

JSON

Opdrachtomschrijving

Als student ICT en Media kom ik in aanraking met het verzamelen en verwerken van gegevens uit externe bronnen. JSON is een veelgebruikt formaat voor het structureren en opslaan van deze gegevens. Deze oefeningen zijn bedoeld om te leren werken met JSON. Tijdens de les hebben we uitleg en opdrachten gekregen over het werken met JSON. We hebben verschillende voorbeelden behandeld. Vanuit deze voorbeelden was het de bedoeling om zelf met JSON een visuele weergave te maken.

Introductie

```
[
  {
    "index": 0,
    "picture": "http://placeholder.it/32x32",
    "age": 30,
    "eyeColor": "green",
    "name": "Theresa Fischer"
  },
  {
    "index": 1,
    "picture": "http://placeholder.it/32x32",
    "age": 35,
    "eyeColor": "blue",
    "name": "Ingrid Flores"
  }
]
```


`{ }` generated.json

JSON staat voor JavaScript Object Notation. Het is een op JavaScript gebaseerde manier om gegevens over te dragen en op te slaan. Om een json-file aan te maken met data heb ik gebruik gemaakt van de website: <https://json-generator.com/>. Om deze data in een bestand te zetten heb ik een file aangemaakt, genaamd generated.json.

```
<ul id="peoplelist"></ul>
```

Dit is een HTML-element, met de id 'peoplelist', dat een lege ongeordende lijst weergeeft. Hierin komen de json gegevens te staan.

```
async function logJSONData() {
  const response = await fetch("generated.json");
  const jsonData = await response.json();
  console.log(jsonData);
}
logJSONData()
```

Loaded new content contentScript.js:6
script.js:4
▼ Array(7) 
▶ 0: {index: 0, picture: 'http://placeholder.it/32x32', ag
▶ 1: {index: 1, picture: 'http://placeholder.it/32x32', ag
▶ 2: {index: 2, picture: 'http://placeholder.it/32x32', ag
▶ 3: {index: 3, picture: 'http://placeholder.it/32x32', ag
▶ 4: {index: 4, picture: 'http://placeholder.it/32x32', ag
▶ 5: {index: 5, picture: 'http://placeholder.it/32x32', ag
▶ 6: {index: 6, picture: 'http://placeholder.it/32x32', ag
length: 7
▶ [[Prototype]]: Array(0)

Deze code bevat de asynchrone functie logJSONData, die JSON-gegevens ophaalt van een extern bestand, ze converteert naar een JavaScript-object en ze in de console weergeeft. Het proces begint met het ophalen van gegevens van "generated.json". Het fetch-gedeelte stuurt een HTTP-verzoek naar de opgegeven URL (in dit geval het JSON-bestand) en wacht tot er een reactie terugkomt. Vervolgens wordt de ontvangen reactie verwerkt en toegewezen aan de variabele response. Daarna wordt het JavaScript-object toegewezen aan de variabele jsonData en weergegeven in de console met console.log(jsonData). De functie logJSONData wordt aangeroepen op het moment dat het script is geladen.

```

async function logJSONData() {
  const response = await fetch("generated.json");
  const jsonData = await response.json();
  jsonData.forEach(data => console.log(data));
}
logJSONData();

```

Loaded new content

```

contentScript.js:6
script.js:14
{index: 0, picture: 'http://placeholder.it/32x32', age:
30, eyeColor: 'green', name: 'Theresa Fischer'}
script.js:14
{index: 1, picture: 'http://placeholder.it/32x32', age:
35, eyeColor: 'blue', name: 'Ingrid Flores'}
script.js:14
{index: 2, picture: 'http://placeholder.it/32x32', age:
26, eyeColor: 'green', name: 'Rivera Murphy'}
script.js:14
{index: 3, picture: 'http://placeholder.it/32x32', age:
20, eyeColor: 'blue', name: 'Jefferson Tyler'}
script.js:14
{index: 4, picture: 'http://placeholder.it/32x32', age:
29, eyeColor: 'green', name: 'Shelia Wong'}
script.js:14
{index: 5, picture: 'http://placeholder.it/32x32', age:
34, eyeColor: 'blue', name: 'Shawn Wiggins'}
script.js:14
{index: 6, picture: 'http://placeholder.it/32x32', age:

```

Aan deze code is de regel 'jsonData.forEach(data => console.log(data))' toegevoegd. Deze regel maakt gebruik van de forEach-methode om door elk element in het jsonData-array te doorlopen. Voor elk element wordt een arrow-functie uitgevoerd, waarin de waarde van het element wordt doorgegeven als de parameter "data". Binnen de arrow-functie wordt de waarde van elk element in het jsonData-array weergegeven in de console met behulp van console.log(data). Met andere woorden, deze regel code logt elk element in het jsonData-array naar de console.

```

const naam = document.getElementById('peoplelist')
fetch('generated.json')
  .then(response => response.json())
  .then(data => jsonopscherm(data));

function jsonopscherm(data) {
  //console.log(data)
  data.forEach(element => {
    console.log(element.name)
    naam.innerHTML += "<li>" + element.name + "</li>"
  })
}

```

- Theresa Fischer
- Ingrid Flores
- Rivera Murphy
- Jefferson Tyler
- Shelia Wong
- Shawn Wiggins
- Klein Gonzales

Deze code selecteert een HTML-element met de id "peoplelist" en haalt asynchroon gegevens op vanuit het bestand "generated.json". Vervolgens worden de JSON-gegevens verwerkt met behulp van response.json() en doorgegeven aan de functie jsonopscherm(data).

De functie jsonopscherm(data) doorloopt de ontvangen gegevens en toont de naam van elk element in de console met console.log(element.name). Daarnaast wordt de naam van elk element toegevoegd aan de innerHTML van het geselecteerde HTML-element met de id "peoplelist", waarbij elk element wordt weergegeven als een lijstitem in HTML.

JSON bij yoga-artikelen

```
[
  {
    "id": 1,
    "picture": "yogamat.jpg",
    "price": "40.00",
    "color": "oranje",
    "name": "Yogamat"
  },
  {
    "id": 2,
    "picture": "bolster.jpg",
    "price": "35.00",
    "color": "blue",
    "name": "Bolster"
  },
  {
    "id": 3,
    "picture": "yogariem.jpg",
    "price": "26.00",
    "color": "yellow",
    "name": "Yogariem"
  },
  {
    "id": 4,
    "picture": "yogawiel.jpg",
    "price": "20.00",
    "color": "black",
    "name": "Yogawiel"
  }
]
```

```
<div id="productlist">
</div>
```

Ik heb een aantal yoga producten toegevoegd aan een nieuw JSON-bestand, genaamd 'products.json'. Elk product heeft een unieke id, foto, prijs, kleur en naam gekregen. In het HTML-bestand heb ik een <div> aangemaakt met een id 'productlist'.

```
const producten = document.getElementById('productlist')
fetch('products.json')
  .then(response => response.json())
  .then(data => jsonopscherm(data));

function jsonopscherm(data) {
  //console.log(data)
  data.forEach(element => {
    console.log(element.name)
    producten.innerHTML += "<li>" + element.name + "</li>"
    producten.innerHTML += "<li>" + element.picture + "</li>"
    producten.innerHTML += "<li>" + element.price + "</li>"
  })
}
```

```
→
• Yogamat
• yogamat.jpg
• 40.00
• Bolster
• bolster.jpg
• 35.00
• Yogariem
• yogariem.jpg
• 26.00
• Yogawiel
• yogawiel.jpg
• 20.00
```

Deze code begint met het selecteren van een HTML-element met de id 'productlist' en wijst dit toe aan de variabele 'producten'. Vervolgens wordt de functie 'fetch' gebruikt om gegevens op te halen vanuit het JSON-bestand 'products.json'. Deze asynchrone functie maakt gebruik van Promise-callbacks. De eerste 'then'-methode wordt uitgevoerd nadat de fetch succesvol is voltooid en zet de ontvangen gegevens om naar een JavaScript-object met behulp van de '.json()' methode. Daarna wordt een tweede 'then'-methode uitgevoerd zodra de JSON-gegevens zijn verwerkt en beschikbaar zijn. Deze methode start als de verwerkte data succesvol is voltooid.

