hogeschool vives

Labo 14 – Verslag JavaScript | Deel 2

Opleidingsonderdeel: **Web Development 1**

Bachelor in de: **Toegepaste informatica**

Afstudeerrichting: **Applicatieontwikkeling**

Specialisatie: **Cybersecurity & infrastructure**

Leerling: **Jari Opsomer**

Docent: **Pim Debaere**

Academiejaar: **2025-2026**

2.4 Opdracht: Dialoogvensters

JS-code:

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

// alert
alert('Dit is een waarschuwing');

// confirm
let result = confirm('Weet je zeker dat je de foto wil verwijderen?');

if (result) {
    // verwijder de foto
} else {
    // doe helemaal niets
}

// prompt
let naam = prompt('Wat is je naam?');

if (naam !== null) {
    alert('Hallo, ' + naam + '!');
} else {
    alert('Er werd niets ingevoerd.');
}

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

confirm → **boolean** (true/false)

prompt → **String** (tekst)

2.5 Elementen uit de DOM tree opvragen

- **DOM & DOM API**
  - hiërarchische boom van de pagina (nodes)
  - via **document**: structuur, stijl, inhoud lezen en wijzigen

- **Nodes vs elementen**
  - node: elk DOM-object (element, tekst, commentaar, attribuut, document, …)
  - element: node die een HTML-tag voorstelt
  - alle elementen zijn nodes; niet alle nodes zijn elementen
  - belangrijke properties: nodeName, nodeType, nodeValue

- **Selectiemethodes (en returntypes)**
  - getElementById(id) → één element of null
  - getElementsByClassName(class) → HTMLCollection
  - getElementsByTagName(tag) → HTMLCollection
  - getElementsByName(name) → HTMLCollection
  - querySelector(selector) → eerste passend element of null
  - querySelectorAll(selector) → NodeList

- **Werken met HTMLCollection / NodeList**
  - hebben length
  - element via index: list[i]
  - vaak gebruiken in een for-loop.

2.6 De innerHTML property

- **innerHTML algemeen**
  - property van een element
  - leest of overschrijft de **volledige** HTML-inhoud van dat element
  - waarde is altijd een string met tekst en/of HTML-tags

- **Mogelijkheden**
  - gewone tekst aan een element geven: div.innerHTML = 'tekst';
  - opgemaakte HTML plaatsen: div.innerHTML = ' Paragraaf ';
  - in één keer een volledige structuur (met geneste elementen) invullen

- **Nadelen / aandachtspunten**
  - altijd de volledige inhoud vervangen, niet één kindelement
  - nieuwe elementen zijn niet automatisch als aparte referentie beschikbaar
  - gevaarlijk met gebruikersinput: HTML wordt geïnterpreteerd, risico op misbruik
  - vuistregel: zo weinig mogelijk gebruiken, zeker nooit rechtstreeks met gebruikersinput

- **Tip uit de cursus**
  - voor grotere/dynamische structuren beter andere technieken gebruiken (bv. nodes aanmaken en toevoegen)

2.6.1 Opdracht: innerHTML

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 id="txtOutput">Hello world!</p>

