

## DEEL 1 + DEEL 2

We gebruiken altijd 'let' om variabelen te declareren en geen 'var'.

```
let tekst2 = "Hello World";
```

- We definiëren onze functies (methods) altijd als arrow functions :

```
const setup = () => {
```

```
...
```

```
}
```

Stop in het index.js bestand de volgende code :

```
const setup = () => {
```

```
// deze code wordt pas uitgevoerd als de pagina volledig is ingeladen
```

```
}
```

```
window.addEventListener("load", setup);
```

```
let globalVar="dit is een globale variabele";
```

```
const mijnFunctie = () => {
```

```
  let localVar="dit is een lokale variabele";
```

```
}
```

Je kunt als volgt een leeg array aanmaken :

```
let leeg = [];
```

Een voorgevuld array aanmaken

```
let teksten = ["een", "twee", "drie"];
```

```
let getallen = [1, 2, 3];
```

```
let gemixt = [true, "hallo", 4]; // verschillende soorten elementen!
```

Let erop dat een array in javascript waarden mag bevatten van verschillende types, zoals het 'gemixt'

voorbeeld hierboven demonstreert.

Het aantal elementen in een array opvragen gebeurt via de .length property :

```
let elementen = ["een", "twee", "drie"];
```

```
console.log ( elementen.length ); // toont 3
```

Elementen in een array worden op basis van hun posities aangeduid, beginnend bij 0. Het eerste element staat dus op indexpositie 0.

Je kan individuele element opvragen op basis van een index :

```
let elementen = ["een", "twee", "drie"];  
console.log( elementen[0] ); // toont "een"  
console.log( elementen[1] ); // toont "twee"  
console.log( elementen[2] ); // toont "drie"
```

Om alle elementen in een array te overlopen kunnen we een for-loop gebruiken :

```
let elementen = ["een", "twee", "drie"];  
for (let i=0 ; i<elementen.length ; i++) {  
  console.log( elementen[i] );  
}
```

Enkele andere nuttige methods om makkelijker met arrays te werken :

`indexOf(element, idx)`

`lastIndexOf(element, idx)`

geeft de laagste/hogste index waarop het element gevonden werd

elementen worden vergeleken met ===

indien het elementen niet wordt gevonden, geeft dit -1 terug

de idx parameter is optioneel en geeft aan op welke index de zoektocht moet beginnen

`push(element)`

voegt het element toe op het einde van het array

geeft de nieuwe lengte terug

`pop()`

verwijdert het laatste element van het array

geeft het verwijderde element terug

`unshift(element)`

voegt het element toe aan het begin van het array

geeft de nieuwe lengte terug

`shift()`

verwijdert het eerste element van het array

geeft het verwijderde element terug

`slice(i1, i2)`

geeft een nieuw array terug met de elementen vanop index i1 tot index i2 (exclusief)

`splice(...)`

een ietwat schizofrene functie waarmee je elementen in een array kunt inlassen of

verwijderen

zie [http://www.tutorialspoint.com/javascript/array\\_splice.htm](http://www.tutorialspoint.com/javascript/array_splice.htm)

**a) Schrijf een simpel programma dat “Hello world” op de console zet. Pas daarna in Webstorm de tekst**

**aan naar “Hello world!!!!!!!!!!” en probeer opnieuw.**

```
const aantal = 4;
const setup = () =>{
    let steretjes = "";
    for (let i = 0; i < aantal; i++) {
        for (let j = 0; j < aantal; j++) {
            steretjes += "*";
        }
        steretjes += "\n";
    }
    return console.log(steretjes);
    console.log(ster);
}
```

```
const sterr = 8;
const ster=()=>{
    let ster = "";
    for (let i = 0; i < sterr; i++) {
        for (let j = 0; j < sterr; j++) {
            ster+="*";
        }
        ster+="\n";
    }
    return console.log(ster);
}
```

```
window.addEventListener('load', setup);
window.addEventListener('load', ster);
```



In javascript kun je drie soorten popups op het scherm plaatsen :

```
window.alert("Dit is een mededeling");
```

Toont de mededeling in een popup met een 'ok' button

```
window.confirm("Weet u het zeker?");
```

Toont de vraag in een popup met een 'ok' en 'cancel' button

De return waarde geeft aan op welke knop de gebruiker duwde.

```
window.prompt("Wat is uw naam", "onbekend");
```

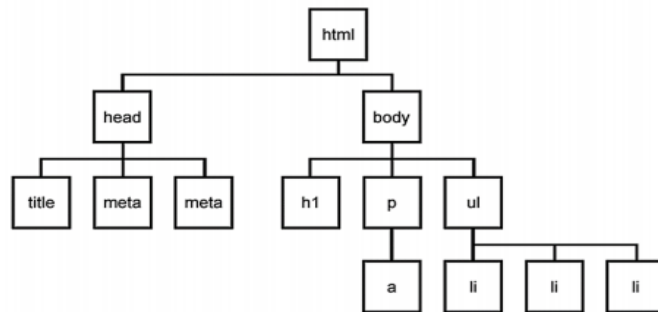
```
let totaal = 0;
for (let i = 1; i < 500; i++) {
  if (i%3===0 || i%5===0 ) {
    console.log(i);
    totaal+=i;
  }
}
console.log("Totaal is " + totaal);
window.alert("Dit is een mededeling");
window.prompt("Put ur bank details please including password");
window.confirm("lol dont test me");
window.addEventListener('load', setup);
```

## DEEL 3

Bij wijze van voorbeeld nemen we eens het volgende HTML document :

```
<!DOCTYPE html>
<html>
  <head>
    <title>White Russian FTW!</title>
    <meta charset="UTF-8">
    <meta name="author" content="Jeff Lebowski ">
  </head>
  <body>
    <h1>The dude abides!</h1>
    <p>Making a <a href="https://en.wikipedia.org/wiki/White_Russian_(cocktail)">White
    russian</a> is easy man, you only need</p>
    <ul>
      <li>Vodka</li>
      <li>Coffee liqueur</li>
      <li>Cream</li>
    </ul>
  </body>
</html>
```

De DOM-tree die hiermee correspondeert kunnen we schematisch als volgt weergeven :



**document.getElementById("abc");**

doorzoekt de DOM-tree naar het element met id="abc" en retourneert een verwijzing naar dit element. Indien geen element wordt gevonden, retourneert de method de null waarde.

- **document.getElementsByClassName("xyz");** let op : getElementS meervoud!!

doorzoekt de DOM-tree naar elementen met class="xyz" en retourneert een lijst met verwijzingen naar deze elementen. De volgorde van de elementen in die lijst is dezelfde als hun onderlinge volgorde in het document. De lijst is een NodeList object en is op dezelfde manier te gebruiken als een array - dus wat ons betreft is het verschil niet belangrijk. Indien geen enkel element wordt gevonden, retourneert de method een lege lijst (vgl. lege array).

- **document.getElementsByTagName("img");**

doorzoekt de DOM-tree naar <img> elementen en retourneert een lijst met verwijzingen

naar deze elementen. Je krijgt hetzelfde soort lijst als bij `getElementsByClassName`, maar de elementen werden gezocht op basis van hun tag i.p.v. hun CSS-classes.

Bijvoorbeeld :

```
<a id="lnkVlaanderen" href="http://www.vlaanderen.be">Vlaanderen</a>
```

...

```
let link = document.getElementById("lnkVlaanderen");
```

```
console.log ( link.textContent ); // toont "Vlaanderen"
```

Bijvoorbeeld :

```
<p class="para">blabla1</p>
```

```
<p class="para">blabla2</p>
```

```
<p class="para">blabla3</p>
```

...

```
let paras = document.getElementsByClassName("para");
```

```
paras[1].textContent = "boemboem2"; // verandert blabla2 in boemboem2
```

Bijvoorbeeld :

```
let paragrafen=document.getElementsByTagName("p");
```

```
for (let i=0;i<paragrafen.length;i++) {
```

```
    paragrafen[i].textContent="Hello world!";
```

```
}
```

```
const setup = () => {
```

```
    // wijzig de tekst van de paragraaf
```

```
    let pElement=document.getElementById("txtOutput");
```

```
    pElement.textContent="Welkom!";
```

```
    let paragraf = document.getElementsByTagName("p");
```

```
    for (let i = 0; i < paragraf.length; i++) {
```

```
        paragraf[i].textContent="Veranderd elementsbytagname";
```

```
    }
```

```
}
```

```
/*
```

```
let iElement = document.getElementById("txtOutput");
```

```
iElement.textContent="Hello world!"*/
```

```
window.addEventListener("load", setup);
```

Je kunt de tekst van een element opvragen door diens `.textContent` property te lezen.

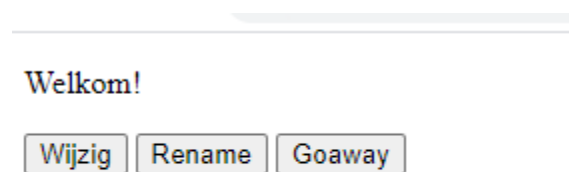
Bijvoorbeeld

```
<a id="lnkVlaanderen" href="http://www.vlaanderen.be">Vlaanderen</a>
```

...

```
let link = document.getElementById("lnkVlaanderen");
```

```
console.log ( link.textContent ); // toont "Vlaanderen"
```



```
const setup = () => {
  let btnWijzig = document.getElementById("btnWijzig");
  btnWijzig.addEventListener("click", wijzig);
  let btnRename = document.getElementById("btnRename");
  btnRename.addEventListener("click", rename);
  let btnGoaway = document.getElementById("btnGoaway");
  btnGoaway.addEventListener("click", goaway);
}

const wijzig = () => {
  let pEl = document.getElementById("txtOutput");
  pEl.textContent = "Welkom!";
}

const rename = () => {
  let pr = document.getElementById("txtOutput");
  pr.textContent = "Hello-World";
}

const goaway = () => {
  let ga = document.getElementById("txtOutput");
  ga.textContent = "Go-Away";
}
```

```
window.addEventListener("load", setup);
```

Bijvoorbeeld

```
<p id="txtDemo">Hier staat een beetje tekst</p>
```

...

```
let txtDemo = document.getElementById("txtDemo");
```

```
txtDemo.innerHTML = 'Make <a href="http://www.vlaanderen.be">Flanders</a> great again.'
```

Het veranderen van de .innerHTML property van een element, verwijdert altijd de kinderen van dit element en vervangt ze door andere!

Bijvoorbeeld

```
<ul id="lstBoodschappen">
```

```
<li>brood</li>
```

```
<li>melk</li>
```

```
</ul>
```

...

```
let lstBoodschappen = document.getElementById("lstBoodschappen");
```

```
lstBoodschappen.innerHTML += "<li>kaas</li>";
```

Dit lijkt enkel een derde list item "kaas" toe te voegen, maar in werkelijkheid zijn ook de "brood" en "melk" items vervangen (door nieuwe elementen met dezelfde tekstuele inhoud).

```
txtDemo.textContent = "dit is een <a href='http://www.example.com'>hyperlink</a>";
```

geeft : dit is een <a href='http://www.example.com'>hyperlink</a>

```
txtDemo.innerHTML = "dit is een <a href='http://www.example.com'>hyperlink</a>";
```

geeft : dit is een hyperlink

Bijvoorbeeld

```
<input type="text" id="txtInput">
```

...

```
let txtInput = document.getElementById("txtInput");
```

console.log( txtInput.value );



Dit zal de tekst in het tekstveld op de console zetten.

`.value` hoort enkel bij `<input>` elementen

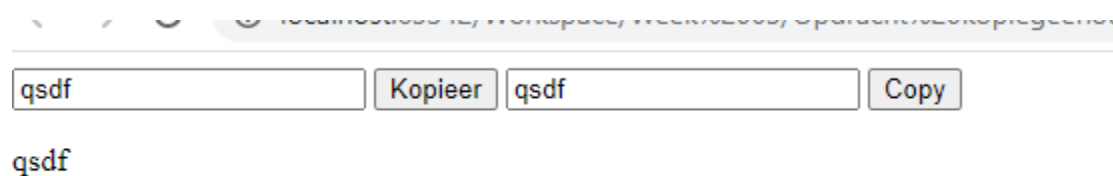
- `.textContent` en `.innerHTML` horen nooit bij `<input>` elementen

o Strikt genomen, “bijna nooit” bv. wel als je de tekst op een button wil veranderen

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <link rel="stylesheet" href="styles/index.css">
  <script src="scripts/index.js"></script>

</head>
<body>
  <input id="txtInput" type="text" />
  <input id="btnKopieer" type="button" value="Kopieer" />

  <input type="text" id="tekstininput"/>
  <input type="button" id="btnCopy" value = "Copy"/>
  <p id="txtOutput"> geen output </p>
</body>
</html>
```



```
const setup = () => {
  // zorg ervoor dat een klik op de knop onze 'kopieer'
  function oproept
  let btnKopieer = document.getElementById("btnKopieer");
  btnKopieer.addEventListener("click", kopieer);

  let btnCopy = document.getElementById("btnCopy");
  btnCopy.addEventListener("click", Copy);
}

const kopieer = () => {
```

```

    // bemachtig een verwijzing naar het input element in de
    DOM-tree
    let txtInput = document.getElementById("txtInput");

    // vraag de tekst op van het input element
    let tekst = txtInput.value;
    let txtoutput = document.getElementById("txtOutput");
    txtoutput.textContent = tekst;
    // zet de tekst op de console
    console.log(tekst);
}

const Copy = () =>{
    let tekstinput=document.getElementById("tekstinput");
    let text = tekstinput.value;
    let txtOutput = document.getElementById("txtOutput");
    txtOutput.textContent= text;
    console.log(text);
}

// zorg ervoor dat onze 'setup' function wordt uitgevoerd als de
pagina is ingeladen (en de DOM-tree is opgebouwd)
window.addEventListener("load", setup);

```

## DEEL 4

Per element in de DOM-tree kunnen we onze code aan een specifiek soort event koppelen met de `addEventListener` method :

```

elementNode.addEventListener("click", ourFunction);

```

Bijvoorbeeld, om een function **printHello** uit te laten voeren wanneer de gebruiker op een <div> met id "abc" klikt moeten we volgende code uitvoeren :

in de HTML file

```
<div id="abc">...</div>
```

in de javascript file

```
let elementNode=document.getElementById("abc");
elementNode.addEventListener("click", printHello)
```

Merk op dat er geen haakjes ( ) staan achter de naam van de callback functie !!!!!



#### Tip van Flip 4

Zet in deze lessenreeks nooit haakjes ( ) achter de naam van de callback functie bij het registreren van een event handler. Doe je dit wel, dan zal de callback functie nooit worden opgeroepen (maar je krijgt helaas geen error op de console).

voorkomt en stop daarin 4 paragrafen (<p> elementen). In  
an de bekende Lorem Ipsum tekst ("Lorem ipsum dolor sit

en lichtgrijze achtergrond geeft en 30px padding, de  
grond.

e <div> klikt, verschijnt er het woordje **klik** op de console.  
> het lichtgrijze gebied als op de gele gebieden te klikken!

sit amet, consectetur adipiscing elit. Architecto  
caecati pariatur perspiciatis qui sunt tenetur voluptates .  
ue sint sunt totam voluptatum.



Uitvoer filteren

Fouten	Waarschuwingen	Logboeken	Info	Debuggen	CSS
Live reload enabled.					
klik op paragraf					
klik op div					
klik op paragraf					
klik op div					
klik op paragraf					
klik op div					
klik op paragraf					
klik op div					
klik op paragraf					
klik op div					
klik op paragraf					
klik op div					

>>

```

const setup = () => {
  // deze code wordt pas uitgevoerd als de pagina volledig is ingeladen
  let div = document.getElementById("klikme");
  div.addEventListener("click", klikopdiv);
  let parag = document.getElementsByClassName("para");
  for (let i = 0; i < parag.length; i++) {
    parag[i].addEventListener("click", klikopparagraf);
  }
}

const klikopdiv = () =>{
  console.log("klik op div");
}

const klikopparagraf = () =>{
  console.log("klik op paragraf");
}

//
window.addEventListener("load", setup);

```

## Opdracht 02

Pas de oplossing van opdracht 01 aan zodat er ook event listeners aan de <div> gekoppeld worden voor de onderstaande events ?

- Event "mouseenter" : de listener toont "enter" op de console
- Event "mousemove" : de listener toont "move" op de console
- Event "mouseleave" : de listener toont "leave" op de console

(let op de kleine letters in de event namen, bv. mouseenter en niet mouseEnter)

In totaal worden er aan het <div> element 4 verschillende event listeners gekoppeld, je moet hiervoor dus 4 functies voorzien.

```

const setup = () => {
  // deze code wordt pas uitgevoerd als de pagina volledig is ingeladen
  let div = document.getElementById("klikme");
  div.addEventListener("click", klikopdiv);
  div.addEventListener("mousemove", muisBeweegt);
  div.addEventListener("mouseenter", muisEnter);
  div.addEventListener("mouseleave", muisLeave);

  let parag = document.getElementsByClassName("para");
  for (let i = 0; i < parag.length; i++) {
    parag[i].addEventListener("click", klikopparagraf);
  }
}

const muisBeweegt = () =>{
  console.log("Muis beweegt");
}

const muisEnter =() =>{
  console.log("Muis Enter");
}

const muisLeave = ()=> {
  console.log("Muis Leave");
}

const klikopdiv = () =>{
  console.log("klik op div");
}

const klikopparagraf = () =>{
  console.log("klik op paragraf");
}

//
window.addEventListener("load", setup);

```

### Opdracht 03

Pas de oplossing van opdracht 1 aan zodat een klik op het <div> element diens achtergrondkleur rood maakt.

Doe dit door de CSS-style property 'backgroundColor' aan te passen.

Merk op dat direct ingestelde CSS-properties altijd voorrang krijgen op waarden die via een CSS regel worden ingesteld uit het CSS bestand. Je kunt dit makkelijk zelf nagaan op het Elements tabblad in Chrome Developer Tools.

Deze voorrangstelling is precies dezelfde als bij "styling via het style attribuut in HTML" t.o.v. "styling via CSS-regels".

```

const setup = () => {

    let parag = document.getElementsByClassName("para");
    for (let i = 0; i < parag.length; i++) {
        parag[i].addEventListener("click", klikopparagraf);
    }
}

const klikopparagraf = (event) =>{
    let gekliktParagraaf = event.target;
    console.log("klik op paragraaf met tekst" + gekliktParagraaf.textContent);
    //gekliktParagraaf.style.color="green";
    // in css is made a property belangrijk with color red and u use classList to
    change the color
    //of paragraph to red or u use above with style.color best practice is classL
    ist
    gekliktParagraaf.classList.add("belangrijk");
}
//
window.addEventListener("load", setup);

```

Bijvoorbeeld

```
<p id="txtOutput" class="abc ">blablabla</p>
```

...

```
let txtOutput = document.getElementById("txtOutput");
```

```
console.log( txtOutput.className ); // toont "abc"
```

```
txtOutput.className = "invalid"; // 1 class instellen
```

```
txtOutput.className = ""; // alle classes verwijderen
```

Gelukkig bestaat er ook een handige .classList property waarmee dit eenvoudiger kan :

```
txtOutput.classList.add("invalid");
```

```
txtOutput.classList.remove("invalid");
```

```
if ( txtOutput.classList.contains("invalid") ) { ... }
```

Dit vermijdt het string knip- en plakwerk.

Meer uitleg over de mogelijkheden van .classList vind je op

<https://developer.mozilla.org/en/docs/Web/API/Element/classList>

paragraaf achtergrond van rood naar groen

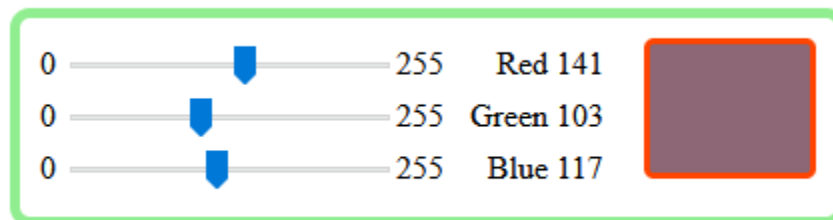
```
const setup = () => {
  // Vraag de DOM-tree om het element met id 'myDiv'
  let myDiv = document.getElementById("myDiv");

  // Zorg ervoor dat de functie printKlik wordt opgeroepen als er op de div wo
  rdt geklikt
  myDiv.addEventListener("click", maakRood);
}

const maakRood = () => {
  // Vraag de DOM-tree om het element met id 'myDiv'
  let myDiv = document.getElementById("myDiv");

  // Geef de div een rode achtergrondkleur
  myDiv.style.backgroundColor="green";
}
//
window.addEventListener("load", setup);
```

COLORPICKER



```
<!DOCTYPE html>
<html>
<head>
  <title>Oefening</title>
  <link rel="stylesheet" type="text/css" href="styles/index.css">
  <script type="text/javascript" charset="utf-
8" src="scripts/index.js"></script>
</head>

<body>
<div class="colorpicker">
  <div class="components">
    <div class="component">
```

```

        <span>0 <input type="range" class="slider" id="sldRed" value="128" min="0" max="255"/>255</span>
        <span>Red</span>
        <span id="lblRood">0</span>
    </div>
    <div class="component">
        <span>0 <input type="range" class="slider" id="sldGroen" value="255" min="0" max="255"/>255</span>
        <span>Green</span>
        <span id="lblGroen">0</span>
    </div>
    <div class="component">
        <span>0 <input type="range" class="slider" id="sldBlauw" value="128" min="0" max="255"/>255</span>
        <span>Blue</span>
        <span id="lblBlauw">0</span>
    </div>

</div>
<div class="swatch" id="swatch"></div>
</div>
</body>
</html>

```

## CSS

```

.slider{
    vertical-align:middle;
}
.component span:nth-of-type(2){
    display:inline-block;
    width:3em;
    text-align:right;
}
.swatch{
    display:inline-block;
    border:3px solid orangered;
    border-radius: 5px;
    margin:0 0 0 1em;
    height:4em;
    width:5em;
}
.components{

```



```

    display:inline-block;
    height:100%;
}
.colorpicker{
    display:inline-block;
    border:5px solid lightgreen;
    border-radius: 10px;
    padding:10px;
}

```

```

const setup = () => {

    let sliders = document.getElementsByClassName("slider");
    for(let i = 0 ; i < sliders.length ; i++)
    {
        sliders[i].addEventListener("change", update);
        sliders[i].addEventListener("input", update);
    }
    update();
}

const update =()=>{
    let red = document.getElementById("sldRed").value;
    let green = document.getElementById("sldGroen").value;
    let blue = document.getElementById("sldBlauw").value;
    document.getElementById("lblRood").innerHTML = red;
    document.getElementById("lblGroen").innerHTML = green;
    document.getElementById("lblBlauw").innerHTML = blue;

    let swatch=document.querySelector("#swatch");
    swatch.style.backgroundColor='rgb('+red+', '+green+', '+blue+')';
}

window.addEventListener("load", setup);

```

Om te testen of een bepaalde number waarde al dan niet NaN is, kun je Number.isNaN(...) gebruiken :

```

let getal = Number.parseInt( txtInput.value, 10 );

if ( Number.isNaN(getal) ) {

    // het is geen getal, doe iets anders

} else {

```

```
// het is wel een getal, bereken iets met 'getal'  
}
```

Als je de tekstvorm wil bekomen van een number kun je diens `.toString()` method gebruiken,

```
let aantal = 10;
```

```
let aantalAlsTekst = aantal.toString(); // aantalAlsTekst is nu "10"
```

maar als je die tekst wil samenplakken met andere tekst kan het eenvoudiger met '+' voor Strings :

```
let tekst = "U kocht " + aantal + " producten";
```

of met een template literal (let op de speciale symbolen rond de tekst, dit zijn backticks of back quotes)

```
let tekst = `U kocht ${aantal} producten`;
```

Om getallen (visueel) te beperken tot een bepaald aantal cijfers na de komma, kun je de

`Number.toFixed(n)` method gebruiken. Deze zal een string produceren van het cijfer met slechts 'n' cijfers na de komma (afgerond).

Bijvoorbeeld :

```
let getal=1234.56789
```

```
console.log( getal.toFixed(2) ); // toont "1234.57" (afronding!)
```

Merk op dat we bij de `console.log` parameter geen `.toString()` moeten doen, de log method kan dat zelf.

### Random getallen

Je kan een willekeurig getal bekomen met de volgende method

```
let getal = Math.random();
```

De variabele 'getal' zal een kommagetal bevatten in  $[0,1[$

Indien je een getal in  $[0,15[$  wil, doe je

```
let getal = Math.random() * 15;
```

Indien je een getal in  $[10,25[$  wil, doe je

```
let getal = 10 + Math.random() * 15;
```

Gehele getallen bekomen

Indien je een geheel getal wil bekomen op basis van een kommagetal (dus zonder fractie), kun je de volgende Math methods gebruiken :

```
Math.floor( kommaGetal )
```

dit zal naar onderen afronden

bv. `Math.floor( 7.67 )` geeft 7

bv. `Math.floor( -7.34 )` geeft -8!

`Math.ceil( kommaGetal )`

dit zal naar boven afronden

bv. `Math.ceil( 7.34 )` geeft 8

bv. `Math.ceil( -7.67 )` geeft -7!

`Math.round( kommaGetal )`

dit zal afronden

bv. `Math.round(7.34)` geeft 7

bv. `Math.round(7.67)` geeft 8

bv. `Math.round(7.5)` geeft 8

bv. `Math.round(-7.5)` geeft 7!

Let goed op met waarden waarvan je niet op voorhand weet of ze positief dan wel negatief zijn!7

Vraag : Waarom is het geen goed idee om het rollen van een dobbelsteen te simuleren met

`Math.round( Math.random()*5 ) + 1`

Als je het niet meteen ziet, probeer dan eens het onderstaande programma uit. Dit telt hoe vaak elk aantal ogen voorkomt als we de virtuele dobbelsteen 600 000 keer laten rollen.

`let aantalKeer = [0,0,0,0,0,0];`

`// genereer random getallen van 1 t.e.m. 6 en tel hoe vaak elk voorkomt`

`for (let i=0 ; i<600000 ; i++) {`

`let getal = Math.round( Math.random() * 5 ) + 1;`

`aantalKeer[ getal-1 ]++; // tel er voor dit getal eentje bij`

`}`

`// toon het resultaat van de telling`

`for (let getal=1 ; getal<=6 ; getal++) {`

`console.log( `getal ${ getal } kwam ${ aantalKeren[getal-1] } keer voor` );`

`}`

Merk op dat het aantal keer dat getal i voorkwam, op positie i-1 staat in het 'aantalKeer' array. Dit komt

omdat we in ons array de posities tellen vanaf 0 maar het aantal ogen tellen vanaf 1.

producten	prijs	aantal	btw	subtotaal
product1	10.00 Eur	<input type="text" value="6"/>	6%	63.60 Eur
product2	15.00 Eur	<input type="text" value="0"/>	21%	0.00 Eur
product3	12.20 Eur	<input type="text" value="0"/>	21%	0.00 Eur
totaal				63.60 Eur

Herbereken

```
const setup = () => {
  let btnHerbereken = document.getElementsByClassName('btnHerbereken')[0];
  btnHerbereken.addEventListener("click", update);

  // vul de UI een eerste keer in
  update();
};

const update = () => {

  let txtBtws = document.getElementsByClassName('btw');
  let txtAantallen = document.getElementsByClassName('aantal');
  let txtPrijzen = document.getElementsByClassName('prijs');
  let txtSubtotalen = document.getElementsByClassName('subtotaal');
  let totaal = 0;

  // Gelukkig staan de elementen van eenzelfde rij, netjes in de dezelfde
  // volgorde in de lijsten txtBtws, txtAantallen, txtPrijzen en txtSubtotalen
  // Zie https://dom.spec.whatwg.org/#old-style-collections
  // We kunnen ze dus gewoon op indexbasis verwerken!
  for (let i = 0; i < txtBtws.length; i++) {
    // parseFloat en parseInt lezen tot aan niet-relevante karakters,
    // dus de ' Eur' en '%' op het einde van prijs en btw values
    // zijn geen probleem.
    let aantal=parseFloat(txtAantallen[i].value, 10);
    let btw=parseFloat(txtBtws[i].textContent, 10); // let op : niet .value!
    let prijs=parseFloat(txtPrijzen[i].textContent, 10); // let op : niet .value!

    let subtotaal=aantal*prijs*(1+(btw/100));
    totaal+=subtotaal;

    // we gebruiken de Number.toFixed(n) method om het gewenste
```

```

        // aantal cijfers na de komma weer te geven (wordt afgerond)
        txtSubtotalen[i].textContent=subtotaal.toFixed(2)+" Eur";

        // merk op dat deze oplossing een groot probleem heeft :
        // de subtotalen worden niet afgerond alvorens ze bij het totaal
        // te tellen, hierdoor kan het zijn dat de optelling in de UI
        // niet klopt.
        // Bv. indien de subtotalen 10.006 en 5.007 zijn is er een verschil :
        // Dan is het totaal nml. 15.01
        // 10.006 + 5.007 = 15.013 (afgerond 15.01)
        // maar als je de afgeronde subtotalen in de UI bekijkt en optelt is het
15.02
        // 10.01 + 5.01
        // eigenlijk moeten we de subtotalen eerst afronden alvorens op te tellen
        .
        //
        // Neem bv. voor prijs product1 en product2 beide 1.00 en 1.02 voor hun a
antallen
        // dan krijg je subtotalen 1.08 en 1.23
        //
        // Netjes afronden op n plaatsen na de komma is in javascript trouwens
        // minder eenvoudig dan je zou verwachten, zie
        // http://stackoverflow.com/questions/11832914/round-to-at-most-2-
decimal-places-in-javascript
    }

    let txtTotaal=document.getElementsByClassName('totaal')[0];
    txtTotaal.textContent=totaal.toFixed(2)+" Eur";
};

window.addEventListener("load", setup);

```

Om teksten aaneen te plakken kun je ofwel de .concat() method gebruiken of wel de '+' operator, bv.

