# AJAX

Asynchronous Javascript and XML

eller

bruk av

XMLHttpRequest objektet

Se for eksempel w3schools,

eller Wikipedia

8.11.2016

# XMLHttpRequest, definisjoner

Synkron utførelse:

- En henvendelse sendes til en server med en URL. Vi venter på svar.

Asynkron utførelse:

- En henvendelse sendes til en server med en URL. Vi venter ikke på svar, men fortsetter annen utførelse på klienten. Vi sjekker fra tid til annen om lasting er fullført, og gjør i så fall det som var planlagt.

# XMLHttpRequest, formål

AJAX, Asynkron Javascript og XML.

Hensikten er også å kunne laste opp (mer) kode fra en URL ved hjelp av et Javascript kall, uten å måtte foreta en full refresh av skjerminnhold når koden er lastet opp, men bare oppdatere den aktuelle del av skjermen.
Det er også meningen at denne opplastingen skal kunne foretas uten at brukeren må vente, men kan fortsette med andre ting i samme skjermbilde/webside, f.eks. fortsette å fylle innhold i input felter.
Vi ser det i dag tydelig i bruk, f.eks. i google søk. Du skriver noe i søkefeltet, og siden blir oppdatert mens du skriver.

# XMLHttpRequest, formål

Funksjonene i Javascript er definert i klassen **XMLHttpRequest**.

Den kan brukes både til synkron og asynkron overføring fra server. Synkron overføring er det enkleste, da det ikke forutsetter sjekking av tilstand i overføringen (gir heller ikke mulighet til det). Vi venter.

Da trengs ikke de funksjonene (metodene) i XMLHttpRequest som er definert for dette formålet.

# XMLHttpRequest metoder

| Kall | Forklaring |
|---|---|
| **abort()** | Avbryt den forrige, uavsluttede request |
| **getAllResponseHeaders()** | Returnerer alle response headere, labeler og verdier, alle key/value par fra response header. |
| **getResponseHeader('header')** | Henter verdien til en angitt 'header'. |

# XMLHttpRequest metoder

| Kall | Forklaring |
|---|---|
| **open(method,url)**<br>**open(method,url,async)**<br>**open(method,url,async,user,passwd)** | Method = POST \| GET \| PUT<br>url kan være relativ eller absolutt.<br>• Se kommentar senere<br>Formål:klargjøring! |
| **setRequestHeader(header,value)** | * Se kommentar senere<br>Open må være kalt, ikke send. |
| **send()**<br>**send(body)** | Sender den faktiske request til server, hvor body er evtentuelle post parametre (som GET parametre ellers), evt. null. |

# Parameter method i metode open

| Metode navn | Funksjon |
|---|---|
| GET | Ber om en representasjon av den angitte ressurs |
| POST | Sender data som skal prosesseres |
| HEAD | Ber om samme respons svar som GET, men uten respons body |
| PUT | Laster opp en representasjon av den angitte ressurs |
| DELETE | Sletter den angitte ressurs |
| TRACE | Gir ekko av mottatt request |
| OPTIONS | Returnerer de HTTP metoder serveren supporterer for aktuell URI |
| CONNECT | Konverterer request collection til en transport TCP/IP tunnel |
| Safe metoder: HEAD, GET, OPTIONS, TRACE (dvs. de gjør ingen endringer på serveren, har ingen sidevirkning) | |

# XMLHttpRequest attributter

| Navn | Forklaring |
|---|---|
| **onreadystatechange** | Settes til en funksjon som skal kalles hver gang readyState forandres (altså en event handler) |
| **readyState**<br>(se nøye på forklaring!) | Holder tilstand under prosessering:<br>0 uninitiated, ikke sendt<br>1 Loading, er åpnet, setRequestHeader kan utføres<br>2 Loaded, alle headere er mottatt<br>3 Interactive, laster<br>4 Complete, ferdig |
| **timeout** | Antall millisekunder før operasjon blir terminert. 0 betyr ingen terminering. |

# XMLHttpRequest attributter

| Navn | Forklaring |
|---|---|
| **responseText** | svar fra server, i form av en string. |
| **responseXML** | svar fra server, i form av XML (DOM) |
| **status** | retur status (http response code)<br>200 OK<br>404 Ikke funnet, for eksempel |
| **statusText** | tilhørende tekstlig forklaring på status |

# XMLHttpRequest oppbygging

## Om metoden open():

Hvis metoden open settes opp med tredje parameter angitt (async), så er dette en angivelse av om lastingen av den angitte fil skal gjøres synkront eller asynkront.

- Synkront innebærer at man venter til lasting er ferdig,
- Asynkront innebærer at prosessering i Javascript vil fortsette.

Om man velger asynkron utførelse, må man da kontrollere at man ikke benytter de hentede data før de er ferdig lastet.
Variabelen async settes **true** hvis lasting skal gjøres asynkront, **false** hvis lasting skal gjøres synkront.
Rutinen onreadystatechange og attributtet readyState er vesentlige i denne sammenheng, men er bare relevante for en asynkron request.

# XMLHttpRequest oppbygging

Vær også oppmerksom på at det ikke starter en response før en request er sendt, dvs. at rutinen **send**() er kalt.

**Alt som gjøres forut for dette er å bygge opp innholdet i requesten.**

Det er bare server requester av type GET (metode) som kan benytte asynkron opplasting. En innloggingsside bruker ikke AJAX, mens f.eks. Google eller yr gjør det.

# setRequestHeader oppbygging

| **setRequestHeader** kan ha et av disse attributtene (header, value) | |
|---|---|
| ***Header*** | ***Value*** |
| Accept-Charset | |
| Content-Length | |
| Cookie | |
| Host | |
| Content-Type | for eksempel ”text/HTML” |
| Charset | for eksempel ”utf-8” |
| User-Agent | |

# XMLHttpRequest: nettlesere?

XMLHttpRequest objektet er avhengig av nettleser. Javascript utføres i nettleser!

Det er IE som har avvik i eldre versjoner, men det meste ordnes på overordnet nivå. Dette er et mindre problem i dag.
Når du skal aktivere objektet, kan du gjøre det som følger (for å kunne håndtere eldre IE versjoner er dette nødvendig):

```
if (window.XMLHttpRequest) { // does the object exist?
        objHTTP = new XMLHttpRequest();
} else if (window.ActiveXObject) { // Internett explorer before version 7
        objHTTP = new ActiveXObject("Microsoft.XMLHTTP");
}
```

# XMLHttpRequest

- Synkron overføring er det enkleste, da det ikke forutsetter noen sjekking av når eller om overføring er ferdig; når mer av koden i siden blir utført, så er overføring ferdig. Brukeren kan ikke gjøre noe på siden mens overføring pågår.
- Vitsen med asynkron overføring er nettopp at man skal kunne gjøre andre ting på siden mens man venter på at overføring skal bli ferdig, at man ikke skal bli avbrutt i det man ønsker å utføre i nettsiden fordi noe skal overføres. Da må du nødvendigvis sjekke om overføring er ferdig, før du kan bruke resultatet.

# XMLHttpRequest: eksempel

Hent et XML dokument inn i en DOM XML node, synkront:

```
if (window.XMLHttpRequest)
{      // code for IE7+, Firefox, Chrome, Opera, Safari
       xmlhttp=new XMLHttpRequest();
} else
{      // code for IE6, IE5
       xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.open("GET","books.xml",false);        // hent books.xml synkront
xmlhttp.send();
xmlDoc=xmlhttp.responseXML;
```

# XMLHttpRequest: eksempel 2

Parse string inn i DOM objekt:

```
txt="<bookstore><book>";
txt=txt+"<title>Everyday Italian</title>";
txt=txt+"<author>Giada De Laurentiis</author>";
txt=txt+"<year>2005</year>";
txt=txt+"</book></bookstore>";
if (window.DOMParser) {
        parser=new DOMParser();
        xmlDoc=parser.parseFromString(txt,"text/xml");
} else {            // Internet Explorer før versjon 7
        xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
        xmlDoc.async=false;
        xmlDoc.loadXML(txt); }
```

```
function fil(myurl,id) {
        teller = 0;
        objHTTP.open("GET", myurl, true);                    // hvilken fil skal åpnes, senere
        if (objHTTP.overrideMimeType) {
                objHTTP.overrideMimeType('text/html;charset=ISO-8859-1');
        }                                                    // tillegg, sett på nettet
        objHTTP.setRequestHeader("Content-Type","text/HTML");
        objHTTP.setRequestHeader("Charset","utf-8");         // vanlige request headere
        objHTTP.onreadystatechange = function() {            // definer funksjon
                if (objHTTP.readyState == 4 && objHTTP.status == 200) {
                        document.getElementById(id).innerHTML +=
                        objHTTP.responseText;                // setter på plass tekst
                }
        }
        objHTTP.send(null);                                  // send request!!!!!
}
```

# Nettleser sikkerhet

Av hensyn til sikkerhet tillater ikke nettlesere i dag Javascript å hente data fra andre domener, dvs. filer som refereres i XMLHttpRequest må være "lokale".

Korreksjon: Det sies at de må ha opphav på samme domene som den siden som etterspør dem. Dette kan sjekkes videre.