hogeschool vives

Labo 19 – Verslag JavaScript | Deel 7

Opleidingsonderdeel: **Web Development 1**

Bachelor in de: **Toegepaste informatica**

Afstudeerrichting: **Applicatieontwikkeling**

Specialisatie: **Cybersecurity & infrastructure**

Leerling: **Jari Opsomer**

Docent: **Pim Debaere**

Academiejaar: **2025-2026**

2.5 Opdracht: hit an object

HTML-code:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="styles/style.css">
    <title>Bom spel</title>
</head>
<body>
<!-- Het speelveld: alles speelt zich hier af -->
<div id="playField">

    <!-- De startknop, verdwijnt zodra het spel begint -->
    <button id="btnStart">START</button>

    <!-- De score die ik bijhoud en toon aan de speler -->
    Aantal hits <span class="score">0</span>

    <!-- Het doelwit: ik verplaats dit icoontje telkens via JavaScript -->
    <img src="images/0.png" class="bom" id="target" alt="sprite">

</div>
<!-- Ik laad het script onderaan zodat de HTML al klaar is als het script
start -->
<script src="scripts/code.js"></script>
</body>
</html>
```

CSS-code:

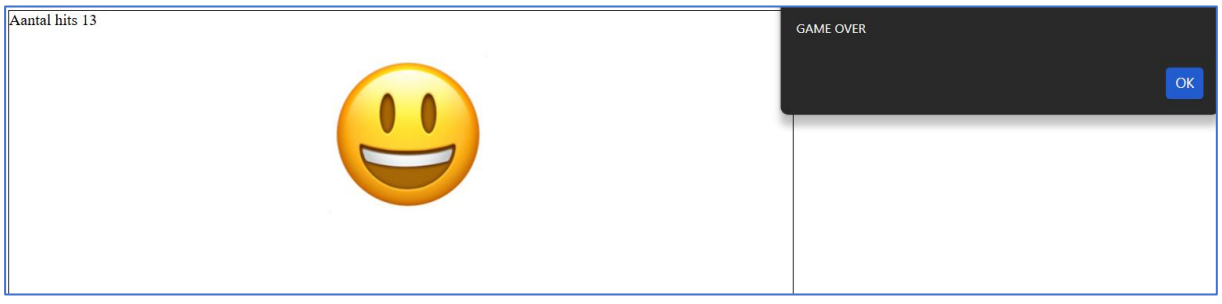
```
/* Het speelveld heeft een vaste grootte zodat het doelwit altijd binnen de
rand blijft */
#playField {
    height: 600px;
    width: 800px;
    border: 1px solid black;
}

/* Ik gebruik position:absolute zodat ik het doelwit zelf kan positioneren
   met left en top via JavaScript. */
#target {
    position: absolute;
}
```

JS-code:

```
// Hier bewaar ik alle instellingen en speldata op één centrale plek
// Zo hoef ik maar op één plek iets aan te passen als ik het spel wil veranderen
let global = {
    IMAGE_COUNT: 5,                     // hoeveel verschillende afbeeldingen er zijn (0
t.e.m. 4)
    IMAGE_SIZE: 48,                     // breedte én hoogte van het doelwit in pixels
    IMAGE_PATH_PREFIX: "images/",       // begin van het pad naar een afbeelding
    IMAGE_PATH_SUFFIX: ".png",          // einde van het pad (de bestandsextensie)

    MOVE_DELAY: 3000,                   // hoelang (in ms) het doelwit op dezelfde plek blijft

    score: 0,                           // het aantal keer dat de speler een niet-bom heeft
geraakt
    timeoutId: 0                        // ik bewaar het id van de timer zodat ik hem kan
annuleren
};

// Deze functie wordt uitgevoerd zodra de pagina volledig geladen is
// Ik koppel hier de startknop aan de startGame-functie
const setup = () => {
    console.log("loaded");
    let btnStart = document.getElementById("btnStart");
    btnStart.addEventListener("click", startGame);
};

// Deze functie start het spel op als de speler op de startknop klikt
const startGame = () => {
    console.log("startgame");

    // Ik verberg de startknop zodat de speler er niet opnieuw op kan klikken
    document.getElementById("btnStart").style.display = "none";

