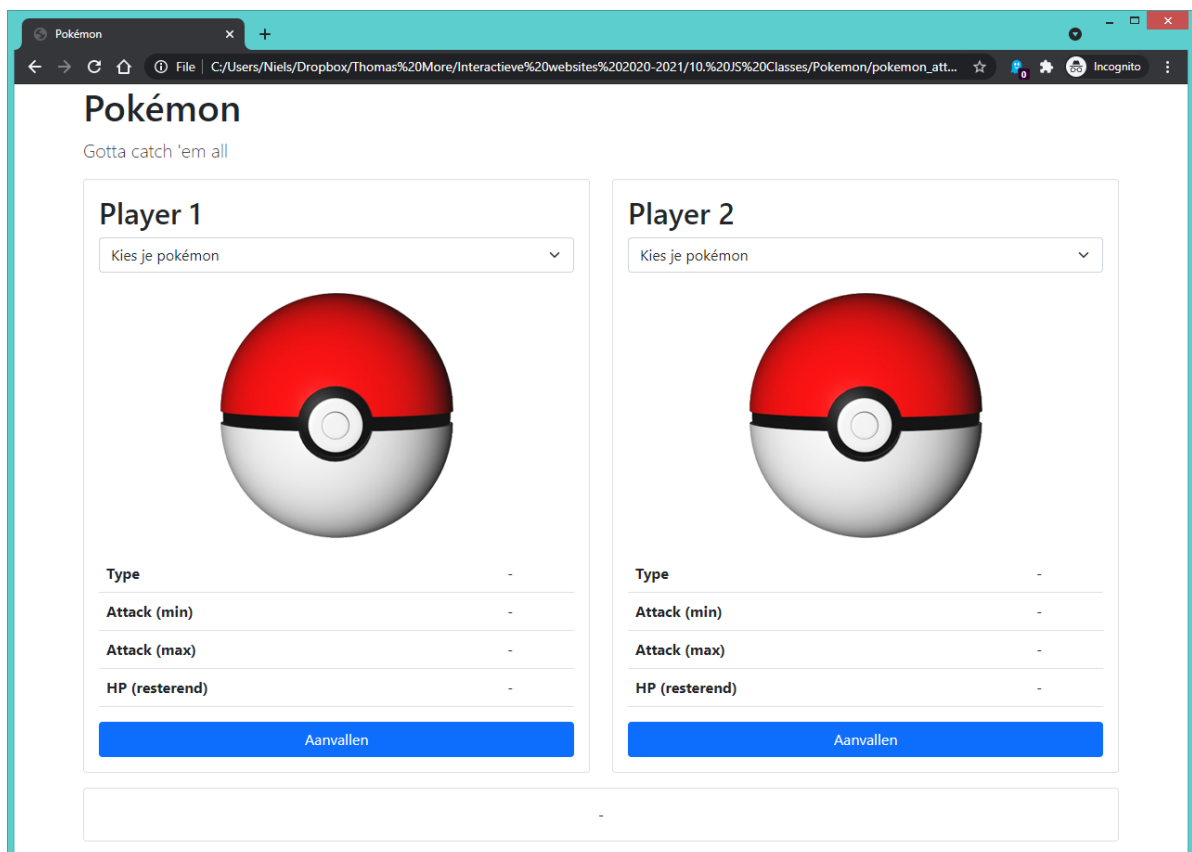
Wanneer een Pokémon een tegenstander aanvalt zal er een random nummer gegenereerd worden tussen de **attack (min)** en **attack (max)**.

Daarna controleer je of de attack **hit** of **missed**. Er is één kans op vier dat een attack missed, hou hier dus rekening mee. Wanneer een attack **hit** zullen de **hitpoints** van de Pokémon die verdedigt verminderen.

