***Hoofdstuk 4***

- In value.color is color de **property van value.** En binnen een array, bijv. array[0], krijg je de property op plek 0. Dit is dus niet letterlijk en de array moet eerst gelezen worden.

- In sequence.push(3) is **push de method van de string.**

- Het is mogelijk om binnen een object een property een waarde te geven / overschrijven door middel van:

Value.wolf = “hoi” (werkt ook als het nog niet bestaat)

- Object.keys geeft een array terug met daarin alle property names.

- Object.assign kopieert alle properties in een ander object (en past aan waar aangegeven):

let objectA = {a: 1, b: 2};

Object.assign(objectA, {b: 3, c: 4});

console.log(objectA);

// → {a: 1, b: 3, c: 4}

- const score = {visitors: 0, home: 0};

// This is okay

score.visitors = 1;

// This isn't allowed

score = {visitors: 1, home: 1};

- Wanneer je geen property definieert, wordt deze vanzelf gekopieerd van de naam:

let journal = [];

function addEntry(events, squirrel) {

journal.push({events, squirrel});

}

- Een andere manier van loopen:

for (let entry of JOURNAL) {

console.log(`${entry.events.length} events.`);

}

- Bij todoList.shift() pak je het eerste item van de array. Bij todoList.unshift() voeg je een array item toe en wordt dit het eerste item.

- console.log([1, 2, 3, 2, 1].indexOf(2));

// → 1

console.log([1, 2, 3, 2, 1].lastIndexOf(2));

// → 3

- Bij slice wordt een deel van array tussen twee waardes gepakt. Het eerste nummer is inclusief, en het laatste nummer exclusief:

console.log([0, 1, 2, 3, 4].slice(2, 4));

// → [2, 3]

console.log([0, 1, 2, 3, 4].slice(2));

// → [2, 3, 4]

Wanneer je het laaste getal weghaald, kopieer je de gehele array vanaf de begin index die je hebt aangegeven.

Een mooi stukje code:

function remove(array, index) {

return array.slice(0, index)

.concat(array.slice(index + 1));

}

console.log(remove(["a", "b", "c", "d", "e"], 2));

// → ["a", "b", "d", "e"]