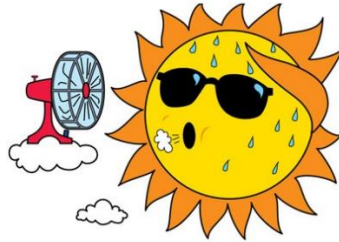


Je eigen weer app

Wil je weten of het lekker weer is om naar buiten te gaan? Om te spelen, te wandelen, te rennen of te shoppen. Of is het weer om lekker binnen te blijven? Je komt erachter met je eigen weer app.



We gaan gebruik maken van een website die weerinformatie geeft, wat Javascript, en het Vue-framework. Hieronder zie je een voorbeeld.

Hoe is het weer?

Den Haag

HAAL HET WEER OP

In Den Haag, Zuid-Holland, Nederland is het nu Helder. De temperatuur is 2,6 graden celsius. Tijd voor een wandeling! Doe wel handschoenen aan.

Wat heb je nodig?

Om te beginnen heb je de volgende dingen nodig:

- Een browser zoals Chrome, Firefox of Edge.
- Een programma om code te schrijven. Wij gebruiken hiervoor Visual Studio Code.

Visual Studio Code

Voor het schrijven van JavaScript code kan je notepad (of kladblok) gebruiken, maar erg prettig is dat niet. We gebruiken liever Visual Studio Code. Ook dit programma staat al geïnstalleerd, maar je kunt het ook gratis downloaden op: <https://code.visualstudio.com>

Webserver

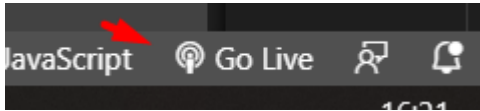
Een webserver zorgt ervoor dat de browser je app kan openen. Vroeger kon je die bestanden ook vaak direct vanaf de computer openen, maar de makers vonden dat niet zo veilig. Je kon er allerlei rare dingen mee doen en virussen mee oplopen. Dat willen we natuurlijk niet!

Je kunt in Visual Studio Code ook een webserver gebruiken. Je installeert hem zo:

Druk in Visual Studio Code op CTRL+P en type

```
ext install ritwickdey.liveserver
```

Druk op ENTER om de installatie te starten. Als het klaar is, zie je rechtsonder in je scherm:



De app kan je starten door op Go Live te klikken.

De onderdelen van je app

Voor het gemak hebben we map gemaakt waar al wat in staat. Start **Visual Studio Code** en kies **File > Open Folder**.

Ga naar de map `\javascript\opdrachten\lekkerweer` en kies **Select Folder**.

Hier zie je bestanden `index.html` en `lekkerweer.js`. In `index.html` staat dit:

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- load MUI -->
    <link href="//cdn.muicss.com/mui-0.10.3/css/mui.min.css"
          rel="stylesheet" type="text/css" />
  </head>
  <body id="weerapp">
    <div class="mui-container">

      </div>
    <script type="module" src="lekkerweer.js"></script>
  </body>
</html>
```

Het bestand `index.html` open je straks in een browser. Er zit een verwijzing in naar een zogenaamde stylesheet.

```
<link href="//cdn.muicss.com/mui-0.10.3/css/mui.min.css"
      rel="stylesheet" type="text/css" />
```

Deze stylesheet zorgt ervoor dat de onderdelen van de pagina er mooi uitzien.

Ook zie je deze regel.

```
<script type="module" src="lekkerweer.js"></script>
```

In het bestand `lekkerweer.js` ga je de code schrijven om weerinformatie op te halen. Het is niet helemaal leeg. Er zitten al vast een paar regels in die anders wel veel werk zouden zijn om over te typen.

Het weer dat we ophalen moeten we een plekje geven in het geheugen van de computer. We geven dit plekje een vorm waarmee je makkelijk gegevens kunt bewaren en weer ophalen. Die vorm noemen we in Javascript een class en die ziet er voor onze weer-app zo uit.

```
class Weer {  
  adres;  
  temperatuur;  
  tekst;  
  plaats;  
  icoon;  
  beschrijving;  
}
```

Zet deze class ook in `lekkerweer.js`. We kunnen dan straks de temperatuur, de plaats, en nog andere informatie bewaren. Een class is dus als een vorm. Net als een vormpje dat je gebruikt om koekjes te bakken. We hebben nu de vorm wel, maar er is nog geen koekje. Dat koekje, of object in Javascript, maken we zo.

