Раніше у обчислювальних системах типу клієнт-сервер кожен додаток мав свою власну програму клієнта, яка служить у якості користувацького інтерфейсу і повинна бути встановленою окремо на компютері кожного користувача.

А оновлення серверної частини вимагало обовязкового оновлення всіх клієнтських програм, що приводить до збільшення вартості підтримки та зниження продуктивності.

На відміну від веб-додатків, веб-документи, написані на стандартних форматах таких як HTML (XHTML) підримуються різними веб браузерами.

Як правило кожна веб-сторінка доставляється клієнту у вигляді статичних документів Але послідовність сторінок забезпечує достатню інтерактивність роботи. В ході сесії веб-браузер інтерпретує сторінки і виступає в якості універсального клієнта для будь - якого веб додатка.

У 1995 році Netscape представила мову сценаріїів на стороні клієнта під назвою JavaScript. Який дозволив програмістам додавати деякі динамічні елементи інтерфейсу користувача, який працював на стороні клієнта. До тих пір всі данні повинні були відправлятись на сервер для обробки, а результати доставлялись за допомогою статичних сторінок користувачеві.

У 1996 році Macromedia представила Flash - програвач векторної анімації. Який міг бути доданий у браузер для відображення анімації вмонтованої у веб-сторінках. Вона дозволила використовувати мову сценаріїв для програми взаємодії на стороні клієнта, без необхідності спілкуватися з сервером.

У 1999 році вроваджено концепцію Java у специфікації сервелету. В цей час JavaScript та xml були вже добре сформовані, а ввесь Ajax був заключений у обєкті XMLHttpRequest який підтримувався лише IE 5 Як частина технології ActiveX

У 2005 році було впроваджено повноцінну технологію Ajax, і такі сервіси як, наприклад, gmail почали розвивати свої клієнтські частини, роблячи їх все більш інтерактивними.

На сьогоднішній день, у доповнення до Ajax, а можливо і на його заміну впроваджується технологія WebSocket.

На відміну від звичайних програм, які в оснвному складаються із одного рівня який знаходиться на машині клієнта, структура веб-додатків дещо складніша. Найчастіше вона трирівнева, і складається із таких рівнів:

1 - "презентація" браузер.

2 - "виконання" (інтерпретація) двигуни що використовують динамічний зміст веб-технологій (ASP, ASP.NET, CGI, ColdFusion, JSP / Java, PHP, Perl, Python)

3 - "збереження" роль безпосередньо бази даних на сервері.

Існує також багато web application frameworks які дозволяють створювати тіж самі продукти використовуючи елементи мови більш високого рівня. Вони дозволяють зробити код більш "оптимальним", а також зменшити число помилок у програмі, або спияти використанню новіших методів та технологій.

Браузерні програми зазвичай включають просте програмне забезпечення офісу (текстові редактори, презентації, електронні таблиці). Найбільш яскравим прикладом тут є серівси google docs. Але зустрічаються і більш розвинені програми , наприклад для роботи із графічними та відео файлами, адміністрування проектів, тайм менеджмент і багато іншого.

Що стосується вимог клієнтської частини, то вони просто мізерні. Браузери зазвичай мають мінімальні вимоги до апаратного забезпечення, автоматично оновлюються і доповнюються. Також надають кросплатформенну сумісність (Windows, Mac, Linux)

При всій своїй зручності, компактності иа швидкості роботи, мають місце і певні застереження щодо web. Враховуючи колосальну їх розповсюдженість, і розмаїття розробників програмного забезпечення, кожен хто хоч трохи відхиилиться від загальноприйнятої тендінції, ризикує "вилетіти з колії" не знайшовши застосування своїм розробкам, або ж викинути із неї декого, започаткувавши нові тенденції, які досить швидко стають загальноприйнятими. Для того щоб процес розвитку не ішов навмання, а мав цілеспрямовний характер, існує ряд спеціальних організації які займаються координацією та управлінням розвитку web.

