Web Development II

Hoofdstuk 06 – Web Storage

HO GENT

Web Storage

- Cookies
- WS Standaard
- WS API attributen en functies
- Voorbeeld Sticky notes
- JSON Object Storage
- WS API Event

HO GENT

06 Web Storage

Cookies

HO
GENT

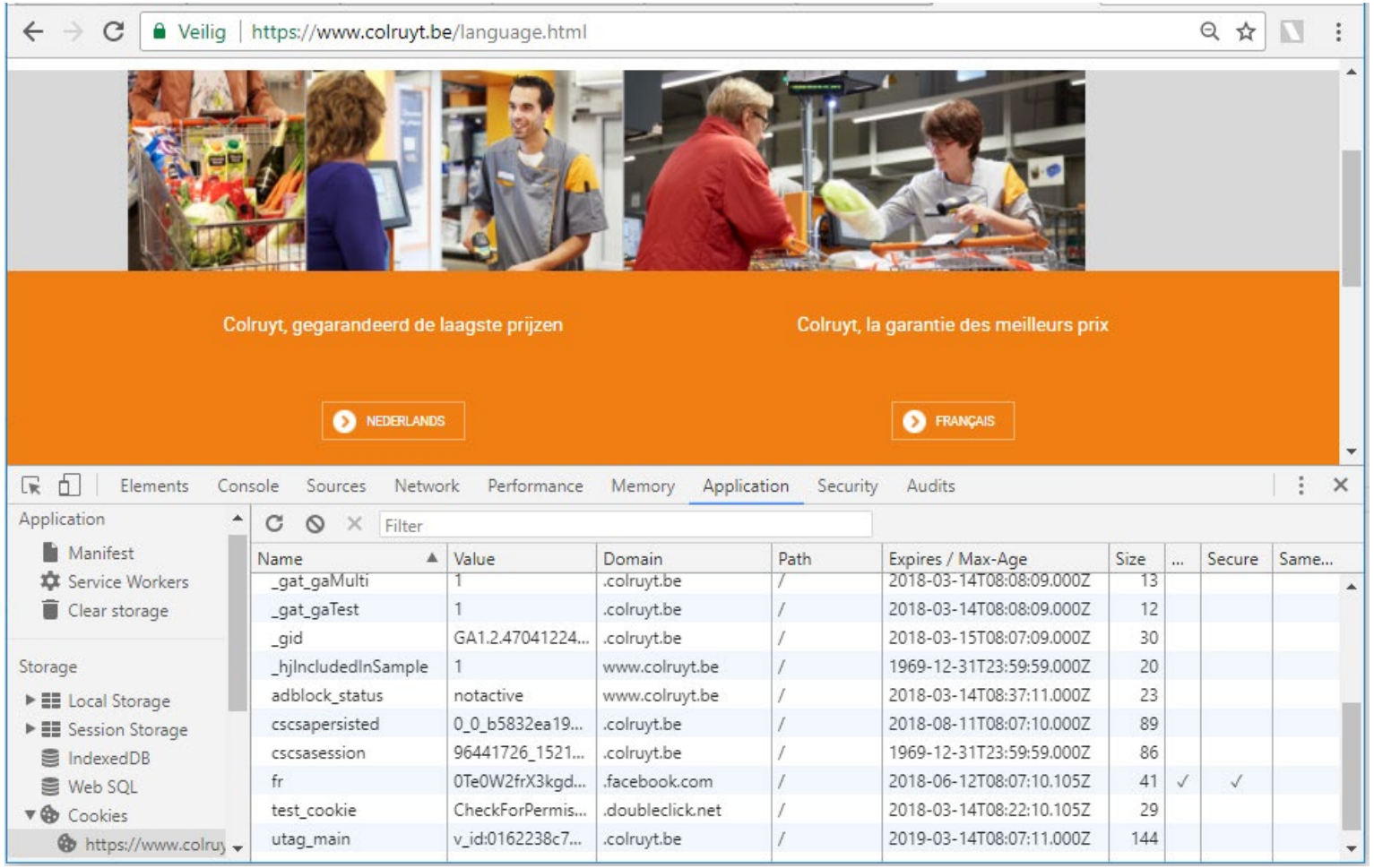
Cookies

- Cookies
  - Ingebouwde manier om kleine tekstbestanden heen en weer te sturen tussen de browser en de server
  - Servers kunnen gebruik maken van de inhoud van deze cookies om informatie over de gebruiker bij te houden over verschillende webpagina's heen.
    Denk aan de taal die de gebruiker op een website kiest, dit kan perfect worden opgeslagen in een cookie.
    De volgende keer dat de gebruiker de website bezoekt, wordt de pagina overgeslagen waarbij de gebruiker om de gewenste taal gevraagd wordt
  - Hoe kan je cookies bekijken?
    - Open de Chrome Developer Tools
    - Application
    - Onder Storage vind je Cookies