```
let s1 = "Hello ";
```

```
let s2 = "world!";
```

```
let s3=s1.concat(s2); // s3 bevat nu "Hello world!"
```

```
let s4=s1+s2; // s4 bevat nu "Hello world!"
```

Doorgaans gebruikt men de '+' operator.

Om de waarde van een variabele middenin een string te stoppen kun je tekst stukjes aaneenplakken

```
let naam = "Jan";
```

```
let leeftijd = 19;
```

```
...
```

```
let tekst = "De persoon "+naam+" is "+leeftijd+" jaar oud";
```

maar vaak is het leesbaarder om een template literal te gebruiken (vgl. met string interpolatie in C#) :

```
let naam = "Jan";
```

```
let leeftijd = 19;
```

```
...
```

```
let tekst = `De persoon ${naam} is ${leeftijd} jaar oud`;
```

Let op : template literals worden afgebakend door backquote (``) symbolen.

Je kunt een zoektekst opzoeken in een string met de volgende twee methods :

- `.indexOf(zoektekst)` // begint zoektocht vooraan de tekst
- `.lastIndexOf(zoektekst)` // begint zoektocht achteraan de tekst

Beide methods geven als resultaat de positie waarop de deeltekst gevonden werd, of -1 indien deze niet gevonden werd.

Bijvoorbeeld, als je je afvraagt of een bepaalde tekst het woordje 'hond' bevat :

```
let tekst = "Man bijt hond met valse tanden";
```

```
if ( tekst.indexOf("hond") != -1 ) {
```

```
// tekst bevat het woordje "hond"
```

```
} else {
```

```
// tekst bevat het woordje "hond" niet
```

```
}
```

Je kan deze methods ook oproepen met een tweede parameter idx :

- `.indexOf(zoektekst, idx)` // begint zoektocht vooraan de tekst
- `.lastIndexOf(zoektekst, idx)` // begint zoektocht achteraan de tekst

Om deelteksten uit een tekst te kunnen gebruiken, bestaan er maar liefst drie verschillende methods :

- `.substring(idx1, idx2)`
- `.substr(idx, length)`
- `.slice(idx1, idx2)` // dit is de betere keuze<sup>13</sup>

De methods `.slice()` en `.substring()` werken min of meer op dezelfde manier, `.substr()` werkt echter iets anders. Zie onderstaande link voor een bespreking :

```
let s1="Hello world";
```

```
console.log( s1.slice(3, 8) ); // dit toont lo wo
```

- `\n` newline
- `\'` single quote
- `\"` double quote
- `\\` backslash
- `\u89ab` voor unicode karakter 89ab14

Yanic Inghelbrecht versie 2021-03-11 10:49

Om de hoofdletter (of kleine letter) versie van een string te verkrijgen kun je de volgende twee methods gebruiken :

- `.toUpperCase()`
- `.toLowerCase()`

```
let tekst=" Jan Janssens ";
```

```
let proper=tekst.trim(); // bevat "Jan Janssens"
```

Tafel van...

Tafel van 2	Tafel van 5	Tafel van 3	Tafel van 4
2 x 1 = 2	5 x 1 = 5	3 x 1 = 3	4 x 1 = 4
2 x 2 = 4	5 x 2 = 10	3 x 2 = 6	4 x 2 = 8
2 x 3 = 6	5 x 3 = 15	3 x 3 = 9	4 x 3 = 12
2 x 4 = 8	5 x 4 = 20	3 x 4 = 12	4 x 4 = 16
2 x 5 = 10	5 x 5 = 25	3 x 5 = 15	4 x 5 = 20
2 x 6 = 12	5 x 6 = 30	3 x 6 = 18	4 x 6 = 24
2 x 7 = 14	5 x 7 = 35	3 x 7 = 21	4 x 7 = 28
2 x 8 = 16	5 x 8 = 40	3 x 8 = 24	4 x 8 = 32
2 x 9 = 18	5 x 9 = 45	3 x 9 = 27	4 x 9 = 36
2 x 10 = 20	5 x 10 = 50	3 x 10 = 30	4 x 10 = 40

## HTML

```
<!DOCTYPE html>
<html>
<head>
  <title>Oefening</title>
  <link rel="stylesheet" type="text/css" href="styles/index.css">
  <script type="text/javascript" charset="utf-
8" src="scripts/index.js"></script>
</head>

<body>
<div>
  <label for="txtGetal">Tafel van...</label>
  <input type="text" name="start" placeholder="Vul getal in" id="txtGetal">
  <input type="button" id="btnGo" value="Go!">
</div>

<div id="divTafels"></div>
</body>
</html>
```

## CSS

```
.tafel {
  margin: 10px;
  display:inline-block;
  border: solid 1px orangered;
}

.tafel p, .tafel h1 {
  margin:0;
  padding:10px;
}

.hoofding {
  font-size: 100%;
  background-color: darkred;
  color: white;
}

.tafel p:nth-of-type(odd) {
  background-color: cornflowerblue;
}

.tafel p:nth-of-type(even) {
```



```
background-color: chartreuse;
}
```

```
const setup = () => {
  // Zorg ervoor dat een klik op de button onze event listener oproept
  let btnGo = document.getElementById("btnGo");
  btnGo.addEventListener("click", addTafel);
  //insertTafelHTML(4);
  //insertTafelHTML(12);
};

const addTafel= () =>{
  // Neem de tekst uit het tekstveld en check of het een getal is
  let txtGetal = document.getElementById("txtGetal");
  let getal = parseInt(txtGetal.value, 10);
  if(isNaN(getal)){
    window.alert("Geen geldig getal");
  } else {
    // het is een getal, dus we kunnen een tafel toevoegen
    insertTafelHTML(getal)
  }
  // maak inputveld leeg
  txtGetal.value = "";
};

const insertTafelHTML = (getal) => {
  // maak de HTML voor de hoofding
  let htmlHoofding=`<h1 class="hoofding">Tafel van ${getal}</h1>`;
  // maak de HTML voor de inhoud
  let htmlInhoud="";
  for (let i=1;i<=10;i++) {
    let product = i*getal;
    let htmlRegel=`<p>${getal} x ${i} = ${product}</p>`;
    htmlInhoud+=htmlRegel;
  }
  // maak een <div> met class tafel die de hoofding en inhoud bevat
  let htmlTafel=`<div class="tafel">${htmlHoofding}${htmlInhoud}</div>`;
  // voeg deze html toe aan het div element voor de tafels
  let divTafels = document.getElementById("divTafels");
  divTafels.innerHTML+=htmlTafel;
};

window.addEventListener("load", setup);
```

Elementen uit de DOM-tree opvragen (bis)

We zagen in 'Javascript deel 03' hoe we bepaalde elementen uit de DOM-tree kunnen te pakken krijgen,

bv. om hun properties te wijzigen. We deden dit met enkele get-methods van het document object :

```
document.getElementById(id)
```

```
document.getElementsByClassName(className)
```

```
document.getElementsByTagName(tagName)
```

Het document object voorziet echter nog een paar andere handige methods om dit te doen.

Twee zeer belangrijke methods zijn querySelector en querySelectorAll

```
document.querySelector(selector)
```

```
document.querySelectorAll(selector)
```

De 'selector' parameter is een string die een CSS-selector bevat, bv. #playfield of .important>img

```
document.querySelector("#playfield")
```

```
document.querySelectorAll(".important>img")
```

Met name

- id selector
- class selector
- descendant combinator
- child combinator
- attribute selector
- pseudo-class selector (zoals :checked en :nth-of-type)

```
document.getElementsByName(naam)
```

Bijvoorbeeld, om een inputveld met name="firstname" in een form met id="frmContact" te selecteren :

```
document.querySelector("#frmContact [name='firstName']")
```

(let op de spatie in die selector!)

```

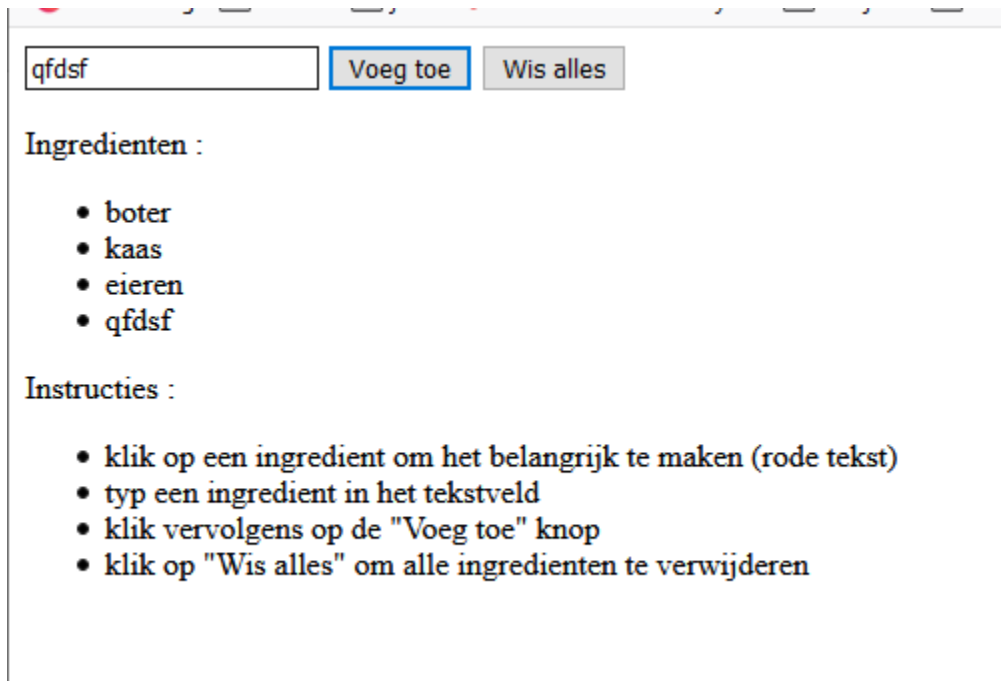
let frmContact=document.getElementById("frmContact");
...
let errorMessages=frmContact.getElementsByClassName("errorMessage");
for (let i=0;i<errorMessages.length;i++) {
    errorMessages[i].textContent=""; // maak de span leeg zodat de foutmelding
    verdwijnt
}

let errorMessages=document.querySelectorAll("#frmContact .errorMessage");
for (let i=0;i<errorMessages.length;i++) {
    errorMessages[i].textContent="";
}

```

maar soms is het wel handiger en/of efficiënter om te beginnen bij een DOM-tree element dat je al

hebt.



qfdsf Voeg toe Wis alles

**Ingredienten :**

- boter
- kaas
- eieren
- qfdsf

**Instructies :**

- klik op een ingredient om het belangrijk te maken (rode tekst)
- typ een ingredient in het tekstveld
- klik vervolgens op de "Voeg toe" knop
- klik op "Wis alles" om alle ingredienten te verwijderen

```

<!DOCTYPE html>
<html>
<head>
    <title>Ingredienten</title>
    <link rel="stylesheet" type="text/css" href="styles/index.css">

```

```

    <script type="text/javascript" charset="utf-
8" src="scripts/index.js"></script>
</head>

<body>
    <input id="txtInput" type="text" placeholder="Type hier een ingredient"/>
    <input id="btnAdd" type="button" value="Voeg toe" />
    <input id="btnClear" type="button" value="Wis alles" />
    <p>Ingredienten :</p>
    <ul id="lstIngredients">
        <li>boter</li>
        <li>kaas</li>
        <li>eieren</li>
    </ul>
    <p>Instructies :</p>
    <ul>
        <li>klik op een ingredient om het belangrijk te maken (rode tekst)</li>
        <li>typ een ingredient in het tekstveld</li>
        <li>klik vervolgens op de "Voeg toe" knop</li>
        <li>klik op "Wis alles" om alle ingredienten te verwijderen</li>
    </ul>
</body>
</html>

```

```

const setup = () => {
    // registreer click event listener 'voegToe' bij #btnAdd
    let btnAdd = document.querySelector('#btnAdd');
    btnAdd.addEventListener("click", voegToe);
    // registreer click event listener 'wisAlles' bij #btnClear
    let btnClear = document.querySelector("#btnClear");
    btnClear.addEventListener("click", wisAlles);
    // registreer click event listener 'maakBelangrijk' bij elke <li> in .lstIngredients
    let lis = document.querySelectorAll("#lstIngredients");
    for (let i = 0; i < lis.length; i++){
        lis[i].addEventListener("click", maakBelangrijk);
    }
}

const voegToe = () => {
    // Lees de tekst uit het tekstveld en voeg nieuw <li> element toe
    let txtInput = document.querySelector("#txtInput");
    let ingredient = txtInput.value;

```

```

    let lstIngredients = document.querySelector("#lstIngredients");
    lstIngredients.innerHTML += `<li>${ingredient}</li>`
    // Probleem A
    // vermits we hier dmv .innerHTML+= eigenlijk alle kinderen van het
    // <ul> element vervangen door kopies (zonder eventlisteners), zal een klik
    // op een bestaand kind geen effect meer hebben.

