JavaScript

Operatoren

	naam	operatoren	
1.	**unaire** *operatoren die een bewerking uitvoeren op 1 getal*	**-** **!** **typeof**	unaire negatie-operator unaire ontkennings-operator geeft het datatype
2.	**multiplicatieve**	***** **/** **%**	vermenigvuldigen delen modulus *(restwaarde v.e. deling)*
3.	**additieve**	**+** **-**	optellen aftrekken
4.	**relationele** *twee uitdrukkingen met elkaar vergelijken*	**<** **>** **<=** **>=**	kleiner dan groter dan kleiner dan of gelijk aan groter dan of gelijk aan
5.	**gelijkheids** twee uitdrukkingen op gelijkheid testen	**===** **!==**	gelijk aan niet gelijk aan
6.	**logische** werkt met booleans	**&&** ||	en of
7.	**toekenning** *de nieuwe waarde van een variabele berekenen op basis van de vorige waarde, in één expressie* *bv. v += b => v = v + b*	**=** **+=** **-=** ***=** **/=** **%=** **++** **--**	toekenning x = x + y x = x - y x = x * y x = x / y x = x % y x = x + 1 x = x - 1

Logische operatoren && en ||

Toegepast op booleans → returnen booleans

Toegepast op andere soorten waarden → returnen één van de twee argumenten

- ✓ || (or): als de linkse waarde *true* evalueert returnt hij deze, anders de rechtse waarde

```
var input = prompt("What is your name?", "Kilgore Trout");
print("Well hello " + (input || "dear"));
```

- ✓ && (and): als de linkse waarde *false* evalueert returnt hij deze, anders de rechts waarde

```
false || alert("I'm happening!");  // De rechtse waarde wordt nooit uitgevoerd
```

Strings

- stellen tekst voor
- staan tussen enkele ‘ of dubbele “ aanhalingstekens (niet gemengd)
- voor speciale karakters moet een backslash \ staan
  - → aanhalingstekens ‘ “, backslashes \
  - → \n = newline
  - → \t = tab
- de + operator voegt strings samen i.p.v. ze op te tellen

`name.split(seperator);`	Een string opsplitsen i.e. array van strings, d.m.v. een bep. seperator
`name.slice(start, end);`	Geeft e. deel v.e. string, beginnend bij `start` en eindigend voor `end`
`name.charAt(position);`	Geeft welk karakter op een bepaalde positie van een string voorkomt
`name.indexOf(character);`	Geeft de positie van het eerste voorkomen van een bep. karakter

Variabelen

var name = value

- → loosely typed: het datatype van de variabele wordt automatisch bepaald en kan veranderen
- → casting: het omzetten van verschillende datatypes
  - ✓ String → Number: *Number()*, *parseInt()*, *parseFloat()*

Comments

Code vergezellen van extra uitleg die genegeerd wordt bij het uitvoeren van de code.

```
// Commentaar van één enkele lijn
/* Commentaar verspreid op
      verschillende lijnen */
<!-- HTML-commentaar -->
```

Functies

```
function naam(parameters) {
    return waarde;
}
```

`arguments.length;`	Geeft het aantal argumenten waarmee een functie werd opgeroepen

isNaN()

Functie die controleert of een waarde een geldig getal is

Controlestructuren

if/else if/else

Een codeblok wordt uitgevoerd als de expressie *true* evalueert.

```
if(expressie) {
   codeblok
} else if(expressie) {
   codeblok
} else {
   codeblok
}
```

