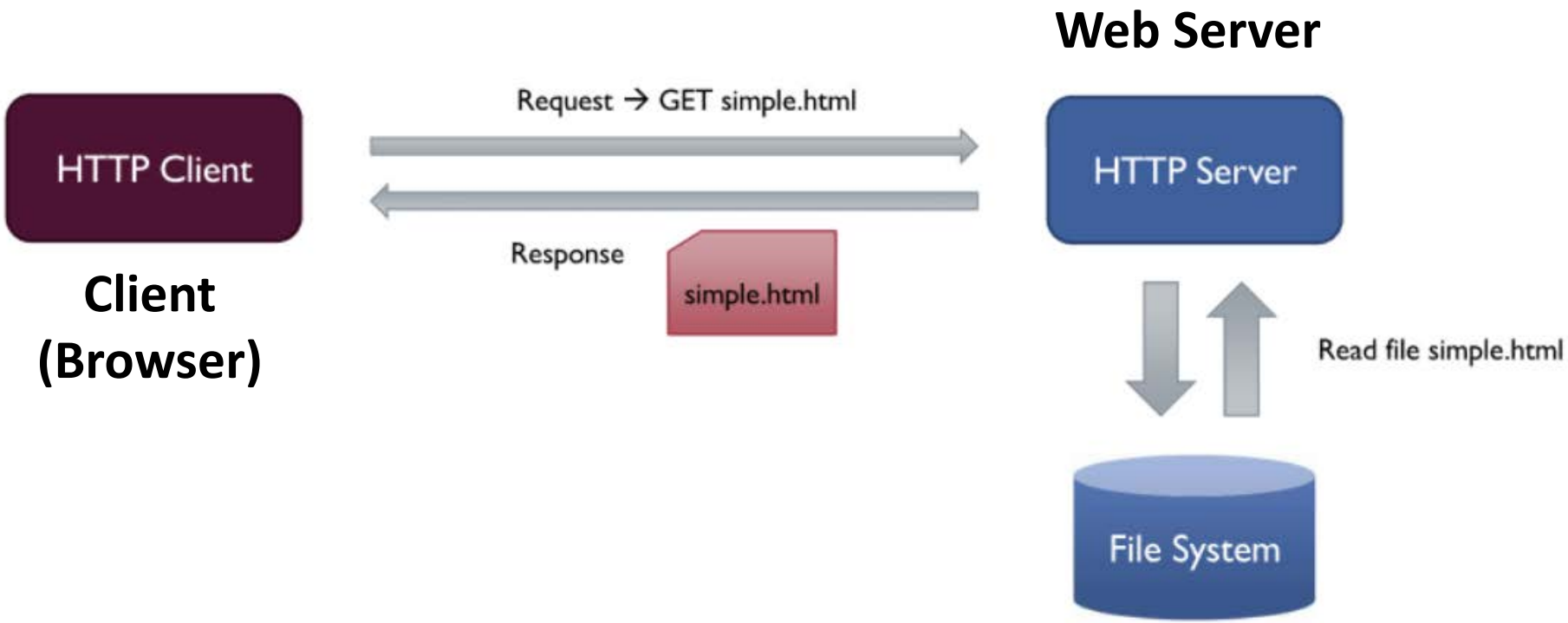
Evoluzioni e soluzioni architetturali per il Web

Summary

- Browser: evoluzioni
- Web Server: evoluzioni
- Soluzioni Architetturali per il WWW
  - Evoluzioni
  - Web Solution Stack

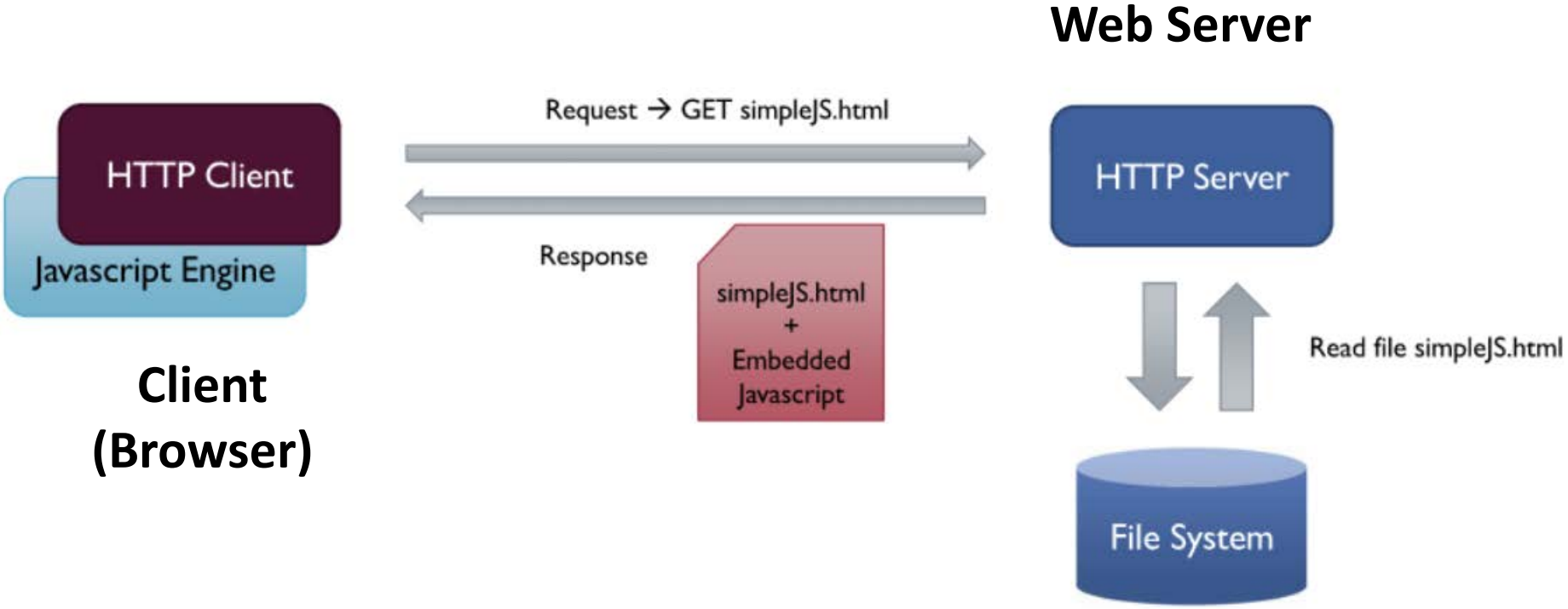
Architettura Web (di base)



Browser

- Il **browser** è client HTTP e un visualizzatore di documenti ipertestuali e multimediali:
  - Inizia l'interazione (come client HTTP)
  - Può renderizzare testi, immagini e semplici interfacce grafiche
  - Può permettere di editare (solo localmente)
  - Può includere:
    - *plug-in* che permettono di visualizzare file in formati particolari (ad esempio: PDF, Word, ecc.)
    - *estensioni* che permettono l'aggiunta di specifiche funzionalità non disponibili originariamente
    - un *linguaggio di programmazione interno* (Javascript) che permette di realizzare semplici verifiche di dati oppure complicate applicazioni autosufficienti (rich client application)

Architettura Web
(con scripting client-side)



Browser: evoluzione

- I precursori dei browser nascono negli anni ‘80, a partire dalle idee di Ted Nelson (MEMEX)
- Il primo browser vero e proprio risale al 1990, sviluppato da Tim Berners-Lee, al CERN: **WorldWideWeb** (successivamente rinominato **Nexus**). Era composto da:
  - Client HTTP
  - Layout engine
  - *Editor WYSIWYG*
- Il primo browser grafico fu **Mosaic**