    // Probleem B
    // vermits we geen event listener koppelen aan het nieuwe kind, zal dit
    // niet reageren op een klik
}

const wisAlles = () => {
    // Wis alle ingredienten
    // Je kunt dit doen door alle de .innerHTML van #lstIngredients een lege stri
ng in te stellen
    let lstIngredients = document.querySelector("#lstIngredients");
    lstIngredients.innerHTML="";
}

const maakBelangrijk = (event) => {
    // Geef het geklikte element de CSS class 'belangrijk'
    let li = event.target;
    li.classList.add("belangrijk");
}

window.addEventListener('load',setup);

```

`firstList.innerHTML += "<li>nieuw item</li>"; // achteraan 'toevoegen'`

`firstList.innerHTML = "<li>nieuw item</li>"+firstList.innerHTML; // vooraan 'toevoegen'`

Om een een nieuw child element toe te voegen kun je beter `.insertAdjacentHTML` gebruiken (zie

`verderop`) i.p.v. `.innerHTML += "..."` te schrijven

BEFOREEND AFTEREND

Alle DOM-tree elementen ondersteunen een method [insertAdjacentHTML](#) waarmee makkelijk content kan toegevoegd worden op welbepaalde plaatsen.

- `element.insertAdjacentHTML(pos, html)`
  - analyseert de html tekst en voegt de DOM-elementen toe relatief t.o.v. het element
  - de 'html' parameter is een string met HTML-tekst
  - de 'pos' parameter geeft aan waar de nieuwe elementen moeten toegevoegd worden
    - `beforebegin`, `afterbegin`, `beforeend` of `afterend`
- indien het element een `<ul>` is met drie `<li>` kinderen, is de positie als volgt :

```
beforebegin
<ul>
  afterbegin
  <li>...</li>
  <li>...</li>
  <li>...</li>
  beforeend
</ul>
afterend
```

Voorbeeld :

```
let firstList = document.getElementsByTagName("ul")[0];
firstList.insertAdjacentHTML("beforeend", "<li>een nieuw item</li>");
```

deze code gebruikt positie "beforeend" om het nieuwe item achteraan de lijst toe te voegen (i.e. als nieuwe laatste `<li>` kind)

Ingrediënten :

- boter
- kaas
- eieren
- qfdsf

Instructies :

- klik op een ingrediënt om het belangrijk te maken (rode tekst)
- typ een ingrediënt in het tekstveld
- klik vervolgens op de "Voeg toe" knop
- klik op "Wis alles" om alle ingrediënten te verwijderen

```

const setup = () => {
  // registreer click event listener 'voegToe' bij #btnAdd
  let btnAdd = document.querySelector('#btnAdd');
  btnAdd.addEventListener("click", voegToe);
  // registreer click event listener 'wisAlles' bij #btnClear
  let btnClear = document.querySelector("#btnClear");
  btnClear.addEventListener("click", wisAlles);
  // registreer click event listener 'maakBelangrijk' bij elke <li> in .lstIngredients
  let lis = document.querySelectorAll("#lstIngredients li");
  for (let i = 0; i < lis.length; i++){
    lis[i].addEventListener("click", maakBelangrijk);
  }
}

const voegToe = () => {
  // Lees de tekst uit het textveld en voeg nieuw <li> element toe
  let txtInput = document.querySelector("#txtInput");
  let ingredient = txtInput.value;
  let lstIngredients = document.querySelector("#lstIngredients");
  lstIngredients.insertAdjacentHTML("beforeend", `<li>${ingredient}</li>`);
  let items = lstIngredients.getElementsByTagName("li");
  let newItem = items[items.length-1];
  newItem.addEventListener("click", maakBelangrijk);
}

const wisAlles = () => {
  // Wis alle ingredienten
  // Je kunt dit doen door alle de .innerHTML van #lstIngredients een lege string in te stellen
  let lstIngredients = document.querySelector("#lstIngredients");
  lstIngredients.innerHTML="";
}

const maakBelangrijk = (event) => {
  // Geef het geklikte element de CSS class 'belangrijk'
  let li = event.target;
  li.classList.add("belangrijk");
}

window.addEventListener('load', setup);

```

De attributen van een HTML-element kun je opvragen en aanpassen met de volgende methods

- `element.getAttribute(a)`

- o retourneert de waarde van attribuut 'a' van het element

- o de 'a' parameter is een string met de naam van het attribuut

- bv. "href", "src", etc.

- o bv. `element.getAttribute("href")`

- `element.setAttribute(a, v)`

- o stelt de waarde van attribuut 'a' in op waarde 'v', voor dit element

- o de 'a' parameter is een string met de naam van het attribuut

- bv. "href", "src", etc.

- o de 'v' parameter is een string met de waarde van het attribuut

- o bv. `element.setAttribute("href", "http://www.example.com");`

- `element.hasAttribute(a)`

- o geeft aan of het element het 'a' attribuut al dan niet heeft

- o te gebruiken indien je niet op voorhand weet of het element het attribuut heeft

- bij sommige browser(versies) geeft `getAttribute` bij afwezigheid namelijk een lege String terug ipv de null waarde

Bijvoorbeeld

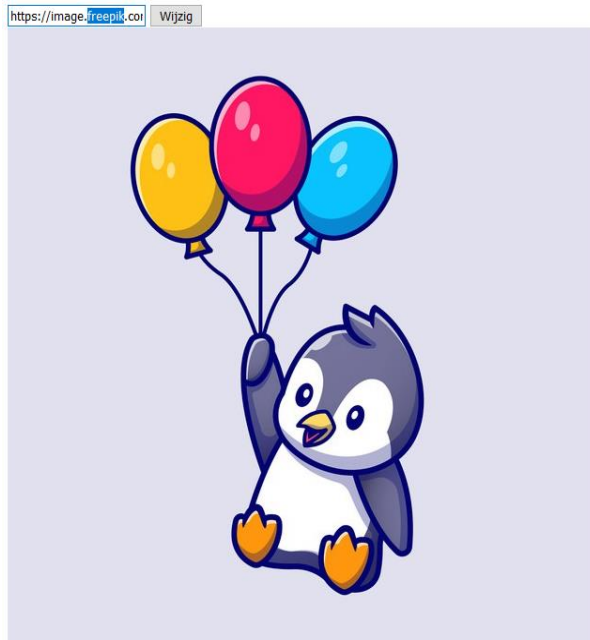
```
let firstImage = document.getElementsByTagName("img")[0];
```

```
let oldSource = firstImage.getAttribute("src");
```

```
firstImage.setAttribute("src", "images/logo.png")
```

WIJZIG





```
const setup = () => {
  // deze code wordt pas uitgevoerd als de pagina volledig is ingeladen
  //kopel de wijzig click event listener aan de button
  let btnWijzig = document.getElementById("btnWijzig");
  btnWijzig.addEventListener("click", wijzig);

  //vul het invulveld met de url van de afbeelding
  //zoek de waarde op van het src attribute van afbeelding
  let image = document.getElementById("imgOutput");
  let url = image.getAttribute("src");
  //Kopieer de url naar het input veld
  let txtInput = document.getElementById("txtInput");
  txtInput.value=url;
}
const wijzig=()=>{
  //verander hte src attribut van de afbeelding naar wat de gebruiker invoerde
  //lees de url in het inputveld
  let txtInput = document.getElementById("txtInput");
  let url = txtInput.value;
  //Stel het src attribuut in van de afbeelding
  let image = document.getElementById("imgOutput");
  image.setAttribute("src", url);
}
window.addEventListener("load", setup);
```

Voorbeeld :

```
element.addEventListener("click", klik);
```

...

```
const klik = (event) => {
```

```
  console.log("u klikte op een "+event.target.name+" element);
```

```
  console.log("de listener is geregistreerd bij een  
"+event.currentTarget.name+" element);
```

Merk op dat je in een click event listener, met

```
if (event.target === event.currentTarget) {
```

```
  // rechtstreeks geklikt op het element met de gekoppelde event  
  listener
```

```
} else {
```

```
  // geklikt op een descendant van het element met de gekoppelde event  
  listener
```

```
}
```

kunt achterhalen of er rechtstreeks op het element werd geklikt dan wel op een van z'n descendants!

```
const setup = () => {  
  // registreer click event listener 'voegToe' bij #btnAdd  
  let btnAdd = document.querySelector('#btnAdd');  
  btnAdd.addEventListener("click", voegToe);  
  // registreer click event listener 'wisAlles' bij #btnClear  
  let btnClear = document.querySelector("#btnClear");  
  btnClear.addEventListener("click", wisAlles);  
  
  // HIERONDER STAAT CODE DIE GEWIJZIGD IS TOV 'oplossing opdracht 02'  
  // registreer click event listener 'maakBelangrijk' op het niveau van het <ul  
> element  
  // Wegens event bubbling komt elk klik event op een <li> sowieso ook bij de c  
lick event  
  // listener van het <ul> element terecht!  
  let lstIngredients = document.querySelector("#lstIngredients");  
  lstIngredients.addEventListener("click", maakBelangrijk);  
  
  // Dit is een aanzienlijk vereenvoudiging, er is maar 1 event listener koppel  
ing nodig
```

```

    // ipv allemaal afzonderlijke op elk <li> element. Bovendien werkt deze zowel
    met kliks
    // op <li>s uit de HTML als met deze die we via javascript toevoegden.
    // Strikt genomen zou deze oplossing ook werken met .innerHTML+=", maar we geve
    en toch de
    // voorkeur aan .insertAdjacentHTML().
}

const voegToe = () => {
    // Lees de tekst uit het textveld en voeg nieuw <li> element toe
    let txtInput = document.querySelector("#txtInput");
    let ingredient = txtInput.value;
    let lstIngredients = document.querySelector("#lstIngredients");
    lstIngredients.innerHTML += `<li>${ingredient}</li>`
    // HIERONDER STAAT DE CODE DIE GEWIJZIGD IS TOV 'oplossing opdracht 02'
    // de code voor het koppelen vd event listener op het nieuwe <li> element is
    verdwenen
}

const wisAlles = () => {
    // Wis alle ingredienten
    // Je kunt dit doen door alle de .innerHTML van #lstIngredients een lege stri
    ng in te stellen
    let lstIngredients = document.querySelector("#lstIngredients");
    lstIngredients.innerHTML="";
}

const maakBelangrijk = (event) => {
    // HIERONDER STAAT DE CODE DIE GEWIJZIGD IS TOV 'oplossing opdracht 02'

    // check of er op een <li> element dan wel rechtstreeks op het <ul> element wer
    d geklikt
    if(event.currentTarget !== event.target){
        // er is niet direct op het <ul> element geklikt, het zal dus op een <li>
        geweest zijn.
        // geef het geklikte element de CSS class 'belangrijk'
        let li = event.target;
        li.classList.add("belangrijk");
    }else{
        // Indien target === currentTarget hebben we rechtstreeks op het <ul> ele
        ment geklikt
        // en niet op een <li> element.
    }
}

```

```
}  
  
window.addEventListener('load',setup);
```



*Spin*



**Proficiat!**

*Spin*



**Zolang u niet stopt bent u geen verliezer!**

Slotmachine

HTML

```
<!DOCTYPE html>  
<html>  
<head>  
  <title>Slot Machine</title>  
  <link rel="stylesheet" type="text/css" href="styles/index.css">
```

```

    <script type="text/javascript" charset="utf-
8" src="scripts/index.js"></script>
</head>

<body>
<div class="controls">
    <!--
- een hyperlink die als een button wordt gestyled, let ook op de href value -->
    <a href="#" class="buttonLink">Spin</a>
</div>
<div class="slots">
    <!--
    we plaatsen de drie img element in de HTML, er is geen enkele reden
    om de extra moeite te doen om ze via Javascript toe te voegen.

    We moeten dan via JS enkel nog het 'src' attribuut veranderen als
    de gebruiker op 'Spin' klikt.
    -->
    
    
    
</div>
<!--
Beide berichten zitten al in de HTML, dat heeft het voordeel dat de teksten
niet in de JS code als literal moeten staan maar gewoon in de HTML
Maar het is natuurlijk nooit de bedoeling dat ze tegelijk getoond worden!
-->
<p class="message winner">Proficiat!</p>
<p class="message loser">Zolang u niet stopt bent u geen verliezer!</p>
</body>
</html>

```

## CSS

```

.controls {
    text-align:center;
}

.slots {
    text-align:center;
}

.slots img {
    height: 100px;
    padding: 10px;
}

```

```

}

.message.show {
    display:block; /* enkel berichten die de .show class hebben worden getoond, d
e andere worden verborgen */
}

.message {
    text-align:center;
    font-family:Arial;
    font-size:28px;
    font-weight:bold;
    margin:0;

    display:none; /* verberg alles berichten via CSS, we zullen ze later via Java
script tonen/verbergen via de .show class, zie hierboven */
}

/* button opmaak gegenereerd met https://www.bestcssbuttongenerator.com/ */
.buttonLink {
    box-shadow: 0px 0px 0px 0px #3dc21b;
    background-color:#44c767;
    border-radius:28px;
    border:1px solid #18ab29;
    display:inline-block;
    cursor:pointer;
    color:#ffffff;
    font-family:Arial;
    font-size:28px;
    font-weight:bold;
    font-style:italic;
    padding:16px 31px;
    text-decoration:none;
    text-shadow:0px 1px 0px #2f6627;
}
.buttonLink:hover {
    background-color:#5cbf2a;
}
.buttonLink:active {
    position:relative;
    top:1px;
}
/* einde button opmaak */

```

JS

```
// een globale variabele met de urls van alle afbeeldingen
const urls = ["images/fruit01.jpg", "images/fruit02.jpg", "images/fruit03.jpg", "images/fruit04.jpg", "images/fruit05.jpg"];

const setup = () => {
  // registreer de 'spin' functie als click event listener van de button (die eigenlijk een hyperlink is!)
  let lnkSpin = document.querySelector(".buttonLink");
  lnkSpin.addEventListener("click", spin);

