JavaScript samenvatting

Deze samenvatting bevat de basis van JavaScript voor het gebruik in de browser.

Structuur, inhoud vorm en interactie

HTML vormt de structuur en de inhoud van een pagina en bestaat uit tags die door de browser worden begrepen. CSS zorgt voor de vormgeving van die inhoud. JavaScript zorgt voor interactie tussen de gebruiker en de website. Denk bijvoorbeeld aan meldingen, slideshows, lightboxes en invoervalidatie.

JavaScript is een programmeertaal, in een programmeertaal schrijf je programmas die uitgevoerd kunnen worden. In dit geval worden de programmas uitgevoerd door de browser.

JavaScript kan ook gebruikt worden om de gebruikerservaring van een site te verbeteren zoals bijvoorbeeld met parallax effecten, smooth scrolling en drag-and-drop.

Daarnaast kent HTML5 een aantal API's die je kunt gebruiken voor bijvoorbeeld geolocating, local storage, canvas en het besturen van audio en video.

Ook wordt JavaScript gebruikt om met de webserver te communiceren (AJAX).

JavaScript in een extern bestand

JavaScript kun je in de HTML code plaatsen tussen `<script>` tags, maar het is beter om je code te scheiden van de HTML en CSS door het in een of meerdere aparte bestanden te plaatsen.

Hieronder een HTML5 pagina met een stylesheet en een JavaScript bestand gekoppeld.

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>HTML5 webpagina</title>
  <link rel="stylesheet" href="style.css">
  <script src="script.js"></script>
</head>
<body>

</body>
</html>
```

JavaScript programmeren

Variabelen

Een variabele gebruik je binnen een programma om aan waardes te refereren. Een variabele heeft een naam en een waarde.

```
var naam = "Niels";
var leeftijd = 34;
var gewicht = 64.5;
var hobbies = ['hardlopen', 'programmeren', 'reizen']; // een
     Array met waarden
var docent = true;
var schatrijk = false;
```

Met variabelen kun je in je programma van alles doen. Je kunt ze veranderen, gebruiken in berekeningen en je kunt nieuwe variabelen maken.

Om een nieuwe variabele te declareren gebruik je het keyword 'var'. Hieronder enkele voorbeelden.

```
var a = 10;
var b = 20;
var c = a + b;

var voornaam = "Walter";
var achternaam = "White";
var volledigeNaam = voornaam + " " + achternaam;

var x = 12;
x = x * 8;
```

Operatoren

Er zijn in JavaScript verschillende operatoren die je kunt gebruiken om variabelen te bewerken. Bijvoorbeeld om te rekenen met getallen of om twee strings samen te voegen.

```
var a = 8.45;
var b = 3.23;
var c = a * b;
var d = a / b;
var e = c - b;
var f = d + a;

var voornaam = "Skyler";
var achternaam = "White";
var volledigeNaam = voornaam + " " + achternaam;
```

Functies

Functies zijn delen van een programma die een specifieke taak hebben. Je kunt bijvoorbeeld een functie maken om twee getallen bij elkaar op te tellen of een functie om te controleren of een ingevoerd e-mailadres geldig is.

Een functie heeft een naam, kan een of meer argumenten mee krijgen en kan een resultaat teruggeven.

```
var waarde = telOp(10, 4);

function telOp(a, b) {
  var resultaat = a + b;
  return resultaat;
}
```

Er zijn binnen JavaScript ook een heleboel ingebouwde functies die je kunt gebruiken. Bijvoorbeeld om een willekeurig getal op te vragen met `Math.random()`, of om een string op te delen met de `split()` functie op een string.

```
var x = Math.random() * 100; // willekeurig getal tussen 0 en
     100;

var zin =  "Hallo ik ben een programmeur";
var woorden = zin.split(' '); // ['Hallo', 'ik', 'ben', 'een', '
     programmeur'];
```

Herhalingen

Vaak wil je handelingen meer dan één keer uitvoeren. Dit kan in JavaScript met `for` loops en met `while` loops. For loops gebruik je als je van tevoren weet hoe vaak je iets wilt herhalen. While loops gebruik je als je dat niet van te voren weet, bijvoorbeeld als je in een lijst op zoek bent naar een naam van iemand of als je het 152e priemgetal na de 10000 zoekt.

Een `for` loop bestaat uit drie gedeeltes, de initialisatie, de conditie en een opdracht die na iedere herhaling wordt uitgevoerd. Onderstaande lus wordt 100 keer herhaald.