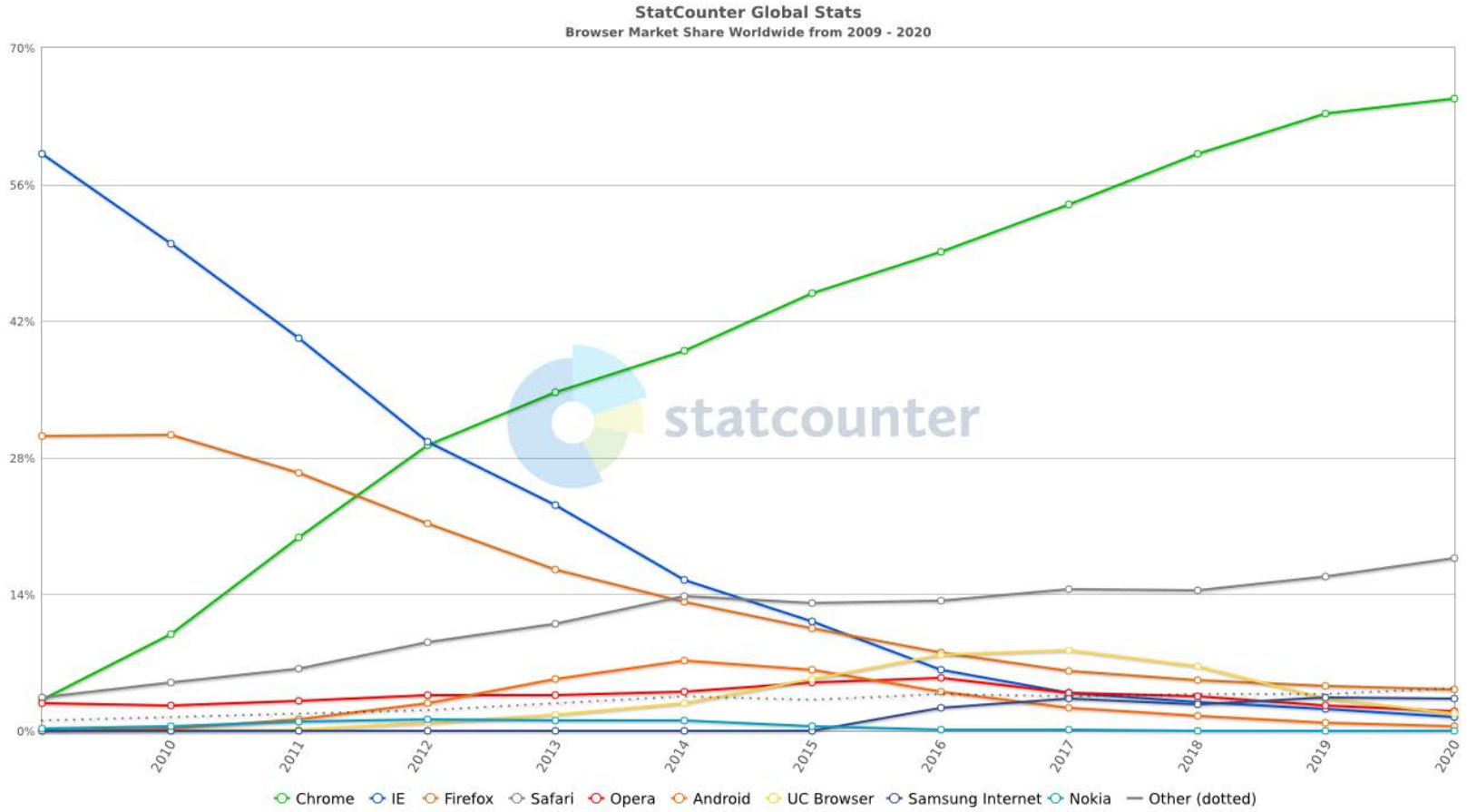
Browser: evoluzione

- Prima guerra dei browser:
  - Indicativamente dal 1994 al 1997
  - Contendenti: Microsoft Internet Explorer e Netscape Navigator
  - Vincitore: Internet Explorer
  - Dal punto di vista tecnico: rilascio continuo di nuove modifiche agli standard, spesso incompatibili tra loro, costringendo l'utente a navigare con un determinato browser per poter fruire delle nuove funzionalità (a seconda di tag e attributi usati dall'autore delle pagine).
  - Azioni commerciali: Microsoft iniziò a distribuire IE includendolo nel sistema operativo Windows 95. Gli utenti avrebbero dovuto installare esplicitamente Netscape Navigator (molti non ne conoscevano l'esistenza)

Browser: evoluzione

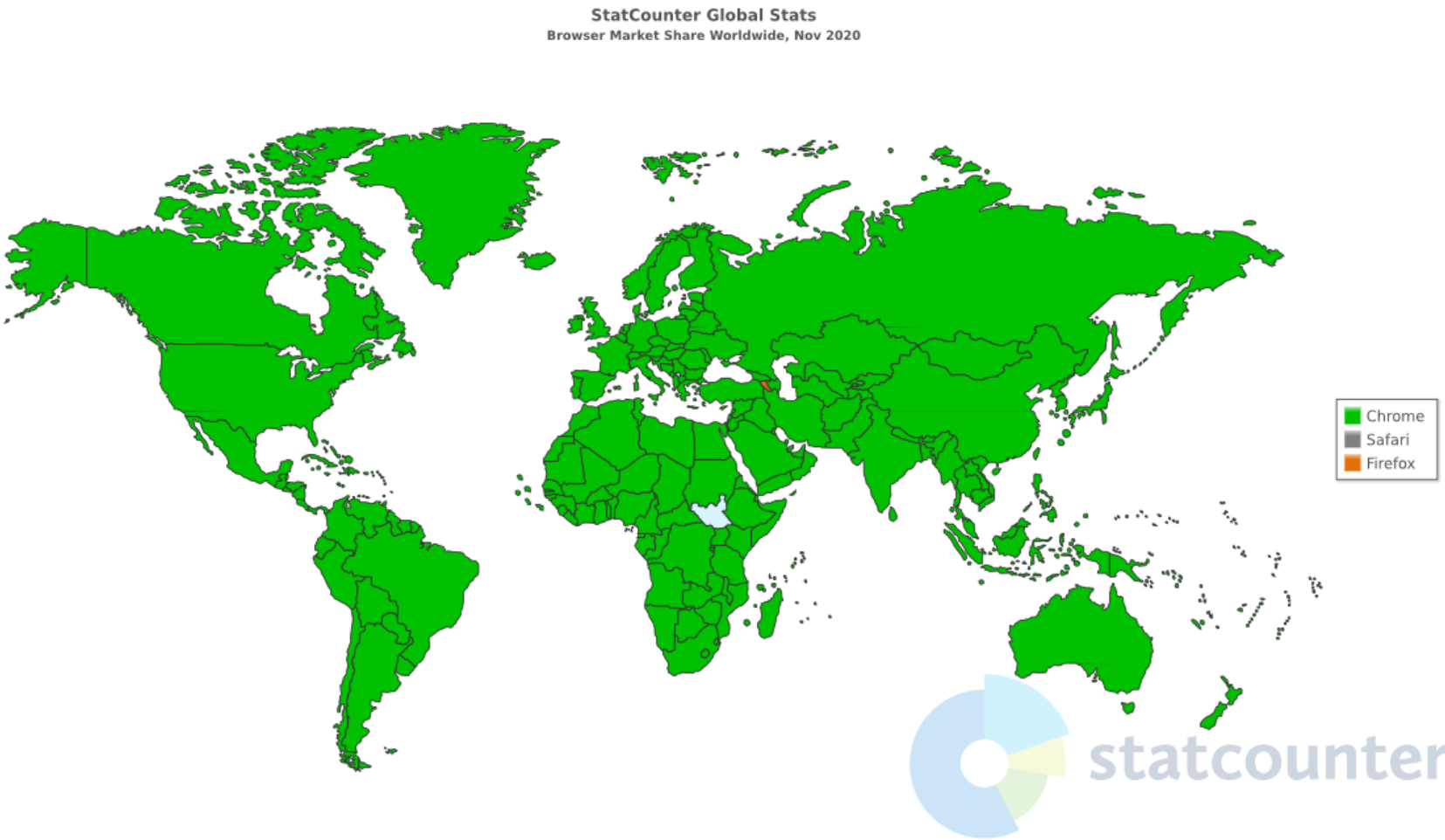
- Seconda guerra dei browser:
  - Indicativamente dal 2004
  - Internet Explorer, senza la minaccia di un avversario competitivo, ha continuato lo sviluppo in maniera irregolare, senza introdurre cambiamenti importanti
  - Iniziano ad affermarsi browser dotati di caratteristiche innovative, con maggiore rispetto degli standard W3C, alcuni gratuiti e open source, conquistando una fetta consistente del mercato
    - Mozilla Firefox
    - Opera
    - Safari
- Nel 2008 viene introdotto Chrome

Diffusione Browser



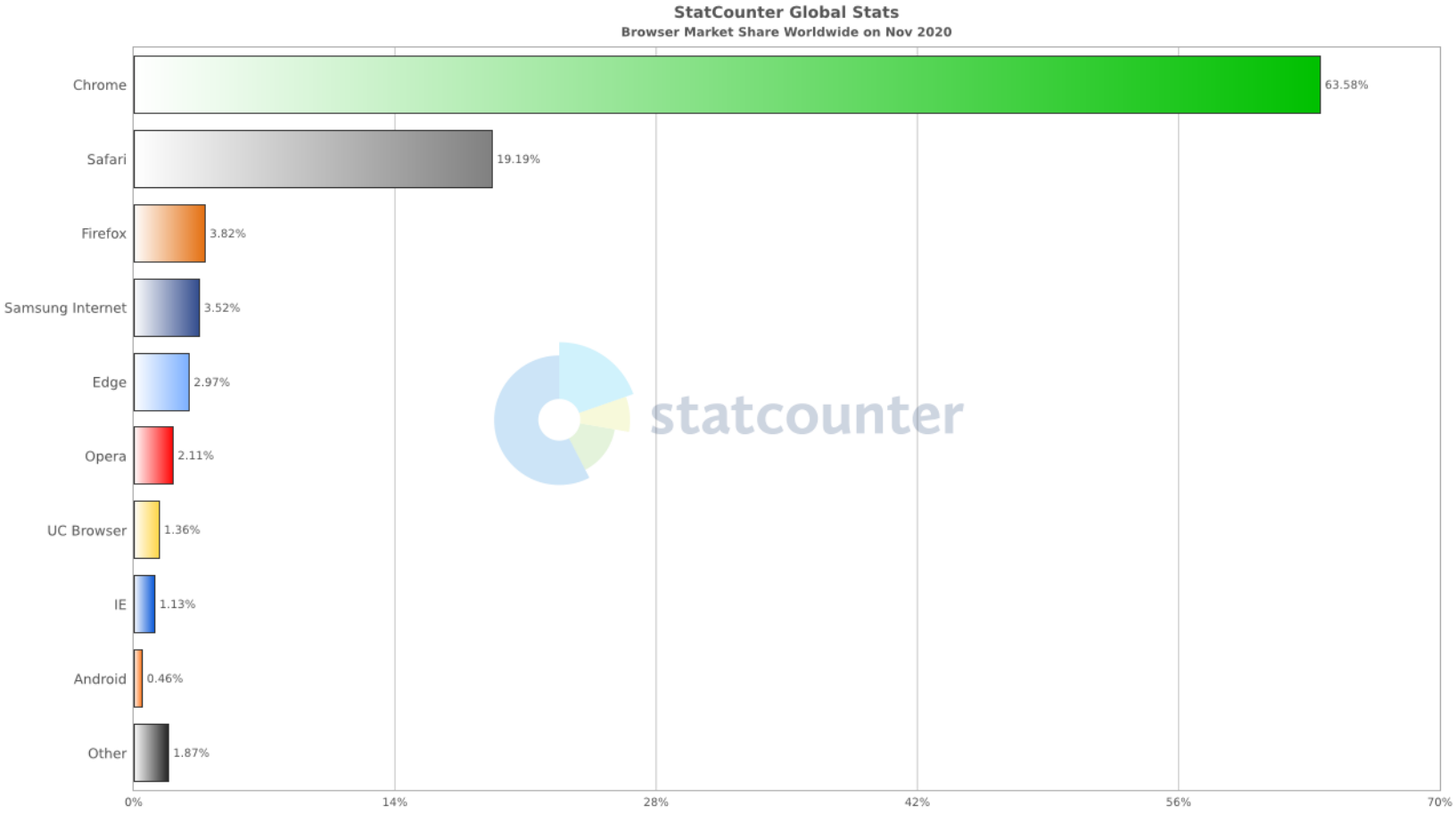
By StatCounter generated image - https://gs.statcounter.com/browser-market-share#yearly-2009-2020, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=96191474

Diffusione Browser



By StatCounter generated image - https://gs.statcounter.com/browser-market-share#monthly-202011-202011-map, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=96191830

Diffusione Browser

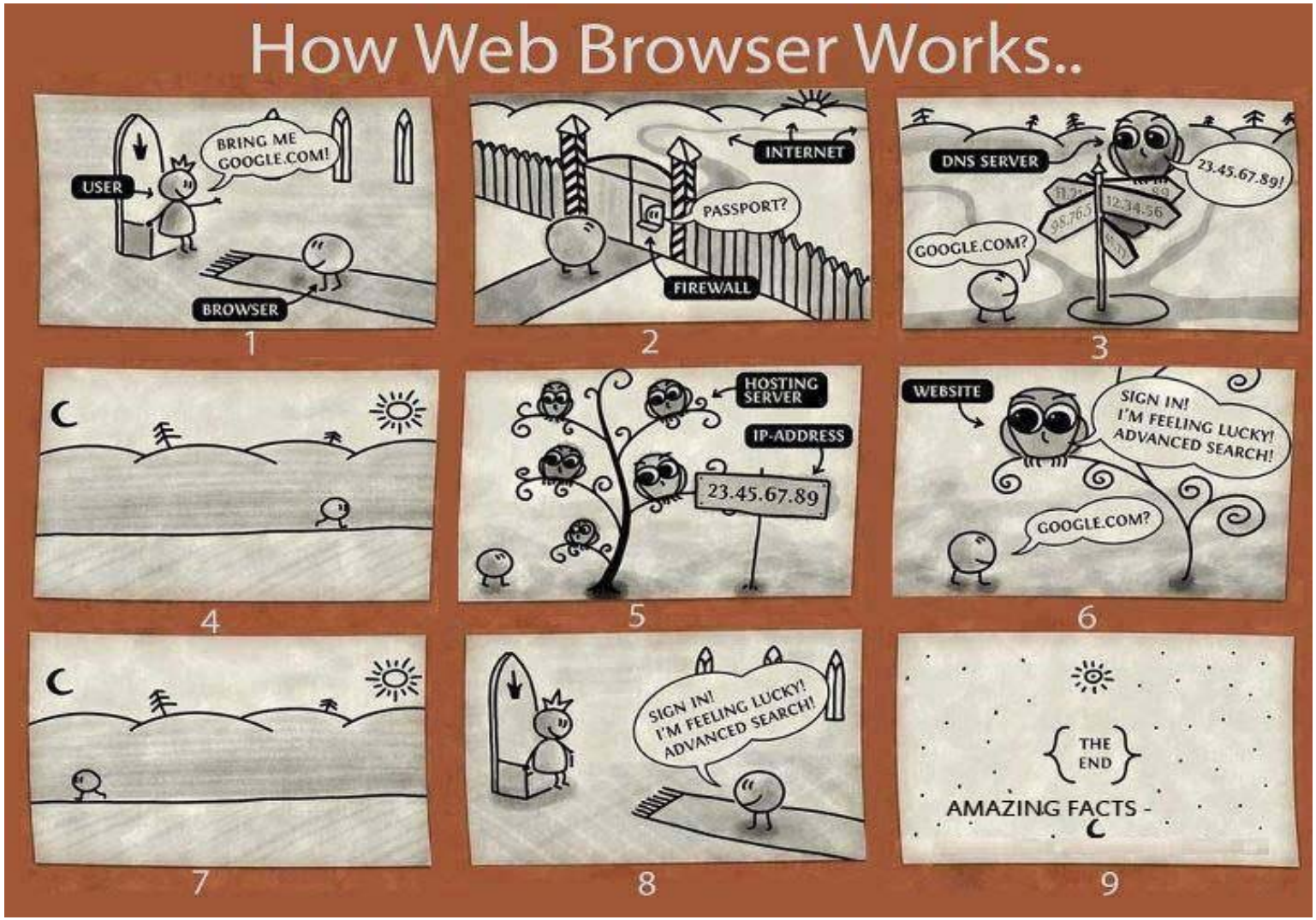


Browser: componenti principali

- Client HTTP
- Interfaccia utente (e backend dell'interfaccia utente)
- Motore di rendering e di layout
- Interprete JavaScript

- Riprenderemo questi componenti e li studieremo in dettaglio nel corso delle prossime lezioni

Browser


How Web Browser Works..
USER
BRING ME GOOGLE.COM!
BROWSER
1
INTERNET
PASSPORT?
FIREWALL
2
DNS SERVER
23.45.67.89!
GOOGLE.COM?
3
4
HOSTING SERVER
IP-ADDRESS
23.45.67.89
5
WEBSITE
SIGN IN!
I'M FEELING LUCKY!
ADVANCED SEARCH!
GOOGLE.COM?
6
7
SIGN IN!
I'M FEELING LUCKY!
ADVANCED SEARCH!
8
THE END
AMAZING FACTS -
9

Q&A

- Ci sono domande?