  // Toon initieel al willekeurige prentjes (in de HTML staat immers 3x src="fruit01.jpg").
  randomizeImages();
}

const spin = () => {
  // Telkens de gebruiker spin klikt, tonen we willekeurige afbeeldingen
  // en controleren of er gewonnen werd. Door dit zo over 2 functions te
  // verdelen (ipv alle code bv. hieronder te schrijven), kunnen we
  // randomizeImages() nu ook gebruiken in setup().
  randomizeImages();
  checkWinnings();
}

const randomizeImages = () => {
  // haal alle relevante <img> elementen op uit de DOM-tree
  // (let op de spatie in de selector, die op descendants (afstammelingen) duidt)
  let imageElements = document.querySelectorAll(".slots img"); // alle img elementen die afstamming zijn van een element met class 'slots'
  for (let i = 0; i < imageElements.length; i++) {
    // bepaal een random index voor het globale 'urls' array
    let imageElement = imageElements[i];
    let randomIndex = Math.floor(Math.random() * urls.length);

    // verander de afbeelding
    setImageFromIndex(imageElement, randomIndex);
  }
}
```

```

}

const setImageFromIndex = (imageElement, index) => {
  // verander de afbeelding op basis van de index in het 'urls' array
  let url = urls[index];
  // wijzig het 'src' attribuut van het <img> element naar de url uit het array
  imageElement.setAttribute("src", url);
}

const checkWinnings = () => {
  // Er is een winnaar als alle drie <img> elementen dezelfde waarde voor hun 'src' attribuut hebben
  let imageElements = document.querySelectorAll(".slots img");

  // we nemen de url van de eerste afbeelding
  let firstUrl = imageElements[0].getAttribute("src");

  // we vergelijken alle andere afbeeldingen met de url van de eerste afbeelding todat we
  // ofwel een verschil opmerken (loser)
  // ofwel alle afbeeldingen gecheckt hebben zonder verschil te vinden (winner)
  let isWinner = true;
  for (let i = 1; i < imageElements.length; i++) {
    let imageElement = imageElements[i];
    let url = imageElement.getAttribute("src");
    if (url !== firstUrl) {
      // niet alle urls zijn gelijk, dus geen winnaar
      isWinner = false;
      break;
    }
  }

  // toon de juiste message en verberg de andere message
  let messages = document.querySelectorAll(".message");
  for (let i=0;i<messages.length;i++) {
    let message = messages[i];
    // we verwijderen sowieso eerst de 'show' class (daardoor wordt het bericht onzichtbaar)
    message.classList.remove("show");
    // indien winner, show winner zoniet show loser
    if (isWinner === message.classList.contains("winner")) { // als er een winnaar is en dit bericht heeft class 'winner'
      message.classList.add("show"); // dan toon dit bericht (alle andere berichten waren al verborgen en blijven verborgen)
    }
  }
}

```



```
}  
}  
  
// roep onze 'setup' functie op eenmaal de DOM-tree klaar is  
window.addEventListener("load", setup);
```

## DEEL07

Bijvoorbeeld, een array van getallen sorteren

```
const compare = (a,b) => {  
  return a-b;  
}  
  
let array=[34, 67, 12, 5, 23];
```

```
array.sort(compare);
```

Bijvoorbeeld, een array van strings sorteren

```
const compare = (a, b) => {  
  return a.localeCompare(b);  
}  
  
let array=["zebra", "aap",
```

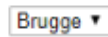
## GemeenteOefening



Geef een gemeente in (of 'stop' om te stoppen)

OK Annuleren

Bv. als de gebruiker achtereenvolgens Gent, Brugge en Kortrijk invoert verschijnt er :



Door op het driehoekje te klikken, klapt het <select> element open en zie je de gemeenten in alfabetische volgorde :



Dit openklappen is standaard gedrag van een <select> element, je hoeft dit dus niet zelf te programmeren!

```
const setup = () => {
  let gemeenten = [];

  let stoppen=false;
  while(!stoppen) {
    // vraag de gebruiker om een tekst
    let input = window.prompt("Geef een gemeente in (of 'stop' om te stoppen)");

    // stop loop indien de gebruiker
    // - de tekst 'stop' typt
    // - op cancel drukt (input==null)
    stoppen=(input==null || input.trim().toLowerCase() == "stop");
    if(!stoppen){
      // voeg gemeente toe achteraan het array
      gemeenten.push(input);
    }
  }
  // sorteer het array met gemeenten
  gemeenten.sort(compare);

  // voeg de nodige HTML toe aan de DOM-tree voor de gemeenten
  voegGemeentenToe(gemeenten);
};

const voegGemeentenToe = (gemeenten) => {
  // Zoek het <select> in de DOM-tree
  let selGemeenten = document.querySelector("#selGemeenten");
```

```

    for(let i=0; i<gemeenten.length; i++){
        // per gemeente een <option> in het <select> element
        selGemeenten.insertAdjacentHTML("beforeend", `<option>${gemeenten[i]}</option>`);
    }
};
const compare = (a,b) => {
    // sorteer alfabetisch (a < z)
    return a.localeCompare(b);
    // om omgekeerd alfabetisch te sorteren (z < a),
    // moet het teken omgekeerd worden (let op het minteken) :
    // return -a.localeCompare(b);
};
window.addEventListener("load", setup);

```

Waarden in een form opvragen

In je HTML cursus heb je wellicht gezien dat er allerlei <input> varianten bestaan die je kunt gebruiken in een invulformulier.

Om de waarde van een input element op te vragen, kunnen we de .value property opvragen van het

corresponderende DOM-tree element. Let er wel op dat dit steeds een string zal zijn, ook al heeft het

<input> element bv. een type="number" attribuut (je zult dit dus nog moeten parsen)!

Bijvoorbeeld

In de HTML code

```
<input type="number" id="txtAmount">
```

In de Javascript code

```
let txtAmount = document.querySelector("#txtAmount");
```

```
let amountAsText = txtAmount.value;
```

```
let amount = Number.parseInt(amountAsText, 10);
```

Voor een checkbox (<input type="checkbox">) kun je de .checked property opvragen. Dit levert een

boolean waarde (true/false) op, al naargelang of de checkbox is aangevinkt.

Bijvoorbeeld

In de HTML code

- `<input type="checkbox" id="chkPriority">`

In de javascript code

```
let chkPriority = querySelector("#chkPriority");
console.log( chkPriority.checked );
```

Bijvoorbeeld

In de HTML code

```
<input type="radio" name="colors" value="red" checked>
<input type="radio" name="colors" value="green">
<input type="radio" name="colors" value="blue">
```

In de javascript code

```
var checkedRadioButton =
document.querySelector("input[name='colors']:checked")
```

De geselecteerde options in een `<select>` element opvragen is wat ingewikkelder. Een DOM-node voor

een `<select>` element heeft volgende relevante properties :

- `.options`

een verzameling DOM-tree elements met de `<option>` elementen uit het `<select>` element. Deze

kan op dezelfde manier als een array gebruikt worden (maar het is strik genomen een `NodeList`).

- `.selectedIndex`

dit is de index (in bovenstaande `.options` lijst) van de eerste geselecteerde option, of `-1` als er

geen enkele geselecteerd is

De truuk is dus via de `.selectedIndex` de geselecteerde option opzoeken in de `.options` verzameling.

Het DOM-element van een <option> heeft volgende nuttige properties :

- `.selected`

een boolean die aangeeft of de option geselecteerd is

- `.value`

de waarde van het value attribuut van de option

- `.text`

de tekst van de option (tussen begin- en eindtag)

Indien een <select> meerdere geselecteerde options toelaat, zul je ze allemaal moeten overlopen en de

waarde van hun `.selected` property nagaan!10

Je kan je bij <select> elementen echter heel wat werk besparen door een slimmere CSS-selector te

gebruiken!

Bijvoorbeeld :

In de HTML code

```
<select id="selfruit" size="4" multiple>
  <option>appel</option>
  <option>peer</option>
  <option>banaan</option>
  <option>kiwi</option>
</select>
```

In de javascript code

```
let selectedOptions=document.querySelectorAll("#selfruit
option:checked");
```

## Opdracht formwaarden

Schrijf een HTML-pagina die er als volgt uitziet

Is roker ☐

Moedertaal ☐ Nederlands ☒ Frans ☐ Engels

Favoriete buurland

Bestelling

- aardappelen
- brood
- melk
- biefstuk
- chips
- krant

De invoermogelijkheden zijn

- Is roker : een checkbox.
- Moedertaal : een radiobutton groep met keuzes "Nederlands", "Frans" en "Engels" met resp. values "nl", "fr" en "en".
- Favoriete buurland : een enkelvoudige select met keuzes "Nederland", "Frankrijk", "Duitsland".
- Bestelling : een multi-select met keuzes "aardappelen", "brood", "melk", "biefstuk", "chips" en "krant".

Als er op de "Toon resultaat" knop geklikt wordt, verschijnen de uitgelezen waarden op de console.

Bijvoorbeeld, met bovenstaande keuzes zal de console de volgende output tonen :

```
is geen roker
moedertaal is nl
favoriete buurland is Frankrijk
bestelling bestaat uit aardappelen melk biefstuk
```

```
const setup = () => {
  let btnToon = document.querySelector("#btnToon");
  btnToon.addEventListener("click", toon);
};

const toon = () => {
  let chkIsRoker=document.querySelector("#chkIsRoker");
  if (chkIsRoker.checked) {
    console.log("is roker");
  } else {
    console.log("is geen roker");
  }
}
```

```

    // Met een wat complexere CSS-selector wordt onze code heel wat eenvoudiger
    let rbtMoedertaal = document.querySelector("input[name='rbtMoedertaal']:checked");
    console.log("moedertaal is "+rbtMoedertaal.value);

    // Met een wat complexere CSS-selector wordt onze code heel wat eenvoudiger
    // Let op, het is de pseudo-
class :checked en niet :selected (die bestaat niet)
    let optionFavorieteBuurland=document.querySelector("#selFavorieteBuurland option:checked");
    console.log("favoriete buurland is "+optionFavorieteBuurland.text);

    // Met een wat complexere CSS-selector wordt onze code heel wat eenvoudiger
    // Let op de gelijkenis met optionFavorieteBuurland hierboven, de selector is
quasi dezelfde
    // maar we verwachten hier (potentieel) meer dan 1 element
    let text="bestelling bestaat uit ";
    let optionsBestelling = document.querySelectorAll("#selBestelling option:checked");
    for (let i=0;i<optionsBestelling.length;i++) {
        text+=optionsBestelling[i].text+" ";
    }
    console.log(text);
};

window.addEventListener("load", setup);

```

## DOM-tree nodes

Elke node 'n' uit de DOM-tree heeft een aantal properties die te maken hebben met de relatie van 'n'

tot sommige andere nodes in de DOM-tree :

- **n.parentNode**

o geeft de parent van n

o geeft null indien er geen is, bv. omdat n nog geen parent heeft (bv. als we de node nog

maar net hebben aangemaakt, zie verderop) of omdat n een document node is

- **n.childNodes**

- o geeft een lijst die alle children van n bevat
- o dit lijst object is geen array, maar je kan wel `.length` en `[idx]` erop toepassen
- `n.firstChild` / `n.lastChild`
- o geeft het eerste/laatste child van n
- o geeft null indien er geen is
- `n.nextSibling` / `n.previousSibling`
- o geeft de volgende/vorige sibling van n
- o geeft null indien er geen is

Meestal zijn we enkel geïnteresseerd in de DOM-tree nodes die HTML-elementen voorstellen.

Bovenstaande methods geven ons echter alle soorten nodes. We moeten dan ietwat omslachtige code

schrijven om enkel de element nodes eruit te pikken en de overige soorten te negeren.

Gelukkig heeft een node heeft ook enkele methods die dit werk voor ons doen :

- `n.children`
- `n.firstChild` / `n.lastElementChild`
- `n.nextElementSibling` / `n.previousElementSibling`

Deze werken op dezelfde manier als de eerdere methods maar leveren enkel de element nodes op.

Let op `n.firstChild` en `n.lastElementChild`, deze komen vaak van pas bij het gebruik van

`insertAdjacentHTML()` op posities "afterbegin" en "beforeend", als verwijzing naar het nieuwe element!

De DOM-tree kan veel verschillende soorten nodes bevatten, bijvoorbeeld

- element nodes (voor HTML-elementen zoals `<h1>`, `<a>`, `<p>`, etc.)
- text nodes (voor de teksten die in en tussen de elementen staan)
- comment nodes (voor HTML commentaren tussen `<!--` en `-->`)



- ...

Elke node heeft een aantal properties die ons vertellen wat voor soort node het is :

- `n.nodeName`

- o geeft het soort element (als string) indien n een element node is, bv. "IMG", "H1", etc.

- Let op, deze string zal altijd in hoofdletters staan

- o geeft "#text" indien n een text node is

- `n.nodeType`

- o geeft een getal naargelang welk soort node n is

- bv. 1 indien n een element node is

- bv. 3 indien n een text node is

- o zie <https://developer.mozilla.org/en-US/docs/Web/API/Node/nodeType>

- `n.nodeValue` (enkel voor text nodes!)

- o geeft de tekst van de node als string, indien n een text node is

- o geeft null indien n een element node is

De text nodes waarvan hierboven sprake is, zijn extra nodes in de DOM-tree die de teksten bevatten die

tussen de element tags staan.<sup>4</sup>

Voorbeeld

Het HTML fragment

```
<p>
hallo
<span>dit is een</span>
tekst
</p>
```

ziet er in de DOM-tree als volgt uit

P ELEMENT NODE

TEXT NODE met `nodeValue="hallo"` (\*)

SPAN ELEMENT NODE

TEXT NODE met nodeValue="dit is een"

TEXT NODE met nodeValue="tekst" (\*)

(\*) in de nodeValue strings zit ook nog whitespace en newlines, maar da's moeilijk om hier te tonen.

Je ziet dus dat element nodes nooit zelf 'hun' tekst bevatten, deze wordt altijd in een text node verpakt!

Voorbeeld

In de HTML code

```
<div class="gallery">  
  
  
  
  
</div>
```

In de Javascript code

```
let gallery = document.querySelector(".gallery");  
...  
let image = document.createElement("img");  
image.setAttribute("src", "images/e.png");  
gallery.appendChild(image); // image toevoegen na laatste kind  
...  
let someImage=gallery.children[3]; // <img> met src="images/d.png"  
gallery.removeChild(someImage); // d.png wordt verwijderd uit de  
gallery
```

## Opdracht 01

Unzip 'beginsituatie opdracht 01.zip' in een nieuw Webstorm project of folder.

De HTML en CSS code voor deze opdracht is al gegeven :

### Oplossing opdracht 01

Ingredienten :

- boter
- kaas
- eieren

Instructies :

- klik op een ingrediënt om het te verwijderen

Voeg de nodige Javascript code toe zodat een klik op een ingrediënt, dit ingrediënt verwijdert.

```
const setup =() => {
  let lis = document.querySelectorAll("#lstIngredients>li");
  for (let i = 0; i < lis.length; i++) {
    let li = lis[i];
    li.addEventListener("click", verwijder);
  }
}

const verwijder =(event) => {
  let list = event.target;
  let ulist = document.querySelector("#lstIngredients");
  ulist.removeChild(list);
  /*
  let li = event.target;
  // let ul = document.querySelector("#lstIngredients");
  //OR up is queryselector and down is with node selector
  let ul = li.parentNode;
  ul.removeChild(li);
  //console.log(li.textContent);*/
}

window.addEventListener('load', setup )
```

## Opdracht 02

Unzip 'beginsituatie opdracht 02.zip' in een nieuw Webstorm project of folder.

De HTML en CSS code voor deze opdracht is al gegeven :

### Oplossing opdracht 02

Ingrediënten :

- boter [verwijder](#)
- kaas [verwijder](#)
- eieren [verwijder](#)

Instructies :

- klik op een 'verwijder' link om een ingrediënt te verwijderen

Voeg de nodige Javascript code toe zodat een klik op een 'verwijder' link, het corresponderende ingrediënt verwijdert (incl. de 'verwijder' link natuurlijk).

```
const setup =()=>{
  let links = document.querySelectorAll("#lstIngredients a");
  for (let i = 0; i<links.length; i++){
    let link = links[i];
    link.addEventListener('click', verwijder);
  }
}
const verwijder =(event) => {
  //console.log(event.target.textContent);
  let link = event.target;
  let li = link.parentNode;
  let ul = li.parentNode;
  ul.removeChild(li);
}
window.addEventListener("load", setup);
```

## Opdracht 03

Unzip 'beginsituatie opdracht 03.zip' in een nieuw Webstorm project of folder.

De HTML en CSS code voor deze opdracht is al gegeven :

### Oplossing 01 voor opdracht 03

Type hier een ingrediënt

Ingrediënten :

- boter [verwijder](#)
- kaas [verwijder](#)
- eieren [verwijder](#)

Instructies :

- typ een ingrediënt in het tekstveld
- klik vervolgens op de "Voeg toe" knop
- klik op een 'verwijder' link om een ingrediënt te verwijderen

Voeg de nodige Javascript code toe zodat

- een klik op een 'verwijder' link, het corresponderende ingrediënt verwijdert (incl. de link)
- er een ingrediënt bijkomt als men de naam in het tekstveld typt en op 'Voeg toe' klikt

Je kunt je hierbij baseren op je oplossing van opdracht 02 in combinatie met de oplossing van een gelijkaardige opdracht uit de vorige les.

```
const setup =()=>{

    let btnAdd = document.querySelector("#btnAdd");
    btnAdd.addEventListener("click", add);

    let links = document.querySelectorAll("#lstIngredients a");
    for (let i = 0; i < links.length; i++) {
        let link = links[i];
        link.addEventListener("click", verwijder);
    }
}

const verwijder=(event)=>{
    let link = event.target;
    let li = link.parentNode;
    let ul = li.parentNode;
    ul.removeChild(li);

    //if u have alot of paragraphs it takes u to up page by clicking verwijder ad
    //this line code to stay at the same
    //place
    event.preventDefault();
}
```

```

}
const add = () =>{
  let ingridient = document.querySelector("#txtInput").value;
  let ul = document.querySelector("#lstIngredients");
  ul.insertAdjacentHTML("beforeend", `<li>${ingridient} <a href="#">verwijder</a></li>`);
  //Voeg event listener toe aan nieuwe hyperlink als er nieuwe geadd wordt mag
  dit dan ook verwijdered worden
  let nieuwelItem = ul.lastElementChild;
  let nieuwelink = document.querySelector("#lstIngredients");
  //OF deze onder gebruiken of de 2lijn er boven its same
  //let nieuwelink = document.querySelector("#lstIngredients >li:last-of-
  type a" );
  nieuwelink.addEventListener("click",verwijder);
}
window.addEventListener("load", setup);

```

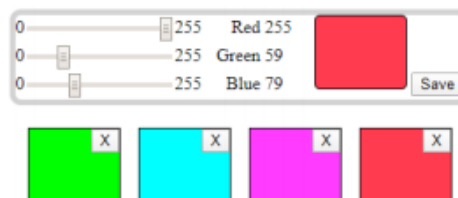
### Opdracht : Colorpicker uitbreiding

Bekijk eerst het korte filmpje dat het eindresultaat demonstreert.

Maak een nieuw project dat een kopie is van de voorbeeldoplossing van de colorpicker opdracht.

Voeg rechts aan de colorpicker component een 'Save' knop toe.

Een klik op de 'Save' knop voegt onderaan een kopie van de swatch toe en een bijbehorende delete knop (met 'X' als opschrift). De swatch is dit keer rechthoekig (niet afgerond) en de delete knop staat netjes in de rechterbovenhoek van deze gekleurde rechthoek.



```

const setup = () => {
  let sliders = document.getElementsByClassName("slider");
  for(let i = 0 ; i < sliders.length ; i++)
  {
    sliders[i].addEventListener("change", update);
    sliders[i].addEventListener("input", update);
  }
  update();
  let btnSave = document.querySelector("#btnSave");
  btnSave.addEventListener("click", saveSwatch);
}

```

```

const saveSwatch =() =>{
  //<div style="background-color:red;" className="swatch">
  //   <input type="button" value="X"></input>
  //</div>
  let div = document.createElement("div");
  div.classList.add("swatch");

  //haal de waarden op van de 3 sliders
  let red = document.getElementById("sldRed").value;
  let green = document.getElementById("sldGreen").value;
  let blue = document.getElementById("sldBlue").value;
  //wijzig de achtergrondkleur van de swatch
  div.style.backgroundColor='rgb('+red+', '+green+', '+blue+')';

  //onthou de rgb waarden in 3 custome attributen
  div.setAttribute("data-red", red);
  div.setAttribute("data-green", green);
  div.setAttribute("data-blue", blue);

  //voeg input element toe
  let input = document.createElement("input");
  input.setAttribute("type", "button");
  input.value="X";
  input.addEventListener("click", deleteSwatch);

  div.appendChild(input);

  //koppel click event listener aan swatch
  div.addEventListener("click", setColorPickerFromSwatch);

  let swatchComponents = document.querySelector("#swatchComponents");
  swatchComponents.appendChild(div);
}

const deleteSwatch =(event) => {
  let input = event.target;
  let div = input.parentNode;
  let section = div.parentNode;
  section.removeChild(div);
  event.stopPropagation();
}

const setColorPickerFromSwatch =()=>{

```

```

    //de X clicken en color van swatch niet veranderen otherwise it will change i
t this if statmenst solves it
    //another solution is
    // (event.target.classList.contains("swatch") the if statment should be aroun
d the code till update
    //    if(event.target === event.currentTarget){
        let swatch = event.target;
        let red = swatch.getAttribute("data-red");
        let green = swatch.getAttribute("data-green");
        let blue = swatch.getAttribute("data-blue");
        let sldRed = document.querySelector("#sldRed");
        let sldGreen = document.querySelector("#sldGreen");
        let sldBlue = document.querySelector("#sldBlue");

        sldRed.value = red;
        sldGreen.value = green;
        sldBlue.value = blue;
        update();
        //or another solution clicking on x doesnt change the sliders isput this
code on deleteswatch event
        // event.stopPropagation();
        //}
    }

const update =()=>{
    let red = document.getElementById("sldRed").value;
    let green = document.getElementById("sldGreen").value;
    let blue = document.getElementById("sldBlue").value;
    document.getElementById("lblRed").innerHTML = red;
    document.getElementById("lblGreen").innerHTML = green;
    document.getElementById("lblBlue").innerHTML = blue;

    let swatch=document.querySelector("#swatch");
    swatch.style.backgroundColor='rgb('+red+', '+green+', '+blue+')';
}
window.addEventListener("load", setup);

```

## DEEL 09

Bekijk die pagina en let daarbij vooral op de verschillende constructoren om Date objecten aan te



maken, alsook de diverse get en set methods waarmee je de maand/jaar/jaar/uur/... onderdelen van

zo'n tijdstip kunt opvragen of aanpassen.

Bijvoorbeeld,

```
let now = new Date();  
let uren = now.getHours(); // uren  
let minuten = now.getMinutes(); // minuten  
console.log( `het is nu ${ uren } uur en ${ minuten } minuten` );  
console.log( `de datum is ${ now.getDate() } / ${ now.getMonth() } `   
); // dag en maand  
console.log( now.toISOString() ); // ISO 8601 formaat
```

Om snel iets uit te proberen kun je echter de tekstvoorstellingen gebruiken die de methods

.toString(), .toISOString(), .toDateString() of .getTimeString() produceren.

Bijvoorbeeld :

```
console.log( now.toString() );  
Sun May 09 2021 23:52:43 GMT+0200 (Central European Summer Time) //   
output  
console.log( now.toDateString() );  
Sun May 09 2021 // output  
console.log( now.getTimeString() );  
23:52:43 GMT+0200 (Central European Summer Time) // output  
console.log( now.toISOString() );  
2021-05-09T21:52:43.296Z // output  
console.log( now.toISOString().substr(0,10) ); // het datum stukje   
eruit halen met substr()  
2021-05-09 // output  
console.log( now.toISOString().substr(11,8) ); // het tijd stukje   
eruit halen met substr()
```

21:52:43 // output

Let erop dat het tijdsdeel van `.toISOString()` als UTC-tijd is uitgedrukt en dus wellicht niet klopt voor de eindgebruiker (verkeerde tijdzone of zomer vs wintertijd).

## Zelf simpele objecten maken

In Javascript kun je heel eenvoudig eigen objecten aanmaken, je hoeft zelfs geen class te definiëren :

```
let student = { }; // maak een nieuw leeg object
student.voornaam = "Jan"; // voeg 'voornaam' property toe
student.familienaam = "Janssens"; // voeg 'familienaam' property toe

let locatie = { }; // maak een nieuw leeg object
locatie.straat = "Koekoekstraat 70"; // voeg 'straat' property toe
locatie.postcode = "90210"; // voeg 'postcode' property toe
locatie.gemeente = "Melle"; // voeg 'gemeente' property toe

locatie.postcode = "9090"; // wijzig waarde van 'postcode' property

student.adres = locatie; // voeg 'adres' property toe

console.log(student.voornaam+" woont in "+student.adres.postcode);
Jan woont in 9090 // output
```

`delete student.voornaam;`

→ We kunnen verwijzingen naar objecten ook gewoon in een array bewaren, bv.

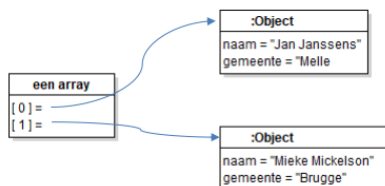
```
let p1 = {}; // een nieuw leeg object
p1.naam = "Jan Janssens";
p1.job = "Melle";

let p2 = {}; // een nieuw leeg object
p2.naam = "Mieke Mickelson";
p2.gemeente = "Brughe";

let personen = []; // een leeg array
personen.push(p1); // voeg p1 achteraan toe aan array (*)
personen.push(p2); // voeg p2 achteraan toe aan array (*)

(*) in het array komen verwijzingen terecht naar de objecten waar 'p1' en 'p2' naar verwijzen
```

Een UML object diagram dat de situatie toont na de uitvoering van deze code:



➡ Verwijzingen kunnen ook als parameter worden doorgegeven bij een functieoproep, bv.

```
const toonGroet = ( persoon ) => {                                // functie 'toonGroet'
  console.log( `Hallo ${ persoon.naam } uit ${ persoon.gemeente }!` );
}

let p = {};                                                       // een nieuw leeg object
p.naam = "Jan Janssens";
p.gemeente = "Melle";

toonGroet( p );           // functieoproep met de verwijzing uit 'p' als parameter
Hallo Jan Janssens uit Melle! // output
```

Als je een ingewikkeld object moet maken met veel properties plus nog wat verwijzingen naar

andere nieuwe objecten, dan zul je behoorlijk veel moeten typen.  
Bijvoorbeeld :

```
let student={};
student.voornaam="Jan";
student.familienaam="Janssens";
student.geboorteDatum=new Date("1993-12-31");
enz.
```

Dit kan veel korter door de object literal notatie te gebruiken, bv.

```
let student = {
  voornaam : "Jan",
  familienaam : "Janssens",
  geboorteDatum : new Date("1993-12-31")
};
```

Let op de dubbele punten, de komma's tussen de properties en de puntkomma op het einde!

Als je een object literal definieert hoeven diens property values niet automatisch ook literals te zijn,

het kunnen bv. ook gewoon waarden zijn die uit andere variabelen komen :

// enkele variabelen met verwijzingen naar de DOM-tree nodes voor  
<input> elementen

```
let txtVoornaam = ...
```

```
let txtFamilienaam = ...
```

```
let txtGeboortedatum = ...
```

```
let student={  
  voornaam : txtVoornaam.value,  
  familienaam : txtFamilienaam.value,  
  geboorteDatum : new Date(txtGeboorteDatum.value)  
};
```

Je ziet dat we de values van de <input> elementen gebruiken als de  
property waarden voor het

student object

Je kan deze notatie ook uitbreiden met objecten en arrays, bv.

```
let student={  
  voornaam : "Jan", // een string  
  familienaam : "Janssens",  
  geboorteDatum : new Date("1993-12-31"), // een Date  
  kotAdres : { // een object  
    straat : "Kerkstraat 12",  
    postcode : "8000",  
    gemeente : "Brugge"  
  },  
  isIngeschreven : true, // een boolean  
  namenVanExen : ["Sofie", "Berta", "Philip", "Albertoooo"], // een  
  array  
  aantalAutos : 0 // een number  
};
```

Je ziet dat een property value eender welk soort waarde kan bevatten : strings, Dates, booleans, andere objecten(!), numbers, arrays(!), etc.

