Denne guide er oprindeligt
udgivet på Eksperten.dk

COMPUTERWORLD

Hent nemt dine undersider med jQuery.load() / Ajax

Ønsker du at hente dine undersider ind i en sektion på din hjemmeside uden at reloade hele siden, og uden at det skaber problemer for søgemaskiner og browsere som ikke understøtter javascript så kig nærmere på denne guide.

Skrevet den **02. nov 2010** af **bkp** I kategorien **Programmering / JavaScript** | ⭐⭐⭐⭐☆

Først vil jeg lige slå fast at denne guide ikke viser hvordan du bygger et flot design, men mere om teknikken du skal benytte for at hente dine underside dynamisk uden at genhente hele siden.

Nogen vil måske benytte IFrame til dette formål, hvilket er en hurtig og endda nemmere løsning, men jeg personligt synes det er bedre at benytte Ajax ved hjælp af jQuery.load() da du herved f.eks. undgår en scrollbar hvis indholdet ikke kan være i rammen, en div som bruges i mit eksempel, vil istedet ændre højden efter indholdet, dette kan selvfølgelig godt skabe problemer hvis man har et fast design som skal passe ind i en fast angiven ramme, så alt dette skal man selvfølgelig tage i betragtning.

Inden jeg beskriver hvordan du gør det, synes jeg lige du skal se dette eksempel, så du er helt med på hvad det går ud på:
http://bkristensen.tumblr.com/jQueryAjaxLink

Hvad er Ajax?
Ajax eller AJAX (en forkortelse for Asynkron JavaScript og XML) er en webudviklingsteknik til at udvikle interaktive webapplikationer. Idéen er at gøre websider mere reaktionsdygtige ved at udveksle små mængder af data mellem klienten og serveren, så hele siden ikke skal genindlæses, hver gang brugeren laver en forespørgsel. Formålet med dette er at øge websidens interaktivitet, hastighed og brugervenlighed.

Nogen sætter lighedstegn mellem javascript og Ajax, men det er ikke fyldestgørende, Ajax er mere en teknik end det er et sprog.

Hvad med Google?
Det er et meget relevant spørgsmål, for hvis indholdet hentes via Javascript, hvordan sikrer vi os så at Google kan indeksere siderne når den ikke eksekverer dit script.

Det er egentlig meget enkelt, du opbygger din menu som almindelige links, og pakker det ind i script som henter de underliggende sider, hvis javascriptet ikke eksekverer, så vil linket virke som det plejer og den underliggende side hentes som det er sket siden html's spæde barndom.

jQuery - hvad er det?
jQuery er et javascript bibliotek som udmærker sig ved at gøre udviklingen af web applikationer lettere, ved at give et simplere og mere kraftfuldt API at arbejde i mod, i forhold til Document Object Model(DOM) og browser specifikke API'er.

Et simpelt eksempel
For at illustrere hvor nemt det er, har jeg bikset et meget simpelt eksempel sammen, det er i html, men kan sagtens bruges i en php eller .Net aspx side.

Hent koden her:
http://static.tumblr.com/6kgdwmv/Mwblb57jm/ajaxdemo.zip

Det første du skal sikre dig, er at der skal være en reference til jQuery biblioteket. Du kan selvfølgelig vælge at hente biblioteket og lægge det på din egen server, men jeg vil anbefale at henvise til en af de flere der ligger på nettet, og gerne en af dem som andre bruger meget, derved oplever du at siden vises hurtigere, da flere af de besøgende allerede har filen liggende i deres cache fordi andre sider linker til den samme fil.

Reference kan placeres i din head sektion og kunne eventuelt se således ud:

```
<head>
  <script src="http://code.jquery.com/jquery-latest.min.js"></script>
</head>
```

Se flere versioner her: http://code.google.com/intl/da-DK/apis/libraries/devguide.html#jquery

Derefter skal du bygge din menu som f.eks. på denne måde:

```
<a href="side1.htm" class="menulink">Side 1</a>
```

Læg mærke til at jeg har givet linket et klasse navn: menulink
det er for at gøre arbejdet lidt lettere senere når vi skal lave en reference til alle vore links, man kunne selvfølgelig godt bygge en click event på alle a tags, men så er det svært at skelne mellem de links der skal fungere som menu, og de links som skal reagere som almindelige links, men det ser du når vi kommer til script delen.

Dernæst indsættes en sektion hvor det generiske indhold skal vises, og den kan således ud:

```
<div id="ajaxbody">Her kommer indholdet !!</div>
```

Nu kommer så det sjove, nemlig vores script:

```
<script type="text/javascript">
    $(document).ready(function () {
        $('.menulink').click(function () {
            // Hent div-body2 fra link ind i ajaxbody
            $('#ajaxbody').load($(this).attr('href') + ' #body2');
            return false;
        });
    });
</script>
```

Det føste du møder i script sektionen er $(document).ready som er en jQuery kommando som fortæller at det der er inden i parantesen (efter ready) først skal køres når siden er klar.

I vores ready sektion opretter vi den funktion som skal køres når siden er klar.

Inden i funktionen kommer det spændende, først henviser vi til alle objekter på siden som har menulink

som klasse og det var jo netop det vi angav i vore links, og hertil kobler vi en klik event som affyres så snart der klikkes på en af linkene.

Inde i klik eventen opretter vi en ny funktion som skal køres når klik eventen køres.

I denne nye funktion henviser vi til den sektion som undersiden skal hentes ind i og det er jo den samme div vi gav en id parameter: ajaxbody, og havelågen lige før navnet angiver at vi taler om en id parameter, hvis det havde været en klasse skal man skrive punktum foran navnet (som i eksemplet med menulink).

Så $('#ajaxbody') henviser altså til det div objekt vi har på siden, og den kalder vi med en load funktion som er en Ajax kommando som henter linket ind i objektet.

Inde i load kommandoen skal vi skrive linket til den side der skal hentes ind i div objektet, men da der er mere end et link, så skal denne url opbygges generisk, og her benytter vi $(this) som er det objekt som kaldte klik eventen, altså det link der blev trykket på, og her læser vi href atributten med attr funktionen.

Da undersiderne også har header og andet der ikke skal hentes ind, har vi på undersiderne lagt indholdet ind i en div med id=body2 og det angiver vi som en ekstra parameter i vores url, så det kun er denne div i undersiden der hentes op i vores ajaxbody.

Til sidst returneres false, for at fortælle browseren at vi har taget os af linket, og at den ikke skal sendes videre til det link der står i <a> href attributten, dette betyder også at hvis scriptet ikke kan afvikles, så virker linket alligevel da browseren ikke får false returneret.

Efter denne forklaring og efter du lige har skimmet nedenstående kode igennem, så tror jeg du forstår hvor smart det egentlig er, hvis der er brug for mere info for at forstå mit eksempel, så se eventuelt ekstra links eller send mig en besked og jeg vil forsøge at uddybe dette nærmere.

http://docs.jquery.com/Tutorials
http://bkristensen.tumblr.com/post/998469255/jquery-hvor-skal-jeg-starte
http://bkristensen.tumblr.com/post/998624407/5-tips-til-bedre-jquery-kode

Index.html

```
<html>
<head>
  <script src="http://code.jquery.com/jquery-latest.min.js"></script>
</head>
<body>
  <h1>Hoved</h1>
  <a href="side1.htm" class="menulink">Side 1</a> -
  <a href="side2.htm" class="menulink">Side 2</a>
  <div id="ajaxbody"></div>
  <script type="text/javascript">
    $(document).ready(function () {
        $('.menulink').click(function () {
            // Hent div-body2 fra link ind i ajaxbody
            $('#ajaxbody').load($(this).attr('href') + ' #body2');
            return false;
        });
    });
  </script>
</body>
```

```
</html>
```

side1.htm

```
<html>
<head>
</head>
<body>
  <h1>Hoved</h1>
  <a href="side1.htm" class="menulink">Side 1</a> -
  <a href="side2.htm" class="menulink">Side 2</a>
  <div id="body2">
    <h1>Dette er side 1</h1>
  </div>
</body>
</html>
```

side2.htm

```
<html>
<head>
</head>
<body>
  <h1>Hoved</h1>
  <a href="side1.htm" class="menulink">Side 1</a> -
  <a href="side2.htm" class="menulink">Side 2</a>
  <div id="body2">
    <h1>Dette er side 2</h1>
  </div>
</body>
</html>
```