HO GENT

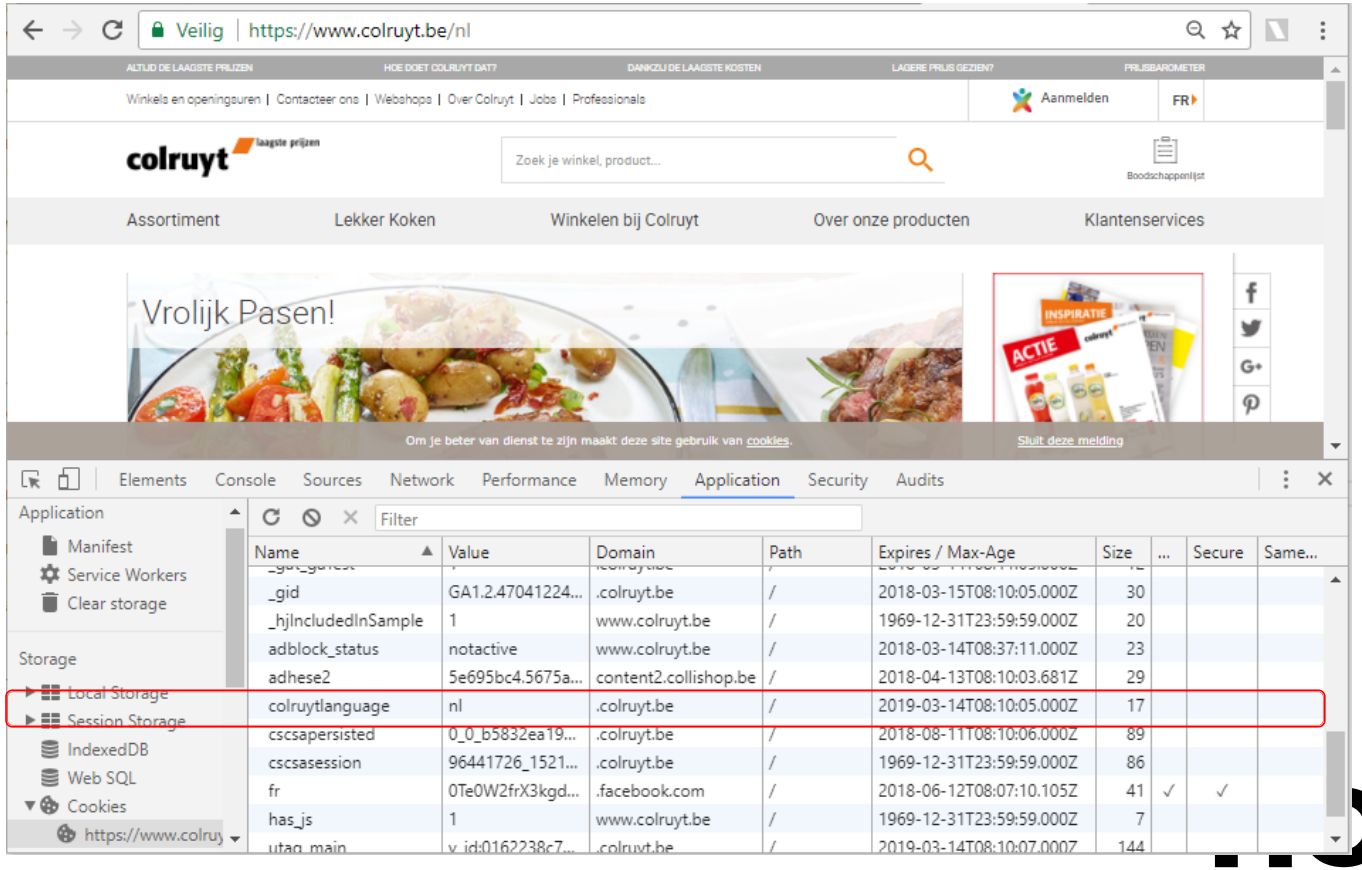
Cookies

- 1ste bezoek aan www.colruyt.be => er is geen cookie voor de taal



Cookies

- Nadat je op Nederlands geklikt hebt: er is een cookie voor de taal. De volgende keer dat je de website bezoekt, krijg je het taalkeuze – scherm niet meer te zien.