```
const setup = () => {
  // deze code wordt pas uitgevoerd als de pagina volledig is ingeladen
  // maak Jans object
  let jan = { }; // maak een nieuw leeg object
  jan.voornaam = "Jan"; // voeg 'voornaam' property toe
  jan.familienaam = "Janssens"; // voeg 'familienaam' property toe

  // maak Jans adres object
  let locatieJan = { }; // maak een nieuw leeg object
  locatieJan.straat = "Koekoekstraat 70"; // voeg 'straat' property toe
  locatieJan.postcode = "9090"; // wijzig waarde van 'postcode' property
  locatieJan.gemeente = "Melle"; // voeg 'gemeente' property toe

  // voeg een 'adres' property toe aan Jans object, die naar Jans adres object
  // wijst
  jan.adres = locatieJan; // voeg 'adres' property toe

  let mieke = { }; // maak een nieuw leeg object
  mieke.voornaam = "Mieke"; // voeg 'voornaam' property toe
  mieke.familienaam = "Mickelson"; // voeg 'familienaam' property toe

  let locatieMieke = { }; // maak een nieuw leeg object
  locatieMieke.straat = "Kerkstraat 12"; // voeg 'straat' property toe
  locatieMieke.postcode = "8000"; // wijzig waarde van 'postcode' property
  locatieMieke.gemeente = "Brugge"; // voeg 'gemeente' property toe

  // voeg een 'adres' property toe aan Miekies object, die naar Miekies adres object
  // wijst
  mieke.adres = locatieMieke; // voeg 'adres' property toe

  // stop de beide objecten in een array
  let array = []; // maak een nieuw leeg array
  array.push(jan);
  array.push(mieke);
}

window.addEventListener("load", setup);
```

## Opdracht 2

Herschrijf de oplossing van opdracht 1 zodat je code de object literal notatie gebruikt om de twee persoon objecten te bouwen die Jan en Mieke voorstellen.

De bijbehorende adres objecten kun je in die persoon objecten stoppen, zoals bij de 'kotAdres' property op de vorige bladzijde.

```
const setup = () => {  
  // deze code wordt pas uitgevoerd als de pagina volledig is ingeladen  
  
  let array = [{  
    voornaam: "Jan",  
    familienaam: "Janssens",  
    adres: {  
      straat: "Koekoekstraat 70",  
      postcode: "9090",  
      gemeente: "Melle"  
    }  
  }, {  
    voornaam: "Mieke",  
    familienaam: "Mickelson",  
    adres: {  
      straat: "Kerkstraat 12",  
      postcode: "8000",  
      gemeente: "Brugge"  
    }  
  }  
  ];  
  
}
```

```
window.addEventListener("load", setup);
```

### Opdracht 3

Begin met een leeg project en voorzie een `setup()` functie voor het `window load` event.

Maak een globale variabele `'personen'` met daarin een array dat de volgende info bevat :

voornaam	familienaam	leeftijd
Jan	Janssens	29
Mieke	Mickelson	31
Donald	Duck	86
Piet	Piraat	54

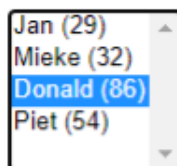
Gebruik voor elke persoon 1 object en stop al die objecten in hetzelfde `'personen'` array.

In de HTML-pagina stop je een `<select>` element met class `'person-list'`. Dit select element toont max. 5 keuzes en laat slechts 1 geselecteerd element toe (geen `'multiple'` attribuut dus).

Daaronder plaats je de elementen voor de details van de geselecteerde persoon :

```
<h1>Details</h1>
<ul class="person-info">
  <li class="firstname"></li>          <!-- hier moet de voornaam komen -->
  <li class="lastname"></li>          <!-- hier moet de familinaam komen -->
  <li class="age"></li>              <!-- hier moet de leeftijd komen -->
</ul>
```

Het eindresultaat zal er zo uitzien, als Donald (86) geselecteerd werd :



## Details

- Donald
- Duck
- 86

```
const setup = () => {
  // deze code wordt pas uitgevoerd als de pagina volledig is ingeladen
}
```

```

    // voeg voor elke persoon een <option> toe
    let select = document.querySelector(".person-list");
    for (let i=0;i<personen.length;i++) {
        let persoon = personen[i];
        // bouw de HTML tekst voor de <option>
        let optionText = `<option data-
index="${i}">${persoon.voornaam} (${persoon.leeftijd})</option>`;
        // voeg de option achteraan toe (i.e. onderaan in de lijst)
        select.insertAdjacentHTML("beforeend", optionText);
    }

    // registreer een 'change' event listener functie
    // die zal worden opgeroepen telkens de selectie wijzigt
    select.addEventListener("change", selecteerPersoon);
};

const selecteerPersoon = () => {
    // de selectie is gewijzigd

    // bepaal welke <option> geselecteerd is
    let select = document.querySelector(".person-list");
    let selectedIndex = select.selectedIndex;
    let option = select.options[selectedIndex];

    // wat is de waarde van het data-index attribuut van die <option>?
    let dataIndexText = option.getAttribute("data-index");
    let dataIndex = Number.parseInt(dataIndexText, 10);

    // bepaal het corresponderende persoon object in het globale 'personen' array
    let persoon = personen[dataIndex];

    // stop de voornaam in de DOM-tree
    let liVoornaam = document.querySelector(".firstname")
    liVoornaam.textContent = persoon.voornaam;

    // stop de familienaam in de DOM-tree
    let liFamilienaam = document.querySelector(".lastname")
    liFamilienaam.textContent = persoon.familienaam;

    // stop de leeftijd in de DOM-tree (eens zonder een variabele te gebruiken)
    document.querySelector(".age").textContent=persoon.leeftijd;
};
window.addEventListener("load", setup);

```



## Opdracht 4

Start met een leeg project dat **een setup functie** zodra de pagina is ingeladen.

Voeg in de Javascript file **een functie toonPersoon** toe.

Deze functie heeft een parameter 'persoon', dit is een verwijzing naar een object met 5 properties :

- voornaam (string)
- familienaam (string)
- geboortedatum (**Date**)
- email (string)
- aantalKinderen (**number**)

Ze toont voor elk van deze properties een geschikte tekstvoorstelling op de console. Deze komen alle 5 naast elkaar op dezelfde regel, gescheiden door komma's.

Probeer je toonPersoon functie uit met het onderstaande code fragment (bv. op de console in de Chrome Developer Tools) :

```
let p = {  
  voornaam : "Jan",  
  familienaam : "Janssens",  
  geboortedatum : new Date("2010-10-15"),  
  email : "jan@example.com",  
  aantalKinderen : 0  
};  
  
toonPersoon( p );  
Jan,Janssens,2010-10-15,jan@example.com,0 // output
```

Als dit goed werkt kun je dit fragment weer wissen, we hebben het verder niet meer nodig.

Definieer **een globale variabele 'personen'** bovenaan je Javascript file. Geef deze variabele een leeg array als beginwaarde.

Schrijf een functie **vulMetDemoData**.

Deze functie heeft geen parameters en vult het 'personen' array met de volgende gegevens :

voornaam	familienaam	geboortedatum	email	aantal kinderen
Jan	Janssens	15 oktober 2010	jan@example.com	0
Mieke	Mickelson	1 januari 1980	mieke@example.com	1
Piet	Pieters	31 januari 1970	piet@example.com	2

Doe dit door 3 objecten aan het globale 'personen' array toe te voegen, één object per persoon.

Schrijf een functie **toonAllePersonen**.

Deze functie heeft een 'personen' parameter, dit is een array van persoon objecten. Of correcter maar saaier : dit is een verwijzing naar een array dat verwijzingen naar objecten bevat. Elk van die objecten heeft dezelfde 5 properties van hiervoor.

De functie overloopt het 'personen' array en roept voor elke persoon object, de **toonPersoon(...)** functie op.

Stop dit codefragment in je **setup** functie :

```
vulMetDemoData();
toonAllePersonen();

Jan,Janssens,2010-10-15,jan@example.com,0           // output
Mieke,Mickelson,1980-01-01,mieke@example.com,1       // output
Piet,Pieters,1970-12-31,piet@example.com,2           // output
```

Herlaad nu de pagina in de browser, als je output klopt dan heb je hoogstwaarschijnlijk alle voorgaande stappen correct uitgevoerd.

Bewerk nu de HTML pagina zodat ze de volgende inhoud toont (de CSS-opmaak is niet zo belangrijk)

Voornaam	<input type="text"/>
Familienaam	<input type="text"/>
Geboortedatum	<input type="text"/>
Email	<input type="text"/>
Aantal kinderen	<input type="text"/>
<input type="button" value="Bewaar"/>	

Als je klaar bent kun je je oplossing met de volgende stappen uitproberen :

### Stap 1

Laad de pagina opnieuw in, er wordt een leeg invulformulier getoond

Voornaam	<input type="text"/>
Familienaam	<input type="text"/>
Geboortedatum	<input type="text"/>
Email	<input type="text"/>
Aantal kinderen	<input type="text"/>
<input type="button" value="Bewaar"/>	

op de console staat nu :

Jan,Janssens,2010-10-15,jan@example.com,0	} output na inladen pagina
Mieke,Mickelson,1980-01-01,mieke@example.com,1	
Piet,Pieters,1970-12-31,piet@example.com,2	

## Stap 2

Vul nu het formulier als volgt :

Voornaam	Joris
Familienaam	Jorissen
Geboortedatum	1945-06-25
Email	joris@example.com
Aantal kinderen	17
<input type="button" value="Bewaar"/>	

en klik op "Bewaar".

Op de console zou nu deze output moeten staan :

Jan,Janssens,2010-10-15,jan@example.com,0	}	output na inladen pagina
Mieke,Mickelson,1980-01-01,mieke@example.com,1		
Piet,Pieters,1970-12-31,piet@example.com,2		
Jan,Janssens,2010-10-15,jan@example.com,0	}	output na klik op "Bewaar"
Mieke,Mickelson,1980-01-01,mieke@example.com,1		
Piet,Pieters,1970-12-31,piet@example.com,2		
Joris,Jorissen,1945-06-25,joris@example.com,17		

Check tot slot nog eens je oplossing, bevat het nieuwe object voor Joris daadwerkelijk

- een verwijzing naar een Date object in de geboortedatum property? (geen string!)
- een number waarde in de aantalKinderen property? (geen string!)

Je kunt dit makkelijk nagaan in de Chrome Developer Tools (bv. bij de debugger).

```
const bewaarPersoon = () => {
  // Maak een leeg object
  const persoon = {};

  // Voeg een string property toe voor de waarde uit het corresponderende tekst
  veld
  const txtVoornaam = document.querySelector("#txtVoornaam");
  persoon.voornaam = txtVoornaam.value;

  // Voeg een string property toe voor de waarde uit het corresponderende tekst
  veld
  const txtFamilienaam = document.querySelector("#txtFamilienaam");
```

```

    persoon.familienaam = txtFamilienaam.value;

    // Voeg een string property toe voor de waarde uit het corresponderende tekst
veld
    const txtEmail = document.querySelector("#txtEmail");
    persoon.email = txtEmail.value;

    // Voeg een Date property toe voor de waarde uit het corresponderende tekstve
ld
    const txtGeboortedatum = document.querySelector("#txtGeboortedatum");
    const geboortedatumAlsTekst=txtGeboortedatum.value;
    persoon.geboortedatum = new Date(geboortedatumAlsTekst);

    // Voeg een number property toe voor de waarde uit het corresponderende tekst
veld
    const txtAantalKinderen = document.querySelector("#txtAantalKinderen");
    const aantalKinderenAlsTekst=txtAantalKinderen.value;
    persoon.aantalKinderen = Number.parseInt(aantalKinderenAlsTekst, 10);

    // toevoegen achteraan het personen array
    personen.push(persoon);

    // toon alle persoonsgegevens op de console
    toonAllePersonen();

    // maak alle invoervelden leeg
    txtVoornaam.value="";
    txtFamilienaam.value="";
    txtEmail.value="";
    txtGeboortedatum.value="";
    txtAantalKinderen.value="";
}

const toonPersoon = (persoon) => {
    // toon de gegevens van een persoon op de console
    const datumTekst = persoon.geboortedatum.toISOString().substr(0,10); // datum
stukje uit ISO 8601 string halen
    console.log(`${persoon.voornaam},${persoon.familienaam},${datumTekst},${perso
on.email},${persoon.aantalKinderen}`);
};

const vulMetDemoData = () => {
    // Voeg de persoonsgegevens toe aan het globale 'personen' array
    // (per persoon 1 object, dus 3 objecten in totaal).

```

```

const jan = {
  voornaam : "Jan",
  familienaam : "Janssens",
  geboortedatum : new Date("2010-10-15"),
  email : "jan@example.com",
  aantalKinderen : 0
};
personen.push(jan);

const mieke = {
  voornaam : "Mieke",
  familienaam : "Mickelson",
  geboortedatum : new Date("1980-01-01"),
  email : "mieke@example.com",
  aantalKinderen : 1
};
personen.push(mieke);

// eens toevoegen zonder extra variabele te introduceren
personen.push({
  voornaam : "Piet",
  familienaam : "Pieters",
  geboortedatum : new Date("1970-01-31"),
  email : "piet@example.com",
  aantalKinderen : 2
});
}

const toonAllePersonen = () => {
  // overloop alle persoonsgegevens in het globale 'personen' array
  for (let i=0;i<personen.length;i++) {
    // toon deze persoon
    let persoon = personen[i];
    toonPersoon(persoon);
  }
}

window.addEventListener("load", setup);

```

## JSON

Bij een webapplicatie is het soms nodig om wat data uit een Javascript programma als tekst voor te

stellen. Deze tekst is niet voor de eindgebruiker bestemd, maar maakt deel uit van de interne

keuken van de applicatie. Bijvoorbeeld om de data mee te sturen in een HTTP response, of te

bewaren in een "permanente" opslagplaats in de browser (bv. local storage).

Het is dan de bedoeling deze tekst later (of elders) te gebruiken om de data te reconstrueren.

Een zeer bekend tekstformaat hiervoor is JSON. Deze afkorting staat voor JavaScript Object Notation

en het is een tekstvoorstelling voor objecten en andere waarden (array, string, number, boolean, ...)

Kort gezegd, het is gewoon een string met daarin de object literal notatie uit het vorige deel!

Je kan makkelijk een object omzetten naar een JSON-tekst met `JSON.stringify()` :

