ELSŐ FEJEZET

Mi az Ajax?

2001 és 2005 között a weben használt technológiák és módszerek terminológiája robbanásszerű növekedésen ment keresztül, hogy ezt a valaha statikus médiumot életre keltse. Ahogy egyre több webes alkalmazás bukkant fel, mint meghatározó része az online célpontoknak, többé már nem online brossúrák és katalógusok uralták az internetet. A webes alkalmazások abban különböztek webhelyőseiktől, hogy azonnali szolgáltatást biztosítottak felhasználóiknak, nem csak információt. Akár az üzleti folyamatok részeként, akár személyes érdeklődésből a fejlesztők arra kényszerültek, hogy új kölcsönhatási paradigmákat hozzanak létre, ahogy a felhasználók egyre több funkciót vártak el.

Kevéssé ismert és kevésbé használt technológiák ösztönzése révén, amelyeket a webböngészők tartalmaztak egy ideig, az internet vakmerő lépést tett előre, amikor összezúzta a hagyományos használati modellt, amely az oldal teljes betöltését követelte meg minden alkalommal, amikor új adatokhoz vagy az alkalmazás logikájának egy új részéhez fértek hozzá. A társaságok elkezdtek kísérletezni a weblapok részeinek dinamikus újratöltésével, kis mennyiségű adatok küldésével a felhasználóknak, amely gyorsabb és vitathatatlanul jobb felhasználói élményt eredményezett.

Ezen mozgalom élén a Google állt. Miután a keresőóriás nyilvánosságra hozta, a Google mérnökei által kitalált új tapasztalatok elkezdtek feltünedezni az oldal egy bizonyos részén, amelynek neve Google Labs (`labs.google.com`). A Google Labs sok projektje, mint a Google Suggest és Google Maps, csak egy weblapot tartalmazott, amely soha nem lett kiürítve, mindazonáltal folyamatosan frissült. Ezen fejlesztéseket, amelyek elkezdték behozni az asztali szoftverinterfészeket a böngészők világába, internetszerte dicsérték, mint egy új korszak nyitányát a webfejlesztésben. Valóban így is volt.

Számos nyílt forráskódú és kereskedelmi termék indult fejlődésnek, hogy kihasználják ezen új webes alkalmazásmodell előnyeit. Ezek a projektek technológiájukat különböző kifejezések használatával magyarázták, mint pl. JavaScript távoli parancsvégrehajtás, távoli internetes eljáráshívások és dinamikus frissítés. Hamarosan azonban egy új kifejezés bukkant föl.

Az Ajax születése

2005 februárjában Jesse James Garrett az Adaptive Path LLC-től, közzétett egy online cikket, amelynek címe: „Ajax: A New Approach to Web Applications" (még mindig hozzáférhető a `www.adaptivepath.com/publications/essays/archives/000385.php` címen). Ebben Garrett elmagyarázza azt a meggyőződését, hogy a webes alkalmazások betöltik az internet és a hagyományos asztali alkalmazások közti szakadékot. Új technológiákat és számos Googleprojektet hozott fel annak példájaként, hogy a hagyományosan windows-alapú felhasználói interakciómodellek most már egyre inkább terjednek el az interneten is. Aztán jött két mondat, amely magasra lobbantotta az érdeklődés, az izgalom és a vita lángját:

> *A Google Suggest és a Google Maps két olyan példája a webes alkalmazások új megközelítésének, amelyet mi itt az Adaptive Path-nál csak Ajaxnak hívunk. Ez a név az Asynchronous JavaScript + XML rövidítése, és alapvető váltást jelképez annak terén, hogy mi lehetséges az interneten.*

Ettől a pillanattól az Ajax-szal foglalkozó cikkek és példaprogramok szökőárként jelentek meg, és internetszerte heves viták kezdődtek: Fejlesztők blogoltak, technológiai magazinok írtak róla, a cégek pedig elkezdték alkalmazni termékeikben. Ám annak megértéséhez, hogy mi az Ajax, először azt kell megértenünk, hogyan vezetett néhány internetes technológia evolúciója a kifejlődéséhez.

A Web evolúciója

Amikor Tim Berners-Lee 1990-ben bemutatta az első weboldalt, az ötlet meglehetősen egyszerű volt: létrehozni az összefüggő információk hálóját a hipertöveg és az egységes erőforrás-azonosító (URI – Uniform Resource Identifiers) használatával. A lehetőség, hogy összekapcsoljunk különböző dokumentumokat szerte a világból, hatalmas potenciált tartogatott a tudományos kutatások számára, ahol az emberek szinte azonnal képesek hozzáférni a hivatkozott anyagokhoz. Valóban, a Hyper Text Markup Language (HTML) első verziója valamivel több szolgáltatást tartalmazott, mint a formázás és a parancsok linkelése, és nem az interaktív programok építésére szolgáló platform volt, hanem azon szöveges és képi információk megőrzése, amelyek uralták a nyomtatás korának végét. Ezekből a statikus weblapokból nőtt ki az internet.

Ahogy az internet fejlődött, az üzletek meglátták a lehetőséget abban, hogy átadhatják a tömegeknek az információkat a termékekről és szolgáltatásokról. Az internet következő generációja meglátta az információk formázásának és megjelenítésének újszerű képességeit, ahogy a HTML is fejlődött, hogy lépést tartson az igényekkel és megfeleljen ezen új média hozzáértő felhasználói elvárásainak. Azonban egy Netscape nevű kis társaság hamarosan sokkal nagyobb ütemben kezdte előmozdítani az internet evolúcióját.

A JavaScript

A Netscape Navigator volt az első sikeres mainstream böngésző, és mint ilyen, gyorsan mozdította elő a webes technológiákat. A Netscape-t azonban gyakran kigúnyolták a szabványszervezetek, amiért új technológiákat és bővítményeket ágyaz be a létező technológiákba, még mielőtt a szabványokat bevezették volna (kb. annyira, mint ahogy mostanában büntetik a Microsoftot, amiért az Internet Explorer fejlesztéseiben figyelmen kívül hagyja a létező szabványokat). Az egyik ilyen technológia volt a JavaScript.