? :

```
voorwaarde ? expressie1 : expressie2
```

- → `expressie1` als test true geeft
- → `expressie2` als test false geeft

while

Herhaalt een codeblok zolang aan een bepaalde voorwaarde voldaan wordt.

```
while(voorwaarde) {codeblok}
```

do..while

Variant op de while loop; voert het codeblok eenmaal uit en herhaalt dat codeblok dan indien en zolang aan een bepaalde voorwaarde voldaan wordt.

```
do {codeblok} while(voorwaarde);
```

for

Voert het codeblok uit zo lang wordt voldaan aan de test-expressie.

```
for (init-expressie; test-expressie; update-expressie) {codeblok}
```

for..in

Loopt door de properties van een object

```
for (variabele in object) {codeblok}
```

switch..case

De uitkomst van de expressie wordt vergeleken met elke *case.* Indien er een overeenkomst wordt gevonden wordt de bijhorende code uitgevoerd, anders wordt de *default* code uitgevoerd.

```
switch(expressie) {
    case n:
        codeblok
        break;
    case m:
        codeblok
        break;
    default:
        codeblok
}
```

Objects

```
var name = {
   property: value,
   property2: value,
};
```

property: waarden die bij een object horen

- → kunnen op twee manieren opgeroepen worden:
  - ✓ brackets notation *(naam v.d. property ≠ geldige variabelenaam)*: `object["property"]`
  - ✓ dot notation *(naam v.d. property = geldige variabelenaam)*: `object.property`
- → de operator `in` wordt gebruikt om een object te testen voor een bep. property

```
var chineseBox = {};
chineseBox.content = chineseBox;
"content" in chineseBox;
→ true
```

- → b.v. `length` property (automatisch bij elke string, geeft de lengte)

method: properties die functies bevatten

Arrays

```
var name = ["item one", "item two", "item three"];
var name = [];
name[0] = "item one";
```

`name.push("string");`	Een item toevoegen op het einde van de array
`name.pop("string");`	Het laatste item van de array opvragen
`name.join(seperator);`	Alle items in een array samenv. in een string, met een bep. seperator

Date

`var datum = new Date();`	Maakt een Date-object voor de huidige tijd
`var datum = new Date(year, month, day, hour, minute, second, millisecond);`	Maakt een Date-object voor de gespecifieerde tijd (alleen de eerste 3 argumenten zijn verplicht) → maanden starten op 0, dagen starten op 1 → wordt intern voorgesteld door het aantal millisec. sinds 1/1/1970
`datum.getHours();`	Haalt specifieke informatie uit een Date-object
`datum.setHours();`	Wijzigt specifieke informatie uit een Date-object
`datum.getTimezoneOffset();`	Zegt hoeveel minuten de huidige tijdzone verschilt van GMT

Math

`Math.cos`, `Math.sin`, `Math.tan`	Trigoniometrie
`Math.PI`	Geeft π
`Math.pow(getal, exponent);`	Geeft de macht van een bepaald getal
`Math.sqrt(getal, exponent);`	Geeft de vierkantswortel van een bepaald getal
`Math.min(arg1, arg2);`	Geeft de kleinste waarde van 2;
`Math.max(arg1, arg2);`	Geeft de hoogste waarde van 2;
`Math.round(getal);`	Rondt getallen af tot het dichtste gehele getal;
`Math.floor(getal);`	Rondt getallen af naar beneden;
`Math.ceil(getal);`	Rondt getallen af naar boven;

Local Storage

= eenvoudig HTML5-alternatief voor cookies

↳ werkt alleen met strings

`localStorage.setItem("name", "value");`	Een item toevoegen aan de local storage
`localStorage.getItem("name");`	Een item oproepen uit de local storage
`localStorage.removeItem("name");`	Een item verwijderen uit de local storage
`JSON.stringify(name);`	Andere datatypes omzetten naar strings
`JSON.parse(localStorage.getItem("name"));`	Items herstellen naar hun oorspronkelijke datatype

Session Storage

werkt op dezelfde manier als Local Storage, maar data gaat verloren als de sessie beëindigd wordt.