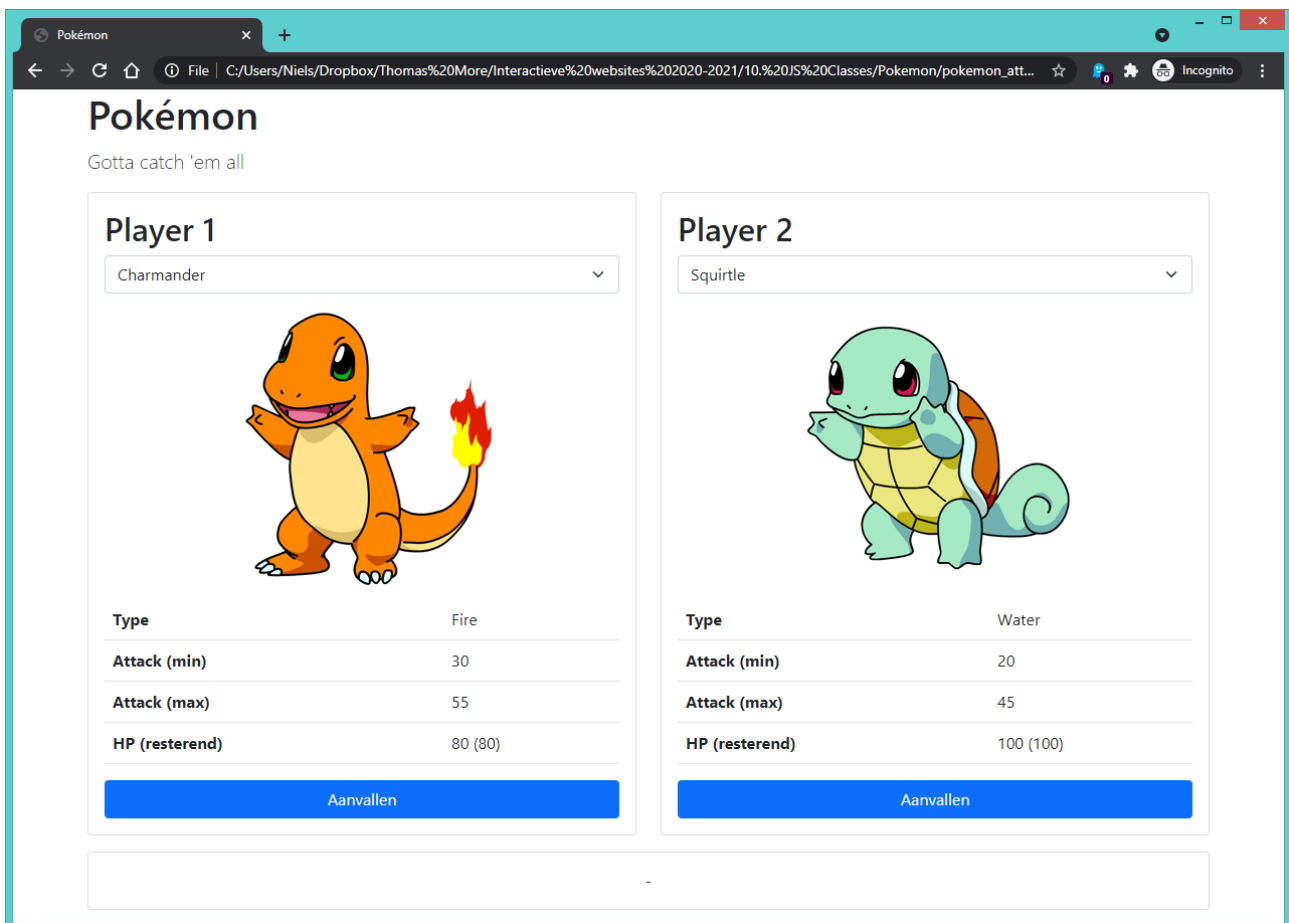
Tips:

- De afbeeldingen voor de Pokémons kan je vinden in de startbestanden.
- Het is niet nodig om zelf CSS te schrijven, in het voorbeeld zijn alleen Bootstrap classes gebruikt.