Server Web

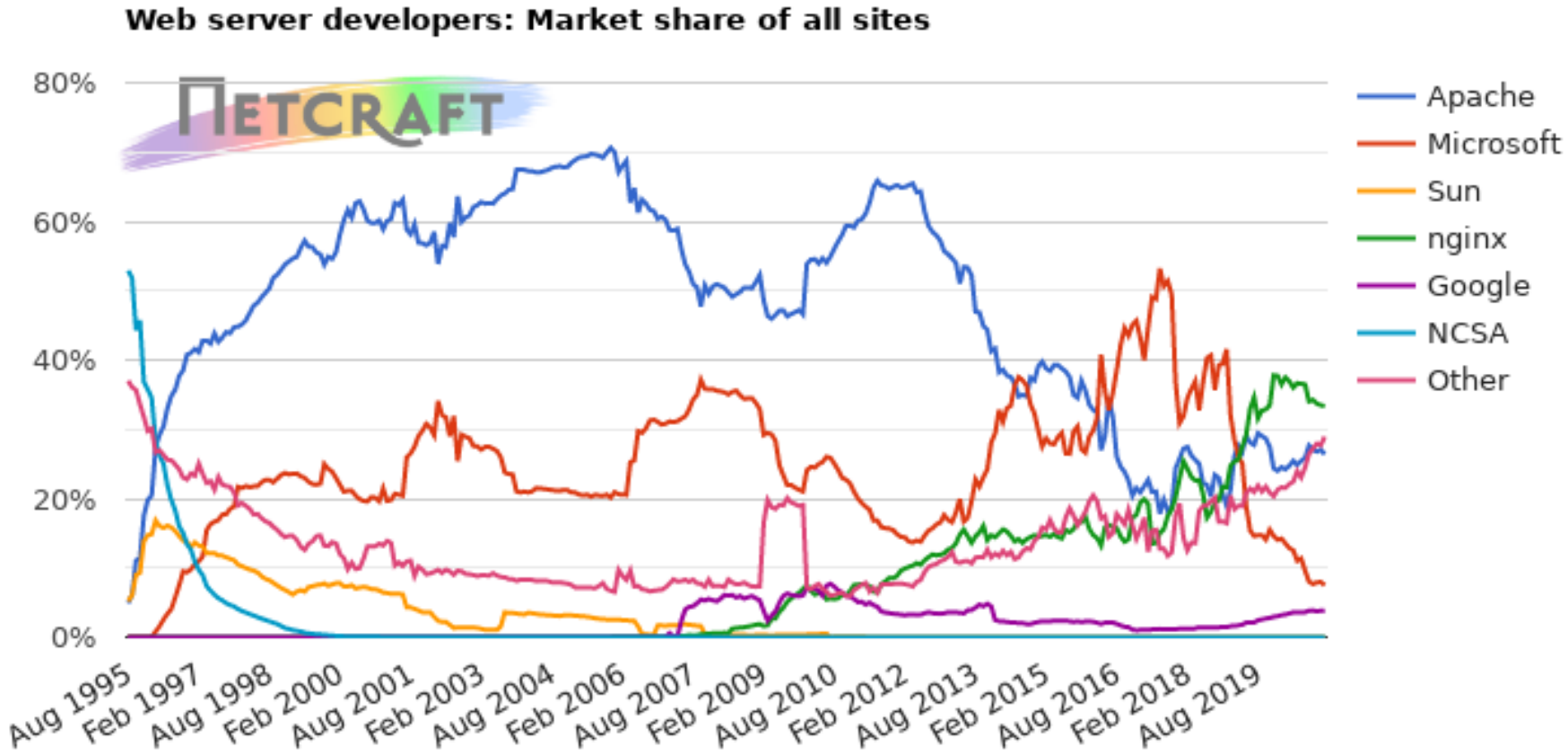
- Il server Web è una applicazione che risponde a richieste di risorse locali (file, record di database, ecc.) individuate da un identificatore univoco
- La funzione primaria del server web è rispondere alle richieste di pagine effettuate dai client (browser)
- Il server può collegarsi ad applicazioni server-side ed agire da tramite tra il browser e l'applicazione che opera sul server in modo che il browser diventi l'interfaccia dell'applicazione

Server Web: principali funzionalità

- Un server web deve ospitare, processare, effettuare il delivery delle pagine Web ai browser
- Server HTTP: riceve le richieste da parte del client e manda le risorse al client in risposta alle richieste ricevute
- Le risorse sono documenti HTML, che possono includere immagini, fogli di stile, script

- Riprenderemo le principali funzionalità dei Server Web nel corso delle prossime lezioni

Server Web: diffusione



https://news.netcraft.com/archives/category/web-server-survey/

Architetture del World Wide Web

- Sito web statico
- Sito web dinamico: server-side include (SSI)
- Sito web dinamico: modello a tre livelli
  - Embedded code
  - Full application
- Sito web dinamico: modello a quattro livelli
- Rich client: applicazioni AJAX (considerando anche API di HTML5)

Sito web statico

- Il server contiene una quantità di file fisici memorizzati in directory e di formati immediatamente riconoscibili dal browser (HTML, GIF, JPEG, ecc.)
- C'è esattamente un file per ogni schermata possibile, ciascuno con un indirizzo (un URL) diverso
- Il client richiede questi file ad uno ad uno e li riceve per la visualizzazione
- Nessun contenuto visualizzato cambia rispetto al documento memorizzato su disco

- **Pregi**: facile da realizzare, non richiede nessuna competenza tecnica
- **Difetti**: totale mancanza di automazione e integrazione. Ogni file è indipendente dagli altri

Siti dinamici: server side include (SSI)

- Il sito comprende una quantità di file fisici memorizzati in directory e di formati immediatamente riconoscibili dal browser
- C'è esattamente un file per ogni schermata possibile, ciascuno con un indirizzo (un URL) diverso
- All'interno dei file HTML ci sono commenti speciali. Ad esempio:

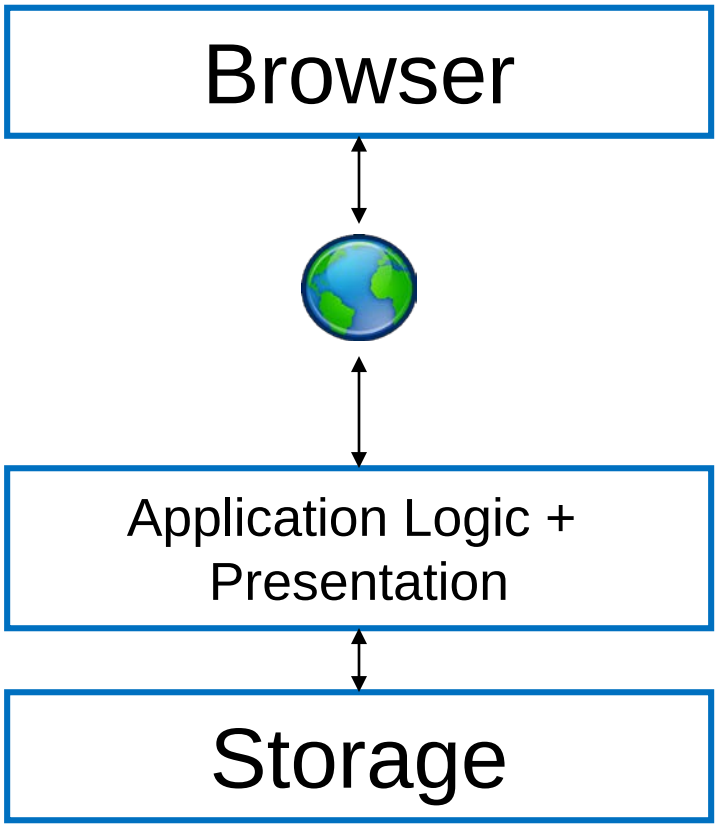
```
<!--#include virtual="header.html"-->
<!--#flastmod virtual="index.html"-->
```

- Questi corrispondono a specifiche azioni che vengono effettuate dal server subito prima della spedizione del file richiesto. C'è un vocabolario molto limitato di azioni che possono essere eseguite

Siti dinamici: server side include (SSI)

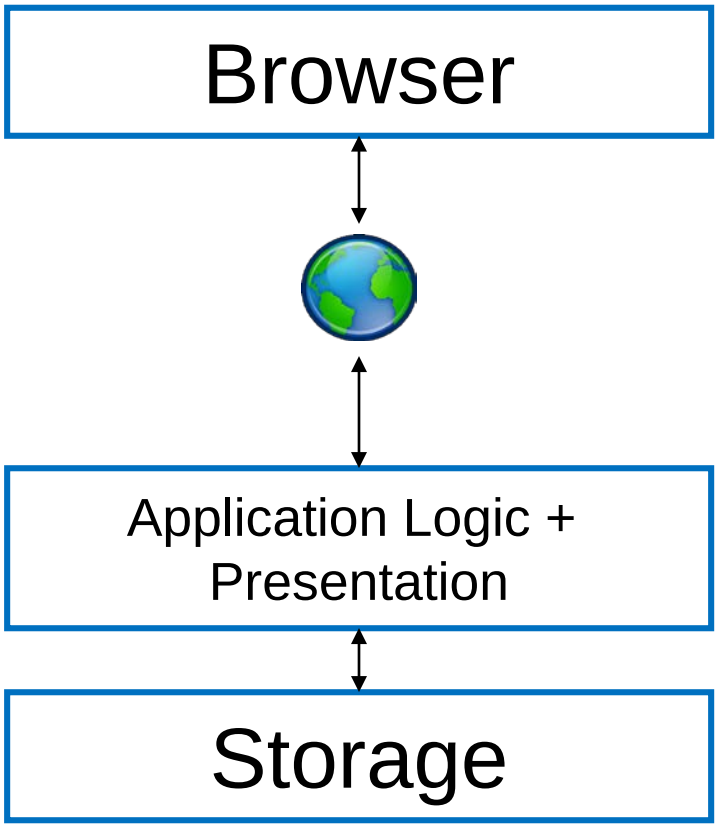
- **Pregi**:
  - fornisce una qualche forma di modularità
  - È ancora molto semplice da usare
- **Difetti**:
  - troppo limitato per essere veramente utile in contesti generali
  - Posso solo modularizzare frammenti che si devono ripetere su più file e realizzare micro-espressioni di contenuti dinamici

Siti dinamici: modello a tre livelli



- Parte dei contenuti è statica e messa su file, ma la parte importante è generata in output da un'*applicazione server-side*
- Questa spesso raccoglie dati da query su un DBMS, le elabora e le trasforma, e le spedisce come risposta al browser

Siti dinamici: modello a tre livelli



- L'*application logic* (a volte business logic) si modularizza rispetto allo storage per poter usufruire di velocità e funzionalità dei DB indipendentemente dalla logica dell'applicazione
  - Embedded code
  - Full application
- Tipicamente associata a modelli applicativi di tipo LAMP

Q&A

- Ci sono domande?

Tre livelli: embedded code

- Il sito Web è in realtà un'applicazione
- Contiene alcuni file fisici memorizzati in directory e formati immediatamente riconoscibili dal browser.
- C'è un file per ogni tipo di funzionalità (servizio) offerto.
- I file HTML contengono commenti speciali con codice inserito in qualche linguaggio di programmazione (PHP, ASP, ecc.).
- La parte puramente HTML è ripetuta su ogni risposta spedita dal server, mentre le istruzioni embedded generano codice HTML di volta in volta diverso.

Tre livelli: embedded code

```
<html>
    <head><title>PHP Test</title></head>
    <body>
     <p>Stai usando il browser
      <?php echo $_SERVER['HTTP_USER_AGENT']; ?>.
     </p>
    </body>
</html>
```

Tre livelli: embedded code

- La singola pagina HTML può contenere molti blocchi di codice embedded
- La parte HTML è il template, mentre il codice fornisce la parte dinamica dell'applicazione
- Si deve fare attenzione a mantenere sincronizzati tutti i template per quel che riguarda la presentazione: un cambio di look al sito richiede di modificare ogni singolo file
- Questa commistione di codice e template può essere considerata fragile e poco pulita

Tre livelli: embedded code

- **Pregi**:
  - Si può realizzare qualunque applicazione server-side con poco sforzo e in maniera fortemente integrata ai template HTML
- **Difetti**:
  - architettura fragile: cambiamenti al codice possono esporre problemi di visualizzazione del template e viceversa modifiche al look possono cambiare il comportamento dell'applicazione

Tre livelli: full application

- Si attua una separazione forte tra logica di applicazione e presentazione.
- Risiedono fisicamente su file diversi e vengono modificati in momenti diversi del processo di produzione, garantendo l'interdipendenza.
- Il server esegue il file di programma e alla fine genera un singolo output dell'intera stringa del documento HTML

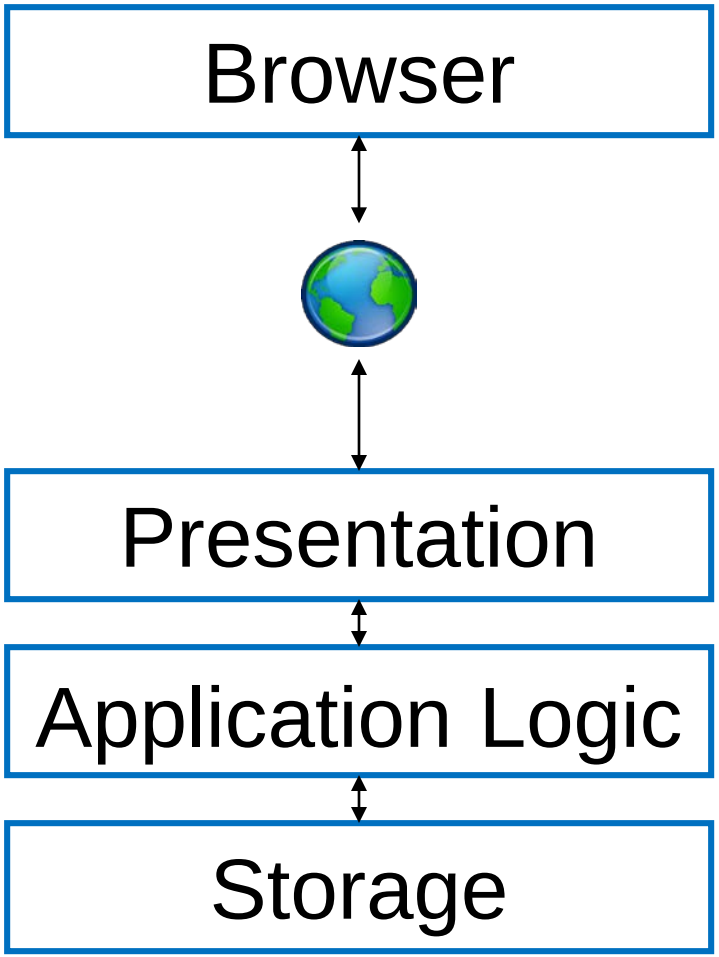
Tre livelli: full application

- Ci sono vari modi per ottenerlo, i più comuni sono:
  - Il template viene generato da istruzioni nel linguaggio usato, ma contenute in un file separato, che viene richiamato all'interno del programma che gestisce la logica principale dell'applicazione
  - Il template viene contenuto in un file HTML statico che viene letto in una variabile del programma e sapientemente mescolato con il contenuto HTML dinamico, generato dal programma lavorando sulle stringhe
  - Si usa un motore di template per PHP, ASP, ecc. Ogni produzione del programma diventa una variabile che viene usata all'interno del template, che è un file HTML. Il motore di template esegue il programma, poi carica il template e ne fa l'unione

Tre livelli: full application

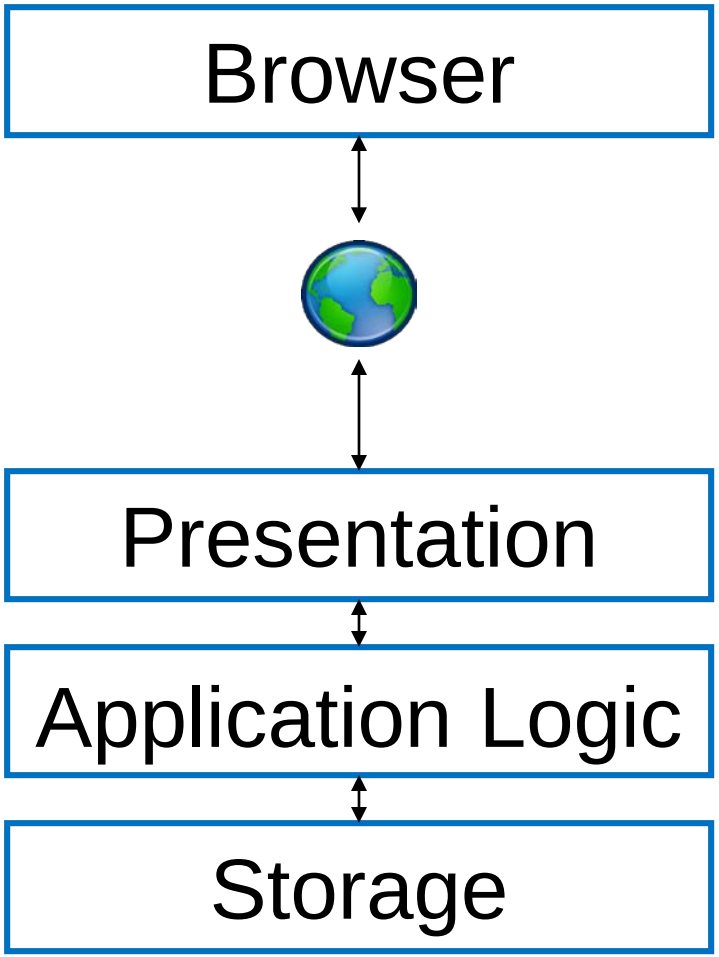
- **Pregi**:
  - si crea un'importante separazione tra application logic e presentazione: permette di separare i processi di generazione e test dell'applicazione da quelli di progettazione della parte visibile del sito
- **Difetti**:
  - Questa separazione non è ancora completa, si applica solo al template (layout complessivo, decorazioni, schemi di colori, parti fisse della pagina, ecc.). Il codice HTML di ogni singolo pezzo di output è ancora deciso dall'applicazione.
    - Ad esempio, che l'output di una query venga restituito come lista o come tabella è ancora una decisione dell'application logic
    - Ad esempio, generare output personalizzati per device molto diversi è ancora complesso