```
let persoon = {  
  voornaam : "Jan",  
  familienaam : "Janssens",  
  aantalKinderen : 0  
};  
let tekst = JSON.stringify( persoon );  
console.log( tekst );  
{"voornaam":"Jan","familienaam":"Janssens","aantalKinderen":0} //  
output
```

Zoals je kunt zien aan de output, is de JSON-tekst gewoon de Javascript code voor een object literal!

Met `JSON.parse()` kunnen we een Javascript object reconstrueren op basis van een JSON-tekst :

- `let tekst = '{"voornaam":"Mieke", "familienaam":"Mickelson", "leeftijd":32}';`
- `// let op de " en ' afwisseling!`

- `let p = JSON.parse( tekst );`
- `console.log( p.leeftijd );`
- `32 // output`

De return value van de `JSON.parse()` oproep is hier een object met drie properties (voornaam, familienaam en leeftijd) dat de info voor Mieke bevat.

We kunnen ook arrays en eenvoudige waarden omzetten naar tekst en reconstrueren, bijvoorbeeld

```
let array = [1, "hallo", false];
console.log( JSON.stringify( array ) );
[1, "hallo", false] // output
let tekst = '[1, "hallo", false]'; // let op de " en ' afwisseling!
let reconstructed = JSON.parse( tekst );
console.log( reconstructed[1] );
hallo // output
```

Vraag : Wat is die "JSON" eigenlijk in een opdracht als `JSON.stringify(...)` of `JSON.parse(...)` ?

Demonstratie

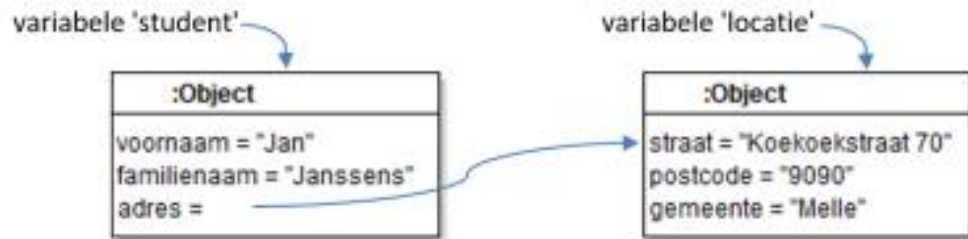
Veronderstel de onderstaande code uit een vorig deel :

```
let student = { };
student.voornaam = "Jan";
student.familienaam = "Janssens";
let locatie = { };
locatie.straat = "Koekoekstraat 70";
locatie.postcode = "9090";
locatie.gemeente = "Melle";
student.adres = locatie;
```

Deze code produceert de volgende situatie in het geheugen :5

Kijken we nu eens naar de JSON-tekstvoorstelling van 'student' :





```
const tekst = JSON.stringify( student );
```

```
console.log( tekst );
```

Als we de output mooier formatteren en inkleuren, wordt de structuur van de JSON-tekst duidelijk :

```
{
  "voornaam": "Jan",
  "familienaaam": "Janssens",
  "adres": {
    "straat": "Koekoekstraat 70",
    "postcode": "9090",
    "gemeente": "Melle"
  }
}
```

Je ziet dat het 'locatie' object automatisch werd opgenomen in de JSON-tekst van het 'student'

object!

## Opdracht 1

Voorspel de output van de onderstaande code :

```
let p1 = {
  naam : 'Jan Janssens',
  gemeente : 'Melle',
};

let p2 = {
  naam : 'Mieke Mickelson',
  gemeente : 'Bruhhe',
};

let personen = [];
personen.push( p1 );
personen.push( p2 );

console.log( JSON.stringify( personen ) );
```

Probeer dit eerst zelf te voorspellen door je JSON-tekst in een document te schrijven.

Voer de code pas daarna uit op de console en vergelijk je voorspelling met de output. Let daarbij vooral op de aanhalingstekens en accenten!

Javascript – deel 10 oplossingen

Oplossing opdracht 1

De JSON-tekst ziet er als volgt uit :

```
[{"naam":"Jan Janssens","gemeente":"Melle"}, {"naam":"Mieke  
Mickelson","gemeente":"Bruhhe"}]
```

Oplossing opdracht 2

## Opdracht 2

Voorspel de output van onderstaande code :

```
let persoon={
  voornaam : 'Jan',
  familienaam : 'Janssens',
  geboortedatum : new Date("1993-12-31"),
  kotAdres : {
    straat : "Kerkstraat 12",
    postcode : '8000',
    gemeente : "Brugge"
  },
  isIngeschreven : true,
  namenVanExen : ['Sofie', "Berta", 'Philip', "Albertoooo"],
  aantalAutos : 0
};

console.log( JSON.stringify( persoon ) );
```

Voer de code pas daarna uit op de console en vergelijk je voorspelling met de output.

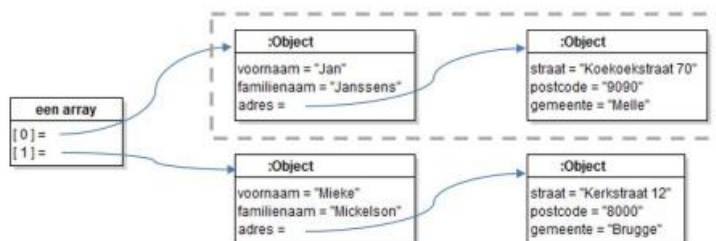
De JSON-tekst ziet er als volgt uit :

```
{"voornaam":"Jan","familienaam":"Janssens","geboortedatum":"1993-12-31T00:00:00.000Z","kotAdres":{"straat":"Kerkstraat 12","postcode":"8000","gemeente":"Brugge"},"isIngeschreven":true,"namenVanExen":["Sofie","Berta","Philip","Albertoooo"],"aantalAutos":0}
```

Oplossing opdracht 3

## Opdracht 3

Wat is de JSON-tekst die overeenkomt met het linkse array uit deze situatie :



Als je je oplossing wil controleren a.d.h.v. een experiment op de console : je vindt de code voor deze situatie in de oplossing van een opdracht uit 'Javascript deel 09'.

De broncode die de getoonde situatie creëert was als volgt (uit JS deel 08)

```
let array = [ {  
  voornaam: "Jan",  
  familienaam: "Janssens",  
  adres: {  
    straat: "Koekoekstraat 70",  
    postcode: "9090",  
    gemeente: "Melle"  
  }  
}, {  
  voornaam: "Mieke",  
  familienaam: "Mickelson",  
  adres: {  
    straat: "Kerkstraat 12",  
    postcode: "8000",  
    gemeente: "Brugge"  
  }  
} ];2
```

We kunnen nu de JSON-tekst produceren met de opdracht

```
console.log( JSON.stringify(array) );
```

En krijgen dan deze JSON-tekst :

```
[{"voornaam":"Jan","familienaam":"Janssens","adres":{"straat":"Koekoek  
straat  
70","postcode":"9090","gemeente":"Melle"}},{ "voornaam":"Mieke","famili  
enaam":"Mickels  
on","adres":{"straat":"Kerkstraat  
12","postcode":"8000","gemeente":"Brugge"}}]
```

## Oplossing opdracht 4

### Opdracht 4 – colorpicker uitbreiding met data-color

Neem de oplossing van opdracht 'Colorpicker uitbreiding' uit 'Javascript deel 08' erbij.

**Pas deze oplossing aan**, zodat elke b-swatch nog maar één enkel data-attribuut 'data-color' meer gebruikt. Dit ene attribuut vervangt de drie aparte data-red, data-green en data-blue attributen.

Bijvoorbeeld, als de originele oplossing dit in de DOM-tree stopte :

```
<div class="swatch" data-red="34" data-green="123" data-blue="90" style="...">...</div>
```

dan moet er nu dit in de DOM-tree terechtkomen :

```
<div class="swatch" data-color="???" style="...">...</div>
```

Op de plaats van de **???** komt er een JSON-tekst die een object beschrijft met een 'red', 'green' en 'blue' property.

De wijzigingen die nodig zijn :

- Op het moment dat ons programma een b-swatch toevoegt, maakt het eerst een simpel kleur object met drie properties waarin de RGB waarden terechtkomen. Daarna wordt van dit kleur object een JSON-tekst gemaakt en die string wordt als waarde voor het **data-color** attribuut gebruikt.
- Als de gebruiker op een b-swatch klikt, haalt ons programma de JSON-tekst uit het custom data-color attribuut. Daarmee reconstrueert het programma vervolgens het kleur object met de RGB waarden en tenslotte worden die waarden gebruikt om de sliders in te stellen.

### - colorpicker uitbreiding met data-color

Deze oplossing zit in een aparte .zip file. Alle wijzigingen staan met commentaar aangegeven in

functies

- saveSwatch
- setColorPickerFromSwatch

## Oplossing opdracht 5a

We hadden al

```
let origineel = {  
  naam : "Jan",  
  geboortedatum : new Date("1993-12-31")  
};
```

De ontbrekende code is dan :

```
let tekst = JSON.stringify( origineel ); // TODO 1  
let kopie = JSON.parse( tekst ); // TODO 2
```

Deze code produceert nu een fout :

```
console.log( kopie.geboortedatum.getFullYear() );
```

Uncaught TypeError: kopie.getFullYear is not a function // output

De reden is dat 'kopie.geboortedatum' geen Date object is, maar een string!

Je kunt dit makkelijk zien door 'typeof' te gebruiken voor die waarden

```
console.log( typeof origineel.geboortedatum );  
object // output
```

```
console.log( typeof kopie.geboortedatum );  
string // output
```

## Oplossing opdracht 5b

De code die de gevraagde situatie creëert, ziet er als volgt uit :

```
// maak Jans object
```

```
let jan = {  
  voornaam : "Jan",  
  familienaam : "Janssens"  
};
```

```
// maak Miekies object  
let mieke = {  
  voornaam : "Mieke",  
  familienaam : "Mickelson"  
};  
  
// maak Miekies adres object  
let locatieMieke = {  
  straat : "Kerkstraat 12",  
  postcode : "8000",  
  gemeente : "Brugge"  
};
```

// voeg een 'adres' property toe aan Miekies object, die naar Miekies adres object wijst

```
mieke.adres = locatieMieke;
```

// voeg een 'adres' property toe aan JANS object, die naar MIEKES adres object wijst

```
jan.adres = locatieMieke;
```

// stop de beide objecten in een array

let origineel = [jan, mieke]; // Ja, dit kan ook zo natuurlijk met een array literal!

Als we de JSON-tekst opvragen met

```
console.log( JSON.stringify(origineel) );
```

krijgen we deze JSON-tekst :

```
[{"voornaam":"Jan","familienaam":"Janssens","adres":{"straat":"Kerkstra  
aat  
12","postcode":"8000","gemeente":"Brugge"}}, {"voornaam":"Mieke","famil  
ienaam":"Mickel  
son","adres":{"straat":"Kerkstraat  
12","postcode":"8000","gemeente":"Brugge"}}]
```

Als we die iets duidelijk formatteren krijgen we :

```
[ {  
  "voornaam": "Jan",  
  "familienaam": "Janssens",  
  "adres": {  
    "straat": "Kerkstraat 12",  
    "postcode": "8000",  
    "gemeente": "Brugge"  
  }  
}, {  
  "voornaam": "Mieke",  
  "familienaam": "Mickelson",  
  "adres": {  
    "straat": "Kerkstraat 12",  
    "postcode": "8000",  
    "gemeente": "Brugge"  
  }  
} ];
```

Je ziet aan de rode tekst hierboven, dat het groene originele 'locatieMieke' object blijkbaar twee keer in de JSON-tekst geplaatst geweest is!

```
const saveSwatch = () => {  
  // <div style="background-color:red;" className="swatch">  
  //   <input type="button" value="X">  
  // </div>  
  let div = document.createElement("div");  
  div.classList.add("swatch");  
  
  // haal de waarden op van de 3 sliders  
  let red=document.getElementById("sldRed").value;  
  let green=document.getElementById("sldGreen").value;  
  let blue=document.getElementById("sldBlue").value;  
  
  // wijzig de achtergrondkleur van de swatch
```



```

div.style.backgroundColor="rgb("+red+", "+green+", "+blue+")";

// BEGIN WIJZIGINGEN tov de oplossing colorpicker uitbreiding uit JS deel 08
// onthou de rgb waarden in 1 custom attribuut
let color = {
  red : red,
  green : green,
  blue : blue
};
let colorText = JSON.stringify( color);
div.setAttribute("data-color", colorText);
// EINDE WIJZIGINGEN tov de oplossing colorpicker uitbreiding uit JS deel 08

// voeg input element toe
let input = document.createElement("input");
input.setAttribute("type", "button");
input.value = "X";
input.addEventListener("click", deleteSwatch);

div.appendChild(input);

// koppel click event listener aan swatch
div.addEventListener("click", setColorPickerFromSwatch);

let swatchComponents = document.querySelector("#swatchComponents");
swatchComponents.appendChild(div);
}

const deleteSwatch = (event) => {
  let input = event.target;
  let div = input.parentNode;
  let section = div.parentNode;

  section.removeChild(div);

  event.stopPropagation();
}

const setColorPickerFromSwatch = (event) => {
  let swatch = event.target;

  // BEGIN WIJZIGINGEN tov de oplossing colorpicker uitbreiding uit JS deel 08
  let colorJSONText = swatch.getAttribute("data-color");
  let color = JSON.parse(colorJSONText);

```

```

    let sldRed = document.querySelector("#sldRed");
    let sldGreen = document.querySelector("#sldGreen");
    let sldBlue = document.querySelector("#sldBlue");

    sldRed.value = color.red;
    sldGreen.value = color.green;
    sldBlue.value = color.blue;
    // EINDE WIJZIGINGEN tov de oplossing colorpicker uitbreiding uit JS deel
08

    update();
}

const update = () => {
    // haal de waarden op van de 3 sliders
    let red=document.getElementById("sldRed").value;
    let green=document.getElementById("sldGreen").value;
    let blue=document.getElementById("sldBlue").value;

    // stop de waarden in de 3 spans
    document.getElementById("lblRed").textContent=red;
    document.getElementById("lblGreen").textContent=green;
    document.getElementById("lblBlue").textContent=blue;

    // wijzig de achtergrondkleur van de swatch
    let swatch=document.getElementById("swatch");
    swatch.style.backgroundColor="rgb("+red+", "+green+", "+blue+)";
};

window.addEventListener("load", setup);

```