    // Ik voeg een click-listener toe aan het doelwit
    // Zo reageert het doelwit als de speler erop klikt
    //
    // Ik gebruik de klasse "bom" om bij te houden of het huidige doelwit een bom is:
    // - heeft het doelwit de klasse "bom" → het is een bom
    // - heeft het doelwit de klasse "bom" niet → het is geen bom
    // Dit is handig omdat ik die klasse makkelijk kan opvragen in de klik-functie
    let target = document.getElementById("target");
    target.addEventListener("click", klik);

    // Ik roep move() op om het eerste doelwit te tonen
    move();
};

// Deze functie wordt aangeroepen als de speler op het doelwit klikt
const klik = (ev) => {
    // Ik controleer of de aangeklikte afbeelding de klasse "bom" heeft
    // indexOf geeft -1 terug als de klasse niet gevonden wordt
    if (ev.target.className.indexOf("bom") != -1) {
        // Het is een bom: spel is voorbij
        gameOver();
    } else {
        // Het is geen bom: de speler scoort een punt
        hit();
    }
};

// Deze functie verplaatst het doelwit naar een willekeurige positie
// en kiest een willekeurige afbeelding (bom of geen bom)
const move = () => {
    let target = document.getElementById("target");
    let speelScherm = document.getElementById("playField");

    // Ik bereken de maximale positie zodat het doelwit altijd binnen het speelveld valt
    // Ik trek de grootte van het doelwit af, anders zou het gedeeltelijk buiten vallen
    let maxLeft = speelScherm.clientWidth - global.IMAGE_SIZE;
    let maxTop = speelScherm.clientHeight - global.IMAGE_SIZE;

    // Ik kies een willekeurig getal tussen 0 en IMAGE_COUNT - 1
    // Afbeelding 0 is de bom, de rest zijn geldige doelwitten
    let nummer = Math.floor(Math.random() * global.IMAGE_COUNT);
```

```
    if (nummer == 0) {
        // Afbeelding 0 is de bom: ik geef het doelwit de klasse "bom"
        target.className = "bom";
    } else {
        // Geen bom: ik verwijder alle klassen van het doelwit
        target.className = "";
    }

    // Ik stel de juiste afbeelding in op basis van het gekozen nummer
    target.setAttribute("src", global.IMAGE_PATH_PREFIX + nummer + global.IMAGE_PATH_SUFFIX);

    // Ik verplaats het doelwit naar een willekeurige positie binnen het speelveld
    target.style.left = Math.floor(Math.random() * maxLeft) + "px";
    target.style.top = Math.floor(Math.random() * maxTop) + "px";

    // Ik plan een nieuwe verplaatsing na MOVE_DELAY milliseconden
    // Ik sla het id op zodat ik de timer later kan annuleren als het spel stopt
    global.timeoutId = setTimeout(move, global.MOVE_DELAY);
};

// Deze functie wordt aangeroepen als de speler op een bom klikt
const gameOver = () => {
    // Ik annuleer de timer zodat het doelwit niet meer beweegt
    clearTimeout(global.timeoutId);
    alert("GAME OVER");
};

// Deze functie wordt aangeroepen als de speler een niet-bom raakt
const hit = () => {
    let scoreSpans = document.getElementsByClassName("score");

    // Ik annuleer de huidige timer want ik ga zelf move() opnieuw oproepen
    clearTimeout(global.timeoutId);

    // Ik verhoog de score met 1
    global.score++;

    // Ik update alle elementen met de klasse "score" zodat de nieuwe score zichtbaar is
    // Er is er maar één in mijn HTML, maar zo werkt het ook als ik er meerdere zou toevoegen
    for (let i = 0; i < scoreSpans.length; i++) {
        scoreSpans[i].innerText = global.score;
    }

    // Ik verplaats het doelwit meteen naar een nieuwe positie
    move();
};

// Zodra de pagina volledig geladen is, roep ik setup aan
window.addEventListener("load", setup);
```

Resultaat:



2.6 Opdracht: matching game

HTML-code:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <title>Matching game</title>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
</head>

<body>
<!-- Het speelveld: hier worden alle kaarten in geplaatst via JavaScript --
>
<div id="playField">
</div>

<!-- Geluidsbestanden die ik gebruik bij het omdraaien en controleren van
kaarten -->
<!-- preload="auto" zorgt dat de browser de geluiden al inlaadt voor ze
nodig zijn -->
<audio id="draai" preload="auto">
    <source src="sound/draai.mp3" type="audio/mpeg">
    <source src="sound/draai.wav" type="audio/wav">
</audio>
<audio id="goed" preload="auto">
    <source src="sound/goed.mp3" type="audio/mpeg">
    <source src="sound/goed.wav" type="audio/wav">
</audio>
<audio id="fout" preload="auto">
    <source src="sound/fout.mp3" type="audio/mpeg">
    <source src="sound/fout.wav" type="audio/wav">
</audio>

<!-- Ik laad het script op het einde zodat de HTML al bestaat als het
script start -->
<script src="script/code.js"></script>
</body>
</html>
```

CSS-code:

```
/* Als het speelveld geblokkeerd is, verander ik de cursor naar een
wachtcursor
   zodat de speler weet dat hij even moet wachten */
#playField.geblokkeerd {
    cursor: wait;
}

/* Als het spel voorbij is, toon ik een "klaar"-afbeelding als achtergrond
*/
#playField.klaar {
    background-image: url(image/klaar.png);
    background-repeat: no-repeat;
    background-size: contain;
}

/* Een vak is de container rondom elke kaart
```

```
   Ik gebruik inline-block zodat vakken naast elkaar staan maar toch een
breedte/hoogte hebben
   overflow:hidden voorkomt dat de marge van .kaart buiten .vak uitsteekt
(collapsing margins) */
.vak {
    display: inline-block;
    width: 100px;
    height: 100px;
    margin: 10px;
    padding: 1px;
    overflow: hidden;
    background-color: white;
}

/* Elke kaart is even groot als zijn vak
   box-sizing:border-box zorgt dat de border mee wordt opgenomen in de
breedte/hoogte
   zodat ik niet zelf +2px hoef te rekenen */
.kaart {
    width: 100px;
    height: 100px;
    border: 1px solid black;
    box-sizing: border-box;
}

/* Achterkant van een kaart: lichtgrijs */
.kaart.achterkant {
    background-color: lightgray;
}

/* Als ik over een achterkant hover, wordt ze zwart
   zo ziet de speler welke kaart hij gaat omdraaien */
.kaart.achterkant:hover {
    background-color: black;
}

/* Voorkant van een kaart: zwarte achtergrond (de afbeelding staat erop) */
.kaart.voorkant {
    background-color: black;
}

/* Als twee kaarten overeenkomen, worden ze groen */
.kaart.voorkant.goed {
    background-color: lightgreen;
}

/* Als twee kaarten niet overeenkomen, worden ze rood */
.kaart.voorkant.fout {
    background-color: red;
}
```

JS-code:

```
// Hier sla ik alle instellingen op die ik door het spel gebruik
// Zo moet ik maar op één plek iets aanpassen als ik het spel wil veranderen
let global = {
    AANTAL_AFBEELDINGEN: 12,            // hoeveel verschillende afbeeldingen er zijn
    AANTAL_KAARTEN_PER_AFBEELDING: 2,   // elke afbeelding komt 2 keer voor
    AANTAL_KAARTEN_HORIZONTAAL: 6,      // hoeveel kaarten er per rij staan
    PREFIX_KAART_PATH: "image/kaart",   // begin van het pad naar een kaartafbeelding
    SUFFIX_KAART_PATH: ".png",          // einde van het pad (de bestandsextensie)
    ACHTERKANT_PATH: "image/achterkant.png", // afbeelding voor de achterkant van een kaart
}

// Deze functie schudt een array door elkaar
// Ik gebruik dit om de kaarten in een willekeurige volgorde te leggen
const shuffle = (array) => {
    // sort() vergelijkt telkens twee elementen
    // door een willekeurig getal terug te geven beslis ik willekeurig wie "groter" is
    // zo komen de elementen in een willekeurige volgorde terecht
    array.sort((a, b) => {
        return Math.random() - 0.5;
    });
}

// Deze functie wordt aangeroepen als twee kaarten overeenkomen
// Ik voeg de klasse "goed" toe zodat de kaarten groen worden via CSS
const toonGoed = () => {
    let kaarten = document.getElementsByClassName("voorkant");
    // speel het geluid af voor een correcte match
    document.getElementById("goed").play();
    // voeg "goed" toe aan de klassenaam van elke omgedraaide kaart
    for (let i = 0; i < kaarten.length; i++) {
        kaarten[i].className += " goed";
    }
}

// Deze functie wordt aangeroepen als twee kaarten NIET overeenkomen
// Ik voeg de klasse "fout" toe zodat de kaarten rood worden via CSS
const toonFout = () => {
    let kaarten = document.getElementsByClassName("voorkant");
    // speel het geluid af voor een foute match
    document.getElementById("fout").play();
    // voeg "fout" toe aan de klassenaam van elke omgedraaide kaart
    for (let i = 0; i < kaarten.length; i++) {
        kaarten[i].className += " fout";
    }
}

// Deze functie draait alle omgedraaide kaarten (met klasse "voorkant") terug om
// Dit doe ik als de twee kaarten niet overeenkomen
const draaiKaartenMetVoorkantNaarAchterkant = () => {
    let kaartenMetVoorkant = document.getElementsByClassName("voorkant");

    // Opgelet: getElementsByClassName geeft een LIVE collection terug
    // Dat betekent dat de lijst automatisch verandert als ik een klasse aanpas
    // Als ik een gewone for-lus zou gebruiken zou ik kaarten overslaan
    // omdat .length kleiner wordt terwijl i groter wordt
    // Met een while-lus pak ik telkens opnieuw het eerste element
    // zodat ik zeker alle kaarten verwerk
    while (kaartenMetVoorkant.length > 0) {
        kaartenMetVoorkant[0].setAttribute("src", global.ACHTERKANT_PATH);
        kaartenMetVoorkant[0].className = "kaart achterkant";
    }

    // Ik verwijder de klasse "geblokkeerd" zodat de speler weer kan klikken
    document.getElementById("playField").className = "";
}

// Deze functie verwijdert alle omgedraaide kaarten als ze overeenkomen
// Ze verdwijnen dus volledig uit het speelveld
const verwijderKaartenMetVoorkant = () => {
    let kaart;
    let kaartenMetVoorkant = document.getElementsByClassName("voorkant");

    // Ook hier gebruik ik een while-lus omdat het een LIVE collection is
    // (zie uitleg bij draaiKaartenMetVoorkantNaarAchterkant)
```

```
    while (kaartenMetVoorkant.length > 0) {
        kaart = kaartenMetVoorkant[0];
        // Ik verwijder eerst de click-listener zodat de kaart niet meer reageert
        kaart.removeEventListener("click", klikOpKaart);
        // Dan verwijder ik de kaart zelf uit de DOM-tree
        kaart.parentNode.removeChild(kaart);
    }

    // Ik verwijder de klasse "geblokkeerd" zodat de speler weer kan klikken
    document.getElementById("playField").className = "";

    // Ik controleer of het spel voorbij is (geen kaarten meer over)
    controleerSpelGedaan();
}

// Deze functie controleert of alle kaarten al gematcht zijn
// Als er geen kaarten meer over zijn, is het spel gedaan
const controleerSpelGedaan = () => {
    let kaarten = document.getElementsByClassName("kaart");
    if (kaarten.length == 0) {
        // Er zijn geen kaarten meer, dus het spel is voorbij
        let playField = document.getElementById("playField");

        // Ik onthoud de hoogte van het speelveld voordat ik het leegmaak
        // zodat de "klaar"-afbeelding even groot blijft als het speelveld was
        let savedHeight = playField.clientHeight;
        playField.innerHTML = "";

        // Na innerHTML="" is de hoogte 0, dus ik stel hem handmatig opnieuw in