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

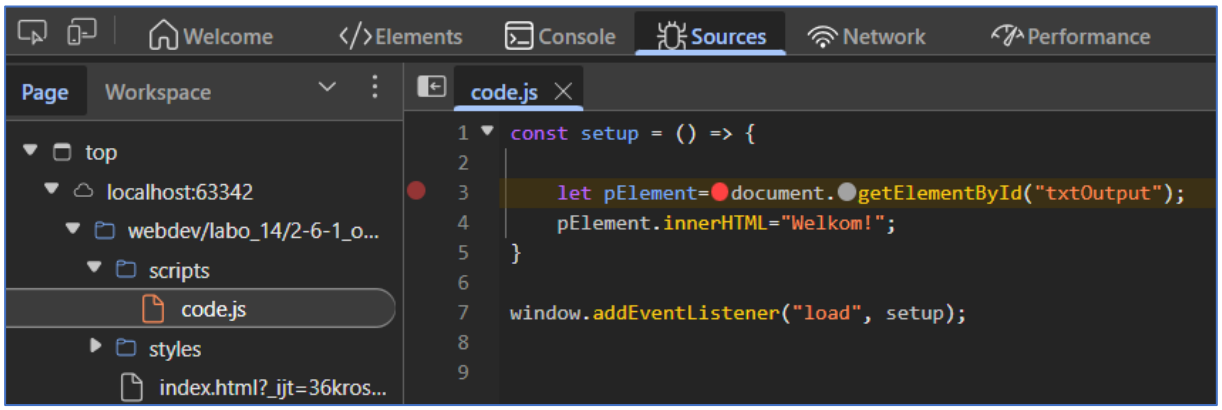
JS-code:

```
const setup = () => {

    let pElement=document.getElementById("txtOutput");
    pElement.innerHTML="Welkom!";
}

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

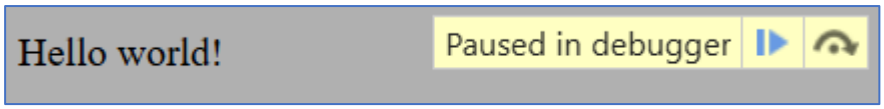
Resultaat:

Welkom!

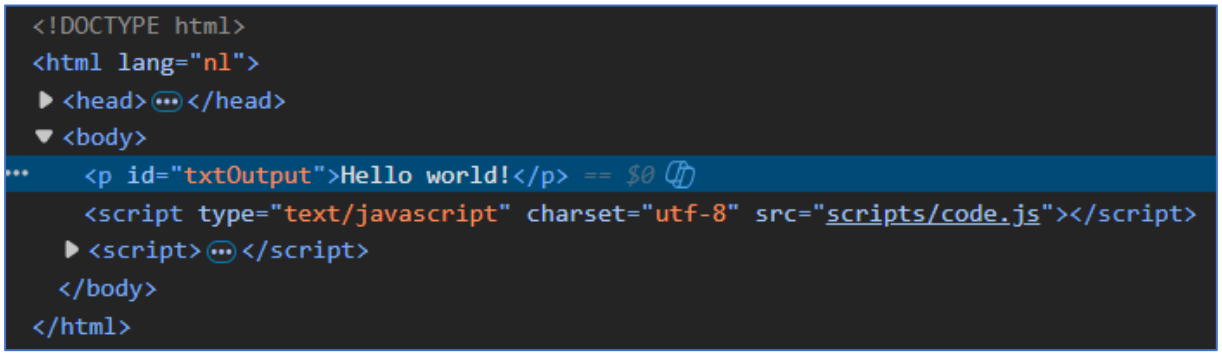
2.6.2 **Opdracht: debuggen**



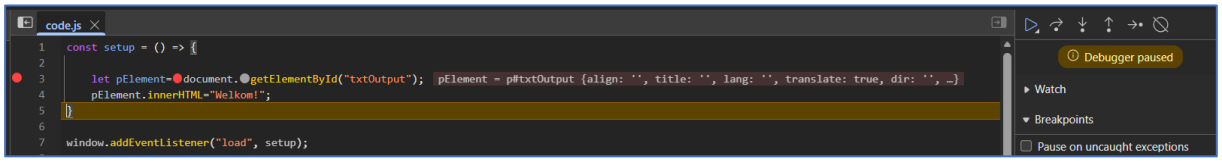
Breakpoint resulteert in **ander resultaat**:



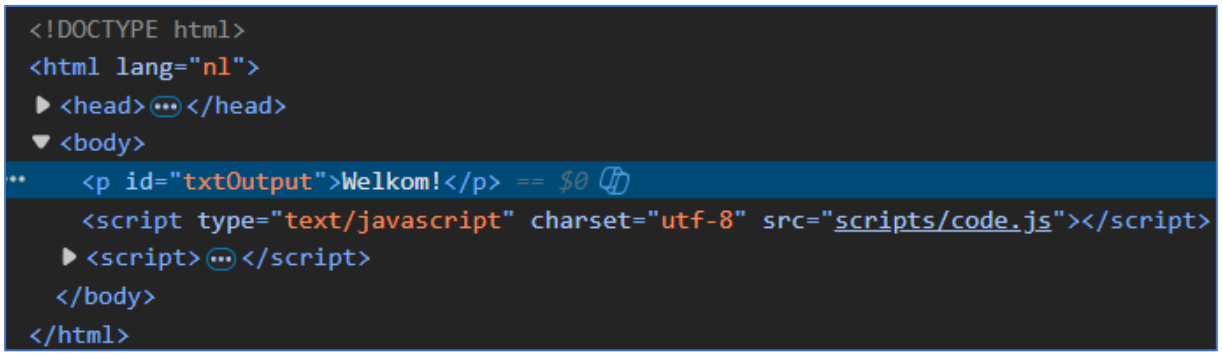
DOM-tree:



Debugger doorlopen:



Resultaat? DOM-tree = nu wel terug aangepast:



2.6.3 Opdracht: innerHTML aanpassing

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 id="txtOutput">Hello world!</p>

<input type="button" id="btnWijzigen" value="Wijzig">

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

JS-code:

```
const setup = () => {

    let btnWijzigen=document.getElementById("btnWijzigen");
    btnWijzigen.addEventListener("click", wijzigen);
}

const wijzigen = () => {

    let pElement=document.getElementById("txtOutput");
    pElement.innerHTML="Welkom!";
}

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

Situatie, **voor** indrukken van knop:

Situatie, **na** indrukken van knop:

2.7 Tekstvelden uitlezen

Demonstratie 'images':

HTML-code:

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

<body>
    <input id="txtInput" type="text" placeholder="Type hier de url van een afbeelding">
    <input id="btnVoegToe" type="button" value="Voeg toe">
    <div class="images" id="divImages"></div>

    <script src="scripts/index.js"></script>
</body>
</html>
```

JS-code:

```
const setup = () => {
    let btnVoegToe = document.getElementById("btnVoegToe");
    btnVoegToe.addEventListener("click", voegToe);
}

const voegToe = () => {
    let txtInput = document.getElementById("txtInput");
    let url = txtInput.value;
    let divImages = document.getElementById("divImages");
    divImages.innerHTML += "<img src='"+url+"'>";
    txtInput.value="";
}

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

Demonstratie 'lijsten:

HTML-code:

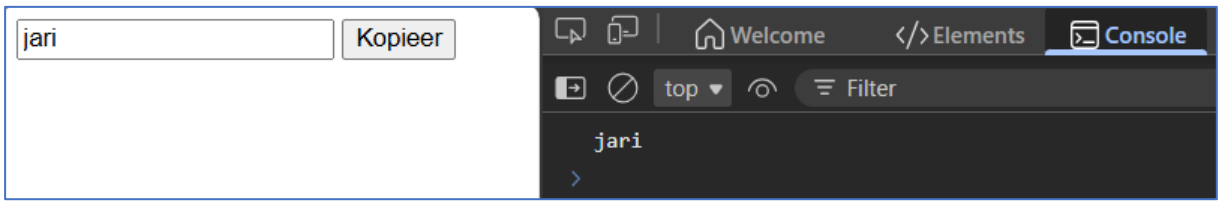
```html
<!DOCTYPE html>
<html lang="nl">
<head>
    <title>Demonstratie lijsten</title>
    <link rel="stylesheet" type="text/css" href="styles/index.css">
</head>

<body>
    <input id="txtInput" type="text" placeholder="Type hier een ingredient">
    <input id="btnVoegToe" type="button" value="Voeg toe">
    <p>Ingredienten :</p>
    <ul id="lstEntries"></ul>

    <script src="scripts/index.js"></script>
</body>
</html>
```

JS-code:

```js
const setup = () => {
    // registeer click event listener bij button
    let btnVoegToe = document.getElementById("btnVoegToe");
    btnVoegToe.addEventListener("click", voegToe);
}

const voegToe = () => {
    // Lees de tekst uit het textveld en voeg nieuw <li> element toe
    let txtInput = document.getElementById("txtInput");
    let lstEntries = document.getElementById("lstEntries");
    let tekst = txtInput.value;
    lstEntries.innerHTML += "<li>"+tekst+"</li>";
}

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

2.7.1 Opdracht: kopieer

Huidige situatie → Bericht wordt getoond in **console**:



Doel → Bericht laten tonen in **p-element**:

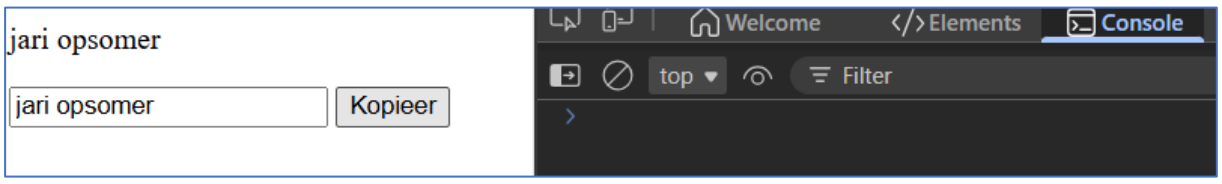
HTML-code (in body):

```
<p id="txtOutput">geen output</p>

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

JS-code:

```
const setup = () => {

    let btnKopieer = document.getElementById("btnKopieer");
    btnKopieer.addEventListener("click", kopieer);
}

const kopieer = () => {

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

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

    txtOutput.innerHTML=txtInput.value;
}

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

Resultaat:



2.8 Opdracht: substring

HTML-code:

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

"<input id="txtInput" type="text" value="appelboom">"
<input id="btnSubstring" type="button" value="substring" />
(
<input id="txtStartIndex" type="number" value="2">
,
<input id="txtEndIndex" type="number" value="5">
) =
<span id="txtOutput">(geen output)</span>

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

JS-code:

```
const setup = () => {

    let btnSubstring = document.getElementById("btnSubstring");
    btnSubstring.addEventListener("click", substring);
}

const substring = () => {

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

    const text = document.getElementById("txtInput").value;
    const start = Number(document.getElementById("txtStartIndex").value);
    const end = Number(document.getElementById("txtEndIndex").value);

    const result = text.substring(start, end);

    if (result !== "") {
        txtOutput.innerHTML = result;
    } else {
        txtOutput.innerHTML = '(geen output)';
    }
}

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

Voorbeeld, resultaat: