hogeschool vives

Labo 15 – Verslag JavaScript | Deel 3

Opleidingsonderdeel: **Web Development 1**

Bachelor in de: **Toegepaste informatica**

Afstudeerrichting: **Applicatieontwikkeling**

Specialisatie: **Cybersecurity & infrastructure**

Leerling: **Jari Opsomer**

Docent: **Pim Debaere**

Academiejaar: **2025-2026**

2 Event based programming

Code **structureren** volgens gebeurtenissen (events) waarop het programma moet reageren **=** Event based programming

Uitgebreid overzicht: developer.mozilla.org/en US/docs/Web/API/Document_Object_Model/Events

Voorbeeld:

Bijvoorbeeld, om een function printHello uit te laten voeren wanneer de gebruiker op een 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)
```

2.1 Window load event

Wachten totdat DOM-tree klaar is:

```
const setup = () => {

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

^ altijd toevoegen

3 Elementen van een HTMLCollection overlopen

- **Nodes vs elementen**: Node = algemeen DOM-object (element, tekst, commentaar, document); element = specifiek type node (HTML-tag)
- **Alle elementen zijn nodes, niet alle nodes zijn elementen**
- **nodeName**: naam node als string (P, BODY, #text, #document)
- **nodeType**: numeriek type node:
  - 1: Element node (HTML-tag)
  - 2: Attribuut node
  - 3: Text node (tekstinhoud)
  - 8: Comment node
  - 9: Document node (root)
- **nodeValue**: tekstwaarde (alleen bij type 3/text nodes); null bij elementen

4 CSS properties instellen op een element

- **style property**: direct CSS-properties instellen op element (camelCase: backgroundColor, paddingLeft)
- **Voorbeeld**: myButton.style.color = "white"; → inline style toegevoegd
- **Nadeel**: styling in JS i.p.v. CSS; beter classes gebruiken (behalve dynamisch, bv. color picker)
- **className**: string met classes (gescheiden door spaties); overschrijven: myButton.className = 'btn secondary';
- **Nadeel className**: lastig toevoegen/verwijderen zonder alle classes te herschrijven
- **classList**: collection voor classes (read-only); methodes:
  - add('class'): toevoegen
  - remove('class'): verwijderen
  - toggle('class'): aan/uit schakelen

4.1 Opdracht: paragrafen

HTML-code:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="styles/style.css">
    <title>Title</title>
</head>
<body>

<p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean ornare.
</p>
<p class="belangrijk">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean ornare.
</p>
<p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean ornare.
</p>
<p class="belangrijk">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean ornare.
</p>

<script type="text/javascript" charset="utf-8"
src="scripts/code.js"></script>
</body>
</html>
```

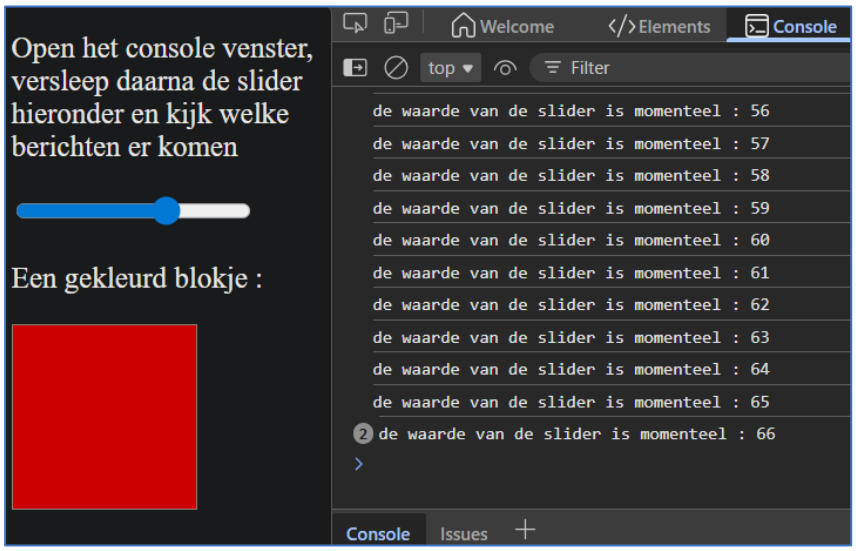
CSS-code:

```
.belangrijk {
    font-weight: bold;
}

.opvallend {
    color: red;
}
```

JS-code:

```
const setup = () => {

    let opvallende = document.getElementsByClassName("belangrijk");
    for (let i = 0; i < opvallende.length; i++) {
        opvallende[i].classList.add("opvallend");
    }

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

4.2 Opdracht: demo slider



Waar wordt de event listener gekoppeld aan de slider?

```
sliders[0].addEventListener("change", update);
sliders[0].addEventListener("input", update);
```

Waarom moeten we die op twee soorten events koppelen?

Chrome en Safari behandelen "change" en "input" verkeerd.

"Input" moet de waarde **live** volgen (bij elke toetsaanslag), terwijl "change" alleen moet triggeren **bij blur** (verlies van focus, zoals Tab of muisklik elders). Maar in Chrome en Safari werken beide events identiek: ze triggeren live bij elke wijziging.