```
for (var i=0; i < 100; i++) {
  // code die 100 keer wordt uitgevoerd
}
```

Een while loop kent alleen een conditie. Zolang die conditie 'waar' is wordt de code herhaald. Als je niet goed oplet hiermee maak je makkelijk een loop die oneindig lang wordt herhaald, waardoor je browser vast loopt.

```
var i = 0;
while (i < 100) {
  // code wordt herhaald
  i++;
}
```

Situatie afhankelijk code uitvoeren

Soms wil je in een programma afhankelijk van een situatie bepaalde code uitvoeren. Dit kan met een `if` statement. Denk bijvoorbeeld aan een spelletje waarbij je wilt controleren of iemand gewonnen of verloren heeft.

```
if (score > 10) {
  alert('goed bezig');
} else {
  alert('goed je best blijven doen');
}
```

JavaScript en de browser

De webbrowser voert het JavaScript programma uit. Vaak wil je vanuit JavaScript de inhoud of de vormgeving van een of meer elementen op een webpagina aanpassen (bijvoorbeeld van een divje). Dit gaat via het Document Object Model (DOM).

Document Object Model (DOM)

De DOM wordt door je browser gemaakt aan de hand van de tags in de HTML code (elementen). Ieder element van een webpagina is een onderdeel van de DOM.

De DOM is een **boomstructuur** met de `HTML` tag als begin. Onder de `HTML` tag komt altijd een `HEAD` tag en een `BODY` tag. Binnen die tags komen weer andere tags zoals de `TITLE` tag in de head en `ARTICLE` en `SECTION` tags in de body.

Wachen tot de DOM geladen is

Voordat je met JavaScript de DOM aan kunt passen, moet je er voor zorgen dat je zeker weet dat de DOM volledig is geladen door de browser. Dit kan met behulp van het `window.onload` event.

```
window.onload = init;

function init() {
  // De DOM is nu geladen
  var blokje = document.getElementById('blokje');
  blokje.style.backgroundColor = 'red';
}
```

Elementen uit de DOM opvragen

Met JavaScript kun je elementen uit de DOM opvragen. Dit kan op verschillende manieren. Als je een uniek element op wilt vragen gebruik je meestal `document.getElementById` of `document.querySelector()`. De query selector is krachtiger en maakt gebruik van de zelfde manier van selecteren als CSS. Hieronder een paar voorbeelden.

```
<div id="blokje"></div>
<ul>
  <li class="item">item 1</li>
  <li class="item">item 2</li>
  <li class="item">item 3</li>
</ul>
```

```
var blokje = document.getElementById('blokje'); // vraag het
     divje op met ID blokje
var blokje = document.querySelector('#blokje'); // zelfde maar
     dan met query selector
```

Je kunt ook meerdere elementen selecteren met `document.getElementsByClassName` en met `document.querySelectorAll`. Deze functies leveren een lijst op met elementen uit de DOM.

```
var items = document.getElementsByClassName('item'); // vraag
     all elementen met class item op
var items = document.querySelectorAll('.item'); // zelfde maar
     dan met query selector
```

Elementen aanpassen

Met `innerHTML` kun je de inhoud van een geselecteerd element aanpassen.

```
var blokje = document.getElementById('blokje');
blokje.innerHTML = "Hallo Wereld!";
```

Je kunt ook de style van een element aanpassen.

```
var blokje = document.getElementById('blokje');
blokje.style.backgroundColor = 'green';
blokje.style.top = '100px';
```

Events afvangen

Om je pagina interactief te maken kun je met JavaScript op bepaalde gebeurtenissen reageren. Dit noemen we event handling. Er zijn heel veel verschillende soorten events, bijvoorbeeld events als de bezoeker ergens op klikt met de muis of als iemand een formulier verstuurd.

Ook het wachten tot de DOM geladen is, is het afhandelen van een event.

```
var knop = document.getElementById('knop');

knop.onclick = volgendeFoto; // koppel de functie volgendeFoto
     aan het klikken op de knop

function volgendeFoto() {
  // toon de volgende foto
}
```