Begin van het spel:



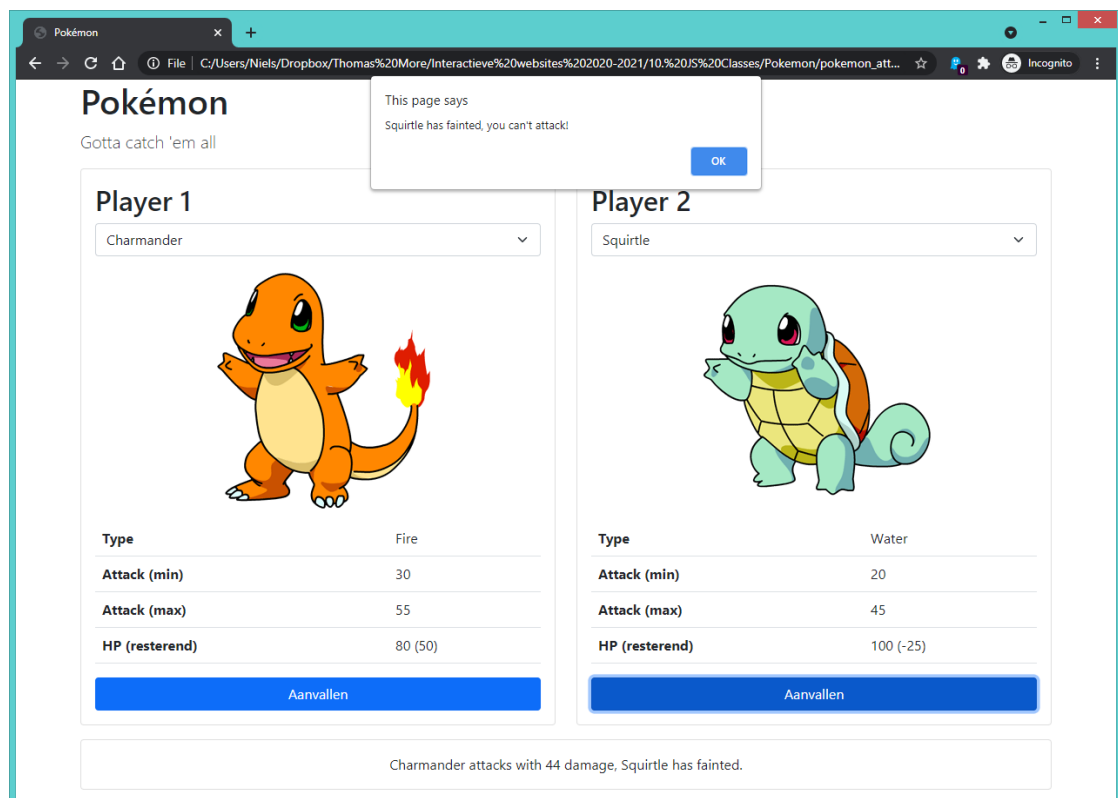
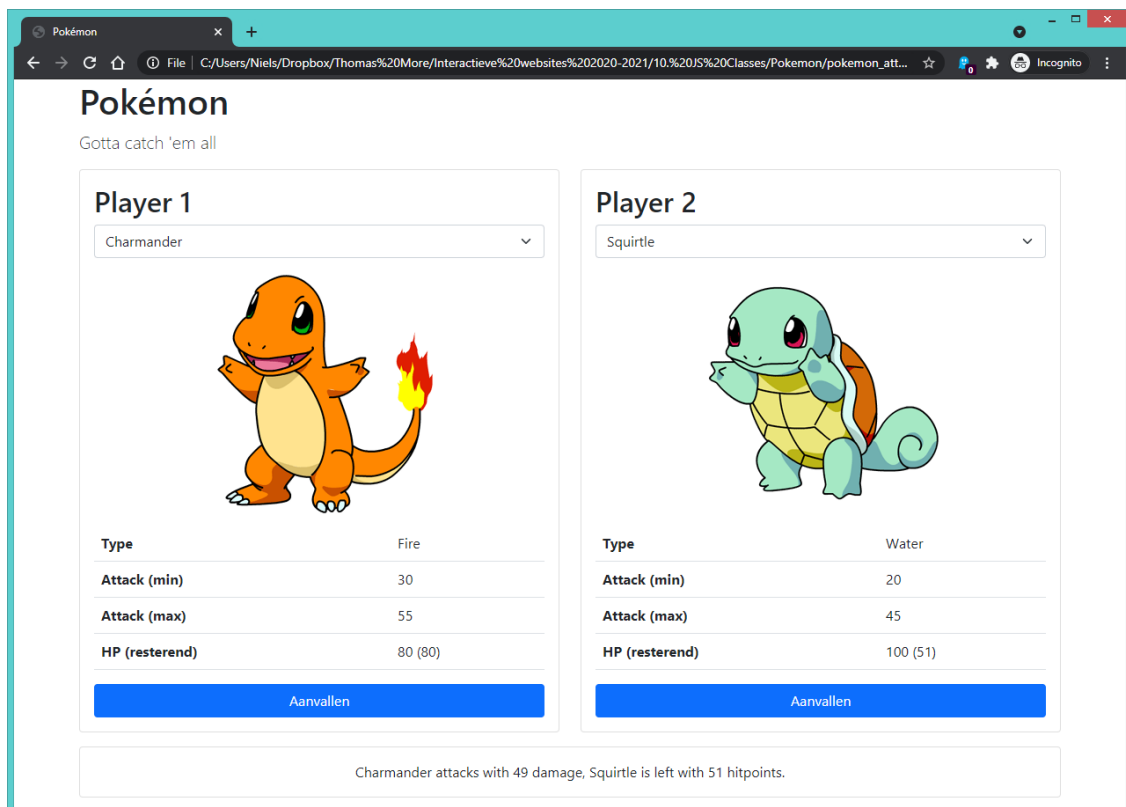
Wanneer het spel begint is het de bedoeling dat beide spelers een Pokémon kiezen. Wanneer een Pokémon gekozen is uit de lijst dan zullen alle eigenschappen van deze Pokémon worden weergegeven op het scherm.