        playField.style.height = savedHeight + "px";

        // Door de klasse "klaar" toe te voegen toont de CSS een afbeelding
        playField.className = "klaar";
    }
}

// Deze functie kijkt of de omgedraaide kaarten dezelfde afbeelding hebben
// Afhankelijk van het resultaat plan ik de juiste acties in via setTimeout
const controleerOpOvereenkomst = (kaarten) => {
    let eersteKaart = kaarten[0];
    let allenGelijk = true;

    // Ik vergelijk elke kaart met de eerste kaart
    // Als één kaart anders is, stop ik meteen en weet ik dat het geen match is
    for (let i = 1; i < kaarten.length; i++) {
        if (eersteKaart.getAttribute("src") != kaarten[i].getAttribute("src")) {
            allenGelijk = false;
            break;
        }
    }

    if (allenGelijk) {
        // De kaarten komen overeen: ik toon groen na 0,5s en verwijder ze na 1s
        window.setTimeout(toonGoed, 500);
        window.setTimeout(verwijderKaartenMetVoorkant, 1000);
    } else {
        // De kaarten komen niet overeen: ik toon rood na 0,5s en draai ze na 1s terug
        window.setTimeout(toonFout, 500);
        window.setTimeout(draaiKaartenMetVoorkantNaarAchterkant, 1000);
    }
}

// Deze functie wordt aangeroepen als de speler op een kaart klikt
const klikOpKaart = (e) => {
    // Ik controleer of het speelveld NIET geblokkeerd is
    // Als er al twee kaarten open liggen, wil ik niet dat er verder geklikt kan worden
    if (document.getElementById("playField").className != "geblokkeerd") {
        let kaart = e.target;
        let kaartenMetVoorkant = document.getElementsByClassName("voorkant");

        // Speel het omdraai-geluid af
        document.getElementById("draai").play();

        // Draai de kaart om: verander de klasse en toon de voorkant-afbeelding
        kaart.className = "kaart voorkant";
        kaart.setAttribute("src", kaart.getAttribute("data-imageSource"));
```

```
        // Opgelet: kaartenMetVoorkant is een LIVE collection
        // De kaart die ik net omdraaide staat er dus al in
        // Daarom test ik op == AANTAL_KAARTEN_PER_AFBEELDING (= 2) en niet op == 1
        if (kaartenMetVoorkant.length == global.AANTAL_KAARTEN_PER_AFBEELDING) {
            // Er liggen nu genoeg kaarten open om te vergelijken
            controleerOpOvereenkomst(kaartenMetVoorkant);
            // Blokkeer het speelveld zodat de speler even moet wachten
            document.getElementById("playField").className = "geblokkeerd";
        }
    }
}

// Deze functie maakt één vak aan met een kaart erin en voegt het toe aan het speelveld
const addVak = (parent, kaartNummer) => {
    let vak = document.createElement("span");
    let kaart = document.createElement("img");

    // Ik stel de klasse en achterkant-afbeelding in voor de nieuwe kaart
    kaart.className = "kaart achterkant";
    kaart.setAttribute("src", global.ACHTERKANT_PATH);

    // Ik sla de voorkant-afbeelding op als data-attribuut
    // zodat ik weet welke afbeelding er achter de kaart zit zonder het te tonen
    kaart.setAttribute("data-imageSource", global.PREFIX_KAART_PATH + kaartNummer +
global.SUFFIX_KAART_PATH);

    // Als er op de kaart geklikt wordt, roep ik klikOpKaart aan
    kaart.addEventListener("click", klikOpKaart);

    // Ik stop de kaart in het vak en het vak in het speelveld
    vak.className = "vak";
    vak.appendChild(kaart);
    parent.appendChild(vak);
}

// Deze functie start het spel op: ze maakt alle kaarten aan en legt ze neer
const initialize = () => {
    let playField = document.getElementById("playField");
    let kaartNummers = [];
    let aantalKaarten = global.AANTAL_AFBEELDINGEN * global.AANTAL_KAARTEN_PER_AFBEELDING;

    // Ik maak een lijst van kaartnummers waarbij elk nummer
    // AANTAL_KAARTEN_PER_AFBEELDING keer voorkomt
    // (bv. 0,1,2,...,11,0,1,2,...,11 voor 2 kaarten per afbeelding)
    for (let i = 0; i < aantalKaarten; i++) {
        kaartNummers.push(i % global.AANTAL_AFBEELDINGEN);
    }

    // Ik schud de kaartnummers willekeurig door elkaar
    shuffle(kaartNummers);

    // Ik voeg alle vakken toe aan het speelveld, rij per rij
    for (let i = 0; i < aantalKaarten; i++) {
        // Aan het begin van elke nieuwe rij (behalve de eerste) voeg ik een regelafbreking
toe
        if (i % global.AANTAL_KAARTEN_HORIZONTAAL == 0 && i != 0) {
            let lineBreakElement = document.createElement("br");
            playField.appendChild(lineBreakElement);
        }
        // Ik voeg een vak toe met het kaartnummer uit mijn geschudde lijst
        addVak(playField, kaartNummers[i]);
    }
}

// Zodra de pagina volledig geladen is, start ik het spel op
window.addEventListener("load", initialize);
```

Resultaat: