OEFENINGEN JAVASCRIPT

01 – POPULARITEIT

"JavaScript is de populairste programmeertaal op internet". Zoek minimaal twee andere artikelen die gaan over 2013, waarin wordt ingegaan op de populariteit van programmeertalen – op internet of in het algemeen. Hoe scoort JavaScript daarin?

02 – EENVOUDIG BEGINNEN

a) Schrijf een script dat twee variabelen bij elkaar optelt en het resultaat in een alert-venster toont.
   - Alternatief: gebruik `console.log('…')`

b) Voorzie je script van nuttig commentaar.

c) Schrijf een statement met en zonder haakjes.
   - bekijk de verschillen in uitvoer:
   - bijvoorbeeld `3 + 4 * 8` vs `(3 + 4)* 8`

d) Wat gebeurt er als je een getal bij een string optelt?

e) Wat gebeurt er als je twee getallen in stringvorm bij elkaar optelt (bijvoorbeeld `'50' + '50'`)?

f) Onderzoek de werking van `parseInt()` en `parseFloat()`.
   - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/parseInt
   - Schrijf er een voorbeeld mee.
   - Noem twee situaties wanneer dit van pas kan komen.
   - Wat is de rol van de (optionele) parameter `radix` bij deze functies?

03 – STATEMENTS

a) Schrijf een script dat de tafel van 3 in een alert (of in de console) toont.
   - Doe dit eerst met een while-lus.
   - Doe daarna hetzelfde met een for-lus.
   - Gebruik een for-in lus om de eigenschappen van je zelfgemaakte object (vorige oefeningen) te tonen.
   - Toon ook alle eigenschappen van de browser (= het object `navigator`) in de console. Bekijk deze.

b) Maak een switch-statement waarin je de bezoeker begroet op basis van zijn geslacht (mevrouw/meneer). Schrijf de uitvoer in de console.

04 – FUNCTIES

a) Schrijf vier functies die respectievelijk twee getallen optellen, aftrekken, vermenigvuldigen en delen.

b) Schrijf in de deelfunctie een controle op het getal 0 in de noemer. Toon een alert als wordt getracht te delen door nul.

c) Schrijf een functie die de parameters voorNaam en achterNaam ontvangt. Laat de functie de totale naam retourneren (met een spatie tussen voor- en achternaam).
   - Roep de functie aan, zodanig dat het resultaat in de console wordt geschreven. (`'Welkom [jouw naam]"`).

d) Schrijf een `window.onload`-functie. Zorg er voor dat de achtergrondkleur van de pagina rood wordt gekleurd bij het laden van de pagina. Zoek zelf informatie over welke eigenschap/style moet worden aangepast.

e) *Gevorderd:* Maak een functie met een variabel aantal argumenten.

- Lees de documentatie hierover in het boek (p. 171 e.v. of op MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions_and_function_scope/arguments)
- Toon in de console hoeveel, en welke argumenten werden meegegeven.

f) *Gevorderd:* Schrijf een functie die een HTML-lijst retourneert in een , op basis van alle (= variabel aantal) meegegeven argumenten. De returnwaarde moet er uitzien als

```
<div>
      <ul>
            <li>arg 1</li>
            <li>arg 2</li>
            <li>arg N </li>
      </ul>
</div>
```

g) *Gevorderd:* Bekijk het YouTube-voorbeeld over de methoden `call()` en `apply()` (http://www.youtube.com/watch?v=GY3qhaJNkLw, boek p. 187). Schrijf hiervoor zelf een – fictieve – toepassing en test de werking.

h) *Gevorderd:* Schrijf een IIFE (*iffy*) en test deze. Laat de functie bijvoorbeeld waarden in de console tonen, of het resultaat van een berekening met parameters.
- Lees het artikel over IIFE op http://benalman.com/news/2010/11/immediately-invoked-function-expression/

05 – ARRAYS

a) Schrijf een script waarin je een array maakt. Voeg aan de array items toe: namen, voertuigen, fruitsoorten, etc.

b) Toon eigenschappen v/d array in de console:
- lengte, item op positie xx
- Voeg via script element toe aan de array. Toon dit in de console

c) Converteer de array naar een string en toon deze in de console. Gebruik als scheidingsteken de komma (`,`). Oefen ook met andere scheidingstekens.

d) Doe het omgekeerde. Maak een kommagescheiden string en zet deze om naar een array.

e) Gebruik de methods `.push()` en `.pop()` om elementen aan de array toe te voegen en te verwijderen.

f) Wat doen de methods `.shift()` en `.unshift()`? Maak hiervan een voorbeeld.

g) Wat gebeurt er als je een array hebt met drie elementen en je voegt een element toe op positie `mijnArray[100]` ? Test dit en maak hiervan een voorbeeld.
- Wat is nu de lengte van de array?

h) Onderzoek de werking van `.slice()` en `.splice()`.
- Wat doen deze functies?
- Schrijf voor beide een codevoorbeeld of Fiddle.

i) Maak een array en zoek binnen de array of een bepaald element voorkomt. Gebruik hiervoor de functie `.indexOf()`.
- Wat wordt geretourneerd?
- Wat wordt geretourneerd als het gezochte element niet is gevonden?

j) Sorteer een array met `.sort()`. Toon de uitvoer Voor en Na sorteren in de console.
- Gevorderd: schrijf een eigen sorteerroutine die de elementen van de array sorteert op de door jou gewenste volgorde (aanwijzingen zie boek: p. 149)

06 – OBJECTEN

a) Maak zelf een object en geef dit leden (*members*)

- Doe dit zowel op de traditionele wijze met `new Object()` als via de JSON-notatie.
- Denk bijvoorbeeld aan vervoermiddelen, je gezin, dieren, je werk
- Maak ook een functie binnen het object die iets nuttigs doet.
- Roep de functie binnen het object aan.
- Test met het doorgeven van parameters aan de functie.
- Toon de uitvoer in de console of via een alert.
- Breid je object uit met members als arrays, en sub-objecten. Test hoe de inhoud hiervan bereikt wordt via dot-notatie of indexnotatie `[]`.
- Houdt het – voorlopig – simpel! Je hoeft nog niet te denken aan meerdere instanties van het object of overerving, etc.

b) Maak een array met (eenvoudige) objecten. Plaats in de array anonieme objecten, bijvoorbeeld als

```
[
      {name: 'Amsterdam', country: 'NL'},
      {name: 'Paris', country: 'FR'},
      {…}
];
```

- Toon in de console verschillende waarden van de inhoud van de array. Hoe wordt bijvoorbeeld de waarde Paris bereikt? En hoe de waarde NL?
- Push een nieuw object naar de array en toon de nieuwe array in de console.
- Pop een object uit de array en toon het object in de console.

07 – DOM

a) Schrijf een script dat een alinea aan de pagina toevoegt en waarin de bezoeker wordt begroet op basis van de huidige tijd: Goedemorgen, goedemiddag, goedenavond, goedenacht.

- Gebruik : het object `Date()` en de methode `.getHours()` en een lus-statement naar keuze.
- `document.createElement()`
- `document.createTextNode()`
- `document.getElementById('xxx').appendChild();`
- MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date, boek p.742)

b) Schrijf een script dat een alinea kopieert en de kopie onder het origineel plakt

- gebruik:
- `document.getElementById('xxx').cloneNode();`
- Kijk ook naar: `insertBefore();`

08 – EVENTS, EVENT HANDLERS EN HET DOM

a) Maak een pagina met een tekstvak. Vang via JavaScript de toetsindrukken binnen dat tekstvak af.

- Toon in de console, of in een element op de pagina op welke toets werd gedrukt.
- Toon in eerste instantie de keyCode (bijvoorbeeld 65, 99, enzovoort); probeer daarna je script zo aan te passen dat de daadwerkelijke toets wordt getoond (A, Z, X, enzovoort).
- Zoek eventueel op `JavaScript capturing key events` of vergelijkbaar voor meer informatie.

b) Maak een script waarbij na een klik op de knop alle links in de pagina een andere achtergrondkleur krijgen. Gebruik:

- Een array met kleuren
- `getElementsByTagName()` of `querySelectorAll()`

- Loop over een NodeList-object

c) Validatie 1 – schrijf een script waarmee je de waarden in een formulier controleert:
   - een textarea mag niet meer dan xx-tekens bevatten. Stel dit aantal zelf in.
   - Laat zien hoeveel tekens de bezoeker nog mag typen
   - Laat zien als het aantal tekens wordt overschreden.
   - de controle vindt plaats na elke geplaatste letter in de textarea

d) Validatie 2 – Tekstvalidatie
   - een inputveld moet een getal bevatten dat kleiner is dan 10.
   - deze controle vindt plaats nadat dit inputveld de focus heeft verloren.

09 – AJAX

a) Basis – maak zelf een JavaScript-ajax call naar een eigen webserver.
   - Laadt externe gegevens in je pagina, naar keuze tekst, HTML of JSON
   - Bij meerdere returnresultaten: toon ze in een lus in de pagina.

b) Vervolg – JavaScript-ajax POST-request
   - Maak een eenvoudig formulier (NAW, selectlist)
   - Stuur de gegevens naar de server; check de flow via Chrome DevTools of Firebug.
   - Optioneel: laat server de request processen. Stuur anders gewoon .txt-bevestiging retour

c) JSONP – Een Ajax-call naar een extern domein
   - Maak zelf een JavaScript-jsonp request
   - Gebruik Yindo URL, of zoek zelf een URL (weather service, Formule 1-data)
     i. http://api.yindo.com/api/book/new/10
     ii. http://www.filltext.com/
     iii. http://openweathermap.org/
   - Toon de gegevens op het scherm.

10 – MODULES

a) Neem je eigen object van cursusdag 1, breidt dit uit op verschillende manieren
   - Prototype pattern
   - Revealing module pattern
   - Revealing prototype pattern
   - Maak een herdistribueerbare module van je object en bespreek/leg uit aan een collega hoe hij wordt gebruikt.

b) Begin met een eigen DOM-library à la jQuery en zorg er voor dat er twee of drie publieke functies zijn die het volgende kunnen:
   - Een element selecteren in het DOM (`.get()`).
   - Een element verbergen (`.hide()`)
   - Een element tonen (`.show()`)
   - Maak een kleine demo van de mogelijkheden, eventueel met een knop en event handler (klikken = element verbergen, nogmaals klikken = element tonen).
   - Gevorderd: voor meer informatie over eigen libraries construeren, zie bijvoorbeeld http://net.tutsplus.com/tutorials/javascript-ajax/build-your-first-javascript-library/. Let hierin voornamelijk op de manier om meerdere elementen in het DOM te selecteren/mappen.

c) Lees meer over JavaScript design patterns bij Addy Osmani: http://addyosmani.com/resources/essentialjsdesignpatterns/book/.
   - Zoek of je de in de cursus behandelde patronen kunt terugvinden en bestudeer de codevoorbeelden (prototype pattern, etc.).
   - Pik er twee of drie patronen uit en maak hiermee een eigen voorbeeld.

d) Maak een eigen, herdistribueerbaar object, bijvoorbeeld een Stopwatch-module, of verzin zelf een object.
   - De stopwatch moet in een pagina meerdere keren te instantiëren zijn (bijvoorbeeld `var sw1 = new StopWatch(), var sw2 = new StopWatch()`, etc).
   - Elke stopwatch-instantie moet methods hebben als `start()`, `stop()`, `reset()`, `currentTime()`, en wat je verder maar handig vindt voor een stopwatch.
   - Maak meerdere knoppen in de pagina, waarmee de stopwatch-instanties gestart, gestopt, enzovoort kunnen worden. De huidige tijd van de stopwatch wordt in een result-div op de pagina getoond.