Cookies

- Er zijn een aantal problemen verbonden aan het gebruik van cookies
  - De maximumgrootte is 4 KB => er kunnen dus geen grote ‘dingen’, zoals bestanden, in opgeslagen worden
  - Elke cookie wordt heen en weer gestuurd voor elk HTTP request
  - Cookies kunnen Data Leakage veroorzaken.
    Cookies kunnen bestaan over de verschillende subdomeinen van één website heen, bijvoorbeeld een cookie kan zichtbaar zijn voor app.test.com en images.test.com. Op dezelfde manier is het mogelijk om een cookie zichtbaar te maken voor bijvoorbeeld álle hosts van ac.be (www.vub.ac.be, www.ua.ac.be, www.ulb.ac.be)

HO GENT

Cookies

```
// In JavaScript wordt een cookie als volgt gecreëerd
document.cookie = 'username=John Doe';

// Je kan ook een datum toevoegen wanneer een cookie
// moet verstrijken
// Een cookie wordt by default verwijderd als de browser wordt gesloten
// Dit kan je ook zien.
document.cookie = 'adblock_status=noactive; expires=Thu, 18 Jul 2019 12:00:00 UTC';

// Met de parameter path, kan je het path opgeven waartoe de
// cookie behoort
// Het path bepaalt op welke pagina's de cookie overal beschikbaar is
// By default, behoort de cookie tot de huidige webpagina
// Met path=/ kan de cookie voor de ganse website gelezen worden
document.cookie = 'language=NL; expires=Thu, 18 Jul 2019 12:00:00 UTC; path=/';
```


HO GENT

Cookies

```
// Een cookie veranderen kan als volgt
document.cookie = 'language=FR; expires=Thu, 18 Jul 2019 12:00:00 UTC; path=/';

// Een cookie verwijderen, kan door de expires date vóór nu te plaatsen
// document.cookie = 'language=FR; expires=Thu, 18 Dec 2018 12:00:00 UTC; path=/';

// Alle cookies lezen
const c = document.cookie;
console.log(c);
// username=John Doe; adblock_status=noactive; language=FR

c.split("; ").forEach((part) => {
    const arrPart = part.split("=");
    console.log(`${arrPart[0]} --> ${arrPart[1]}`);
})
// username --> John Doe
// adblock_status --> noactive
// language --> FR
```


HO GENT

Cookies

- Met de ontwikkeling van HTML5, was er een grote vraag om lokaal (= op de client) data te kunnen stockeren.
- Momenteel zijn er 4 types, eigenlijk 3 als je Local Storage en Session Storage samenneemt
  - (1) Web Storage = Local Storage en Session Storage
  - (2) Web SQL Storage: wel in Chrome, niet meer ondersteund in FF en Edge, support bij andere browsers is niet gegarandeerd in de toekomst
  - (3) IndexedDB = JavaScript-based object-oriented database

Controleer de ondersteuning op https://caniuse.com/

HO GENT

06 Web Storage

Web Storage: String in Storage

HO GENT

Web Storage standaard

- https://html.spec.whatwg.org/multipage/webstorage.html
- Web Storage
  - Dit wordt tegenwoordig vaak gebruikt ipv cookies (= upgrade voor cookies)
    - Maximum grootte = 5 MB (sommige browsers vergroten zelf de capaciteit indien vol of vragen om de capaciteit te verhogen)
    - Wordt niet meegestuurd met elke HTTP Request
    - Goed ondersteund door de huidige browsers
  - We maken een onderscheid tussen
    - localStorage: beschikbaar over meerdere vensters van eenzelfde domein (website) + persistent (= blijft bestaan nadat de browser wordt herstart)
    - sessionStorage: is gekoppeld aan het venster waarin de website geopend is (=> in 2 vensters dezelfde url geopend => verschillende sessionStorage) + niet persistent (= als het venster wordt gesloten, verdwijnt de inhoud)
  - Implementeren dezelfde interface Storage

HO GENT

Web Storage standaard

- Web Storage
  - Houdt de data bij onder de vorm van **sleutel – waarde** (key – value) paren

Key	Value
id	L001
voornaam	Ilse
familienaam	Van Acker

  - Er kunnen enkel **strings** opgeslagen worden. We zullen hiermee rekening moeten houden als we objecten of arrays van objecten willen opslaan in de storage!
  - De inhoud kan door de gebruiker bekeken, gewijzigd en verwijderd worden [Google Chrome Developer Tools > Application > Storage > Local Storage / Session Storage]
  - Elk domein (site) heeft een eigen storage
  - Vanuit .js zijn beide toegankelijk via window object:
    - window.localStorage
    - window.sessionStorage

HO GENT

Web Storage API attributen en functies

```
interface Storage {
  readonly attribute unsigned long length;
  DOMString? key(unsigned long index);
  getter DOMString getItem(DOMString key);
  setter creator void setItem(DOMString key, DOMString value);
  deleter void removeItem(DOMString key);
  void clear();
};
```

- length: het aantal key – value paren opgeslagen in het storage object
- key(index): geeft de key op een bepaalde positie.
  De index van de keys gaat van 0 tot length-1.
  Dit geeft null terug als index >= length
- getItem(key): om een value op te vragen bij een bepaalde key. Er wordt null teruggegeven als de opgegeven key niet bestaat.
- setItem(key, value): om key – value paren in te stellen. Je krijgt een QUOTA_EXCEEDED_ERR als storage uit staat voor deze site of als de storage helemaal vol is.
- removeItem(key): verwijdert een key – value paar als de opgegeven key aanwezig is. Als deze niet aanwezig is gebeurt er niets.
- clear(): verwijdert alle key – value paren uit de storage lijst.