Найбільш значущими із них є:

W3C - (world Wide Web Consortium) - Консорціум світової павутини, розробляє і впроваджує технологічні стандарти до www.

ISOC (Internet Society) Співтовариство інтернету. Надає організаційну основу для багатьох інших груп. Заснована для забезпечення корпоративної структури організацій що займаються розвитком інтернету, і які часто є неформальними, потребують фінансової підтримки і правового статусу. (такі як IETF)

IESG (The Internet Engineering Steering Group) Група по розробці інженерного регламенту інтернету. Яка відповідає за технічне керівництво діяльністю IETF і процесом стандартизації інтернету. Як підрозділ ISOC відповідає за прийняття нових специфікацій в якості стандартів інтернету із дотриманням всіх встановлених процедур.

IAB (Internet Architecture Bound) Рада по архітектурі інтернету. Група технічних радників ISOC яка здійснює:

- консультації ISOC по технічній архітектурі і процедурних питаннях.
- нагляд за архітектурою інтернету.
- нагляд за тсворенням нових стандартів.
- редакцію і публікацію RFC

IRTF (Internet Research Task Force) Дослідна група інтернету. Технологічний підрозділ IAB який виконує довгострокові дослідні програми повязані з питаннями розвитку архітектури, базових протоколів і мережевих доповнень інтернет.

IETF (Internet Engineering Task Force) Відкрите міжнародне співтовариство проектувальників, учених, мережевих операторів і провайдерів створене IAB у 1986 році яке займається розвитком протоколів і архітектури інтернету. Результати їх роботи оформляються у вигляді робочих проектів (Internet Drafts) які потім використовуються ISOC.

RFC (Request For Comments) Запит коментарів. Документ що містить технічні специфікації та стандарти. Ним безпосередньо користуються у всесвітній мережі.

Такими ж відповідними Драфтами та RFC оформлені нещодавно затверджені стандарти PHP5, які вкоючають у себе технології, та рекомендації до застосування нових розробок, у тому числі і webSocket.

Не зважаючи на те, що у роботі Web-сервісів значна на сьогодні обробляється на клієнтській стороні, Вузьким місцем усе ж залишається канал спілкування із сервером. Який здійснюється через HTTP протокол. І навіть при застосуванні технології Ajax для обміну даними потрібно мимоволі передавати значні (на рівні програми) об'єми, які займає службова інформація для ініціалізації з'єднання та коректної передачі даних.

Також паралельно із передачею запитів і отриманням відповідей по основному каналі спілкування із сервером, існує потреба в оперативному отриманні деяких побічних даних від сервера.

Для таких цілей було розроблено так звану технологію Comet або як її ще називають HTTP Streaming.

Ідея її застосування досить проста - відкрити і підтримувати додаткове TCP/IP зєднання між сервером та клієнтом, по якому сервер буде за необхідності "скидати" нові повідомлення, а клієнт у свою чергу буде отримувати їх та обробляти. З точки зору користувача - все ідеально. Браузер майже миттєво реагуватиме на зміну на сервері. Але з іншої сторони. Ця технологія може бути несумісною із деякими видами слієнт-серверних систем. А також потребує значного ускладнення клієнтського і серверного коду. Що є не зовсім практичним.

Подібні реалізації, це фактично деяка надбудова для роботи із уже існуючою технологією, а краще сказати екстенсивний її розвиток.

Зовсім недавно компанія google яка уже звично стала ініціатором "модних" тенденцій web середовища оголосила про реалізацію технології WebSockets у своєму браузері Chrome версії 4.0.249.0. Ця технологія буде доступною і ввімкненою за замовчуванням.

Розглянемо що ж принципово нового пропонує нам google.

На відміну від XMLHttpRequest сокети забезпечують реальний двонаправлений канал звязку у браузері.

Для цього необхідно згенерувати зовсім нескладну процедуру "спілкування" :

```
if ("WebSocket" in window) {
    var ws = new WebSocket("ws://example.com/service");
    ws.onopen = function() {
        // Web Socket підключено. Можна відправляти дані методом send().
        ws.send("Повідомлення для відправки");
    };
    ws.onmessage = function (evt) { var received_msg = evt.data; ... };
        //викликається щоразу як сервер отримує якісь данні
    ws.onclose = function() { // websocket закритий. };
} else {
    //Повідомлення для браузерів які не підтримують WebSocket.
}
```

Підключення відбувається як і при звичайному HTTP запиті. Браузер підключається по протоколу TCP на порт 80 сервера і віддає GET запит:

```
GET /demo HTTP/1.1
Upgrade: WebSocket
Connection: Upgrade
Host: site.com
Origin: http://site.com
```

Якщо на сервері здійснюється підтримка Сокетів, він відповідає:

```
HTTP/1.1 101 Web Socket Protocol Handshake
Upgrade: WebSocket
Connection: Upgrade
WebSocket-Origin: http://site.com
WebSocket-Location: ws://site.com/demo
```

Якщо браузера це влаштовує, то він просто залишає зєднання відкритим, і все. Канал зєднання залишається відкритим

Як тільки ви відкриєте зєднання веб сокета, можете спокійно передавати дані від браузера до сервера викликаючи метод send(), та отримувати дані від сервера за допомогою обробника подій OnMessage.

Інформація в обидві сторони передається за допомогою короткої текстової стрічки до якої з дівох боків приставлено нулевий та кінцевий байти:

0x00, <тескт із кодуванням utf-8>, 0xff

При цьому зміст повідомлення може бути будь-яким. І щоразу як браузер отримуватиме повідомлення від сервера, буде задіяно OnMessage.

У сокетах також передбачено передавання бінарних даних:

Поки розробники працюють над впровадженням підтримки на своїх серверах та браузерах.

0x80 <байтова довжина> <повідомлення>

Довжина повідомлення встановлюється послідовністю байтів, у кожному з яких, старший біт встановлюється 1, а кінець позначається старшим бітом рівним 0. Таким чином не встановлюється ніяких обмежень на довжину повідомлення, і одночасно не витрачаються байти на її резервування.

Завдяки такому механізму, протокол працює миттєво, і не потребує передавання додаткових заголовків, та затрат на втсановлення зєднання з сервером.

Ще однією перевагою сокетів є необмежений час встановленого TCP з'єднання. Тобто воно може необмежений час знаходитися відкритим у пасивному стані, а отже не потрібно витрачати зайві ресурси на його оновлення та підтримку каналу в активному стані.

Крім того не існує обмеження на кількість одночасно відкритих сокетів. Адже, як відомо, у HTTP передбачено обмеження на число одночасно відкритих сесій до одного сервера. Через це, якщо на вашій сторінці багато різних асинхронних блоків, доводиться використовувати мультиплексорні схеми, щоб уникнути відмови в обслуговуванні. Із використанням Socket можна відкривати скільки завгодно каналів звязку.

У сокетах також відсутні проблеми крос-доменного зв’язку. Технологія здійснюється не за принципом "із того ж джерела", а "із дозволеного джерела", і визначається не на клієнті а на сервері. Тобто при встановленні з’єднання передається інформація звідки хочуть підключитись до вашого сокету, і якщо ця адреса вас не влаштовує, то у з’єднанні їй буде відмовлено.

Отже дізнавшись про всі переваги цієї чудо-технології може виникнути питання, чи можна просто зараз її випробувати. Адже це новинка, і як уже було сказано вище, підтримується вона лише на Chrome. А інші розробники тільки працюють над впровадженням підтримки на своїх серверах та браузерах.

Але тут google знову про все подбав.

Для підтримки на серверній стороні, був розроблений спеціальний пакет pywebsocket який можна використовувати як розширення для сервера Apache, або запустити як автономний сервер.

А для браузерів створено бібліотеку web-socket-js яка емулює роботу веб сокетів за допомогою флеш.

Таким чином, кожен бажаючий може випробувати можливості нової технології, особисто переконатися в її перевагах та допомогти виправити можливі недоліки.