Az eredetileg LiveScript nevű JavaScriptet a Netscape-nél dolgozó Brendan Eich hozta létre, és a böngésző 2.0-s verziójában (amit 1995-ben adtak ki) már alkalmazták is. A fejlesztők most először voltak képesek befolyásolni, hogyan hathat a weblapra a felhasználó. Ahelyett, hogy folyamatosan ide-oda mentek volna az adatok a szerverhez és vissza, olyan egyszerű feladatoknál is, mint az adatellenőrzés, lehetségessé vált áthárítani az adatfeldolgozás ezen kis részét a böngészőre. Ez a képesség nagyon fontos volt abban az időben, amikor a legtöbb felhasználó 28.8 kpbs sebességű modemmel csatlakozott az internethez, ami minden egyes szerverhez intézett kérést türelemjátékká változtatott. Azon idő csökkentése, amelyet a felhasználónak a válaszra való várakozással kellett eltöltenie, volt az első nagyobb lépés az Ajax-szemlélet felé.

Keretek

A HTML eredeti verziója azt irányozta elő, hogy minden dokumentum egyedülálló legyen, és csak a HTML 4.0 idején jelentek meg hivatalosan a keretek. Azon ötlete, hogy egy weblap megjelenítése több dokumentum között is felosztható, radikális volt, és nagy vita kerekedett arról, hogy a Netscape megvalósította ezt a szolgáltatást, még mielőtt a HTML 4.0 szabvány készen lett volna. A Netscape Navigator 2.0 volt az első böngésző, amely együtt támogatta a kereteket és a JavaScriptet. Mint később kiderült, ez fontos lépés volt az Ajax fejlődésében.

Amikor az 1990-es évek végén megkezdődött a böngészők háborúja a Microsoft és a Netscape között, mind a JavaScript, mind a keretek formalizáltak lettek. Ahogy egyre több szolgáltatás került bele mindkét technológiába, a kreatív fejlesztők elkezdtek kísérletezni a kettő együttes használatával. Mivel a keret egy teljesen elkülönült kérés volt a szerverhez, az a képesség, hogy JavaScript segítségével vezéreljük a keretet és annak tartalmát, a lehetőségek izgalmas tárházát nyitotta meg.

A rejtettkeret-módszer

Ahogy a fejlesztők elkezdték megérteni, hogy hogyan kezeljék a kereteket, felbukkant egy új technika, amely megkönnyítette a kliens-szerver kommunikációt. A rejtettkeret-módszer tartalmaz egy olyan keretkészletet, ahol egy keret szélessége vagy magassága 0 képpontra volt állítva, amelynek egyedüli célja, hogy elkezdje a kommunikációt a szerverrel. Ez a rejtett keret tartalmazhat egy sajátságos űrlapmezőkkel rendelkező űrlapot, amelyet JavaScripttel dinamikusan ki lehetett tölteni és visszaküldeni a szervernek. Ha a keret visszatért, akkor meghívott egy másik JavaScript függvényt, hogy figyelmeztesse a hívó oldalt arra, hogy az adatok visszatértek. A rejtettkeret-módszer jelentette az első aszinkron kérés/válasz modellt a webes alkalmazások számára.

Míg ez volt az első Ajax kommunikációs modell, egy másik technológiai fejlődés már a küszöbön állt.

A Dynamic HTML és a DOM

1996-ban az internet még mindig inkább statikus világ volt. Noha a JavaScript és a rejtettkeret-módszer felélénkítette a kölcsönhatást a felhasználóval, még mindig nem volt mód arra, hogy megváltoztassák egy oldal kinézetét anélkül, hogy újra kéne azt tölteni, nem is beszélve az űrlapok mezőiben tartalmazott értékek módosításairól. Aztán megjelent az Internet Explorer 4.0.

Az Internet Explorer ekkorra behozta technológiai lemaradását a piacvezető Netscape Navigatorral szemben, sőt, a Dynamic HTML (DHTML) bemutatásával egy fontos szempontból még föl is múlta azt. Bár még mindig csak fejlesztési fázisban volt, a DHTML jelentős előrelépés volt a statikus weblapok idejében, és lehetővé tette, hogy egy betöltött oldal bármely részét megváltoztassák a JavaScript segítségével. A Cascading Style Sheets (CSS) felbukkanásával a DHTML újraélesztette a webfejlesztést annak ellenére, hogy a Microsoft és a Netscape milyen eltérő utat követett a korábbi években. A fejlesztői társadalomban megmutatkozó izgatottság beigazolódott, ugyanis a DHTML kombinálása a rejtettkeret-módszerrel azt jelentette, hogy az oldalakat bármikor lehetett frissíteni a szervertől származó információkkal. Ez valódi paradigmaváltás volt az internet számára.

A DHTML soha nem lett szabvány, noha a Microsoft befolyása erősen érződött a Document Object Model (DOM) bemutatásával, mint a szabványosításra való törekvés központi darabjával. A DHTML-lel ellentétben, amely csak a weblap részeinek módosítását kutatta, a DOM ambiciózusabb céllal rendelkezett: struktúrát akart biztosítani az egész weblap számára. Ezen struktúra manipulálása aztán lehetővé teszi a DHTML-szerű módosításokat az oldalon. Ez volt a következő lépés az Ajax felé.

Az iFrame-mechanizmus

Noha a rejtettkeret-módszer hihetetlenül népszerűvé vált, volt egy hátránya is: az embernek előre kellett terveznie, és a keretkészletet úgy kellett elkészítenie, hogy előre látta a rejtett keretek használatát. Amikor 1997-ben bemutatták a `<frame/>` elemet, mint a HTML 4.0 hivatalos részét, az egy másik jelentős lépés volt az internet fejlődésében.

Keretkészletek meghatározása helyett a fejlesztők elhelyezhettek iFrame-elemeket bárhol az oldalon. Ez lehetővé tette a fejlesztők számára, hogy egyáltalán ne használjanak keretkészleteket, hanem egyszerűen elhelyezzenek láthatatlan iFrame-elemeket (CSS használatával) az oldalon a kliens-szerver kommunikáció kiszolgálására. Mikor a DOM-ot végül megvalósították az Internet Explorer 5-ös és a Netscape 6-os verziójában, az lehetővé tette annak a képességét, hogy menet közben, dinamikusan hozzunk létre iFrame-elemeket, ami azt jelenti, hogy JavaScript függvény segítségével hozhatunk létre egy iFrame-elemet, intézhetünk egy kérést, és kaphatjuk meg a választ – anélkül, hogy további HTML kerülne az oldal kódjába. Ez vezetett a rejtettkeret-módszer következő generációjához: a rejtett iFrame módszerhez.

XMLHttp

A Microsoft böngésző-fejlesztői bizonyára rádöbbentek a rejtettkeret és a rejtett iFrame módszerek népszerűségére, ugyanis úgy döntöttek, hogy biztosítanak egy jobb eszközt a fejlesztők számára a kliens-szerver kölcsönhatás érdekében. Ez az eszköz, egy ActiveX objektum formájában, 2003-ban érkezett, és a neve: XMLHttp.

A Microsoft egyik JavaScript bővítménye tette lehetővé az ActiveX-vezérlőelemek, a Microsoft-tulajdonú programozási objektumok létrehozását. Amikor a Microsoft megkezdte az XML támogatását egy MSXML nevű osztálykönyvtár révén, az tartalmazta az XMLHttp objektumot is. Noha az XML név benne van, ez az objektum több volt, mint egy újabb módja az XML-adatok kezelésének. Ez valóban több volt, mint egy ad hoc HTTP-kérés, amely JavaScriptből vezérelhető. A fejlesztők hozzáfértek a HTTP-állapotkódokhoz és -fejlécekhez, valamint a szervertől visszakapott összes adathoz. Az

adat lehetett strukturált XML, előformázott HTML, sorosított JavaScript-objektum vagy bármilyen más formátum, amire a fejlesztőnek szüksége volt. Rejtett keretek vagy iFrame elemek használata helyett most már lehetséges volt hozzáférni a szerverhez programozott módon, csupán JavaScript használatával, függetlenül az oldal töltés/újratöltés ciklusától. Az XMLHttp objektum hatalmas siker volt az Internet Explorer fejlesztői számára.

A népszerűség növekedésével a nyílt forráskódú Mozilla projekt fejlesztői nekiláttak a saját XMLHttp portjuknak. Ahelyett, hogy engedélyezték volna a hozzáférést az ActiveX-hez, a Mozilla fejlesztői lemásolták az objektum legfontosabb metódusait és tulajdonságait egy natív böngészőobjektumba, az `XMLHttpRequest`-be. Azáltal, hogy mindkettő nagyobb böngésző támogatja az XMLHttp valamilyen formáját, az Ajax-típusú interfészek fejlesztése igazán lendületet vett, és arra kényszerítette a kisebb böngészőket, az Operát és a Safarit, hogy szintén támogassák az XMLHttp valamilyen formáját (mindkettő a natív támogatást választotta, a Mozillához hasonlóan egy `XMLHttpRequest` objektummal). Igen ironikus, hogy ezen XMLHttp-klón népszerűsége a Microsoftot is elérte, így az Internet Explorer 7-ben bemutatták a natív `XMLHttpRequest` objektumot.

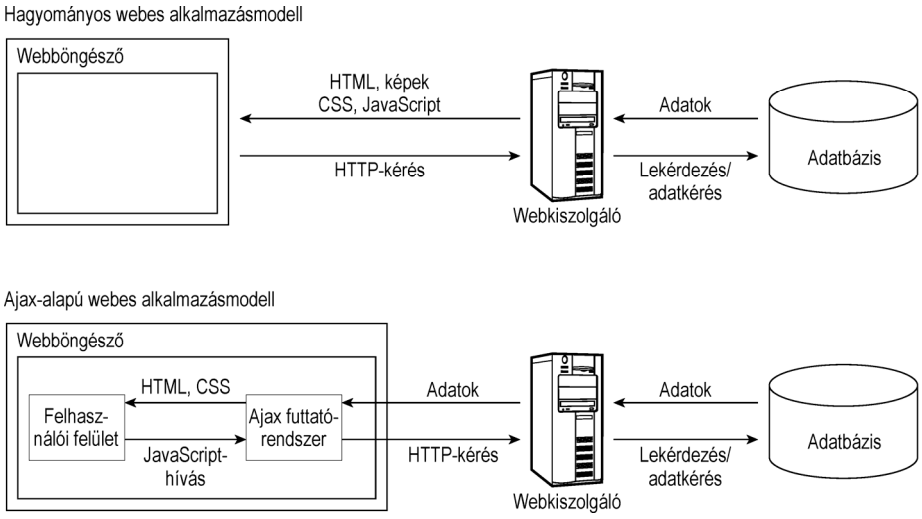
Az igazi Ajax

A Garrett írásának végéhez kapcsolt, gyakran ismételt kérdések ellenére némi zavar még mindig létezik azzal kapcsolatban, hogy valójában mi is az Ajax. Egyszerűen mondva az Ajax nem más, mint a webes interakció egyfajta megközelítése. Ez a megközelítés csak kis mennyiségű információ átvitelét tartalmazza a szerverhez és vissza; annak érdekében, hogy a felhasználónak a lehető legmegfelelőbb élményt nyújtsa.

A hagyományos webes alkalmazásmodell helyett, amelyben maga a böngésző felelős a szerverhez intézett lekérdezések kezdeményezéséért, valamint a kapott kérések feldolgozásáért, az Ajax egy közbülső réteget biztosít – Garrett ezt Ajax-motornak hívja – ezen kommunikáció vezérlésére. Az Ajax-motor valójában csak egy JavaScript objektum vagy függvény, amelyet akkor hívnak meg, ha információt kell kérni a szervertől. A hagyományos modell helyett, mely szerint hivatkozást kell biztosítani a másik erőforráshoz (pl. egy másik weblaphoz), minden hivatkozás az Ajax-motort hívja meg, amely beütemezi és végrehajtja a lekérdezést. A lekérdezés aszinkron módon történik, ami azt jelenti, hogy a kódvégrehajtás nem vár válaszra, mielőtt folytatódna.