In het voorbeeld zijn er maar vier Pokémons beschikbaar:

```
pokemon.push(new Pokemon("Squirtle", "Water", 20, 45, 100, "Squirtle.gif"));
pokemon.push(new Pokemon("Bulbasaur", "Grass", 25, 40, 120, "Bulbasaur.gif"));
pokemon.push(new Pokemon("Charmander", "Fire", 30, 55, 80, "Charmander.gif"));
pokemon.push(new Pokemon("Onix", "Ground", 20, 30, 150, "Onix.gif"));
```

Eens beide spelers een Pokémon gekozen hebben kunnen ze elkaar aanvallen zolang een Pokémon in leven is. Wanneer een Pokémon minder dan 0 HP heeft is de Pokémon dood en zal deze niet meer kunnen aanvallen.

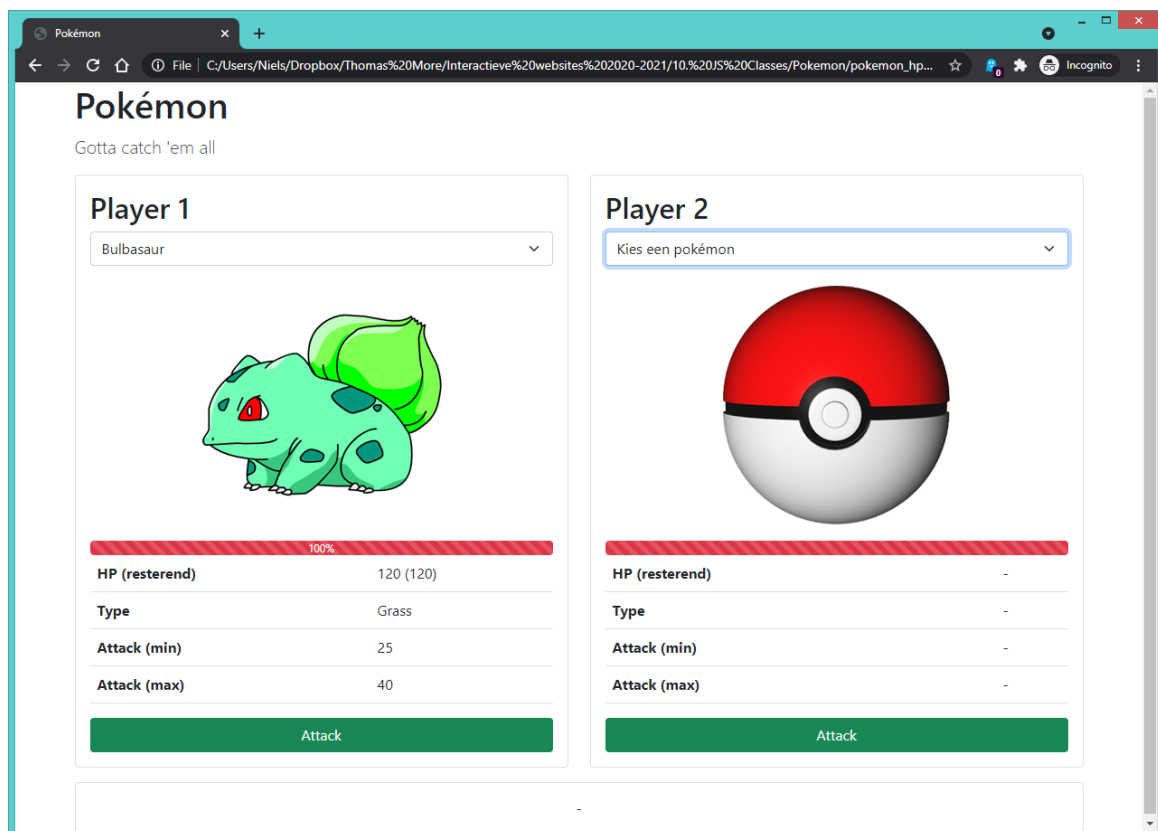


Bewaar deze oefening als **pokemon_attack.html**