Siti dinamici: modello a quattro livelli



- Nel modello a quattro livelli, il livello di application logic genera un output completamente privo di aspetti presentazionali (ad esempio, un file XML) e lo dà in pasto a un altro livello, che chiamiamo *presentazione* (anche *presentation logic*)

Siti dinamici: modello a quattro livelli



- *Presentation* modifica completamente il documento posizionando le varie parti dell'output all'interno del template in maniera completa e assoluta, prendendo anche micro-decisioni riguardo a quale HTML usare di volta in volta
- Template diversi possono applicarsi allo stesso output per personalizzarsi su specifici tipi di device mantenendo assolutamente intatta l'application logic

Siti dinamici: modello a quattro livelli

- Esistono motori di template che funzionano in questa maniera
- Una soluzione utilizzata in passato era basata sull'uso di XSLT per generare il documento HTML finale
- Un processo della catena di produzione dell'output o l'ultima istruzione della application logic del programma richiamava il motore XSLT sul documento XML prodotto, e restituiva in output il documento HTML finale
- **Pregi**: completa e definitiva separazione tra application logic e presentation logic; architettura semplice, modulare, ripetibile
- **Difetti**: i difetti di tutte le applicazioni server-side

Q&A

- Ci sono domande?

Rich client: Ajax e il web 2.0

- Tradizionalmente la programmazione client-server ha sempre criticato la presenza di application logic sul client.
  - Poco controllo sull'ambiente operativo
  - Difficoltà di distribuzione di versioni e patch
- Il WWW ha alcune obiezioni a queste critiche
  - ogni passo dell'applicazione richiede di consultare il server, eseguire una funzione dell'application logic, generare l'HTML finale, riceverlo e visualizzarlo. Questo può portare via molto tempo.
  - Ogni passo dell'applicazione ha un proprio URL, che richiede specifiche politiche di naming e di caching piuttosto complesse da gestire
- Altre risposte specifiche alle critiche di cui sopra:
  - I browser sono ambienti indipendenti dal sistema operativo, e (più o meno) sono standardizzati
  - Il codice comunque può risiedere sul server ed essere distribuito al client ogni volta, in modo da garantire che giri sempre l'ultima versione

Rich client: Ajax e il web 2.0

- All'inizio dell'esecuzione del servizio, il browser carica una normale pagina HTML, che contiene codice Javascript e alcune librerie Ajax
- Scopo delle librerie Ajax è duplice:
  - Garantire di fornire una libreria di funzioni comuni e indipendenti dallo specifico sistema operativo e browser utilizzato
  - Fornire librerie utili per la realizzazione di applicazioni client side in maniera facile (ad esempio, l'interazione con il server o la generazione di frammenti di output)

Rich client: Ajax e il web 2.0

- Il documento HTML contiene solo le parti fisse (layout, ecc.) e le funzioni javascript.
- Il programma parte e attraverso un'apposita chiamata di libreria (*XMLHttpRequest*) chiede al server dati presentation-independent da convertire in HTML e visualizzare sulla pagina. Qui possiamo avere:
  - Esiste comunque un'applicazione server-side che genera l'XML e lo passa al client: cioè si sposta sul client solo la presentation logic
  - Sul server esiste solo il minimo indispensabile per fare interrogazioni ai server e ottenere le risposte alle query (si spostano sul client sia la presentation logic sia la application logic)

I framework

- Librerie che rendono più ricco, sofisticato e semplice l'uso di una tecnologia, come un linguaggio server-side, un linguaggio client-side o le specifiche grafiche di una pagina web
- Framework Server-side: esistono dalla fine degli anni novanta, e hanno reso più semplice la programmazione a tre livelli
- Framework Client-side: si sono sviluppati a partire dal 2002, su CSS e Javascript, con scopi molto differenti, difformi l'uno dall'altro

Framework server-side

Struts, Django, Ruby on Rails, Symfony, Yii, Spring MVC, Stripes, Play, CodeIgniter, etc.

- Architettura a tre livelli, con una struttura concettuale Model-View-Controller, forniscono una varietà di servizi come:
  - Gestione di autenticazione e sicurezza
  - Accesso a database
  - Mappatura di URL
  - Scaffolding: fornire strutture semplificate e generiche di applicazione che vanno poi riempite caso per caso dei dettagli

Framework client-side - CSS

Foundation, YAML, Twitter Bootstrap, etc.

Un mix di classi CSS e strutture predefinite HTML per:

- Gestione del layout di pagina (colonne, aree, stili integrati, ecc.)
- Gestione dei layout responsive (che si adattano alla dimensione dello schermo)
- Semplificazione e omogeneizzazione di feature frequenti (blocchi evidenziati, tab, barre di navigazione, menu dropdown, ecc.)

Framework client-side - Javascript

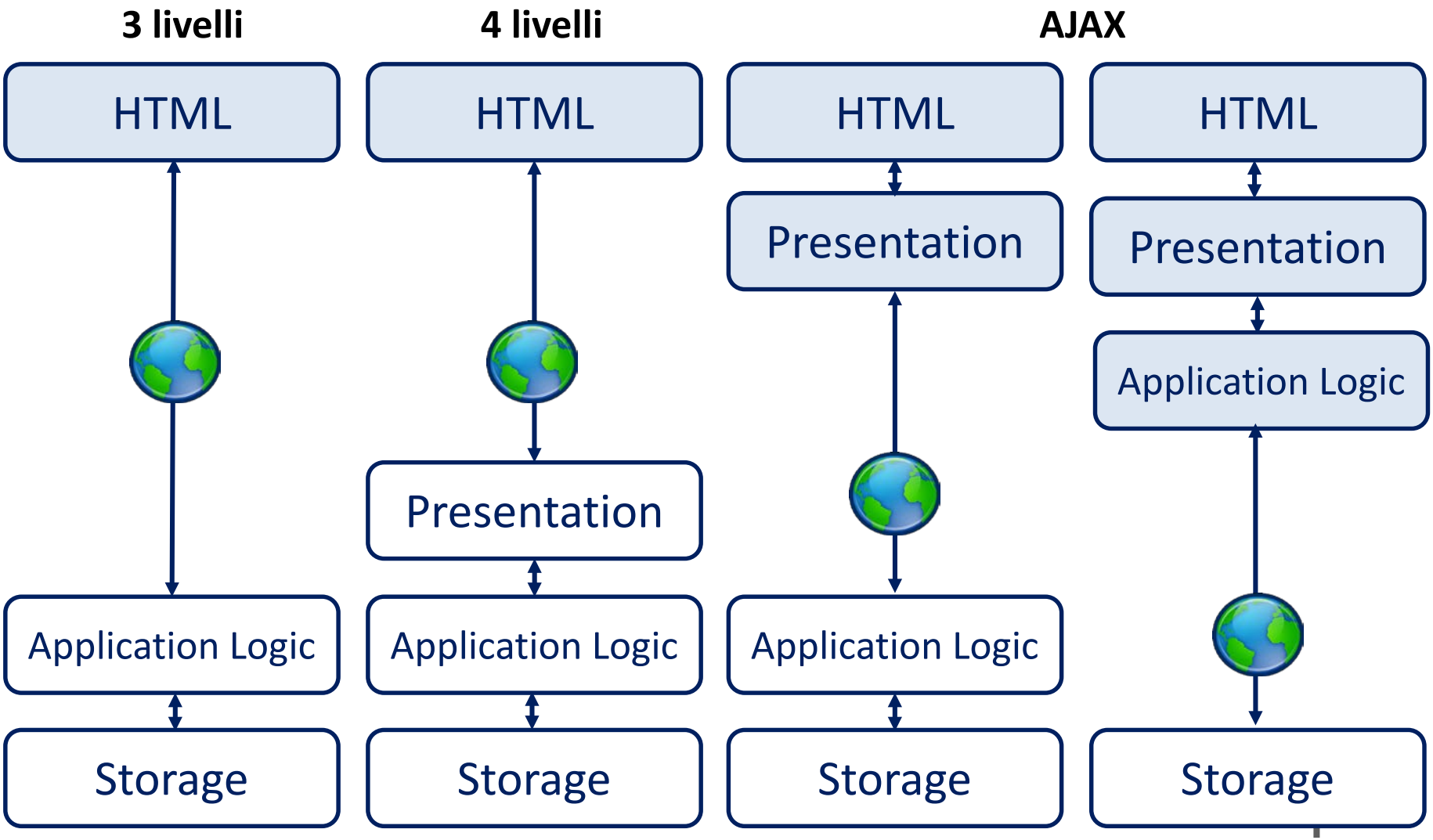
Prototype, Dojo, GWT, jQuery, ExtJs, Angular, Vue.js, React, etc.