e) Maak een Parent-Child relatie tussen classes, waarbij de child-klasse alle members en methods uit de parentclass overerft.
   - Denk aan: voertuigen, gebouwen, personen, producten, etc.
   - Instantieer verschillende instanties en check of je in de child-classes zowel members als methods uit de parent-class kunt gebruiken, als uit de child-class zelf.
   - Override een methode uit de parent-class in een child-class en test hoe dit werkt.
   - Optioneel: geef zowel de parent- als de child-class een member met dezelfde naam (afgeraden!). Test in de instanties hoe dit werkt. Welke waarde heeft de member gekregen?
   - Lees: Hoofdstuk 6 JavaScript Ninja.

11 – OBJECT.OBSERVE()

Data binding is een hot item in webapps. Er zijn tal van frameworks voor ontworpen (Knockout, Ember, AngularJS). Deze binden op basis van verschillende concepten modellen (objecten) aan views. ECMAScript 6 gaat hiervoor mogelijk native ondersteuning bieden in de vorm van `Object.observe()`.

a) Lees het artikel/tutorial van Addy Osmani over O.o op www.html5rocks.com/en/tutorials/es7/observe/.

b) Maak een eigen implementatie van O.o waarbij aan de volgende requirements wordt voldaan:
   - Een app ontvangt een array met producten, de voorraad. Elk product heeft een id, een naam, prijs, een aantal (quantity) en eventuele extra properties die je zelf verzint.
   - Een customer kan producten kopen. Elke keer als een product wordt gekocht, moet het aantal beschikbare producten in de voorraad met één worden verminderd.
   - Als een product is uitverkocht, moet een melding worden getoond.
   - De winkelier kan kortingen geven op een bepaald product. Maak een routine die elk product met xx-% in prijs verlaagt. De actuele prijzen moeten worden getoond. Optioneel: maak een van/voor-routine, waarbij ook de oude prijzen worden getoond ('U bespaart EUR xx.xx!')
   - Als de voorraad wordt aangevuld, moet de status van een product van uitverkocht op beschikbaar worden gezet.
   - Als een product wordt verwijderd uit de voorraad (discontinued) wordt dit gemeld, als een nieuw product wordt toegevoegd aan de voorraad wordt dit ook gemeld.
   - Maak naar keuze een (eenvoudige) web-UI in HTML, of toon de uitvoer in de console.

c) Meer documentatie over O.o vind je eventueel op
   http://wiki.ecmascript.org/doku.php?id=harmony:observe
   http://readwrite.com/2014/07/24/object-observe-javascript-api-impact

d) Zorg voor een browserversie die O.o ondersteunt!

12 – JAVASCRIPT "EXAMEN"

Er is niet één wereldwijde examen- of certificeringsstructuur voor JavaScript. Rebecca Murphy maakte een JavaScript assesment-test.

a) Installeer (indien nodig) Node.js en Git op je machine en kloon de repository op github.com/rmurphey/js-assessment.
b) Installeer de js-assessment testsuite met `npm install`, zoals beschreven in de readme-pagina's. draai de tests op `localhost:4444`.
c) Pas de `*.js`-bestanden in `\app` aan zodat de tests passeren. Probeer 100% te scoren.
d) Modules die voor je werk niet van toepassing zijn, of over technieken gaan waar je niet mee te maken krijgt (misschien *modules.js*, of *regex.js*) mag je overslaan.

13 – MEER DESIGN PATTERNS

a) Ga naar http://www.pluralsight.com en maak een free account aan voor één maand gratis toegang.
b) Open de training JavaScript Design Patterns, op http://www.pluralsight.com/training/Courses/TableOfContents/javascript-design-patterns en bekijk de modules (met headset)
    - Timer Patterns,
    - Asynchronous Module Definitions
    - Pub/Sub Design Pattern
    - Promises
c) Maak voor elk van de patronen zelf een use case en maak een korte demo waarin het betreffende design pattern wordt toegepast. Documenteer je code en demonstreer hem aan een collega.