AJAX

Voor 2005 was het uiteraard wel mogelijk om gegevens van een server op te halen (anders zou er geen internet bestaan), maar om de opgehaalde gegevens te kunnen tonen, moest een hele nieuwe pagina geladen worden met daarin die gegevens. AJAX maakt het mogelijk opgehaalde gegevens in een pagina te tonen, zonder de pagina in zijn geheel te hoeven vernieuwen. Veel grote websites als Google en Facebook maken gebruik van deze techniek.

Een voorbeeld

Een niet-spectaculair, maar wel eenvoudig voorbeeld waarin de essentie van AJAX is terug te vinden, ziet u in de code op deze pagina.

De code staat hieronder:

```
<!DOCTYPE html>
<html>
<head>
<script>
function loadXMLDoc() {
  var xmlhttp;
  if (window.XMLHttpRequest) {
      // code for IE7+, Firefox, Chrome, Opera, Safari
      xmlhttp=new XMLHttpRequest();
  }
  else { // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.onreadystatechange=function() {
    if (xmlhttp.readyState==4 && xmlhttp.status==200) {
      document.getElementById("myDiv").innerHTML =
                                    xmlhttp.responseText;
    }
  }
  xmlhttp.open("GET","ajax_info.txt",true);
  xmlhttp.send();
}
</script>
</head>
<body>
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">
  Change Content</button>
</body>
</html>
```

(Bron: www.vruplanet.com.)

Het is een pagina met een knop. Zodra u op de knop klikt, wordt de functie `loadXMLDoc()` aangeroepen.

De functie maakt een lokale variabele met de naam `xmlhttp`. Het if-else-statement dat daarop volgt, maakt onderscheid tussen moderne browsers en oudere versies van IE.

In de moderne versie staat er:
```
xmlhttp=new XMLHttpRequest();
```

Hiermee wordt een nieuw object gemaakt van het type `XMLHttpRequest`. Dat is een object waarmee de communicatie met de server tot stand komt. In oude versies van IE verliep dit via een `ActiveXObject`.

De voorwaarde in het if-else-statement
```
if (window.XMLHttpRequest)
```
controleert of het `window`-object over de constructor-functie met de naam `XMLHttpRequest` beschikt, zo niet, dan levert deze uitdrukking `false`.

De opdracht na het if-else-statement luidt:
```
xmlhttp.open("GET","ajax_info.txt",true);
```

Met dit statement wordt het `XMLHttpRequest`-object gevuld met de juiste informatie.

Zoals u wellicht weet, zijn er verschillende manieren om met het http-protocolgegevens van een server op te halen. De meest gebruikte zijn GET en POST. Op deze pagina wordt het verschil tussen beide uitgelegd.

In de meeste gevallen is GET sneller en eenvoudiger en dat is hier gebruikt. Het tweede argument van `open()` geeft de URL van de op te vragen informatie. In zijn eenvoudigste vorm, zoals hier, is dat een bestandsnaam van een bestand dat op de server staat. Het laatste argument (`true`)geeft aan dat de informatie asynchroon (op de achtergrond, daar gaat het bij AJAX juist om) moet worden opgehaald.
Met de opdracht
```
xmlhttp.send();
```
wordt het verzoek aan de server verzonden.

Om in de gaten te houden wat er met het verzoek gebeurt, heeft het `XMLHttpRequest`-object een property met de naam `readyState`. Deze kan vijf verschillende waarden aannemen, van 0 tot en met 4, afhankelijk van de status van het verzoek, zie deze pagina.

Wij zijn natuurlijk speciaal geïnteresseerd in de waarde 4 voor `readyState` en de waarde 200 voor `status`, want dan is alles in orde. Bij de verandering van `readyState` treedt er een `onreadystatechange`-event op die in de event-handler wordt opgevangen:

```
xmlhttp.onreadystatechange=function() {
  if (xmlhttp.readyState==4 && xmlhttp.status==200) {
    document.getElementById("myDiv").innerHTML =
                                  xmlhttp.responseText;
  }
}
```

(Bron: www.vtuplanet.nl.)

De server kan XML terugsturen of tekst die geen XML is. Als het XML is, kunt u die vinden in de property `responseXML` van het `XMLHttpRequest`-object. Tekst die geen XML is in de property `responseText`.

Een voorbeeld van het opvragen van een XML-file en het weergeven van een deel van de data uit deze file vindt u hier.

Een opdracht als:

```
var x = xmlhttp.responseXML;
```

zorgt ervoor dat x een XML-DOM-object is. Zo'n object kan met JavaScript op vergelijkbare wijze benaderd worden als een HTML-DOM-object. Het voert te ver om daar in deze module op in te gaan. Zie eventueel deze pagina voor een cursus op dit gebied.

Omdat XML erg geschikt is om gegevens op een platformonafhankelijke manier te transporteren, kan XML ook gebruikt worden om gegevens uit een database op te vragen.

Aan de clientzijde voert u bijvoorbeeld in een webpagina de naam van een persoon in en met een klik op de knop wordt vanuit JavaScript via een `XMLHttpRequest` informatie over die persoon aan de server gevraagd.
Aan de serverkant is een programma (script) aanwezig dat de koppeling met de database maakt, de gegevens daaruit ophaalt, en die gegevens als XML-bestand verzendt naar de browser van de client. Met JavaScript kan de informatie uit de property `responseXML` gehaald worden en ergens op de pagina van de client worden getoond.