De functie genaamd 'jsonopscherm' maakt gebruik van de parameter 'data'. Binnen de functie wordt een loop uitgevoerd over elk element in de 'data' array. Voor elk element worden de volgende stappen uitgevoerd. Ten eerste wordt de naam van het element afgedrukt naar de console. Er wordt een element geselecteerd met de id 'producten'. Aan dit element worden de naam, afbeelding en prijs van het huidige element in de loop toegevoegd. Op deze manier worden de namen, afbeeldingen en prijzen van elk element in de 'data' array weergegeven op een webpagina in de vorm van een lijst.

```
function jsonopscherm(data) {  
  data.forEach(element => {  
    console.log(element.name);  
    producten.innerHTML += "<li>" + element.name + "<img src='" + element.picture + "'/>" + element.price + "</li>";  
  });  
}
```



Ik heb de code ingekort door de regel: producten.innerHTML += "" + element.name + "" + element.price + "".

Door "" te gebruiken wordt de afbeelding op de pagina getoond in plaats van de naam van de afbeelding.

```
const producten = document.getElementById('productlist')
fetch('products.json')
  .then(response => response.json())
  .then(data => jsonopscherm(data));

function jsonopscherm(data) {
  data.forEach(element => {
    const productBox = document.createElement('div');
    productBox.className = 'product-box';

    productBox.innerHTML = `
      <h2>${element.name}</h2>
      
      <p>Price: €${element.price}</p>
    `;

    producten.appendChild(productBox);
  });
}
```

In de code wordt voor elk element een nieuw div-element aangemaakt en aan de variabele productBox toegewezen. Dit div-element heeft de className 'product-box'. Vervolgens wordt de innerHTML-eigenschap van het productBox-element gebruikt om HTML-inhoud te genereren.

Door middel van de template literals (``) kunnen de huidige elementen ingevuld worden in het template voor productBox.innerHTML. De gegevens van het huidige element, zoals de naam (element.name), afbeelding (element.picture) en prijs (element.price), worden ingevuld. Tot slot wordt het productBox-element toegevoegd aan de producten-container met behulp van de appendChild()-methode.

```
#productlist {
  display: flex;
  flex-wrap: wrap;
  column-gap: 20px;
  justify-content: center;
  margin-top: 100px;
}

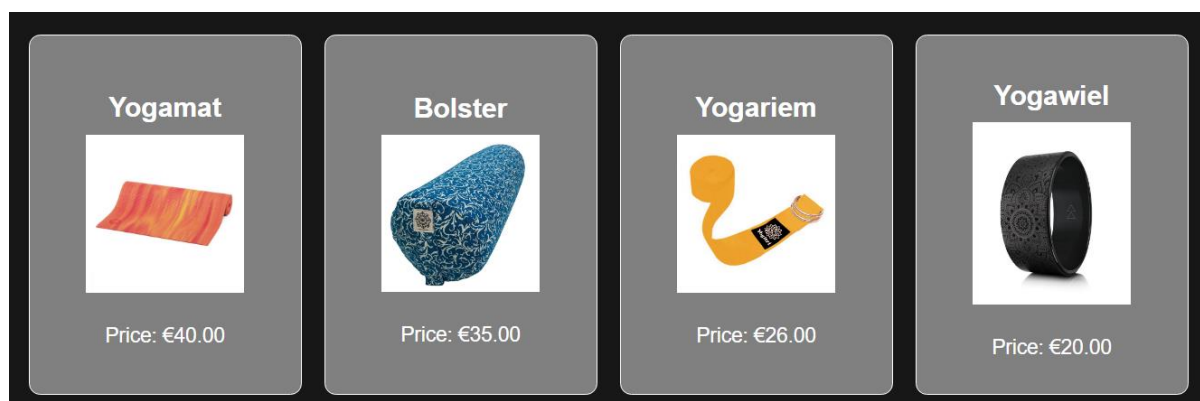
.product-box {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  border: 1px solid white;
  background-color: grey;
  padding: 20px;
  margin-bottom: 20px;
  width: 200px;
  border-radius: 10px;
}

h2 {
  font-size: 24px;
  margin-bottom: 10px;
}

img {
  max-width: 70%;
  height: auto;
  margin-bottom: 10px;
}

p {
  font-size: 18px;
  margin-bottom: 10px;
}
```

De HTML-elementen productlist, product-box, h2, img en p hebben een CSS opmaak gekregen.



Dit is hiervan het resultaat.