A szerver – amely hagyományosan HTML-t, képeket, CSS-t vagy JavaScriptet szolgáltat – úgy van beállítva, hogy az Ajax-motor által használható adatokat adjon vissza. Az adat lehet normál szöveg, XML, vagy bármilyen más adatformátum, amelyre szükségünk van. Az egyetlen követelmény az, hogy az Ajax-motor megértse és értelmezni tudja az adatot.

Amikor az Ajax-motor megkapja a szerver válaszát, akcióba lép, gyakran elemzi az adatokat és módosításokat hajt végre a felhasználói felületen a kapott információk alapján. Mivel ez a folyamat kevesebb információ átvitelét jelenti, mint a hagyományos webes alkalmazásmodellnél, a felhasználói felület frissítése gyorsabb, és a felhasználó is gyorsabban képes végezni a munkáját. Az 1.1. ábra a Garrett cikkében levő kép adaptálása, amely jelzi a hagyományos és az Ajax webes alkalmazásmodellek közti különbséget.



1.1. ábra

Az Ajax alapelvei

Mint új webes alkalmazásmodell, az Ajax még mindig kezdeti szakaszában jár. Azonban számos webfejlesztő kihívásként értelmezi ezt az új fejlesztést. A kihívás annak meghatározása, hogy mi tesz jóvá egy Ajax webes alkalmazást, és mi teszi rosszá vagy középszerűvé. Michael Mahemoff (`www.mahemoff.com`) szoftverfejlesztő és felhasználhatósági szakértő számos kulcselvét meghatározta a jó Ajax-alkalmazásoknak, amelyeket érdemes megtekinteni:

- **Minimális forgalom:** Az Ajax-alkalmazásoknak olyan kevés adatot kell küldeniük a szervernek és visszakapniuk attól, amennyire csak lehetséges. Röviden, az Ajax minimalizálhatja a kliens és a szerver közti forgalom mennyiségét. Annak biztosítása, hogy az Ajax-alkalmazásunk nem küld és kap felesleges információt, hozzáadódik erőteljességéhez.

- **Nincsenek meglepetések:** Az Ajax-alkalmazások tipikusan más felhasználói kölcsönhatásmodelleket mutatnak, mint a hagyományos webes alkalmazások. A kattints-és-várakozz internetes szabvánnyal ellentétben néhány Ajax-alkalmazás más felhasználói felületparadigmákat alkalmaz, pl. húzás vagy kettős kattintás. Nem számít, hogy milyen felhasználói kölcsönhatásmodellt használunk, legyünk következetesek, hogy a felhasználó tudja, mire számítson legközelebb.
- **Meghonosodott szokások:** Ne fecséreljük az időt olyan új felhasználói kölcsönhatásmodellek kifejlesztésére, amelyeket a felhasználók nem ismernek. Kölcsönözzünk bátran a hagyományos webes- és asztali alkalmazásoktól, így kevesebbet kell tanulni.
- **Nincsenek zavaró tényezők:** Kerüljük el az olyan szükségtelen és zavaró elemeket az oldalon, mint pl. az ismétlődő animációk és villogó oldalrészek. Az ilyen fogások elvonják a felhasználó figyelmét arról, amit végre akar hajtani.
- **Hozzáférhetőség:** Gondoljuk át, hogy kik lesznek az elsődleges és másodlagos felhasználók, és hogyan fognak hozzáférni az Ajax-alkalmazásunkhoz. Ne programozzuk egy sarokba, hogy a váratlan új közönség teljesen kirekesztődjön belőle. Használni fognak a felhasználóink régi böngészőket vagy speciális szoftvereket? Tudjuk meg előre, és készítsünk rá tervet.
- **Teljes oldalletöltések elkerülése:** A kezdőoldal letöltése után minden szerverrel kapcsolatos kommunikációt az Ajax-motornak kell kezelnie. Ne romboljuk a felhasználói élményt azzal, hogy egyik helyen kevés adat töltődik le, máshol viszont az egész oldal újratöltődik.
- **A felhasználó az első:** Az Ajax-alkalmazást úgy kell megtervezni, hogy a felhasználó az elsődleges szempont. A szokványos felhasználási esetek elvégzését igyekezzünk könnyűvé tenni, és ne azzal törődjünk, hogy hogyan illeszthetünk be hirdetéseket vagy frankó effekteket.

Mindezen elvek közös vonása a használhatóság. Az Ajax elsődlegesen a felhasználó internetes élményeinek fokozására szolgál; a mögötte álló technológia pusztán egy módszer ehhez. Ha ragaszkodunk az előző elvekhez, egészen biztosak lehetünk benne, hogy Ajax-alkalmazásunk hasznos és használható lesz.

Az Ajax mögött álló technológiák

Garrett cikke számos olyan technológiát említ, amelyet ő az Ajax-megoldások részének tekint. Ezek:

- **HTML/XHTML:** Elsődleges tartalomleíró nyelvek
- **CSS:** Stílusbeli formázást biztosít az XHTML részére
- **DOM:** A betöltött oldal dinamikus frissíthetősége
- **XML:** Adatcserélési formátum
- **XSLT:** Az XML-t XHTML-lé alakítja (CSS által formázva)
- **XMLHttp:** Elsődleges kommunikációs közvetítő
- **JavaScript:** Szkriptnyelv az Ajax-motor programozásához

Valójában mindezen technológiákat lehet használni az Ajax-megoldásokban, de csak három szükséges: az XML/XHTML, a DOM és a JavaScript. Az XHTML az információk megjelenítéséhez szükséges, míg a DOM az XHTML oldal egyes részeinek módosításához az egész oldal újratöltése nélkül. Az utolsó, a JavaScript a kliens-szerver kommunikáció elkezdéséhez és a DOM kezeléséhez kell, hogy az frissítse a weblapot. A többi technológia az Ajax-megoldások finomhangolásában segíthetnek, de nem szükségesek.

Van egy fontos összetevő, amit Garrett nem említett cikkében: a szerver oldali feldolgozás szükségessége. Az előbb felsorolt technológiák mindegyike közvetlenül a kliens oldali Ajax-motorhoz kapcsolódik, azonban nincs Ajax egy stabil, megfelelően visszaható szerver nélkül, amely arra vár, hogy tartalmat küldjön a motornak. Erre a célra a választásunk szerinti alkalmazásszervert használhatjuk. A szerver oldali összetevőket akár PHP oldalként, akár Java szervletként, akár .NET összetevőként írjuk meg, minden, amit biztosítanunk kell az, hogy a megfelelő adatformátum kerüljön visszaküldésre az Ajax-motorhoz.

> *A könyvben levő példák a lehető legtöbb szerver oldali technológiát alkalmazzák, hogy elég információt adjanak az Ajax kommunikációs rendszerének beállításáról a különböző szervereken. A könyvben szereplő példák többsége elérhető PHP, JSP és ASP.NET verziókban a* `www.wrox.com` *címen.*

Ki használja az Ajaxot?

Számos kereskedelmi webhely használ Ajax-módszereket a felhasználói élmények javítására. Ezek az oldalak tényleg inkább webes alkalmazások, mint hagyományos brosúrajellegű webhelyek, amelyek csupán információt jelenítenek meg, mert egy bizonyos cél elérése érdekében látogatjuk meg őket. A következőkben néhány olyan jól ismert és jól kidolgozott webes alkalmazásról lesz szó, amely Ajaxot használ.

Google Suggest

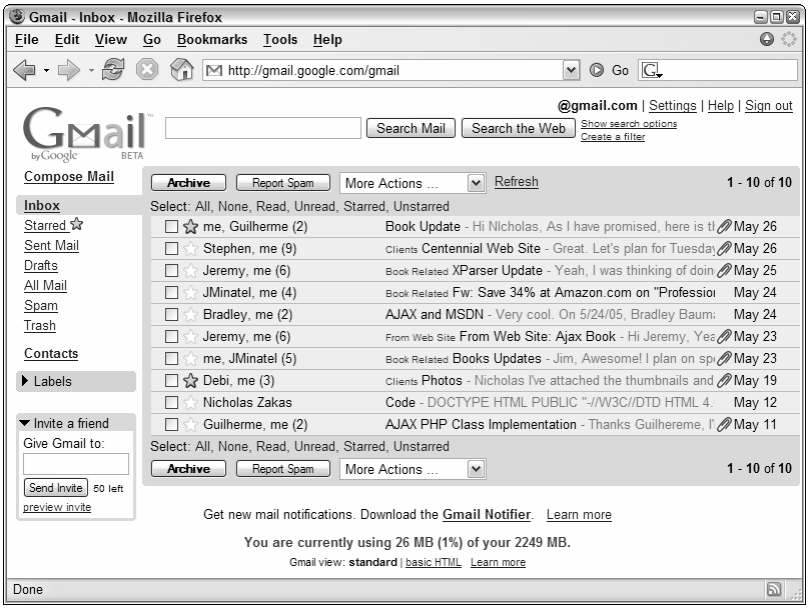
Az első példák egyike, amit a fejlesztők említenek az Ajaxról beszélve, a Google Suggest (`www.google.com/webhp?complete=1`). Az interfész a fő google interfész másolata, amely szembetűnően egy olyan szövegmezőt biztosít, amelybe keresési kifejezéseket írhatunk. Teljesen ugyanolyannak tűnik addig, amíg el nem kezdünk gépelni a szövegmezőbe. Gépelés közben a Google Suggest ajánlásokat kér a szervertől, mutatva egy legördülő listát a keresési kifejezésekről, amelyek érdekelhetnek minket. Minden javaslat az adott kifejezéshez rendelkezésre álló eredmények számával jelenik meg, hogy segítsen a döntésben (ld. 1.2. ábra).

1.2. ábra

Ezen egyszerű kliens-szerver kölcsönhatás nagyon hatásos és hatékony anélkül, hogy tolakodó lenne. Az interfész jóval érzékenyebb, mint amit megszokhattunk a webes alkalmazásoktól; attól függetlenül frissít, hogy milyen gyorsan gépelünk, és az asztali programoknál megszokott automatikus kiegészítésszolgáltatásokhoz hasonlóan a fel-le nyilakkal emelhetünk ki és jelölhetünk ki elemeket a javaslati listából. Noha még béta-verzió, várhatóan végül a fő Google oldalra is be fog kerülni.

Gmail

A Gmail-t, a Google ingyenes e-mail szolgáltatását az ügyfél-kiszolgáló kölcsönhatás csodájaként ünneplik az Ajax korában. Amikor először jelentkezünk be a Gmail-be, egy felhasználói felületmotor töltődik be az alkalmazás által használt néhány iFrame elem egyikébe. A szerverhez irányuló minden további lekérdezés ezen a felhasználói felületen, egy XMLHttp objektumon keresztül történik. Az átvitt adat JavaScript-kód, amely gyorsabb végrehajtást eredményez, amint egyszer letöltötte a böngésző. Ezen kérések utasításokként szolgálnak a felhasználói felület motorjának arra vonatkozóan, hogy mit kell frissíteni a képernyőn.