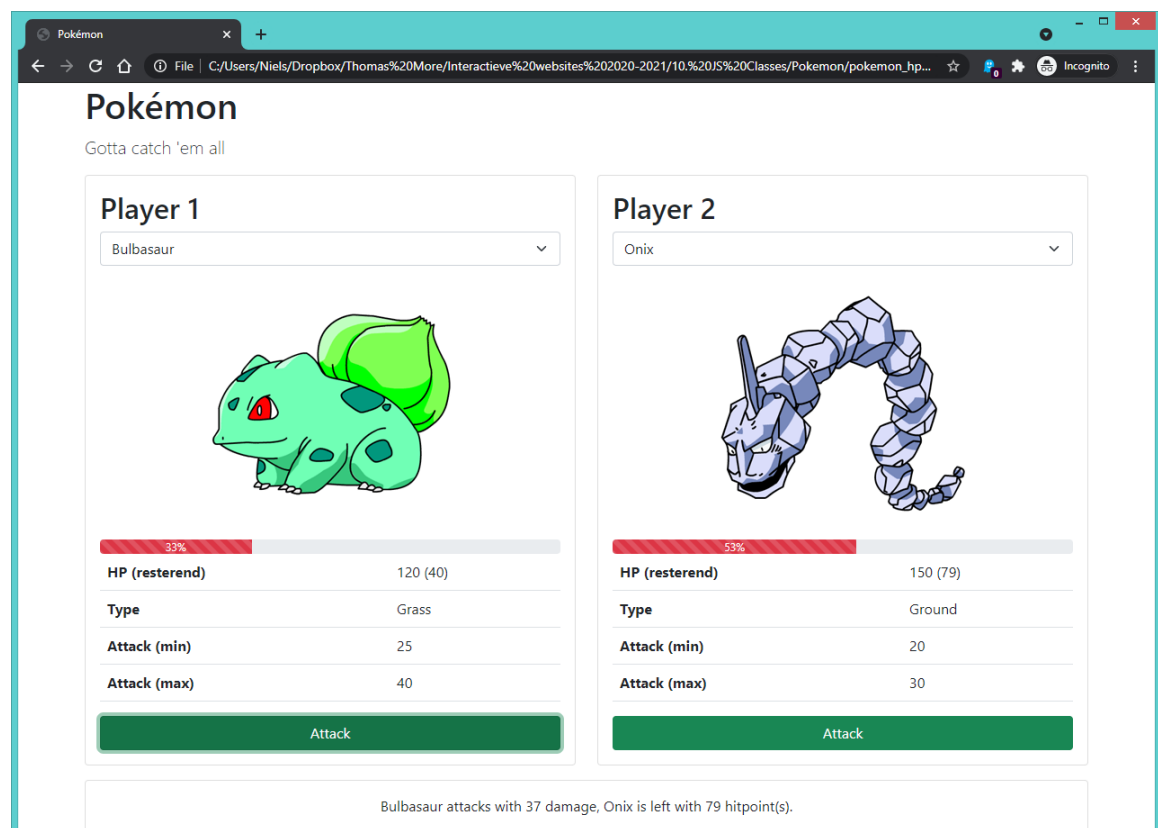
Versie 2

Zorg voor een beter visuele weergave van de **hitpoint bar** door gebruik te maken van een **progress component** van Bootstrap.

<https://getbootstrap.com/docs/5.0/components/progress/>



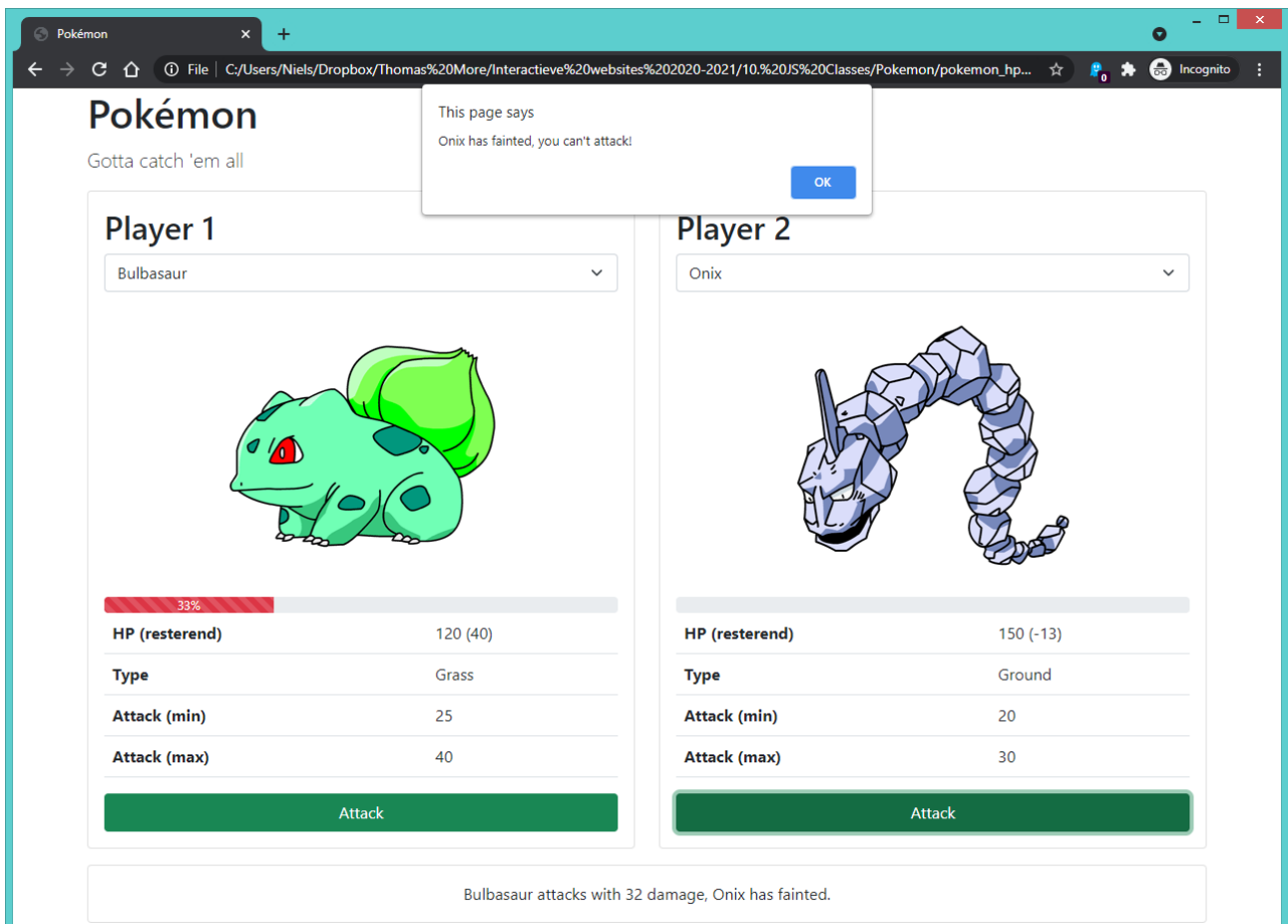
Wanneer een Pokémon is aangevallen zal zijn hitpoint bar natuurlijk wijzigen.