- Librerie Javascript per:
  - Omogeneizzare le versioni Javascript dei vari browser
  - Aggiungere funzionalità utili al linguaggio (Ajax, query su DOM, funzionalità Object Oriented, ecc.)
  - Fornire librerie di effetti grafici sofisticati (animazioni e transizioni)
  - Integrare piattaforme tradizionali (PC) e mobili (tablet e smartphone)
  - Fornire widget grafici di uso frequente (grid, tree, menu, form, date picker, ecc.)
  - Gestire template per frammenti HTML riusati frequentemente

Single-page web site
Single-page Application

- Sito web complesso e sofisticato composto in realtà da un unico documento HTML piuttosto ricco e complesso. E' possibile grazie a:
  - Aumento della velocità di rete
  - Disponibilità di Ajax
  - Sofisticazione dei framework Javascript e CSS
- Questo sta avvenendo sia per siti informativi sia per siti di applicazioni
  - Gestione semplificata
  - Connessioni HTTP semplificate e unificate
  - Estrema flessibilità di reazione dell'applicazione
  - Offline editing

Confronto

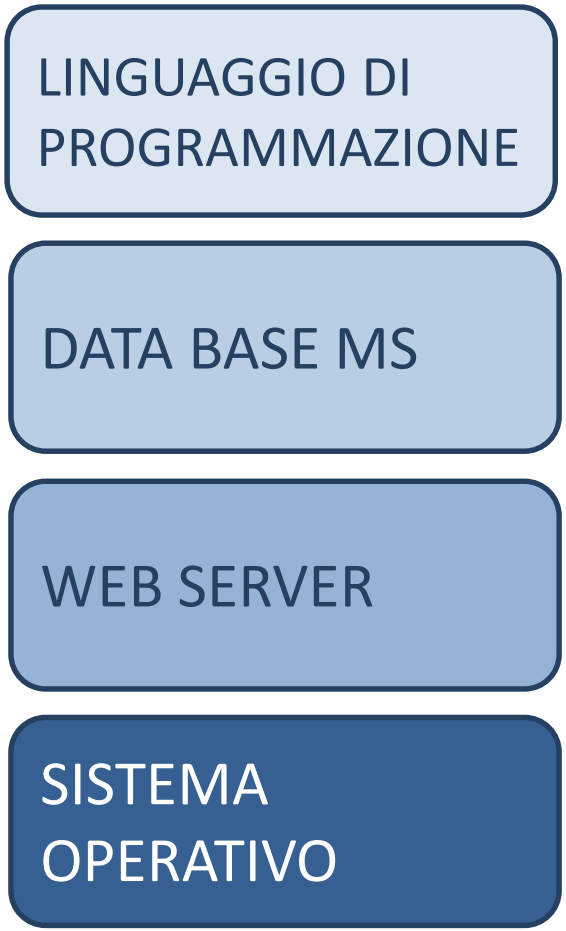

3 livelli
HTML
Application Logic
Storage
4 livelli
HTML
Presentation
Application Logic
Storage
AJAX
HTML
Presentation
Application Logic
Storage
HTML
Presentation
Application Logic
Storage

Q&A

- Ci sono domande?

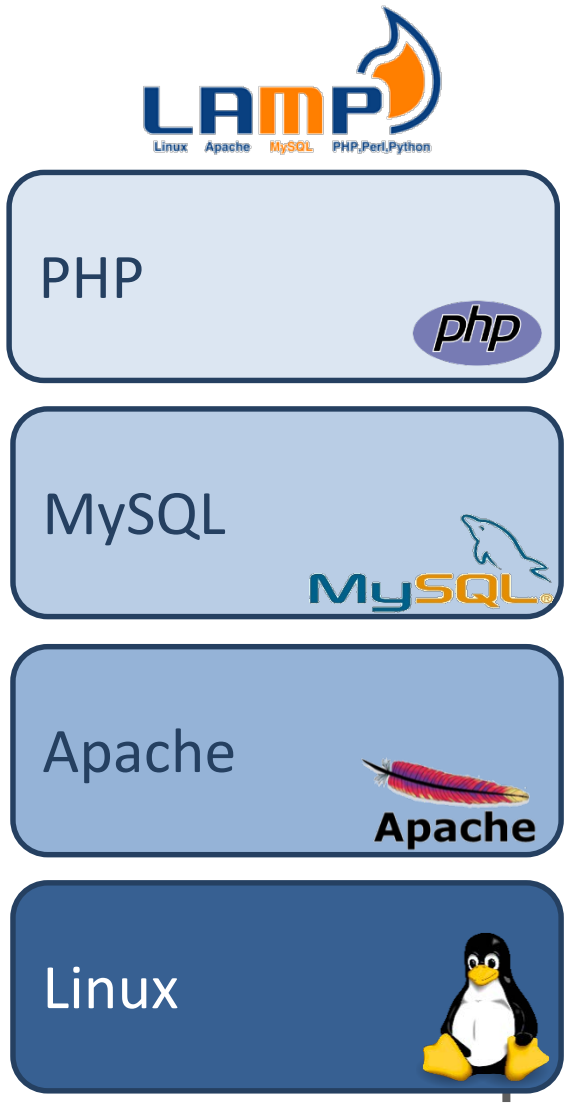
Web solution stack

- Un **solution stack** è un insieme di componenti o sottosistemi software che sono necessari per creare una piattaforma completa basata su **architetture a 3 livelli**, in modo che nessun software aggiuntivo sia **indispensabile** allo sviluppo di applicazioni.
- Per esempio per lo sviluppo Web un solution stack è **solitamente** costituito da 4 elementi: sistema operativo, web server, database, e linguaggio di programmazione.



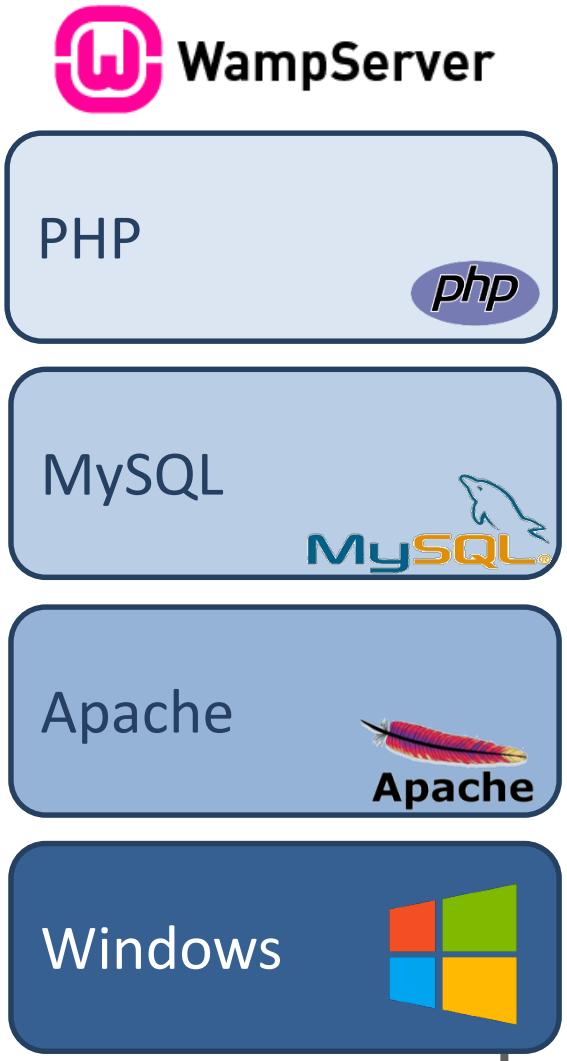
LAMP

- Uno dei solution stack più noti è LAMP, in cui:
  - **L**inux è il sistema operativo.
  - **A**pache è il server web.
  - Il DBMS è **M**ySQL.
  - Il linguaggio di programmazione comunemente è **P**HP ma vengono usati anche **P**erl e **P**yton.
- Tutto il solution stack è completamente opensource.