```
let hetweer = new Weer();
```

Zet deze regel ook in je code. In het object `hetweer` stoppen we straks weerinformatie van een door jou ingevulde plaats. Je kunt ook alvast een plaatsnaam invullen:

```
hetweer.plaats = "Den Haag";
```

Zet deze regel er ook maar vast in. Je kunt natuurlijk ook een andere plaats invullen. Misschien wil je liever weten hoe koud het in Rovaniemi, de woonplaats van de kerstman, is.

Het weer ophalen

Het weer halen we op met een functie. Zet de onderstaande code vast in `lekkerweer.js` en dan bekijken we daarna wat hier gebeurt.

```
function haalHetWeerOp() {  
  var ophaaladres = url + hetweer.plaats + parameters;  
  
  fetch(ophaaladres).then(antwoord => {  
    if (!antwoord.ok) {  
      alert("Hé, dat ging fout! Dit is de foutcode: " + antwoord.status);  
    }  
    return antwoord.json();  
  }).then(weerdeata => vulWeer(weerdeata));  
}
```

Eerst zorgen we ervoor dat het webadres wordt gemaakt om het weer op te halen. Het adres bestaat uit een URL, de plaats en tot slot wat extra instructies. Die URL en die extra instructies staan boven in je Javascript-bestand. Kijk maar even.

We kunnen het weer ophalen met de instructie `fetch`. Na het ophalen van de gegevens kijken we of het goed ging. Daarom staat er `then`. We kijken naar het antwoord. Als het antwoord niet OK is, dan is het dus... niet OK! Met de functie `alert` kan je zien wat er fout ging. Misschien heb je een tikfout gemaakt? Of is de internetverbinding stuk?

Als het antwoord wel goed is, dan stuur je dit door met de instructie `return antwoord.json()`. We vragen van het antwoord om `json` omdat je daarmee makkelijk de juiste stukjes weerinformatie in ons Weer-object kan stoppen. Dit json ziet er namelijk zo uit.

```
{
  "latitude": 52.0841,
  "longitude": 4.31732,
  "resolvedAddress": "Den Haag, Zuid-Holland, Nederland",
  "currentConditions": {
    "datetime": "10:16:08",
    "datetimeEpoch": 1671095768,
    "temp": 1.0,
    "feelslike": -0.7,
    "conditions": "Licht bewolkt"
  }
}
```

Er zit nog veel meer in, maar je krijgt zo wel een idee. Deze informatie sturen we weer door naar een functie om het weer in ons Weer-object te krijgen. Deze functie ziet zo uit.

```
function vulWeer(weerdata) {
  hetweer.temperatuur = weerdata.currentConditions.temp;
  hetweer.beschrijving = weerdata.currentConditions.conditions;
  alert("De temperatuur is nu " + hetweer.temperatuur + " graden.");
}
```

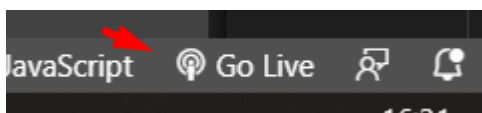
We pakken uit het resultaat de temperatuur en zetten deze in het veld temperatuur van ons Weer-object. Met de functie `alert` krijg je een melding op het scherm.

Je kunt de code nu al testen.

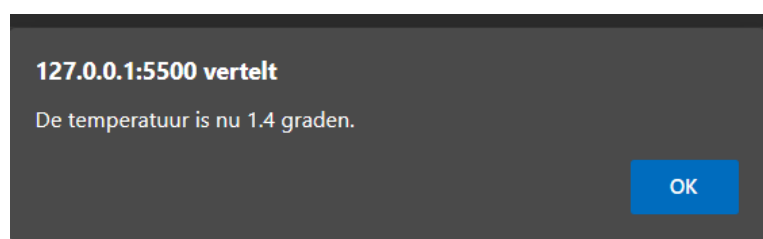
Zet onderaan in je Javascript-code deze regel.

```
haalHetWeerOp();
```

Start je programma via de knop Go Live. Zie je Go Live niet, maar wel Port: 5500, klik dan daarop. Dan verschijnt Go Live. Klik er nu op zodat je gelijk naar de browser gaat waar je programma staat.



Als alles goed is gegaan krijg je een melding.



Het is natuurlijk leuk om te weten hoe warm of koud het is, maar je hebt al gezien dat er meer te vertellen valt over het weer. Is het zonnig of bewolkt? En telkens dezelfde plaats is ook een beetje saai. Om onze weer-app nuttiger te maken moeten we meer doen.

Een pagina in HTML

Om het weer van andere plaatsen te kunnen ophalen hebben we een formulier nodig. Een formulier waar je een plaatsnaam kunt invullen. Dat formulier ziet er zo uit.

```
<form class="mui-panel" onsubmit="return false">
  <h1>Hoe is het weer?</h1>
  <div class="mui-textfield">
    <input type="text" placeholder="Plaatsnaam"
      v-model="plaats">
  </div>
  <button type="submit" @click="ophalen();"
    class="mui-btn mui-btn--primary mui-btn--raised"> Haal
    het weer op</button>
</form>
```

Zet deze regels in het bestand index.html tussen deze regels.

```
<div class="mui-container">
  ... hier staat je formulier ...
</div>
```

Het formulier lijkt wat ingewikkeld uit, maar als we deze wat simpeler maken is dit de basis.

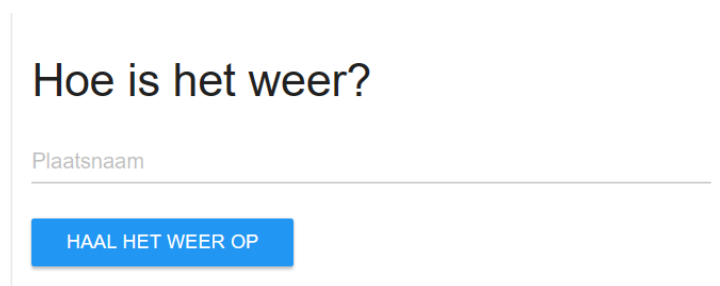
```
<h1>Hoe is het weer?</h1>
<input type="text">
<button type="submit">Haal het weer op</button>
```

Hier zie je dat het eigenlijk gaat om:

- 1) Een stukje tekst
- 2) Een veld om wat in te vullen
- 3) Een knop om op te klikken

Overal waar je **class** ziet geven we instructies aan de browser om teksten en knoppen een kleur en grootte te geven. We stoppen het invoerveld en de knop in een **form** zodat je ook op de ENTER knop kan drukken om de gegevens op te halen. Maar we willen niet dat de pagina dan helemaal opnieuw laadt, daarom staat er ook **onsubmit="return false"**.

Als je het goed is ziet je pagina er nu zo uit



Terug naar Javascript. Haal eerst de alert-instructie uit de functie `vulWeer()`. Anders krijg je bij het open van de pagina telkens een melding.

Werken met Vue

We gaan nu **Vue** gebruiken. **Vue** is een bibliotheek van Javascript-functies waarmee je makkelijk een HTML-pagina kan vullen met gegevens die in Javascript staan. In je Javascript-code staat al de verwijzing naar Vue.

```
// Gebruik Vue
import {
  createApp,
  reactive
} from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'
```

Deze regel haalt twee functies op van Vue: `createApp` en `reactive`. Verderop zie je hoe we deze twee functies gebruiken. Zoals gezegd, deze regels staan al in je code, dus die hoeft je niet over te typen.

Met de functie `createApp` maken we van je HTML-pagina een 'app'.

```
// Vue maakt van de pagina een soort App met gegevens en functies
let app = createApp({
  data() {
    return hetweer;
  },
  methods: {
    ophalen() {
      haalHetWeerOp();
    }
  }
}).mount('#weerapp');
```

Deze 'app' wordt met de functie `mount` gekoppeld aan je HTML-pagina. Kijk maar eens of je kunt vinden waar `weerapp` in je HTML pagina staat.

Deze HTML-pagina moeten we nog vullen met code om te laten zien wat we aan weerinformatie hebben opgehaald. Zet daarvoor de volgende HTML onder de regel waar je `</form>` ziet staan.

