Opdracht 1.1 no jquery

Voordelen JQUERY

Wat: Gigantisch veel “kant-en-klare” mogelijkheden in de vorm van plug ins

Waarom is dit een voordeel: Het scheelt veel tijd, je hoeft de code alleen maar in je html te implementeren en je bent klaar

---

Wat: Het is makkelijk te gebruiken

Waarom is dit een voordeel: I.p.v. Jquery te gebruiken zou je natuurlijk ook zelf javascript kunnen schrijven. Dit is alleen veel lastiger om onder de knie te krijgen en kost hierdoor veel meer tijd.

Wat: De gigantische community rond Jquery

Waarom is dit een voordeel: Hierdoor zijn er veel tutorials, plugg-ins beschikbaar en kan je op veel websites om support vragen wanneer je iets niet begrijpt of advies nodig hebt

---

Wat: Het is cross browser

Waarom is dit een voordeel: Dit houdt in dat het script op (bijna) alle browsers werkt, je hoeft dus niet van alles in javascript te tweaken om het werkend te krijgen.

Wat: Het is een klein bestandje

Waarom is dit een voordeel: Omdat het een klein bestandje is wordt het snel ingeladen, dit scheelt natuurlijk tijd. Er zit alleen wel een grote **maar** aan.

Nadelen JQUERY

Wat: Onnodige code = slecht voor het milieu.

Waarom is dit een nadeel: In jquery staat vrijwel altijd code die niet gebruikt wordt. Deze code wordt toch geladen

---

Wat: De vele updates van Jquery

Waarom is dit een nadeel: Dit kan verwarrend zijn voor de programmeur, welke moet hij/zij nou downloaden? In een update verandert natuurlijk ook de code, dit kan voor een verandering zorgen in het gedrag van een toepassing.

Wat: Het kan zijn dat de laatste update niet goed ondersteund wordt.

Waarom is dit een nadeel: Stel je voor er is vorige week een nieuwe versie van jquery beschikbaar gesteld, dan is hier nog niet veel support voor. Wanneer je nu tegen een probleem aanloopt is er een kans dat er nog geen oplossing gevonden is voor dit probleem.

Wat: Als beginner doe je maar wat.

Waarom is dit een nadeel: Als beginner heb je natuurlijk geen idee wat de code in Jquery inhoud, dit kan ervoor zorgen dat je verkeerde keuzes maakt en dus bijvoorbeeld een slechte plugg-in installeerd.

Wat: Niet alle functionaliteiten zijn uit te voeren via jquery

Waarom is dit een nadeel: Het spreekt natuurlijk voor zich, veel is mogelijk in jquery, maar om van alle mogelijke functies gebruik te maken heb je toch echt kennis van javascript nodig.

Opdracht 1.2 functionaliteiten JQUERY

1. Jquery interacteerd met de DOM
2. Het voert Ajax requests uit
3. Het creeren van effecten (als animaties)
4. Het toevoegen van Eventlisteners
5.

Opdracht 1.3 Microlibraries

1.
2. https://github.com/ForbesLindesay/ajax
3.
4.
5.

Opdracht 2.1

```
<script>


function Persoon(name) {
	this.speak = function() {
		console.log('hallo mufpeer, mijn naam is ' + name + ' en ik ben een beetje raar
XD' );
		}
	}
var bob = new Persoon('Bob');
bob.speak();




</script>
```

Opdracht 2.2

```
<script>

function Persoon(name) {
	this.name = name;
	this.speak = function() {
		console.log('hallo mufpeer, mijn naam is ' + name + ' en ik ben een beetje raar
XD' );
		}
	}


Persoon.prototype.walk = function() {
		console.log('hallo mufpeer, mijn naam is ' + this.name + ' en ik ben een rondje
aan het lopen, ik moet aan mijn conditie werken ;)' );
		}

Persoon.prototype.eat = function() {
		console.log('hallo mufpeer, mijn naam is ' + this.name + ' en ik ben aan het eten,
laat mij met rust!' );
		}

var bob = new Persoon('Bob');
bob.eat();



</script>
```

Extra oefening

```
<script>
function Object(name) {
	this.name = name;
	this.eat = function() {
		console.log('Hallo, mijn naam is ' + this.name + ', ik eet graag computers');

	}

	this.walk = function() {
			console.log('Ik loop nu naar jouw harde schijf, om deze te laten charshen,
		muhahahahah');
	}

	this.infect = function() {
			console.log('Your computer has been infected by a ' + this.name);
	}
}

var virus = new Object('Virus');


</script>
```

Opdracht 2.3

```
<script>

var Persoon = {
	name: 'Bob',
```

```
	speak: function () {
		console.log('hallo mufpeer, mijn naam is ' + this.name + ' en ik ben een beetje raar XD' ),

	walk: function () {
		console.log('hallo mufpeer, mijn naam is ' + this.name + ' en ik ben een rondje aan het lopen, ik moet aan mijn conditie werken ;)' );
	},

	eat: function () {
		console.log('hallo mufpeer, mijn naam is ' + this.name + ' en ik ben aan het eten, laat mij met rust!' );

	}
}
</script>
```

Extra oefening

```
<script>

var virus  = {
	this.name: 'Virus',

	walk: function () {
	console.log('Ik loop nu naar jouw harde schijf, om deze te laten charshen, muhahahahah')
	},


	infect: function () {
	console.log('Your computer has been infected by a ' + this.name);
	},

	eat: function () {
	console.log('Hallo, mijn naam is ' + this.name + ', ik eet graag computers')
	}
}
</script>
```

Opdracht 3.1

```
function local() {

var iterator = 10;
var max = 10;
var min = 10;

}
```

An Iterator is an object that knows how to access items from a collection one at a time, while keeping track of its current position within that sequence. In JavaScript an iterator is an object that provides a `next()` method which returns the next item in the sequence. This method can optionally raise a `StopIteration` exception when the sequence is exhausted.

Opdracht 3.2

```
var iterator = 10;
var max = 10;
var min = 10;
```

Opdracht 3.3

Alle functies in javascript bevatten closures. Een closure kan wel variabelen etc. van buitenaf oproepen, maar van buitenaf kan niks uit de closure(in dit geval dan function) opgeroepen worden. Vandaar het woord closure.
Closure wordt gebruikt zodat je afzonderlijke regels code kan schrijven.

```
function geenMacht() {
        var max = 10;
        var min = 1;
        function welMacht() {
                if (true) { var max = 20 };
        }
}
```

Opdracht 4.1

```
// Test of GPS beschikbaar is (via geo.js) en vuur een event af
function init(){
}

// Start een interval welke op basis van REFRESH_RATE de positie updated
function _start_interval(event){
}

// Vraag de huidige positie aan geo.js, stel een callback in voor het resultaat
function _update_position(){
}

// Callback functie voor het instellen van de huidige positie, vuurt een event af
function _set_position(position){
}

// Controleer de locaties en verwijs naar een andere pagina als we op een locatie zijn
function _check_locations(event){
}

// Bereken het verchil in meters tussen twee punten
function _calculate_distance(p1, p2){
}

// GOOGLE MAPS FUNCTIES
function generate_map(myOptions, canvasId){
}

function isNumber(n) {
  return !isNaN(parseFloat(n)) && isFinite(n);
}

function update_positie(event){
        // use currentPosition to center the map
        var newPos = new google.maps.LatLng(currentPosition.coords.latitude,
currentPosition.coords.longitude);
        map.setCenter(newPos);
        currentPositionMarker.setPosition(newPos);
}
```

```
// FUNCTIES VOOR DEBUGGING

function _geo_error_handler(code, message) {
}

function debug_message(message){
}

function set_custom_debugging(debugId){
}
```

Opdracht 4.2.1 (constructor versie)

```
<script>
//Functies voor GPS

function gps() {
	this.init = function(){ console.log('YEAAAAAAAAAH, het werkt!')
	};
	this.startInterval = function(event){
	}
	this.updatePosition = function(){
	}
	this.setPosition = function(position){
	}
	this.checkLocations = function(event) {
	}
	this.calculateDistance = function(p1, p2) {
	}
}

//Functies voor Googlemaps

function googleMaps() {
	this.generateMap = function(myOptions, canvasId) {
	};
	this.isNumber = function(n) {
	};
	this.updatePositie = function(event){
```

```
        }; //checken of position kan, closure
}

//Functies voor Debugging

function debugging() {
        this.geoErrorHandler = function(code,message){
        };
        this.debugMessage = function(message) {
        };
        this.setCustomDebugging = function(debugId) {
        }

}

var GPS = new gps();
var MAPS = new googleMaps();
var DEBUG = new debugging();

</script>
```

Opdracht 4.2 (literal)

```
var gps {

        init : function () {
        },
        startInterval : function () {
        },
        updatePosition : function () {
        },
        setPosition : function () {
        },
        checkLocations : function () {
        },
        caclulateDistance : function () {
        },
```

```
}

var Googlemaps {

	generateMap : function () {
	},
	isNumber : function () {
	},
	updatePositie : function () {
	},
}

var debugging {

	geoErrorHandler : function (code, message) {
	},
	debugMessage : function (message) {
	},
	setCustomDebugging : function () {
	},
}
```

Opdracht 4.2.2

```
<script>
<script>
var SANDBOX = "SANDBOX";
var LINEAIR = "LINEAIR";
var GPS_AVAILABLE = 'GPS_AVAILABLE';
var GPS_UNAVAILABLE = 'GPS_UNAVAILABLE';
var POSITION_UPDATED = 'POSITION_UPDATED';
var REFRESH_RATE = 1000;
var currentPosition = currentPositionMarker = customDebugging = debugId = map = interval
=intervalCounter = updateMap = false;
var locatieRij = markerRij = [];
var GPS = new gps();
var MAPS = new googleMaps();
var DEBUG = new debugging();
```

```
// Event functies - bron: http://www.nczonline.net/blog/2010/03/09/custom-events-in-javascript/
Copyright (c) 2010 Nicholas C. Zakas. All rights reserved. MIT License
// Gebruik: ET.addListener('foo', handleEvent); ET.fire('event_name'); ET.removeListener('foo',
handleEvent);
function EventTarget(){this._listeners={}}
EventTarget.prototype={constructor:EventTarget,addListener:function(a,c){"undefined"==typeof
this._listeners[a]&&(this._listeners[a]=[]);this._listeners[a].push(c)},fire:function(a){"string"==type
of a&&(a={type:a});a.target||(a.target=this);if(!a.type)throw Error("Event object missing 'type'
property.");if(this._listeners[a.type]instanceof Array)for(var
c=this._listeners[a.type],b=0,d=c.length;b<d;b++)c[b].call(this,a)},removeListener:function(a,c){if
(this._listeners[a]instanceof Array)for(var b=
this._listeners[a],d=0,e=b.length;d<e;d++)if(b[d]===c){b.splice(d,1);break}}}; var ET = new
EventTarget();


//Functies voor GPS

function gps() {
		this.init = function(){
		debug_message("Controleer of GPS beschikbaar is...");

		ET.addListener(GPS_AVAILABLE, _start_interval);
		ET.addListener(GPS_UNAVAILABLE, function(){debug_message('GPS is niet
beschikbaar.')});

		(geo_position_js.init())?ET.fire(GPS_AVAILABLE):ET.fire(GPS_UNAVAILABLE);
	};
	this.startInterval = function(event){
			debug_message("GPS is beschikbaar, vraag positie.");
		_update_position();
		interval = self.setInterval(_update_position, REFRESH_RATE);
		ET.addListener(POSITION_UPDATED, _check_locations);

	}
	this.updatePosition = function(){
		intervalCounter++;
		geo_position_js.getCurrentPosition(_set_position, _geo_error_handler,
{enableHighAccuracy:true});
	}
	this.setPosition = function(position){
		currentPosition = position;
		ET.fire("POSITION_UPDATED");
```

```
                debug_message(intervalCounter+" positie lat:"+position.coords.latitude+"
long:"+position.coords.longitude);
        }
        this.checkLocations = function(event) {
                // Liefst buiten google maps om... maar helaas, ze hebben alle coole functies
                for (var i = 0; i < locaties.length; i++) {
                        var locatie = {coords:{latitude: locaties[i][3],longitude: locaties[i][4]}};

                        if(_calculate_distance(locatie, currentPosition)<locaties[i][2]){

                                // Controle of we NU op die locatie zijn, zo niet gaan we naar de
betreffende page
                                if(window.location!=locaties[i][1] &&
localStorage[locaties[i][0]]=="false"){
                                        // Probeer local storage, als die bestaat incrementeer de
locatie
                                        try {

(localStorage[locaties[i][0]]=="false")?localStorage[locaties[i][0]]=1:localStorage[locaties[i][0]]++;
                                        } catch(error) {
                                                debug_message("Localstorage kan niet
aangesproken worden: "+error);
                                        }

        // TODO: Animeer de betreffende marker

                                        window.location = locaties[i][1];
                                        debug_message("Speler is binnen een straal van "+
locaties[i][2] +" meter van "+locaties[i][0]);
                                }
                        }
                }
        }
        this.calculateDistance = function(p1, p2) {
                var pos1 = new google.maps.LatLng(p1.coords.latitude, p1.coords.longitude);
                var pos2 = new google.maps.LatLng(p2.coords.latitude, p2.coords.longitude);
                return
Math.round(google.maps.geometry.spherical.computeDistanceBetween(pos1, pos2), 0);
        }
}

//Functies voor Googlemaps
```

```
function googleMaps() {
		this.generateMap = function(myOptions, canvasId) {
	// TODO: Kan ik hier asynchroon nog de google maps api aanroepen? dit scheelt calls
		debug_message("Genereer een Google Maps kaart en toon deze in #"+canvasId)
		map = new google.maps.Map(document.getElementById(canvasId), myOptions);

		var routeList = [];
		// Voeg de markers toe aan de map afhankelijk van het tourtype
		debug_message("Locaties intekenen, tourtype is: "+tourType);
		for (var i = 0; i < locaties.length; i++) {

			// Met kudos aan Tomas Harkema, probeer local storage, als het bestaat,
voeg de locaties toe
			try {

(localStorage.visited==undefined||isNumber(localStorage.visited))?localStorage[locaties[i][0]]=fal
se:null;
			} catch (error) {
				debug_message("Localstorage kan niet aangesproken worden:
"+error);
			}

			var markerLatLng = new google.maps.LatLng(locaties[i][3], locaties[i][4]);
			routeList.push(markerLatLng);

			markerRij[i] = {};
			for (var attr in locatieMarker) {
				markerRij[i][attr] = locatieMarker[attr];
			}
			markerRij[i].scale = locaties[i][2]/3;

			var marker = new google.maps.Marker({
				position: markerLatLng,
				map: map,
				icon: markerRij[i],
				title: locaties[i][0]
			});
		}
	// TODO: Kleur aanpassen op het huidige punt van de tour
		if(tourType == LINEAIR){
			// Trek lijnen tussen de punten
			debug_message("Route intekenen");
			var route = new google.maps.Polyline({
```

```
                        clickable: false,
                        map: map,
                        path: routeList,
                        strokeColor: 'Black',
                        strokeOpacity: .6,
                        strokeWeight: 3
                    });

                }

                // Voeg de locatie van de persoon door
                currentPositionMarker = new google.maps.Marker({
                    position: kaartOpties.center,
                    map: map,
                    icon: positieMarker,
                    title: 'U bevindt zich hier'
                });

                // Zorg dat de kaart geupdated wordt als het POSITION_UPDATED event
afgevuurd wordt
                ET.addListener(POSITION_UPDATED, update_positie);
                this.isNumber = function(n) {
                return !isNaN(parseFloat(n)) && isFinite(n);
                };
        this.updatePositie = function(event){
                // use currentPosition to center the map
                var newPos = new google.maps.LatLng(currentPosition.coords.latitude,
currentPosition.coords.longitude);
                map.setCenter(newPos);
                currentPositionMarker.setPosition(newPos);
        } //checken of position kan, closure

        }
        }

//Functies voor Debugging

function debugging() {
        this.geoErrorHandler = function(code,message){
                debug_message('geo.js error '+code+': '+message);
        };
        this.debugMessage = function(message) {
                (customDebugging &&
```

```
debugId)?document.getElementById(debugId).innerHTML:console.log(message);
	};
	this.setCustomDebugging = function(debugId) {
		debugId = this.debugId;
		customDebugging = true;
	}

	}

</script>
```

Object literals are usually the way to go. They only need to be parsed when loading the script, which can introduce various optimizations by the scripting engine.

Constructors need to be executed. That means they will be slower, but you can easily add some validation code etc. to them, and they allow the construction of complex objects with public, privileged methods and private "attributes" hidden in the constructors scope. Also, they of course construct objects that share a prototype, which you might find useful.