Omdat niet elke Pokémon 100 **hitpoints** heeft zal je de **current hitpoints** op 100 moeten berekenen voor de progress component. De progress component heeft een bepaalde width nodig, manipuleer hiervoor het style attribuut. Eens je weet hoeveel percentage een Pokémon nog over heeft kan je de progress component correct instellen. Het percentage berekenen kan je zo doen:

```
HPConvertedTo100() {  
    return Math.round((this.currentHP / this.hitpoints) * 100);  
}
```

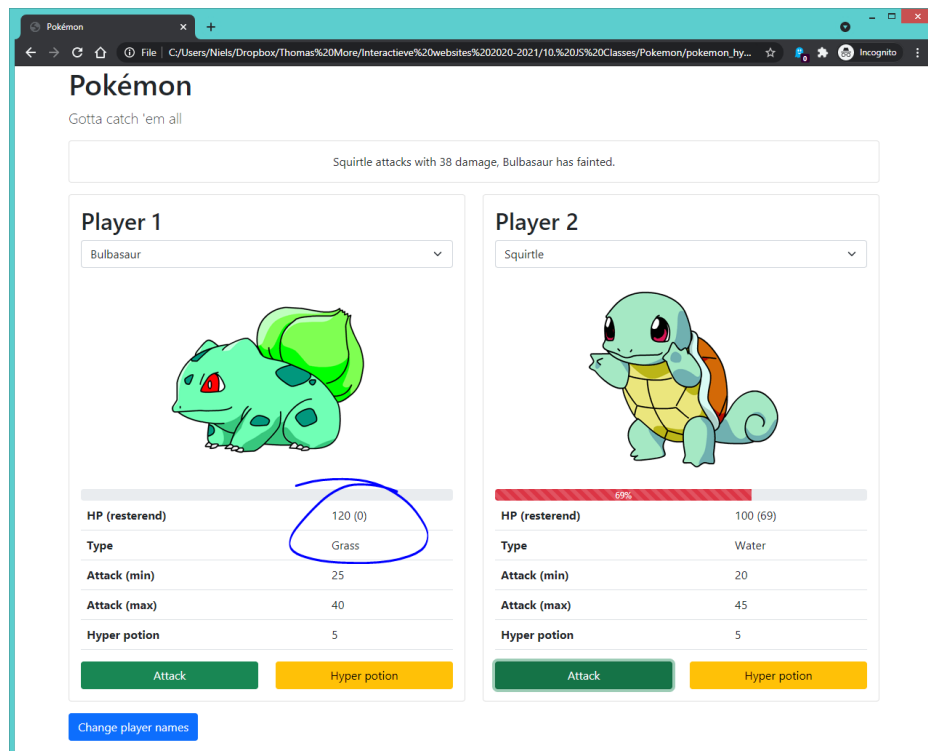
Het resultaat als een Pokémon onder 0 hitpoints gaat:



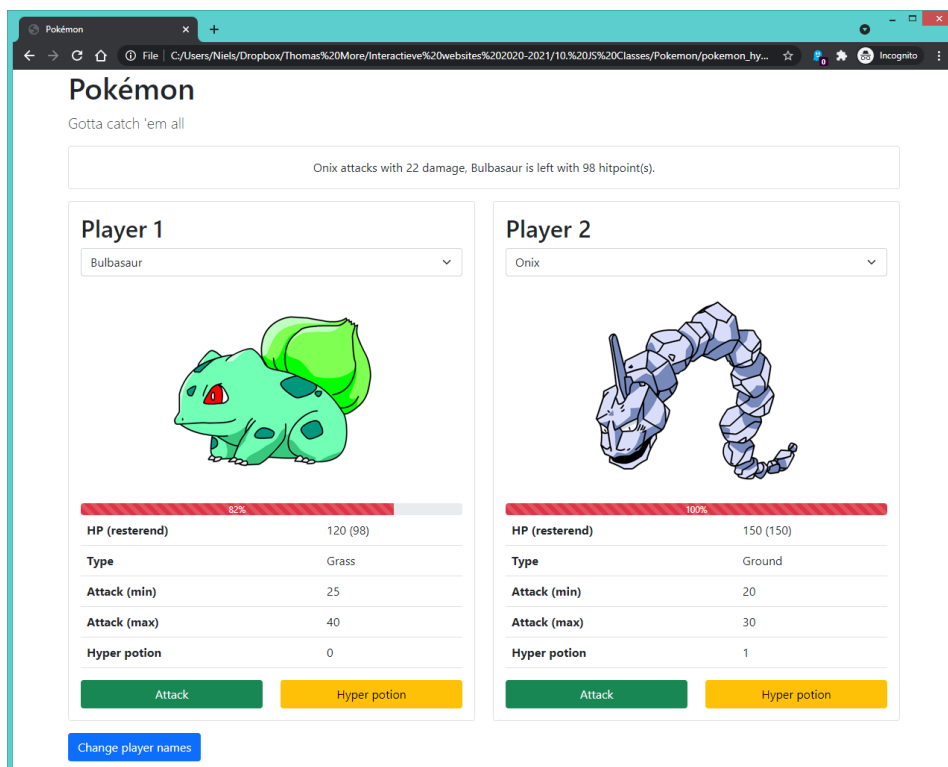
Bewaar deze oefening als **pokemon_hp.html**

Versie 3

Een Pokémon kan nog steeds **visueel** onder de **0 hitpoints** gaan in versie 2, los dit eerst op met behulp van een simpel **if-statement**.



Als laatste zorg je ervoor dat elke speler **5 hyper potions** krijgt. Een **hyper potion** kan een Pokémon **50 hitpoints** bijgeven. Wanneer een speler op de gele knop drukt zal het aantal hyper potions voor die speler verminderen met 1.



Bewaar deze oefening als **pokemon_hyperpotion.html**