



Saptamana 13

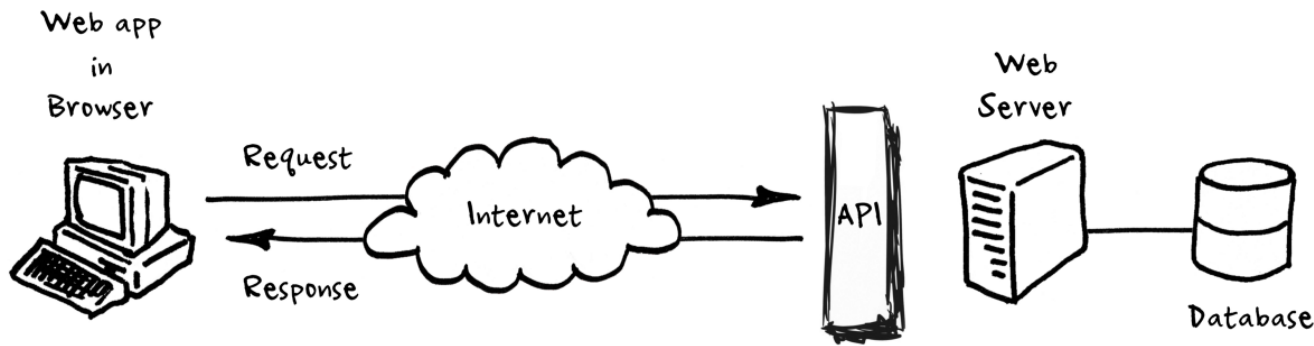
Partea 2

Programare Front-End

1. APIs

What's an API ?

- **Application Programming Interface**
- ofera aplicatiilor o modalitate de a “comunica” intre ele
- in cazul aplicatiilor web, API-ul reprezinta punctul de acces extern al unui server; un **web API** se ocupa cu returnarea de date intr-un anumit format (ex: **JSON**) ca si **response** pentru un **request** initiat de catre un client (browser)



What's an API ?

- in cazul unui API implementat la nivel de browser, acesta reprezinta o interfata implementata in JavaScript prin intermediul careia se pot initia si accesa la nivel de cod anumite functionalitati legate de: comunicare la nivel de retea, cronometrare si masuratori de timp, stocare de date, localizare, utilizarea bluetooth pentru device-uri, acces la elemente ale interfetei si proprietati ale acesora, etc...

Exemple:

- **Fetch API**
- **Navigation Timing API**
- **Web Storage API**
- **Geolocation API**
- **Bluetooth API**
- **DOM**
- ...

1.1 JavaScript Fetch API

Fetch API

- ofera o interfata JavaScript pentru accesarea si manipularea anumitor parti (pasi) din *pipeline*-ul HTTP (comunicarea cu un web server), precum **request** si **response**
- pune la dispozitie metoda globala **fetch()** ce ofera o modalitate usoara de transmitere si primire a resurselor in mod asincron prin intermediul retelei
- are la baza 4 interfete:
 - **Body**
 - **Headers**
 - **Request**
 - **Response**

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

fetch()

- returneaza un **Promise**
- *Promise*-ul returnat nu face reject in cazul in care raspunsul HTTP are un status de tip eroare (400, 500)
- *Promise*-ul returnat va face reject doar in cazul in care se produce un *fail* la nivel de retea sau daca a aparut o problema care a impiedicat *request*-ul de la a fi completat
- *Promise*-ul returnat contine un obiect de tip **Response**; apelul metodei **json()** a acestuia va intoarce *body*-ul raspunsului, in format **JSON**

```
fetch('http://example.com/movies.json').then(function(response) {  
    return response.json();  
}).then(function(myJson) {  
    console.log(JSON.stringify(myJson));  
});
```



REQUESTS

- Method - GET, POST, PUT, DELETE, HEAD
- Url
- Headers
- Referrer
- Mode (cors,no-cors,same-origin)
- Credentials
- Redirect
- Integrity
- cache

GET REQUESTS

- clone()
- json()
- redirect()
- text()
- arrayBuffer()
- blob()
- formData()

```
fetch('https://api.github.com/users/Oviduzz')  
  .then(response => response.json())  
  .then(data => {  
    console.log(data) // Prints result from `response.json()`  
  })  
  .catch(error => console.error(error))
```




Request Headers

```
fetch('https://api.github.com/users/Oviduzz', {  
  headers: new Headers({  
    'User-agent': 'Mozilla/4.0 Custom User Agent'  
  })  
})  
.then(response => response.json())  
.then(data => {  
  console.log(data)  
})  
.catch(error => console.error(error))
```

Post requests

```
fetch('https://my-url.com', {  
  method: 'post',  
  body: 'JSON.stringify(contentObject)',  
  headers: { 'Content-Type': 'application/json' }  
})  
.then(res => res.json())  
.then(res => console.log(res));
```

fetch() – custom options and data

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch#Supplying_request_options

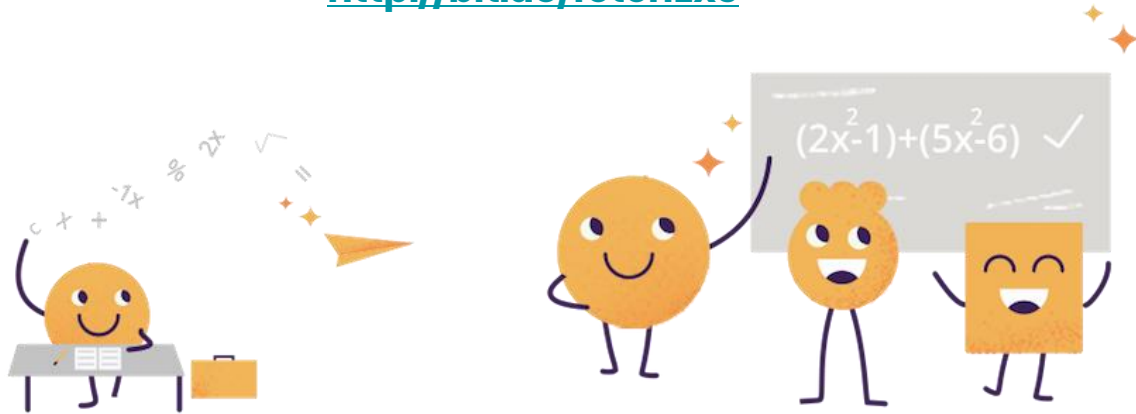
<https://css-tricks.com/using-fetch/>

PRACTICE: JavaScript `fetch()`, Promises and modules

<http://bit.do/fetchEx1>

<http://bit.do/fetchEx2>

<http://bit.do/fetchEx3>





Cookies

vs

Local Storage

vs

Session Storage

Cookies, localStorage, sessionStorage

- 3 metode de stocare de date la nivel de client - browser (client-side storage)
- localStorage si sessionStorage au fost introduse in HTML5 - sunt valabile doar la nivel de browser
- *cookie*-urile sunt valabile si la nivel de server

Cookies

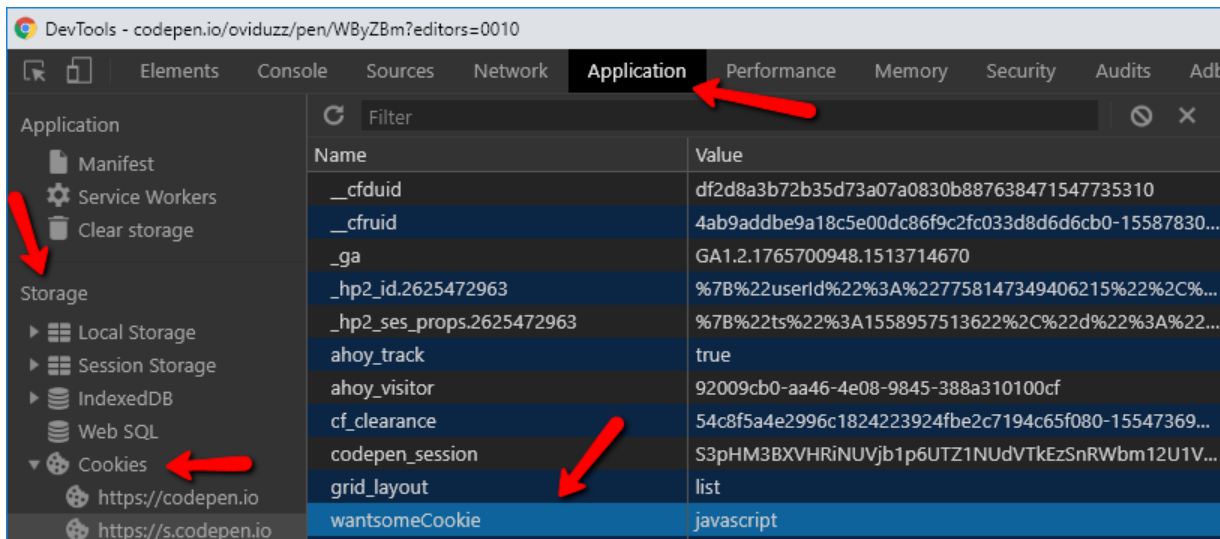
- au un timp de expirare
- maxim ~4kb ca si spatiu pentru date
- datele stocate intr-un *cookie* sunt prezente la nivel de fiecare request HTTP, lucru care mareste traficul, volumul de date transferate intre *client* si *server*
- functioneaza pentru *browsere*-le vechi