In de CSS file wordt nergens een rode kleur opgegeven, waar wordt dan wel de rode kleur van het blokje ingesteld?

```
colorDemos[0].style.backgroundColor="red";
```

Waarom schrijven we telkens sliders[0] en colorDemos[0] en niet gewoon sliders of colorDemos?

sliders[0] en colorDemos[0] verwijzen naar het **eerste (en enige) element** van die collecties. Zonder [0] probeer je een methode aan te roepen op de hele collectie, wat niet werkt.

4.3 Opdracht: colorpicker

HTML-code:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <title>Demonstratie slider</title>
    <link rel="stylesheet" href="styles/styles.css">
</head>
<body>
<div class="container">
    <div class="sliders">
        <div>
            <input type="range" id="red-slider" min="0" max="255" />
            <span>
            <span>Rood </span>
            <span id="red-value">255</span>
          </span>
        </div>
        <div>
            <input type="range" id="green-slider" min="0" max="255" />
            <span>
            <span>Groen </span>
            <span id="green-value">128</span>
          </span>
        </div>
        <div>
            <input type="range" id="blue-slider" min="0" max="255" />
            <span>
            <span>Blauw </span>
            <span id="blue-value">128</span>
          </span>
        </div>
    </div>
    <div id="color-box"></div>
</div>
    <script src="scripts/code.js"></script>
</body>
</html>
```

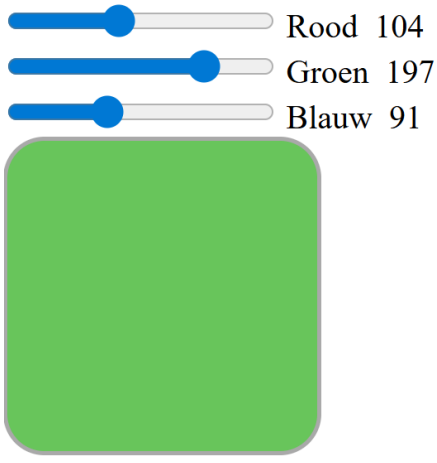
CSS-code:

```
#color-box {
    width: 150px;
    height: 150px;
    border: darkgray solid 2px;

    border-radius: 20px;
}
```

JS-code:

```
const setup = () => {

    // vind alle sliders en hun waardeteksten
    const redSlider = document.getElementById("red-slider");
    const greenSlider = document.getElementById("green-slider");
    const blueSlider = document.getElementById("blue-slider");

    const redValue = document.getElementById("red-value");
    const greenValue = document.getElementById("green-value");
    const blueValue = document.getElementById("blue-value");

    const colorBox = document.getElementById("color-box");

    const updateColor = () => {

        // haal RGB-waarden op en toon ze
        const r = redSlider.value;
        redValue.textContent = r;

        const g = greenSlider.value;
        greenValue.textContent = g;

        const b = blueSlider.value;
        blueValue.textContent = b;

        // Stel kleur in
        colorBox.style.backgroundColor = `rgb(${r}, ${g}, ${b})`;
    };

    // luister naar de sliders
    redSlider.addEventListener("input", updateColor);
    greenSlider.addEventListener("input", updateColor);
    blueSlider.addEventListener("input", updateColor);

    // start met huidige waarden
    updateColor();
};

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

Resultaat:



4.4 DOM element *property style vs. className vs. classList*

- **Manieren styling wijzigen**:
  - .style: directe CSS-properties (camelCase)
  - .className: classes als string overschrijven
  - .classList: add/remove/toggle classes

- **Voorkeur**: classes (styling in CSS houden)

- **Uitzonderingen (.style nodig)**:
  - Te veel variaties (bv.: colorpicker: RGB-kleuren)
  - Dynamische waarden (bv.: spellen: left/top positionering)

4.5 Opdracht: kleurenwisselaar

HTML-code:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="styles/style.css">
    <title>Title</title>
</head>
<body>

<div>
    <button>Button 1</button>
    <button>Button 2</button>
    <button>Button 3</button>
</div>

<script type="text/javascript" charset="utf-8"
src="scripts/code.js"></script>
</body>
</html>
```

CSS-code:

```css
button {
    background-color: white;
    border: 3px solid black;
    padding: 5px;
    font-size: 20px;
}

button.isingedrukt {
    background-color: dodgerblue;
}
```

JS-code:

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

    // vind alle buttons
    const buttons = document.getElementsByTagName("button");

    // zet om naar echte array
    const buttonsArray = Array.from(buttons);

    // voeg klik aan elke button toe
    for (let i = 0; i < buttonsArray.length; i++) {
        const button = buttonsArray[i];
        button.addEventListener("click", function() {
            button.classList.toggle("isingedrukt");
        });
    }
};

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

Resultaat:

5 DOM element property textContent

textContent haalt **alle tekst** op uit een element (inclusief tekst uit verborgen elementen).

Methode	Wat doet het?	Voordelen	Nadelen
innerHTML	Tekst **EN** HTML-tags	Kan HTML toevoegen	**Gevaarlijk** bij gebruikersinput (XSS-aanval mogelijk)
innerText	Alleen **zichtbare** tekst	Negeert verborgen elementen	Mist tekst uit display: none elementen
textContent	**Alle** tekst (ook verborgen)	Veilig, precies	Inclusief alle witruimtes

Goed gebruik:

```
const para = document.querySelector("p");  // Element ZONDER kinderen
para.textContent = "Nieuwe tekst";         // ☑ Veilig en simpel
```

Vermijd bij:

- Elementen met **kinderen** (lijsten, divs met paragrafen) → rommelige whitespace
- HTML-tags nodig → gebruik innerHTML (maar voorzichtig!)

Gebruik textContent voor **simpele tekst-elementen** zonder kinderen.

6 Number

```
let getalAlsTekst="123";
let getalAlsNumber=parseInt(getalAlsTekst, 10);
```

OF:

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

Overzicht van Perplexity:

JavaScript getallen: alles is Number

Belangrijk: JavaScript heeft **GEEN** `int`, `float`, `double` zoals Java. **Alle getallen** zijn `Number` (komma-getallen met hoge precisie). `developer.mozilla`

String → Number omzetten

Functie	Wat doet het?	Voorbeeld
`parseInt(tekst, 10)`	Maakt **geheel getal**	`parseInt("123.45", 10)` → `123`
`parseFloat(tekst)`	Maakt **komma-getal**	`parseFloat("123.45")` → `123.45`

javascript

```
let tekst = "123.45";
let getal = parseFloat(tekst);  // ✅ 123.45
```

Number → String omzetten

javascript

```
let getal = 123;
let tekst = getal.toString();     // "123"
let zin = "Je hebt " + getal;     // "Je hebt 123" ✅ Automatisch!
```

Cijfers na komma beperken

javascript

```
let prijs = 19.999;
console.log(prijs.toFixed(2));    // "20.00" (afronding!)
```

Let op: `toFixed()` geeft **string** terug, geen Number! `developer.mozilla`

6.1 Opdracht: producten

HTML-code:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="styles/style.css">
    <title>Title</title>
</head>
<body>

<table>
    <tbody>
    <tr>
        <th>producten</th>
        <th>prijs</th>
        <th>aantal</th>
        <th>btw</th>
        <th>subtotaal</th>
    </tr>
    </tbody>
    <tr>
        <td>product1</td>
        <td>10.00 Eur</td>
        <td><input type="number" min="0" /></td>
        <td>6 %</td>
        <td>0.00 Eur</td>
    </tr>
    <tr>
        <td>product2</td>
        <td>15.00 Eur</td>
        <td><input type="number" min="0" /></td>
        <td>21 %</td>
        <td>0.00 Eur</td>
    </tr>
    <tr>
        <td>product3</td>
        <td>12.20 Eur</td>
        <td><input type="number" min="0" /></td>
        <td>21 %</td>
        <td>0.00 Eur</td>
    </tr>
    <tr>
        <td colspan="4">totaal</td>
        <td id="totalPrice">0.00 Eur</td>
    </tr>
</table>
<button id="calculate">Herbereken</button>

<script type="text/javascript" charset="utf-8"
src="scripts/code.js"></script>
</body>
</html>
```

CSS-code:

```css
table,
th,
td {
    border: 1px solid;
}

td {
    text-align: left;
}
```

JS-code:

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

    // luister naar klik op "Bereken" knop
    document.getElementById("calculate").addEventListener("click", berekenTotaal);
};

const formatValuta = (bedrag) => {

    // maak bv. Eur 12,34 van 12.34
    return `${bedrag.toFixed(2).replace(".", ",")} Eur`;
};

const berekenTotaal = () => {

    let totaal = 0;

    // doorloop alle rijen in tabel (behalve kop)
    const rijen = document.querySelectorAll("table tr");
    for (let i = 1; i < rijen.length; i++) {
        const rij = rijen[i];

        // sla rijen met te weinig kolommen over
        if (rij.cells.length < 5) continue;

        // haal waarden op uit kolommen
        const prijsTekst = rij.cells[1].textContent.replace(" Eur", "").replace(",", ".");
        const prijs = parseFloat(prijsTekst);

        const hoeveelheid = parseInt(rij.cells[2].querySelector("input").value) || 0;

        const btwTekst = rij.cells[3].textContent.replace("%", "").trim();
        const btw = parseFloat(btwTekst) / 100;

        // bereken sub-totaal en toon
        const subTotaal = hoeveelheid * prijs * (1 + btw);
        rij.cells[4].textContent = formatValuta(subTotaal);

        // tel op bij groot totaal
        totaal += subTotaal;
    }

    // toon groot totaal
    document.getElementById("totalPrice").textContent = formatValuta(totaal);
};

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

Resultaat:

producten	prijs	aantal	btw	subtotaal
product1	10.00 Eur	5	6 %	53,00 Eur
product2	15.00 Eur	9	21 %	163,35 Eur
product3	12.20 Eur	3	21 %	44,29 Eur
totaal				260,64 Eur

Herbereken