```
<div class="mui-panel" v-show="temperatuur!=null">
  <div>
    In {{adres}} is het nu {{beschrijving}}.
    De temperatuur is {{temperatuur}} graden
    celsius.
  </div>
</div>
```

Zie je het attribuut `v-show`? Dat is ook van Vue en hiermee kan je de browser vertellen dat het stukje tekst alleen zichtbaar mag zijn als er een temperatuur is gevuld. Dat is netter want anders staat er al tekst in je pagina zonder weerinformatie. Daar hebben we niks aan.

Alle teksten die tussen `{{` en `}}` staan verwijzen naar velden in het Weer-object. Dat zijn dus velden als `temperatuur` en `beschrijving`.

We zijn er bijna. Ga nu in je Javascript-code naar de regel waar dit staat:

```
let hetweer = new Weer();
```

Deze regel vervang je door de volgende.

```
let hetweer = reactive(new Weer());
```

Hiermee zorgen we ervoor dat de HTML-pagina reageert (**reactive**) op veranderingen in het Weer-object. Als we de temperatuur hebben opgehaald, dan zetten we die immers in het veld temperatuur. Dat is zo'n verandering. De browser moet weten dat iets veranderd is zodat deze de pagina kan bijwerken.

Oké, als alles goed is gegaan heb je nu een pagina waarmee je het weer van een plaats kunt ophalen.

Hoe is het weer?

Den Haag

HAAL HET WEER OP

In is het nu Licht bewolkt. De temperatuur is 4.4 graden celsius.

Valt je ook iets gekk op? Het lijkt wel of er iets mist in de tekst. Dat klopt! We hebben het adres niet gevuld. Zet daarom de volgende regel in de functie **vulWeer**.

```
hetweer.adres = weerdata.resolvedAddress;
```

Probeer het nog eens en nu zal je zien dat er een plaatsnaam met provincie en land getoond wordt.

Hoe is het weer?

Den Haag

HAAL HET WEER OP

In Den Haag, Zuid-Holland, Nederland is het nu Licht bewolkt. De temperatuur is 4.3 graden celsius.

Je kunt ook je eigen straat invullen. Maar soms zijn er wel veel straten met dezelfde naam in Nederland. Dus dan moet je ook de plaats erbij zetten. Als de plaats niet gevonden kan worden krijg je een melding met foutcode 400.

Je app uitbreiden

Als je er geen genoeg van kunt krijgen, kan je je app nog uitbreiden. Je kunt bijvoorbeeld een plaatje toevoegen die laat zien hoe het weer is: zonnig, bewolkt, regen of sneeuw. Voeg daarvoor de volgende regel toe in **vulWeer**.

```
hetweer.icoon = plaatjes + weerdata.currentConditions.icon + ".png";
```

In de HTML-pagina geef je het plaatje ook een plek. Dat doe je met deze regel.

```
<div class="mui--text-center"></div>
```

Ook kan je beslissen wat je wil doen met het weer. Bijvoorbeeld als het koud of warm is. Hieronder staat een voorbeeld.

```
if (hetweer.temperatuur < -10)
    hetweer.tekst = "Het is veel te koud buiten!";
if (hetweer.temperatuur < 0)
    hetweer.tekst = "Misschien kan je wel schaatsen!";
if (hetweer.temperatuur >= 0)
    hetweer.tekst = "Tijd voor een wandeling! Doe wel handschoenen aan.";
if (hetweer.temperatuur > 10)
    hetweer.tekst = "Prima weer voor een fietstochtje!";
if (hetweer.temperatuur > 20)
    hetweer.tekst = "Zoek alvast een plekje op het strand!";
if (hetweer.temperatuur > 30)
    hetweer.tekst = "Oef, het is warm! Ga zwemmen om af te koelen.";
if (hetweer.temperatuur > 40)
    hetweer.tekst = "Veel te heet buiten. Heb je een ventilator?";
```

Weet je nu ook hoe je deze tekst in je pagina kunt tonen? Hint: gebruik de tekens {{ en }}.

We kunnen veel meer weerinformatie krijgen. Open in je browser het volgende adres

<http://cddh.nl/weer>

Je krijgt dan een heleboel tekst te zien. Het is een beetje onoverzichtelijk. Selecteer alle tekst met CTRL-A en kopieer deze naar het klembord van je computer met CTRL-C.

Ga nu naar <https://jsonbeautifier.org/> en plak met CTRL-V de tekst in het linker paneel. Klik dan op **Beautify**. Nu ziet het er wat overzichtelijker uit. Probeer nu je Weer-object uit te breiden met andere gegevens zoals gevoelstemperatuur of het weer van morgen.