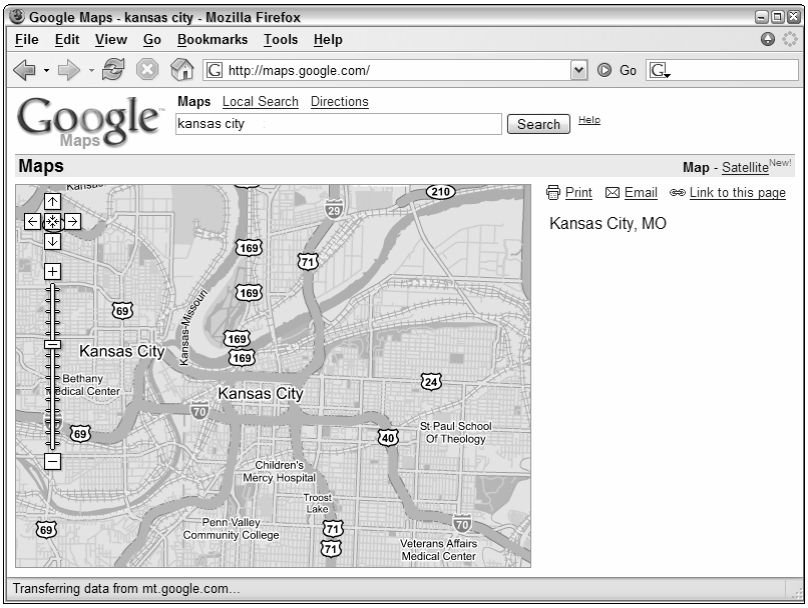
1.3. ábra

Emellett a Gmail-alkalmazás számos keretet és iFrame elemet használ, hogy kezelje és gyorsítsa a felhasználói felület változásait. A keretek szerfelett bonyolult használata lehetővé teszi, hogy a Gmail megfelelően működjön a Vissza és Előre gombokkal, ami a keretek vagy iFrame elemek használatának egyik előnye az XMLHttp használata helyett vagy amellett (később erről még lesz szó).

A Gmail legnagyobb eredménye a használhatósága. A felhasználói felület – ahogy az 1.3. ábrán is látható – egyszerű és rendezett. Az interakció a felhasználóval és a kommunikáció a szerverrel zökkenőmentes. Még egyszer: a Google az Ajax használatával fejlesztett tovább egy már egyszerű koncepciót, hogy kivételes felhasználói élményeket biztosítson.

Google Maps

A Google domináns Ajax webes alkalmazásainak egy másik része a Google Maps (`maps.google.com`). A jól megalapozott térképoldalakkal való versenyre tervezett Google Maps Ajaxot használ annak elkerülésére, hogy frissítse főoldalát (ld. 1.4. ábra).



1.4. ábra

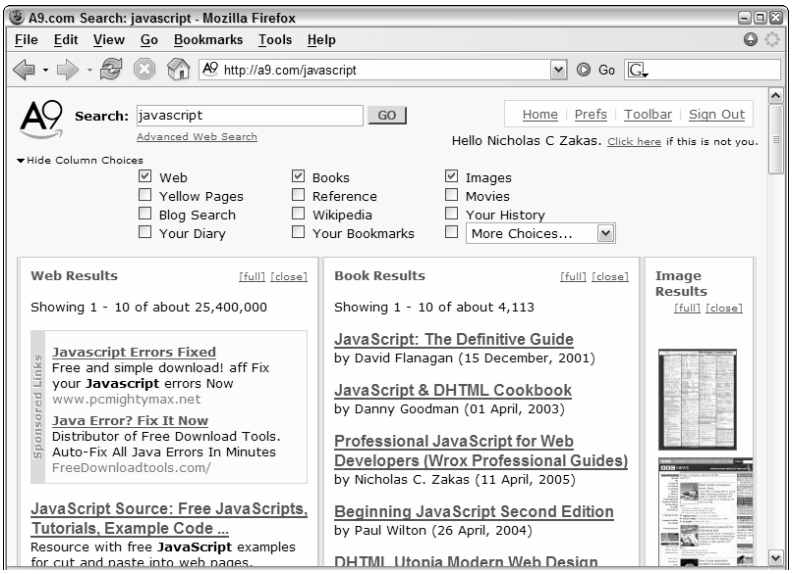
A többi webes térképalkalmazással ellentétben a Google maps lehetővé teszi, hogy a térképet húzással mozgassuk különböző irányokba. A húzást szolgáló kód nem új a JavaScript fejlesztőknek, de a térkép átfedése és a látszólag végtelen görgetési effekt már más történet. A térkép fel van osztva képek so-

rozatára, amelyek fedik egymást, hogy egy folyamatos kép érzetét keltsék. A térképet megjelenítő képek száma véges, mivel az új képek létrehozása minden egyes alkalommal, amikor a felhasználó megmozdítja az egeret, hamar memóriaproblémákhoz vezetne. Ehelyett ugyanazokat a képeket használja újra és újra a térkép különböző részeinek megjelenítésére.

A kliens-szerver kommunikáció egy rejtett iFrame-elemen keresztül történik. Amikor keresünk vagy új irányba mozgunk, az információ azon a rejtett iFrame-elemen keresztül távozik és tér vissza. Az adat XML-formátumban tér vissza, és átkerül egy JavaScript függvényhez (az Ajax-motorhoz), ami kezeli. Ezt az XML-t aztán különböző módokon lehet felhasználni: néhányat a térkép helyes képeinek meghívására használnak, néhányat XSLT használatával átalakítanak HTML-lé, és megjelenítik a főablakban. Az alsó sor az, hogy ez az összetett Ajax-alkalmazás 2006 vége óta a második számú célpont az internetes térképek között.

A9

Az Amazon.com világszerte híres online piactér szinte bármihez, de amikor kiadott egy keresőmotort, az kevés dicséretet és figyelmet kapott. Az A9 (`www.a9.com`) bemutatása a kibővített keresést fitogtatta, lehetővé téve különböző információk keresését egyszerre. Az internetes és képkeresésekhez az MSN-t használja. Emellett keres könyveket az Amazon.com-on és filmeket az IMDB-n (Internet Movie Database – kb. internetes filmes adatbázis). A Yellow Pages, Wikipedia és Answers.com keresések 2005. közepén debütáltak.



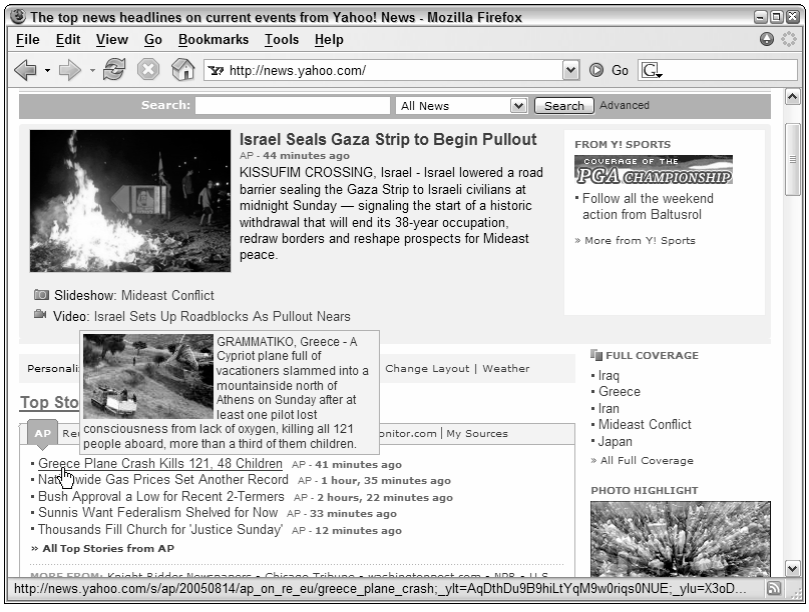
1.5. ábra

Ami egyedivé teszi az A9-et, az a felhasználói felületének működése. Amikor elvégzünk egy keresést, a különböző típusú eredmények az oldal különböző területein jelennek meg (ld. 1.5. ábra).

A keresési eredmények oldalon megvan a lehetőségünk arra, hogy más kereséseket hajtsunk végre ugyanazon feltételek használatával. Amikor kiválasztunk egy jelölőnégyzetet a keresés típusának megfelelően, a keresés a színfalak mögött zajlik rejtett iFrame-elemek és XMLHttp kombinációjának használatával. A felhasználói felület megváltozik, hogy több helyet adjon a további keresési eredményeknek, amelyek azonnal betöltődnek, amint visszatérnek a szervertől. Az eredmény sokkal információdúsabb keresési eredmény oldal, amelyet ráadásul nem kell újra betölteni, ha különböző típusú információkat akarunk keresni.

Yahoo! News

Szintén 2005-ben mutatták be a Yahoo! News oldal új dizájnját (`news.yahoo.com`). Az új dizájn érdekes fejlesztéssel szolgál: ha az egeret egy címsor fölé visszük, felbukkan egy kis doboz összefoglalással és tetszőlegesen egy képpel, amelyet az adott történethez társítottak (ld. 1.6. ábra).



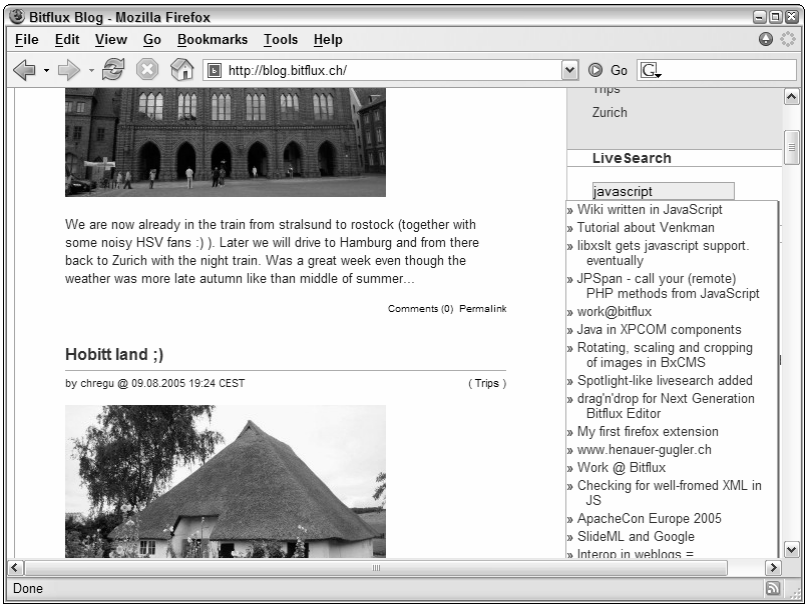
1.6. ábra

A fotóinformációt és az összefoglalást XMLHttp használatával keresik vissza a szerveren, és dinamikusan teszik be az oldalba. Ez tökéletes példája annak, hogy az Ajax hogyan használható egy weblap gazdagításához. Ahelyett, hogy az Ajax lenne az elsődleges használati mód, a Yahoo! News oldal tökéletesen használható Ajax nélkül; az Ajax-funkciót csak arra használják, hogy megfelelőbb felhasználóélményt biztosítsanak az azt támogató böngészőkben. Alatta egy szemantikailag helyes HTML-oldal van, amelyet úgy viteleztek ki logikailag, hogy még CSS-formázás sincs.

Bitflux Blog

Az Ajax mint puszta bővítmény használatának egy másik nagyszerű példája a Bitflux Blog (`blog.bitflux.ch`), amely a LiveSearch nevű technológiát szolgáltatja. A LiveSearch az oldalon levő keresőmezővel együtt működik. Ha gépelünk a mezőbe, a lehetséges keresési eredmények listája jelenik meg közvetlenül alatta (ld. 1.7. ábra).

A keresési eredményeket XMLHttp használatával keresik vissza, mint olyan HTML-sztringgel, ami be van építve az oldalba. A régi módszerrel is kereshetünk az oldalon: a szövegmező kitöltésével és az Enter megnyomásával. A LiveSearch Ajax-funkció csak egy bővítmény az oldalhoz, és nem szükséges a kereséshez.



1.7. ábra

Zűrzavar és ellentmondások

Az Ajaxnak népszerűsége ellenére is megvan a maga szép számú ellenzője és vitatója. Néhányan azt hiszik, hogy az Ajax egy tévút azon az úton, amerre az internet haladt, mielőtt az Ajax belépett volna a képbe. A szemantikai HTML-tervezés, a hozzáférhetőség és a tartalom és prezentáció elkülönítésének védelmezői sokan vannak a webfejlesztők között, és néhányan azt hiszik, hogy az Ajax népszerűsége a háttérbe szorította ezt a mozgalmat. Ezen ellenzők hite szerint az Ajax a prezentációk JavaScripten belüli létrehozását pártolja, ezáltal egy zavaros mixtúrává téve azt, a szerver oldali szkriptírás korai napjaihoz hasonlóan. Sokan azt hiszik, hogy a hozzáférhetőség kárt fog szenvedni, ha több fejlesztő is az Ajax felé fordul.