Error Handling

Programmer mistakes

= fouten in de syntax van de code, schuld van de programmeur (oplossen met validators, bv. *JSHint/JSLint*)

Run-time errors

= verkeerde input of het niet werken van een derde partij, niet de schuld van de programmeur

→ de programmeur kan wel zorgen dat de functie kan **omgaan met onverwachte input**

- ✓ een speciale waarde returnen (bv. undefined; kan echter niet bij elk soort functie)
- ✓ exception handling:
  - *try* laat je een codeblok testen voor errors
  - *catch* laat je de error verwerken als *try* er een gevonden heeft
  - *throw* laat je toe zelf error-berichten te maken *(throw an exception)*
  - *finally* laat je code uitvoeren na *try* en *catch*, ongeacht het resultaat (meestal om een functie iets te laten opkuisen op het einde, ongeacht er errors zijn)
  - custom errors maken met de *Error-constructor*:

```
var InvalidInputError = new Error("Invalid numeric input");
throw InvalidInputError;
```

DOM (Document Object Model)

= opdeling van de HTML-pagina in een boomstructuur

DOM Objects

- **Node** (document, element, attribuut, tekst of commentaar zijn nodes)
- **Document** (representeert hele HTML pagina, is de root-node van de DOM-tree)
- **Element** (elk element van het document)
  - ✓ heeft child nodes als tekst, elementen, ...
  - ✓ kan attributen bevatten
- **HTMLElement** (elk element, bevat specifieke eigenschappen van HTML-elementen)
- **Attr** (attribuutnode)
- **Anchor**, **Body**, **Form**, **Image**, **Text**, ...
- **Style** (elke HTMLElement node bevat een Style object)

Selecteren van HTML

`document.getElementById(value);`	Element selecteren via zijn id
`document.getElementsByTagName(value)[i];`	Array van elementen met een bep. tag-naam
`document.getElementsByName(value)[i];`	Array van elementen met een bep. name attribute
`document.getElementsByClassName(value)[i];`	Array van elementen met een bep. klassenaam
`document.querySelector(value);`	Element selecteren via CSS-query
`document.querySelectorAll(value)[i];`	Array van elementen via een CSS-query

Navigeren door nodes en elementen

`bepaaldElement.parentNode`	Parent node van een bepaald element
`bepaaldElement.parentElement`	Parent element (≠ tekst nodes) v.e. bepaald element
`bepaaldElement.childNodes`	Array van child nodes van een bepaald element
`bepaaldElement.children`	Array van child elements van een bepaald element
`bepaaldElement.firstChild`	Eerstvolgende child node van een bepaald element
`bepaaldElement.firstElementChild`	Eerstvolgende child element v.e. bepaald element
`bepaaldElement.lastChild`	Laatste child node v.e. bepaald element
`bepaaldElement.lastElementChild`	Laatste child element v.e. bepaald element
`bepaaldElement.nextSibling`	Node die exact na een bepaald element staat, binnen een bepaald niveau in de DOM
`bepaaldElement.nextElementSibling`	Element die exact na een bepaald element staat, binnen een bepaald niveau in de DOM
`bepaaldElement.previousSibling`	Node die exact voor een bepaald element staat, binnen een bepaald niveau in de DOM
`bepaaldElement.previousElementSibling`	Element die exact voor een bepaald element staat, binnen een bepaald niveau in de DOM

Manipuleren van HTML/CSS

Element wijzigen

`bepaaldElement.innerHTML = value;`	De inhoud van een element veranderen (bv. p, a, …)
`bepaaldElement.style.bepProperty = value;`	Een CSS property v.e. bepaald element veranderen
`bepaaldElement.bepaaldeAttribute = value;`	Een HTML attribute v.e. bepaald element veranderen
`bepaaldElement.setProperty("p", "v");`	Een HTML attribute *p* met een value *v* aan een bepaald element toevoegen

Nieuw element aanmaken

```
document.createElement("tagName");
```

Elementen toevoegen aan de pagina

`bepaaldElement.appendChild(nieuwElement)`	Een element toevoegen als laatste child van een bestaand element
`bepaaldElement.insertBefore(nieuwElement)`	Een element toevoegen voor een bestaand element
`bepElement.parentElement.insertBefore (nwElement, bepElement.nextElementSibling)`	Een element toevoegen na een bestaand element
`bepaaldElement.insertAdjacentHTML(p, text)`	Een element toevoegen als string *text* op een positie *p* ✓ 'beforebegin': voor het element ✓ 'afterbegin': voor het eerste child v.h. element ✓ 'beforeend': na het laatste child v.h. element ✓ 'afterend': na het element

Element verwijderen van de pagina

`parentNode.removeChild(bepElement)`	Een child element van een bepaald element verwijderen van de pagina

Element vervangen op de pagina

`parentNode.replaceChild(e1, e2)`	Een element vervangen door een nieuw element

DOM Events

`bepaaldElement.onclick`	Wanneer op een bep. element geklikt wordt
`bepaaldElement.ondblclick`	Wanneer op een bep. element dubbel geklikt word
`bepaaldElement.onmouseover`	Wanneer over een bep. element gehoverd wordt
`bepaaldElement.onkeydown`	Wanneer een toets wordt ingedrukt (*keyCode*)
`bepaaldElement.onkeyup`	Wanneer een toets wordt losgelaten
`bepaaldElement.onfocus`	Wanneer een element focus krijgt
`bepaaldElement.onblur`	Wanneer een element focus verliest
`bepaaldeForm.onsubmit`	Wanneer een form gesubmit wordt (*return: false* om submit te annuleren)
`bepaaldeInput.onchange`	Wanneer de inhoud v.e. invoerveld verandert
`window.onload`	Wanneer de volledige pagina ingeladen is

```
element.eventHandler = function
   → bv. window.onload = function() {}
```

Formuliervalidatie

- (Snelle) client-side controle op invoer
- Geen vervanging voor (trage) server-side validatie
- **Controle gebeurt na event** (versturen formulier, verlaten invoerveld, typen in een veld, ...)
  - ✓ submit event (wanneer het formulier verzonden wordt)
    - `return false` bij foutieve invoer (vermijdt versturen van formulier naar server)
    - `preventDefault` methode van het submit-event voorkomt het standaardgedrag (hetgeen het versturen van de formulierdata naar de server betekent)
    - `form.submit()` methode om formulier toch te verzenden bij geldige invoer
  - ✓ blur event (wanneer een invoerveld verlaten wordt)
- **Valideren van invoerwaarden**
  - ✓ value-attribuut van input-elementen
  - ✓ innerHTML van textarea
  - ✓ checked-attribuut van input met type checkbox
  - ✓ selected-attribuut van option-element
- **Valideren op**
  - ✓ al dan niet ingevuld door waarde te vergelijken met lege String
  - ✓ geldige numerieke waarde controleren, eerst controle isNaN en casting naar Number
  - ✓ geldige alfanumerieke waarde controleren via String-methodes en properties (length, substr, substring, contains, indexOf, toLowerCase, toUppercase , …)
  - ✓ via reguliere expressie

HTML5-validatie

`<input type="text">`	Gewone tekst
`<input type="button">`	Knop
`<input type="checkbox">`	Checkbox (meerdere tegelijk mogelijk)
`<input type="radio">`	Radio button (slechts één tegelijk mogelijk)
`<input type="email">`	E-mailadres
`<input type="submit">`	"Verzenden"-knop
`<input type="reset">`	Reset-knop
`<input type="url">`	URL
`<input type="hidden">`	Verborgen invoerveld
`<input type="password">`	Wachtwoord
`<input type="select">`	Drop-down keuzemenu
`<input maxlength="number">`	De maximumlengte van de invoer
`<input required>`	Veld is verplicht in te vullen
`<input pattern="regexp">`	De invoer checken d.m.v. een reguliere expressie
`<input novalidate>`	De invoer wordt niet gevalideerd bij het submitten
`<input min="number|date">`	Bepaalt de minimumwaarde voor *number*, *range*, *date*, *datetime*, *datetime-local*, *month*, *time* en *week*-velden
`<input max="number|date">`	Bepaalt de maximumwaarde voor *number*, *range*, *date*, *datetime*, *datetime-local*, *month*, *time* en *week*-velden

→ probleem: browser support voor nieuwe HTML5 input-velden
→ oplossing: fallback via JavaScript