HO GENT

Web Storage API attributen en functies

- Session/localStorage kan je ook behandelen als associatieve array (object literal)

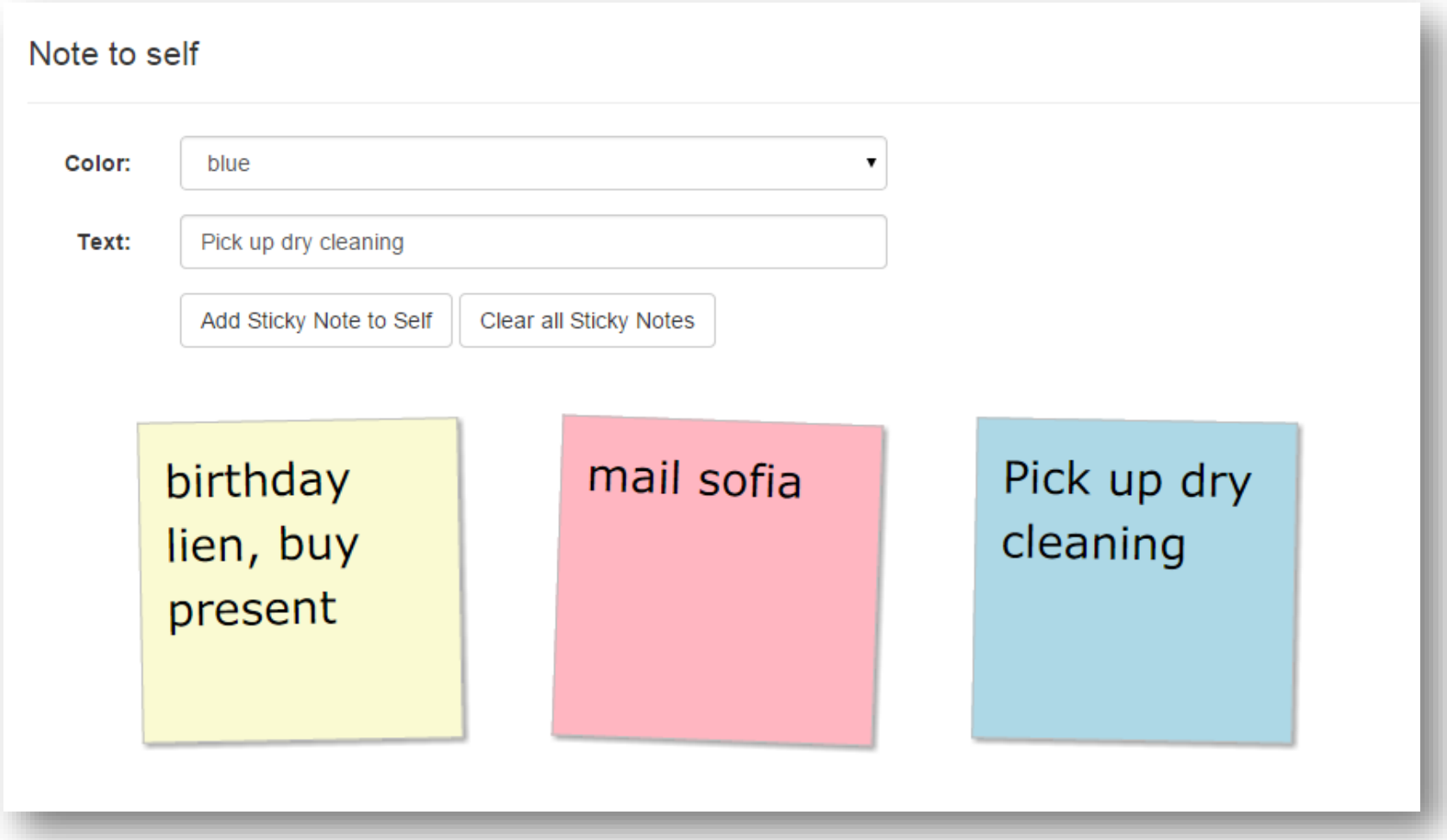
```
//toevoegen of wijzigen
localStorage.setItem("sticky_0", "Pick up dry cleaning");
localStorage["sticky_0"] = "Pick up dry cleaning";
localStorage.sticky_0 = "Pick up dry cleaning";

//opvragen
const sticky= localStorage.getItem("sticky_0");
const sticky= localStorage["sticky_0"];
const sticky= localStorage.sticky_0;
```


HO GENT

Voorbeeld

- Sticky notes



HO GENT

Voorbeeld

- Bekijk index.html en css
  - Ga naar noteToSelfV1.js
    - Vergeet niet: pas eerst index.html aan

```
<script src="scripts/notetoselfV1.js"></script>
```

    - De **init** fucntie wordt opgeroepen als de pagina klaar is. Deze initializeert de **StickyComponent** wat alle logica bevat.
    - De functie **initializeEventHandlers** controleert of de browser storage ondersteunt en definieert eventhandelingen: wat er gebeurt bij een klik op de knop add of clear.
    - Om te checken welke browsers storage ondersteunen, kan je ook kijken op http://caniuse.com/#search=web%20storage

HO GENT

Voorbeeld

```
26      #initializeEventHandlers() {
27        const addButton = document.getElementById('add');
28        const clearButton = document.getElementById('clear');
29        const noteText = document.getElementById('notetext');
30
31        if (!this.#storage) {
32          alert('browser ondersteunt geen storage');
33          addButton.disabled = true;
34          clearButton.disabled = true;
35          return;
36        }
37        addButton.onclick = () => {
38          this.#addSticky(noteText.value);
39          noteText.value = '';
40          this.#toHTML();
41        };
42        clearButton.onclick = () => {
43          this.#clear();
44        };
45
46      }
```


HO GENT

Voorbeeld

- Merk op dat we voor onclick gebruik maken van de arrow functies. We schrijven

```
addButton.onclick = () => {
    this.#addSticky(noteText.value);
    noteText.value = '';
    this.#toHTML();
};
```

- In plaats van

```
addButton.onclick = function() {
    this.#addSticky(noteText.value);
    noteText.value = '';
    this.#toHTML();
};
```

- In het laatste geval verwijst het this keyword binnen een event handler naar het object dat het event heeft getriggered (de button). In het eerste geval (mogelijk sinds ES6) wordt this opgezocht in de scope zoals elke andere variabele. Dit is meestal gemakkelijker.

HO GENT

Voorbeeld

- In noteToSelfV1.js
  - De class StickiesComponent bevat alle data en acties die op de pagina plaatsvinden. De class bevat volgende properties en methodes :
    - storage: verwijst naar localStorage of sessionStorage (die meegegeven wordt)
    - #addSticky: maakt een nieuwe sticky aan en plaatst die in de storage
    - #toHtml: overloopt alle stickies in storage en toont deze in een alert.

```
#storage;
constructor(storage) {
        this.#storage = storage;
}
get storage() {
        return this.#storage;
}
#addSticky(note) {
        const key = 'sticky_' + Math.random().toString(36).substring(2);
        //unieke sleutel
        this.#storage.setItem(key, note);
}
```


HO GENT

Voorbeeld

- Class StickiesComponent
  - Methode toHtml : overloopt alle stickies in storage en geeft ze weer via een alert

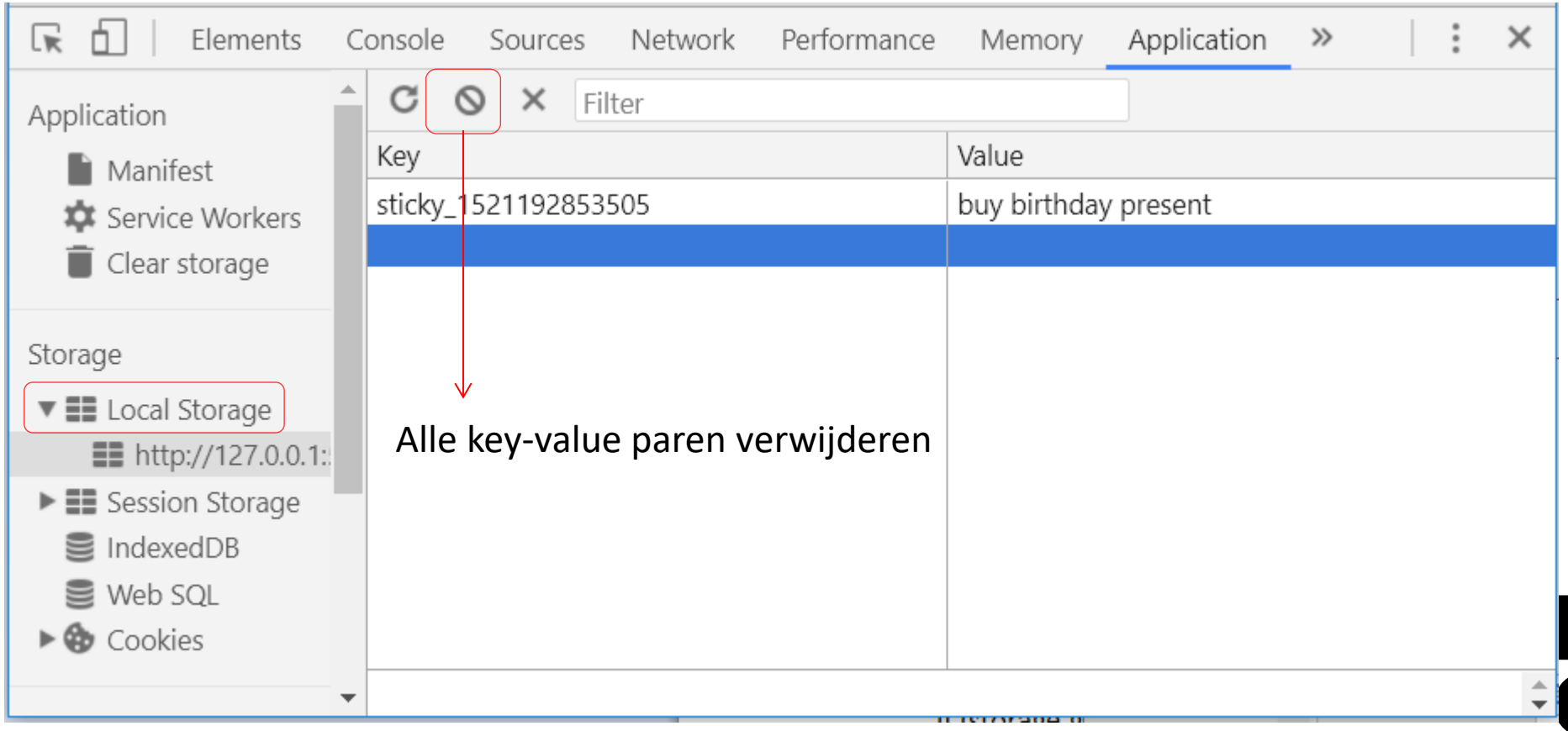
```
#toHTML() {
    const allStickies = Object.entries(this.#storage).reduce(
      (result, [key, value]) => (result + `${key}: ${value}\n`),
      ''
    );
    alert(allStickies);
  }
```

Meer info over Object.entries: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/entries

HO GENT

Voorbeeld

- Run de applicatie
  - Je kan de storage bekijken via Google Chrome Developer Tools > Application > Storage > Local Storage / Session Storage



HO GENT

Voorbeeld : session <-> localStorage

- sessionStorage <-> localStorage
  - sessionStorage : is gekoppeld aan het venster waarin de website geopend is + niet persistent
    - Open de website in ander venster => verschillende sessionStorage
    - Sluit de browser af en open opnieuw => sessionStorage is leeg
  - localStorage: beschikbaar over meerdere vensters van eenzelfde domein (website) + persistent (= blijft bestaan nadat de browser wordt herstart)
    - Pas storage property aan bij de aanroep in init

```
const init = function() {
    new StickiesComponent(sessionStorage);
    ....
```

- Open de website in een ander venster
- Sluit de browser ook eens af

HO GENT

06 Web Storage

JSON

HO
GENT

JSON

- Er kunnen enkel Strings opgeslagen worden in de storage. Dit betekent bijvoorbeeld dat een object zal opgeslagen worden als [object Object].
- Wil je een object of array opslaan, dan moet je het eerst converteren naar JSON formaat
- JSON staat voor JavaScript Object Notation en is een deelverzameling van de programmeertaal JavaScript.
- Opslag beelden : converteren naar data_url
  - Meer op https://hacks.mozilla.org/2012/02/saving-images-and-files-in-localstorage/

- **JSON** is een lichtgewicht data-uitwisselingsformaat. Het is gemakkelijk voor mensen om te lezen en te schrijven. Het is gemakkelijk voor machines om het te analyseren en te genereren. Het is gebaseerd op een subset van de JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is een tekstformaat dat volledig taalonafhankelijk is.
- http://JSON.org

HO GENT

JSON

- JSON's basic types zijn:
  - Number (double precision floating-point format)
  - String (double-quoted Unicode met backslash escaping)
  - Boolean (true or false)
  - Array (Een geordende lijst van waarden, gescheiden door komma’s en omsloten door vierkante haken ([]). De waarden moeten niet noodzakelijk van hetzelfde type zijn.)
  - Object (een ongeordende verzameling van key:value paren, gescheiden door een komma en omsloten door accolades ({}); de key moet een string zijn)
  - Null

HO GENT

JSON

```
{
        "firstName": "John",
        "lastName": "Smith",
        "age": 25,
        "student": true,
        "address": {
                "streetAddress": "21 2nd Street",
                "city": "New York",
                "state": "NY",
                "postalCode": "10021"
        },
        "phoneNumber": [
                {"type": "home", "number": "212 555-1234"},
                {"type": "fax", "number": "646 555-4567" }
        ]
}
```

Dit is het object in JSON dat informatie bevat over een persoon. Het object heeft string fields voor first name en last name, een getal voor age, een boolean voor student, bevat een object voor het address van de persoon, en een lijst(een array) van telefoonnummer objecten.