Kommentar af repox d. 01. nov 2010 | 1

Jeg bider mærke i at du skriver dette:
"*Nogen vil måske benytte IFrame til dette formål, men jeg synes det er mere smart at benytte Ajax ved hjælp af jQuery.load().*
"
Jeg mangler en begrundelse for hvorfor AJAX er smartere end iFrames andet end at 'du synes det er smartere'? Du konstaterer også senere i din guide at det er 'smartere', men vi ved stadig ikke hvorfor.

Jeg kan ikke helt finde ud af hvilket niveau af udviklere du henvender dig til - nogle ting går du meget i detaljer med som f.eks. jQuery's document ready bliver forklaret meget godt, mens et javascript event nævnes som begreb og du er hurtigt videre som om man burde vide hvad det er.

Sidste minus fra min side er dine link forkortelser, som for mig virker som et dårligt forsøg på at skabe mertrafik på din side, uden du egentlig vil indrømme det - hvis du gerne vil linke til tekster på din side, så gør det; vi læser ikke guiden for at få let overleverbare URL'er, men for at lære noget.

Af positive ting kan jeg nævne at det er ganske rart at du linker til en ZIP fil med dit eksempel pakket sammen og klar til test og brug; det gør at man kan følge dig igennem artiklen som ellers også er ganske pædagogisk formuleret så selv nybegyndere har en chance for at følge med.

Ligeledes er det et stort plus at du trækker Google ind i billedet, som jo netop er relevant for mange udviklere og de kunder/arbejdsgivere de nu har.

Stort set en god guide - hvis du kan begrunde og argumentere for din konstatering om at AJAX er smartere til netop dette formål giver jeg en stjerne mere, blot for at kunne begrunde valget.

En sidste ting, som er mere praktisk - gør din artikel mere letlæselig ved at bruge flere tags - en ganske fremragende guide skrevet af en anden bruger på sitet kan hjælpe dig:
http://www.eksperten.dk/guide/1325

Kommentar af bkp d. 02. nov 2010 | 2

Hej repox, jeg takker for dit konstruktive svar som jeg vil tage i betragtning.
Jeg vil først slå fast at jeg ikke forsøger at nedgøre IFrame, det er bare en personlig ting at jeg ikke bryder mig om IFrame, på samme måde som nogen måske ikke bryder sig om farven rød. Så lad os lige slå fast at IFrames kan være en nem og hurtig måde at opnå samme effekt på.
Jeg kunne måske godt have gjort artiklen mere fyldestgørende omkring de forskellige teknikker jeg nævner men jeg har fokuseret primært på jQuery.load() men jeg vil inden for nærmeste fremtid redigere artiklen og uddybe det nærmere enten med links eller flere detaljer.
Ang. mine links, så er de forkortet fordi jeg syntes de blev alt for lange men det vil jeg ændre min procedure på, da jeg absolut ikke ønsker at være til gene for nogen der gerne vil se hvor linket fører hen.
Tak for dit link ang. tags det vil jeg kigge nærmere på.

Kommentar af bkp d. 02. nov 2010 | 3

Jeg vil dog lige nævne 2 ting som gør at jeg "næsten" aldrig benytter IFrames.

1. IFrame har fast højde og det betyder at hvis din underside fylder mere så får du scroll bars inde midt i siden, det er selvfølgelig en smagssag, men en Div vil automatisk ændre højden hvis det er nødvendigt og man ikke har valgt noget andet i sin style.

2. Desuden "nedarver" undersiden samme CSS m.m. som hovedsiden ved at hentes dynamisk ind i din Div, dette sker ikke i en IFrame, her skal du huske at linke til dine stylesheets.

Så det kan sagtens lade sig gøre at få samme effekt med en IFrame hvis man bare husker ovenstående.

Og hånden på hjertet, så er det selvfølgelig bedst at have en del erfaring med Javascript før man giver sig i kast med denne teknik, det er ikke optimalt hvis man kun kopierer fra mit eksempel og ikke ved hvorfor det virker.

Kommentar af softspot d. 04. nov 2010 | 4

Fin artikel! Jeg har selv benyttet jQuery i nogen tid nu, men jeg har ikke været opmærksom på den specifikke teknik med at loade delvis indhold fra andre sider med load-funktionen, så det var godt lige at få det på plads :-)

En ting jeg noterer mig er, at dine undersider (side1.htm og side2.htm) ikke indeholder scriptet til at loade via AJAX og det er vel ikke hensigtsmæssigt i og med man sagtens kunne forestille sig, at der linkes direkte til disse sider og så skal den "smarte" metode vel stadig fungere.

Jeg er klar over det er en detalje, men for komplethedens skyld... :-)

Kommentar af bkp d. 04. nov 2010 | 5

Det kan du have ret i, men dette er kun et eksempel, og hvis jeg f.eks. skulle lave det samme i en Asp.net webform, ville jeg lave alt scriptarbejdet på master siden, og derved er det løst.

Husk også at scripts normalt lægges ud i en script fil, og derved kan undersiderne jo nemt linke til scriptet uden du behøver at gentage dit script på alle siderne.

Kommentar af softspot d. 05. nov 2010 | 6

Jeg er lige faldet over dette link til en eBook der uddyber noget af det du er inde på og giver samtidig nogle konkrete forslag til de udfordringer jeg nævnte i #6:

http://www.thefutureoftheweb.com/blog/free-book-unobtrusive-ajax

Det er godt nok en lidt gammel bog (2007), men den giver, ikke desto mindre, stadig med en god indgangsvinkel til, hvordan man kan få sine sider til at fungere for brugere der ikke understøtter javascript (og CSS for den sags skyld), såvel som dem der understøtter/tillader det hele...

Kommentar af softspot d. 05. nov 2010 | 7

...og det indlæg #6 jeg henviser til er så åbenbart ikke blevet registreret alligevel (til trods for at jeg har kunne se det indtil flere gange i min browser!?!?).

Grundlæggende omhandlede det indlæg (som ikke blev registreret) den relativt lille gevinst man har ved at bruge den foreskrevne metode uden videre.

For mig at se, er det kun på klienten man slipper for at se på flimmer når der opdateres, for alle andre steder i opdateringsprocessen, ved klik på et link, skal der stadig transporteres og genereres samme mængde data, som hvis man bare undlod at benytte metoden.

Den nævnte eBook giver dog, som nævnt, nogle konkrete bud på hvordan man kan klare dette, men det er altså på serveren dette skal ske for at det giver en egentlig gevinst...

En anden udfordring med denne metode er, at det kan være svært for brugeren at gennemskue hvor han/hun er på sitet, da url'en ikke vil skifte når indhold loades på denne måde. Dette gør det samtidig svært at lave direkte links til et specifikt indhold. Dette kan der jo nok også findes en løsning på (et hint findes da også i e-bogen).

Kommentar af bkp d. 05. nov 2010 | 8

Mange tak for indput softspot, det tjekker jeg lige ved lejlighed.

Jeg har arbejdet lidt med Fiddler for at tjekke performance og pakkestørelser, og her kunne jeg se god performance samt mindre pakker ved f.eks. at benytte jQuery.getJson(), her er det godt nok Json der sendes frem og tilbage, men alligevel sparer du på en del af de headers m.m. som browseren sender sammen med en request.

Men da jeg ikke har læst bogen du henviser til, vil jeg ikke afvise at der er gode facts at hente i den, så tak for linket.

Kommentar af olebole d. 24. mar 2011 | 9

<ole>

Ved kritikløst at hente og bruge andres JavaScript direkte over nettet, pålægger man sig et enormt ansvar overfor sine brugere, derved at man slipper enhver kontrol med koden - og dens potentielt skadelige virkninger.

Ved at undervise andre i denne fremgangsmåde pålægger man sig selv et endnu større ansvar. Med al respekt: Det er ikke særlig klogt!

/mvh
</bole>

Kommentar af bkp d. 07. apr 2011 | 10

Hej Ole #9
Er din kommentar myntet på min artikel eller på en kommentar?
Uddyb lige hvad du henviser til ;-)