Reguliere expressies

Manier om geavanceerde patronen in Strings op te zoeken (bv. bij validatie)

- ✓ expressie staat tussen twee slashes /
- ✓ gevolgd door optionele modifiers:
  - i geen rekening houden met hoofdletters/kleine letters
  - g alle overeenkomsten vinden i.p.v. stoppen bij de eerste
  - m overeenkomsten vinden over verschillende regels (na newline characters \n)
- ✓ controle a.d.h.v.
  - `regExp.test(string)` returnt een boolean
  - `regExp.exec(string)` returnt een array van waarden
  - `string.search(regExp)` returnt de index van de eerste match
  - `string.match(regExp)` returnt een array van waarden
  - `string.replace(regExp, str)` vervangt de resultaten v.d. expressie door een string

Syntax

Jokertekens

.	Komt overeen met elk willekeurig teken (letter, cijfer of symbool)	**goo.gle** komt overeen met **goooogle**, **goodgle**,**goo8gle**
*	Komt overeen met nul of meer van het vorige item	Het standaard vorige item is het vorige teken.**goo*gle** komt overeen met **goooogle**, **gooooogle**
+	Net als een sterretje, behalve dat een plusteken overeen moet komen met ten minste één vorig item	**gooo+gle** komt overeen met **gooooogle**, maar nooit met **google**.
?	Komt overeen met geen of één van het vorige item	**labou?r** komt overeen met zowel **labor** als **labour**
|	Hiermee kunt u een of-vergelijking maken	**a|b** komt overeen met **a** of **b**

Ankers

^	Vereist dat uw gegevens zich aan het begin van het veld bevinden	**^site** komt overeen met **site** maar niet **mijnsite**
$	Vereist dat uw gegevens zich aan het eind van het veld bevinden	**site$** komt overeen met **site** maar niet **sitescan**
^$	Letterlijke match met de hele string (start tot einde)	^site$ komt enkel overeen met site

Groeperen

()	Gebruik haakjes om een item te maken in plaats van de standaard te accepteren.	**Als(jeblieft|tublieft)** komt overeen met zowel **Alsjeblieft** als **Alstublieft**
[]	Gebruik vierkante haakjes om een lijst met items te maken waarmee een overeenkomst moet worden vastgesteld (**^** inverteert de selectie)	**[abc]** maakt een lijst met **a**, **b** en **c** erin
-	Gebruik streepjes met vierkante haakjes om uw lijst uit te breiden	**[A-Z]** maakt een lijst voor het alfabet met hoofdletters

Quantifiers

n+	Komt overeen met een string die ten minste één *n* bevat	o+ komt overeen met **google**, maar niet **ggle**.
n*	Komt overeen met een string die 0 of meer voorkomens van *n* bevat	lo* komt overeen met He**llo** wor**l**d!
n?	Komt overeen met een string die 0 of 1 voorkomen van *n* bevat	10? komt overeen met **1** en **10**
{n}	Komt overeen met exact *n* aantal voorkomens van het karakter dat ervoor staat (gebruik haakjes om strings te testen)	o{2} komt overeen met **google**, maar niet met **goooogle**.
{n, m}	Komt overeen met ten minste *n* en maximum *m* voorkomens van het karakter dat ervoor staat.	o{2,3} komt overeen met **goooogle**, maar niet met **gogle**.
{n,}	Komt overeen met ten minste *n* voorkomens van het karakter dat ervoor staat.	o{2,} komt overeen met **goooogle**, maar niet met **gogle**.
?=n	Komt overeen met een string die wordt gevolgd door een string *n*	is(?= all) komt overeen met *Is th**is** all there is*
?!n	Komt overeen met een string die niet wordt gevolgd door een string *n*	is(?! all) komt overeen met ***Is** this all there **is***

Overige

\	Hiermee verandert u een teken dat een reguliere expressie vormt in een normaal teken	mysite\.com voorkomt dat de punt wordt geïnterpreteerd als jokerteken