HO GENT

06 Web Storage

Web Storage: Object in Storage

HO
GENT

Class declaration

- Ga naar noteToSelfV2.js
  - vergeet niet: pas eerst index.html aan

```
<script src="scripts/notetoselfV2.js"></script>
```

- Maak de storage leeg, anders krijg je problemen

HO GENT

JSON

- Hoe objecten opslaan in webstorage?
- Het object wordt omgezet naar een JSON object
  - JSON.stringify(): zet een object om naar string
  - JSON.parse(): zet een string om naar object literal

```
const data;
// Object in storage plaatsen
function saveData() {
        sessionStorage.setItem("myStorageKey“, JSON.stringify(data));
}
```


```
// Object ophalen uit storage
function loadData() {
        data = JSON.parse(sessionStorage.getItem("myStorageKey"))
}
```

HO GENT

Voorbeeld

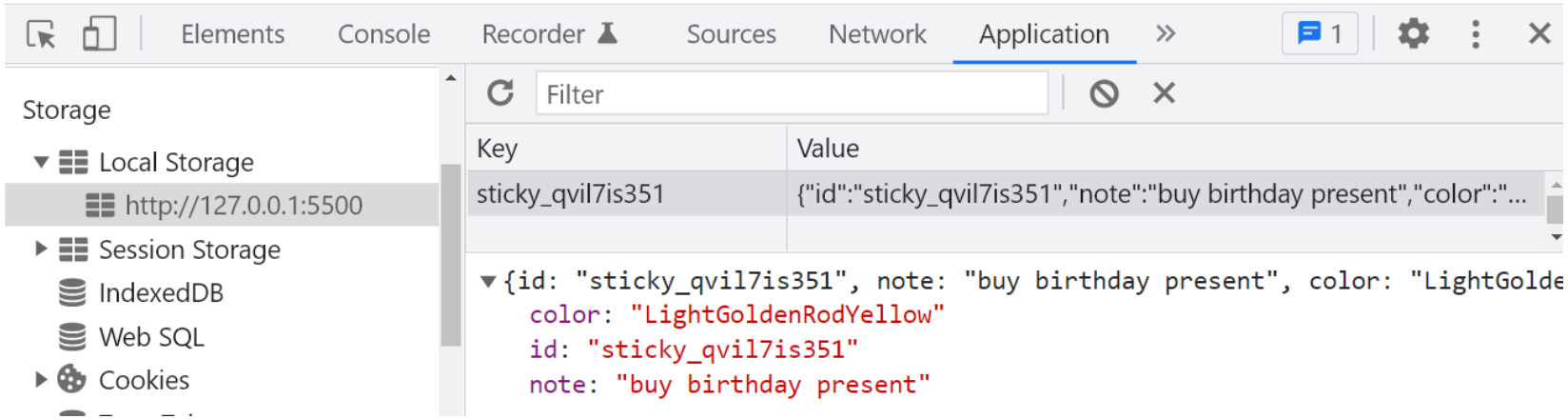
- Ga naar noteToSelfV2.js
- Maak de klasse **Sticky** aan met de volgende eigenschappen:
  - **id** (private)
  - **note** (getter + setter)
  - **color** (getter + setter)
  - **constructor** die zelf de key bepaald (zie v1) en de andere nodige waarden zet.
  - methode **toJSON** om het object om te zetten naar een object literal:

  Omdat de **stringify** methode ook bij ES6 classes zou werken, moeten we een methode **toJSON** toevoegen die bepaalt welke properties worden gebruikt in de JSON string (= het resultaat van de stringify methode).

HO GENT

Voorbeeld

- Class **StickiesComponent**
- Vervolledig de methode **addSticky(note,color)**:
  - Maak op basis van de parameters een nieuw **Sticky object.**
  - Seraliseer het **Sticky-object** naar **JSON** en sla het op in **localStorage**. Gebruik het **id** van de **Sticky** als sleutel.
    - Tip: `JSON.stringify(data)`



Voorbeeld

- Class **StickiesComponent**
- Vervolledig de **#toHtml** method (doet net het omgekeerde)
  - Tip: Converteert object literal naar object van class Sticky en gebruik deze.

HO GENT

06 Web Storage

Web Storage: Array van objecten in Storage

HO GENT

Class declaration

- Ga naar noteToSelfV3.js
  - vergeet niet: pas eerst index.html aan

```
<script src="scripts/notetoselfV3.js"></script>
```

- Maak de storage leeg, anders krijg je problemen
- De methode **#toHTML** is gegeven. Dit maakt de notes aan in de HTML pagina. De details komen aan bod in volgend hoofdstuk.

- Wat als er ook andere informatie in de storage wordt opgeslagen en je wenst enkel de stickies op te halen. Daarvoor zullen we gebruik maken van een array van stickies en niet meer de stickies elk apart opslaan in de storage.
- We willen ook één sticky kunnen verwijderen en alle stickies samen kunnen verwijderen.

HO GENT

Voorbeeld

- De **StickiesComponent** is veranderd. Er wordt een array stickies toegevoegd.

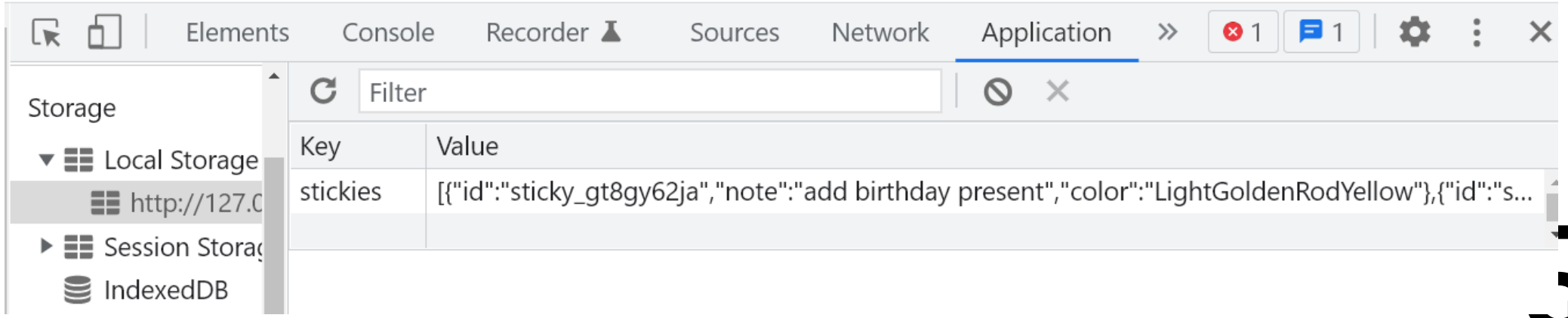
```
37  class StickiesComponent {
38    #storage;
39    #stickies = [];
40    constructor(storage) {
41      this.#storage = storage;
42      this.#initializeEventHandlers();
43      this.#getStickiesFromStorage();
44      this.#toHTML();
45    }
46    get storage() {
47      return this.#storage;
48    }
49    get stickies() {
50      return this.#stickies;
51    }
52    // ...
```


HO GENT

Voorbeeld

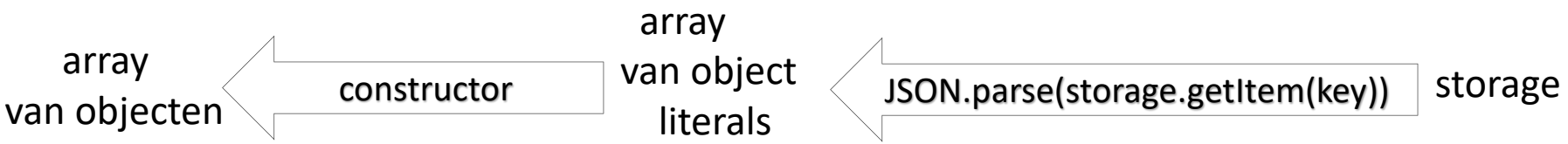
- Plaatsen van array in storage (toevoegen of overschrijven)
- Er kunnen enkel strings opgeslagen worden in de storage. Hiermee moet rekening gehouden worden als we een array van objecten willen opslaan in de storage


```
this.#storage.setItem('stickies', JSON.stringify(this.#stickies));
```



Voorbeeld

- Ophalen van array uit storage




```
74  #getStickiesFromStorage() {
75    //
76
77    this.#stickies = [];
78    if (this.#storage.getItem('stickies')) {
79      this.#stickies = JSON.parse(this.#storage.getItem('stickies')).map(
80        (s) => {
81          return new Sticky(s.note, s.color);
82        }
83      );
84    }
85  }
```


HO GENT

Voorbeeld

- Vervolledig de volgende methoden in de **StickyComponent**
- **#addSticky**(note,color)
  - Maak een nieuw **Sticky** object aan.
  - Voeg het object toe aan de **stickies.**
  - Roep de methoden #setStickiesInStorage en #toHTML op.
- **#clearStickies**()
  - Maak de array van **stickies** leeg,
  - Verwijder in localStorage het **stickies** element
  - Rerender de component door **toHtml** op te roepen
- **#deleteSticky**(key)
  - Verwijder het element met de gegeven key
  - Roep **setStickiesInStorage** op
  - Rerender de component door **toHtml** op te roepen
    - In de toHtml is de eventhandler reeds voorzien.

HO GENT

Voorbeeld

- Vervolledig de volgende methoden in de **StickyComponent**
- **#setStickiesInStorage**
  - Gebruik de **JSON.stringify** om de **stickies** in de storage te plaatsen met de sleutel **stickies**

HO GENT

setItem

- als de opslag vol is wordt er een **DOMException** met de naam **QuotaExceededError** geworpen.

```
setStickiesInStorage() {
 try {
   this.#storage.setItem('stickies', JSON.stringify(this.#stickies));
 }
 catch (e) {
   if (e.name === 'QuotaExceededError')
      alert('Kan sticky niet bewaren, opslag is vol!');
 }
}
```


HO GENT