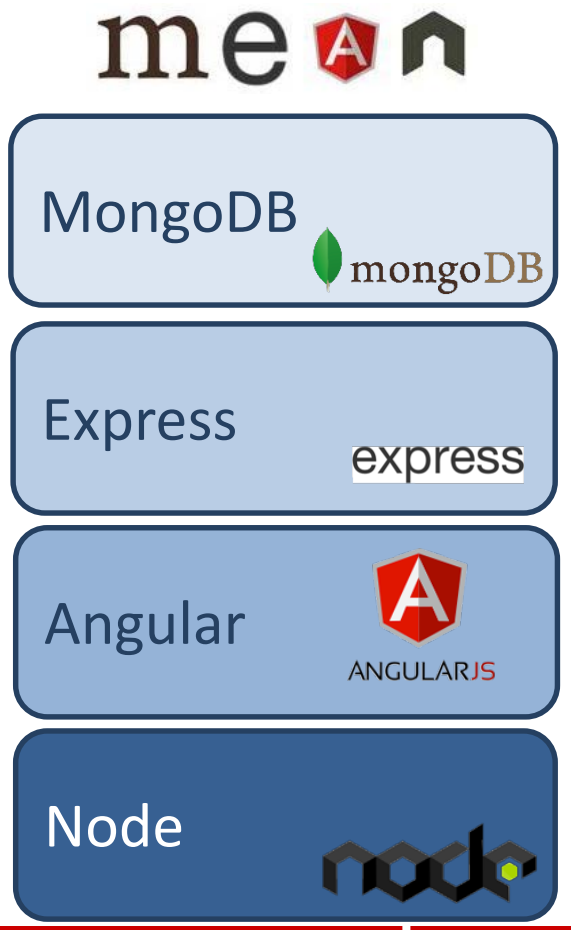
WAMP

- Variante Windows di LAMP:
  - **W**indows è il sistema operativo.
  - **A**pache è il server web.
  - Il DBMS è **M**ySQL.
  - Il linguaggio di programmazione comunemente è **P**HP ma vengono usati anche **P**erl e **P**yton.



MEAN

- Mean è un solution stack con una struttura diversa.
- Non fa riferimento a sistema operativo
  - **M**ongoDB, come data Base (NoSQL database)
  - **E**xpress.js, come framework di sviluppo JavaScript lato **server** (esegue su Node.js).
  - **A**ngular JS, come framework di sviluppo JavaScript lato **client**.
  - **N**ode.js, ambiente di esecuzione per applicazioni server-side (permette di eseguire codice Javascript server-side all'esterno del browser)

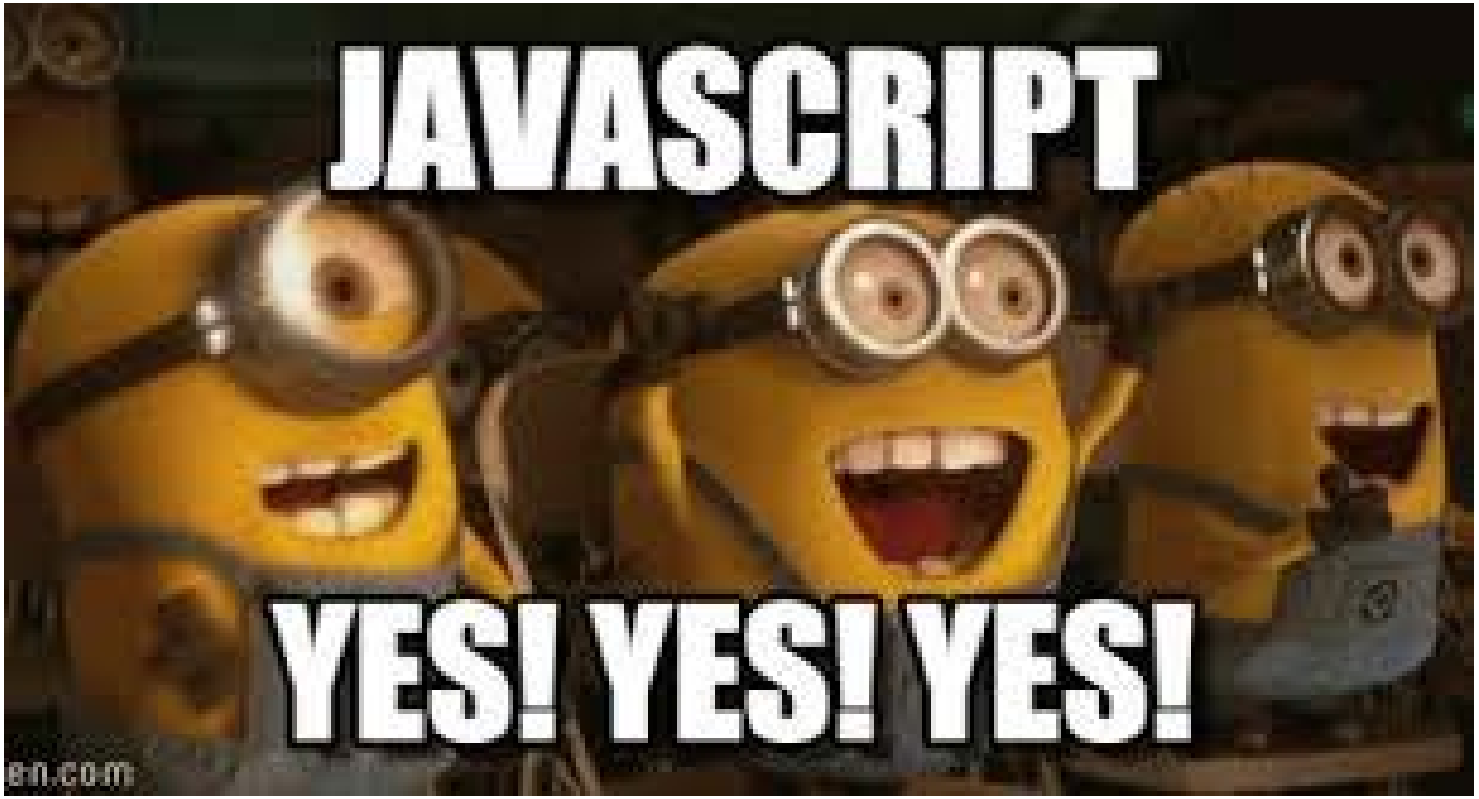


Confronto MEAN-L(W)AMP

- LAMP e WAMP fanno riferimento a uno specifico sistema operativo. MEAN nasce **multipiattaforma**
- MEAN usa Node.js come ambiente di esecuzione delle applicazioni server-side, esegue codice JS al di fuori del browser (**paradigma: «JavaScript everywhere»***)
- MEAN usa un DB **non relazionale** (MongoDB) al posto di MySQL, DB relazionale di L(W)AMP.
- MEAN fornisce **due supporti di programmazione** (uno **client** e uno **server,** Angular e Express) basati sullo stesso linguaggio **Javascript**.
- L(W)AMP si occupano invece di definire solo lo stack lato server, usando per programmare PHP/Python e Perl.

Paradigma JavaScript Everywhere

Paradigma JavaScript Everywhere



Paradigma JavaScript Everywhere

Q&A

- Ci sono domande?

Riferimenti

- *Architecture of the World Wide Web, Volume One* (*https://www.w3.org/TR/2004/REC-webarch-20041215/*)
- *WEB ARCHITECTURE* (*https://www.w3.org/standards/webarch/*)
- Roads and Crossroads of the Internet History *http://www.netvalley.com/cgi-bin/intval/net_history.pl?chapter=1*
- HTTP - Hypertext Transfer Protocol: https://www.w3.org/Protocols/, https://tools.ietf.org/html/rfc2616, https://www.tutorialspoint.com/http/index.htm

- *Ajax: A New Approach to Web Applications* articolo di Jesse James Garrett http://www.adaptivepath.com/publications/essays/archives/000385.php
- *AJAX: Getting Started* http://developer.mozilla.org/en/docs/AJAX:Getting_Started
- http://www.ecma-international.org/publications/standards/Ecma-262.htm
- *Survey Results: JavaScript Frameworks*, Kyle Hayes, 29 Marzo 2009, http://www.kylehayes.info/2009/03/29/survey-results-javascript-frameworks:/

Q&A

- Ci sono domande?