Mások jelentős időt töltenek Garrett cikkének elemzésével és feltételezéseinek megcáfolásával. Például a cikk az XML és XMLHttp ismétlődő használatát említi, mint az Ajax-modell magját, azonban a felsorolt példák nagy része nem használja ezeket. A Gmail és a Google Maps takarékosan bánik ezen technológiákkal; a Google Suggest csak XMLHttp-t használ, adatcseréhez pedig JavaScript tömböket XML helyett. A kritikusok arra is rámutatnak, hogy az Ajax technikai magyarázata a cikkben teljesen félrevezető, idézve számos technológiát, amely nemcsak hogy szükségtelen (mint az XML és XMLHttp), de sok esetben kifejezetten nem is szeretik használni őket (mint az XSLT-t).

Az Ajax-szal és Garrett Adaptive Path cikkével szembeni másik jelentős érv az, hogy ez pusztán új név egy olyan módszer számára, amelyet már használnak egy ideje. Noha ezen típusú adat-visszakeresés a Netscape Navigator 2.0-ban is használható volt, igazán jelentőssé csak 2001-2002 körül vált, különösen egy, az Apple's Developer Connection oldalon publikált cikk után, amelynek címe: „Remote Scripting With IFRAME" (megtekinthető a `http://developer.apple.com/internet/webcontent/iframe.html` címen). Sokak szerint ez az első mainstream cikk, amelyet az Ajax-szerű módszerekről publikáltak. A remote scripting kifejezés nem honosodott meg annyira, mint az Ajax.

Megint mások az Ajax-kifejezésen és Garrett cikkén gúnyolódnak, azt gondolván, hogy ez az alkotás nem más, mint Garrett cégének, az Adaptive Path LLC-nek egy marketingfogása. Egyesek szerint egy új név megalkotása egy már létező módszer számára kétszínű és a beteges szándék csalhatatlan jele. Tekintet nélkül erre és az Ajaxot övező más vitákra, a megközelítésnek most már olyan neve van, amellyel a fejlesztők gyorsan megismerkednek, és amely jobb megértést és magyarázatot igényel, hogy a lehető legjobb módon lehessen használni.

Az Ajax és a Web 2.0

Nem sokkal az Ajax-kifejezés megalkotása után egy másik kifejezés is felbukkant. A Web 2.0 eredetileg az O'Reilly Media és a CMP Media által közösen, 2005 végén szervezett konferencia neve volt. Ezután a Web 2.0 kifejezés önálló életre kelt, és internetszerte felbukkant olyan írásokban, amelyek az internet változásairól szólnak. Hogy megpróbálja féken tartani a kifejezést, mielőtt az irányíthatatlanná válna, Tim O'Reilly (az O'Reilly alapítója és elnöke) írt egy cikket „What is Web 2.0" címmel (a `www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html` címen olvasható), amelyben leírja, hogy szerinte milyen koncepciót képvisel a Web 2.0. Ez a koncepció tartalmazza:

- Az internetet mint szolgáltatást, nem mint programot.
- Az internet csoportmentalitását – a felhasználókat bátorítani kell a részvételre (címkézéssel, blogolással, hálózati műveletekkel stb.).
- Az adatok és a prezentáció elkülönítését – az adatokat akárhány módszerrel lehet jelképezni, és azok kombinálhatók más adatforrásokkal (amelyek neve mashups).
- Gazdagabb, megfelelőbb felhasználói élményt.

Az Ajax az utolsó ponthoz kapcsolódik, gazdagabb élményt adva a felhasználónak. Hogy világosak legyünk, az Ajax nem a Web 2.0 szinonimája és a Web 2.0 nem csak az Ajaxról szól; a Web 2.0 az internet igazi karakterisztikájának változását jelenti. Amíg az Ajax fontos része azon következő generációs felhasználói élmény létrehozásának, amit a Web 2.0 jelent, mégis csak egy darabja egy jóval nagyobb kirakós játéknak.

Összefoglalás

Ez a fejezet mutatja be az Ajax alapelvét. Az Asynchronous JavaScript + XML rövidítéseként használt Ajax-kifejezést Jesse James Garrett vezette be az Adaptive Path LLC webhelyén publikált cikkében. A cikk az Ajaxot egy olyan új felhasználói kölcsönhatásmodellként mutatta be a webes alkalmazások számára, amelyben a teljes oldalak betöltése többé nem szükséges.

Ez a fejezet feltárja az internet fejlődését azon technológiák függvényében, amelyek lehetővé tették, hogy az Ajax mára valóság legyen. Az Ajax a létezését a JavaScriptnek és a webböngészőkben levő keretek bemutatásának köszönheti, amelyek aszinkron adat-visszakeresést végeznek JavaScript használatával, ami elvileg már a Netscape Navigator 2.0-ban is jelen volt.

Az új internetes technológiák evolúciója révén olyan új Ajax-metódusok fejlődtek ki, mint a rejtett keret módszer. Az iFrame-elemek és az XMLHttp bemutatása volt az Ajax fejlődésének igazi mozgatórugója.

Noha az Ajax segítségével sokféle dolgot végre lehet hajtani, legfontosabb alkalmazási módja a felhasználói élmény javítása ahelyett, hogy frankó effekteket biztosítson. A fejezet tárgyalt néhány Ajax-alapelvet, melyek mindegyike arra utal vissza, hogy a felhasználók igényei minden másnál fontosabbak a webes alkalmazások fejlesztésében.

Néhány népszerű Ajax-alkalmazásról is volt szó, beleértve a Google Suggest-et, a Gmail-t, a Google Maps-t, a Yahoo! News-t és a Bitfux Blog-ot.

Végül a fejezet áttekintette az Ajax-szal, Garrett cikkével és az Ajax interneten elfoglalt helyével kapcsolatos vitákat. Néhányan úgy gondolják, hogy az Ajax népszerűsége a hozzáférhetőség általános hiányához fog vezetni, míg mások megkérdőjelezik Garrett motivációját híres cikkének megírására. Mint minden megközelítés, az Ajax is akkor legjobb, ha logikus bővítményként használják egy jól megtervezett webes alkalmazásban.