Cookies, localStorage, sessionStorage

Set cookie

```
document.cookie = "cookienname=cookievalue"
```

```
document.cookie = "wantsomeCookie = javascript"
```



The screenshot shows the Chrome DevTools Application tab. On the left sidebar, the 'Storage' section is expanded, and 'Cookies' is selected. A red arrow points to the 'Cookies' item. The main panel displays a table of cookies. A red arrow points to the 'Application' tab header, and another red arrow points to the 'wantsomeCookie' entry in the table.

Name	Value
__cfduid	df2d8a3b72b35d73a07a0830b887638471547735310
__cfuid	4ab9adbe9a18c5e00dc86f9c2fc033d8d6d6cb0-15587830...
_ga	GA1.2.1765700948.1513714670
_hp2_id.2625472963	%7B%22userid%22%3A%227758147349406215%22%2C%...
_hp2_ses_props.2625472963	%7B%22ts%22%3A1558957513622%2C%22d%22%3A%22...
ahoy_track	true
ahoy_visitor	92009cb0-aa46-4e08-9845-388a310100cf
cf_clearance	54c8f5a4e2996c1824223924fbc2c7194c65f080-15547369...
codepen_session	S3pHM3BXVHRiNUVjb1p6UTZ1NUdVTkEzSnRWbm12U1V...
grid_layout	list
wantsomeCookie	javascript

Cookies, localStorage, sessionStorage

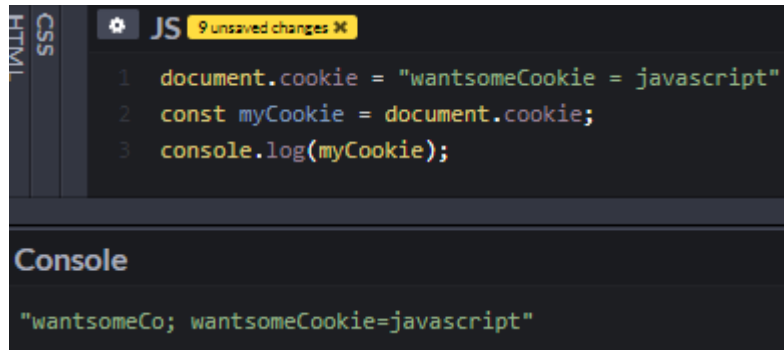
Set cookie

```
document.cookie = "cookienname=cookievalue; expires= Thu, 21 Aug 2019 20:00:00 UTC "
```

```
document.cookie = "cookienname=cookievalue; expires= Thu, 21 Aug 2019 20:00:00 UTC ; path=/"
```

Get cookie

```
const myCookie = document.cookie
```

A screenshot of a web browser's developer console. The top panel shows a JavaScript file with three lines of code: 1. document.cookie = "wantsomeCookie = javascript" 2. const myCookie = document.cookie; 3. console.log(myCookie); The bottom panel, labeled 'Console', shows the output of the log statement: "wantsomeCo; wantsomeCookie=javascript". The browser's sidebar on the left shows 'HTML', 'CSS', and 'JS' tabs, with 'JS' selected. A yellow notification bar at the top of the JS tab says '9 unsaved changes'.

Cookies, localStorage, sessionStorage

Delete cookie

- Pentru a sterge un cookie, trebuie ca acesta sa ia o valoare “goala” si o data de expirare din trecut

```
document.cookie = "cookienam= ; expires= Thu, 1 Jan 1970 00:00:00 GMT"
```


Cookies, localStorage, sessionStorage

localStorage

- datele raman stocate pana in momentul in care sunt sterse
- spatiu de stocare de pana la ~5MB
- datele sunt stocate doar la nivel de client
- datele le gasim doar in acelasi domeniu - de exemplu, indiferent de pagina pe care navigam in cadrul aplicatiei web *facebook*, vom avea acces la datele stocate in **localStorage** pentru *facebook.com*

sessionStorage

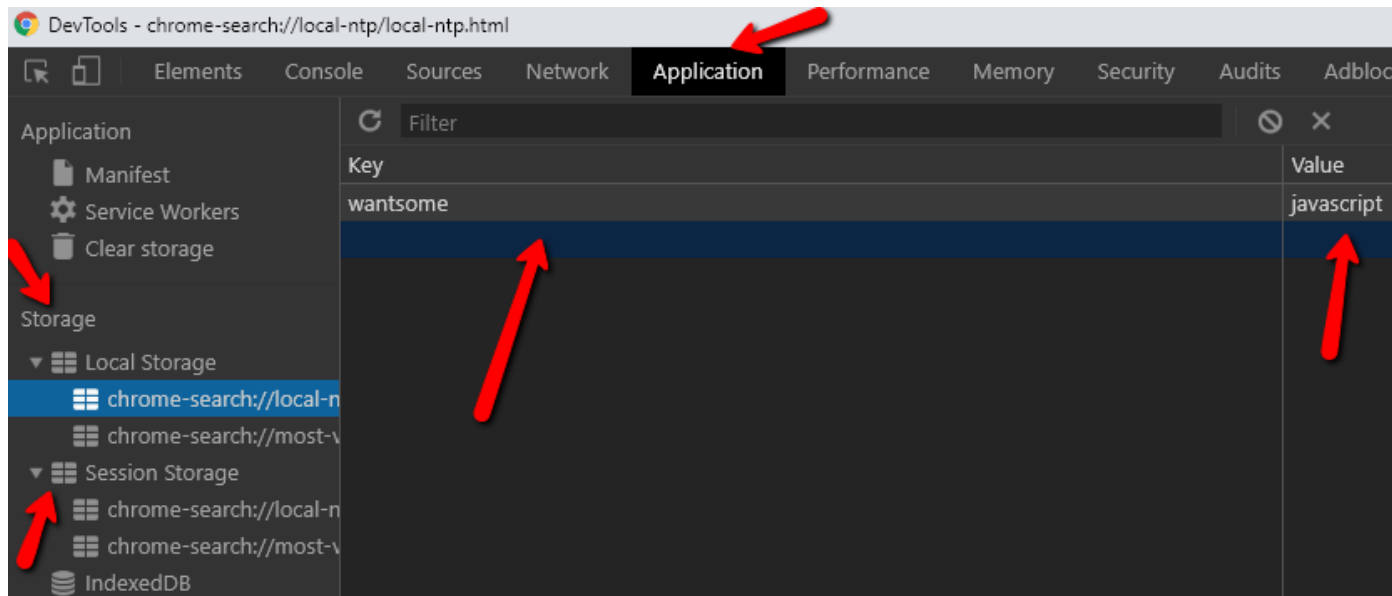
- similar cu **localStorage**, insa datele se pierd in momentul in care sesiunea se incheie - utilizatorul inchide *tab-ul*, *site-ul*, *browser-ul*
- Si **localStorage** si **sessionStorage** folosesc pentru stocare perechi de forma **key-value**

Cookies, localStorage, sessionStorage

Set localStorage/sessionStorage

```
localStorage.setItem("key", "value");
```

```
> localStorage.setItem("wantsome", "javascript")  
< undefined  
> sessionStorage.setItem("wantsome", "frontend")  
< undefined
```



Cookies, localStorage, sessionStorage

localStorage/sessionStorage

- `localStorage.getItem("key");`
- `localStorage.setItem("key", "value")` - valoarea cheii deja setate poate fi actualizata
- `localStorage.removeItem("key")`
- `localStorage.clear()` - sterge toate datele din localStorage

Pentru **sessionStorage** se foloseste aceeași sintaxa la nivel de metode, doar că acestea vor fi apelate din cadrul obiectului **sessionStorage**

[Cookie, localStorage and sessionStorage explained](#)

[cookie/local/session](#)

PRACTICE: Cookies, localStorage and sessionStorage

<http://bit.do/CookieEx>

<http://bit.do/LocalStorageEx>

<http://bit.do/fetchExx>

<http://bit.do/CookieBonus>

