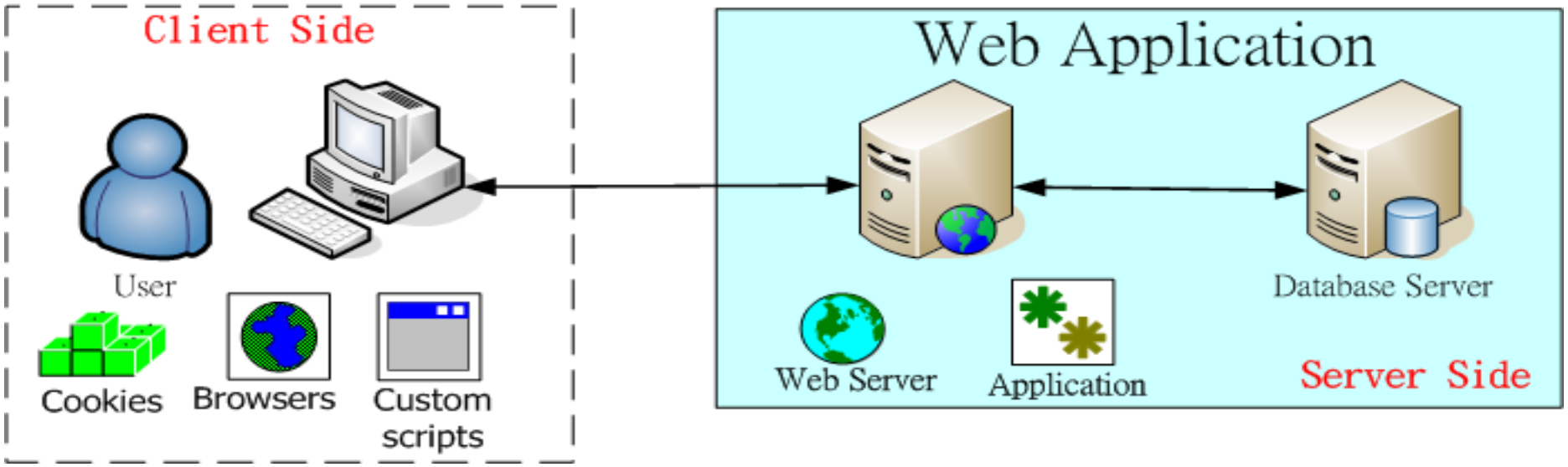
Web Performance/Loading Testing

Web Performance



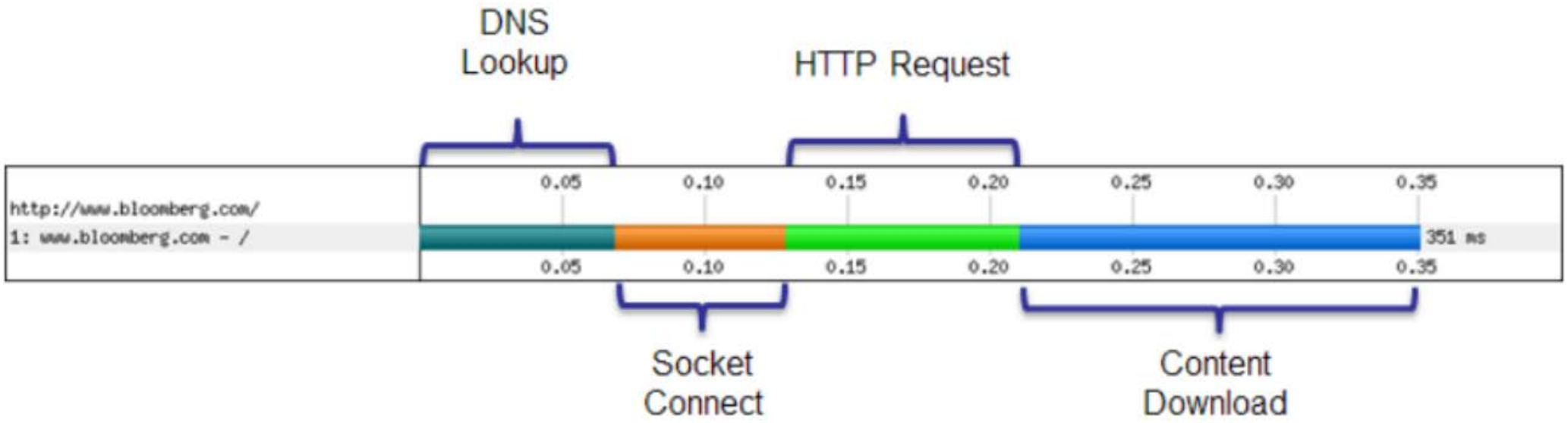
Web Application Performance

- Web Page Loading
  - First Loading
  - Script
  - Picture…etc.,
- Network
- Web Server Response
- Application Run Time
- OS Loading

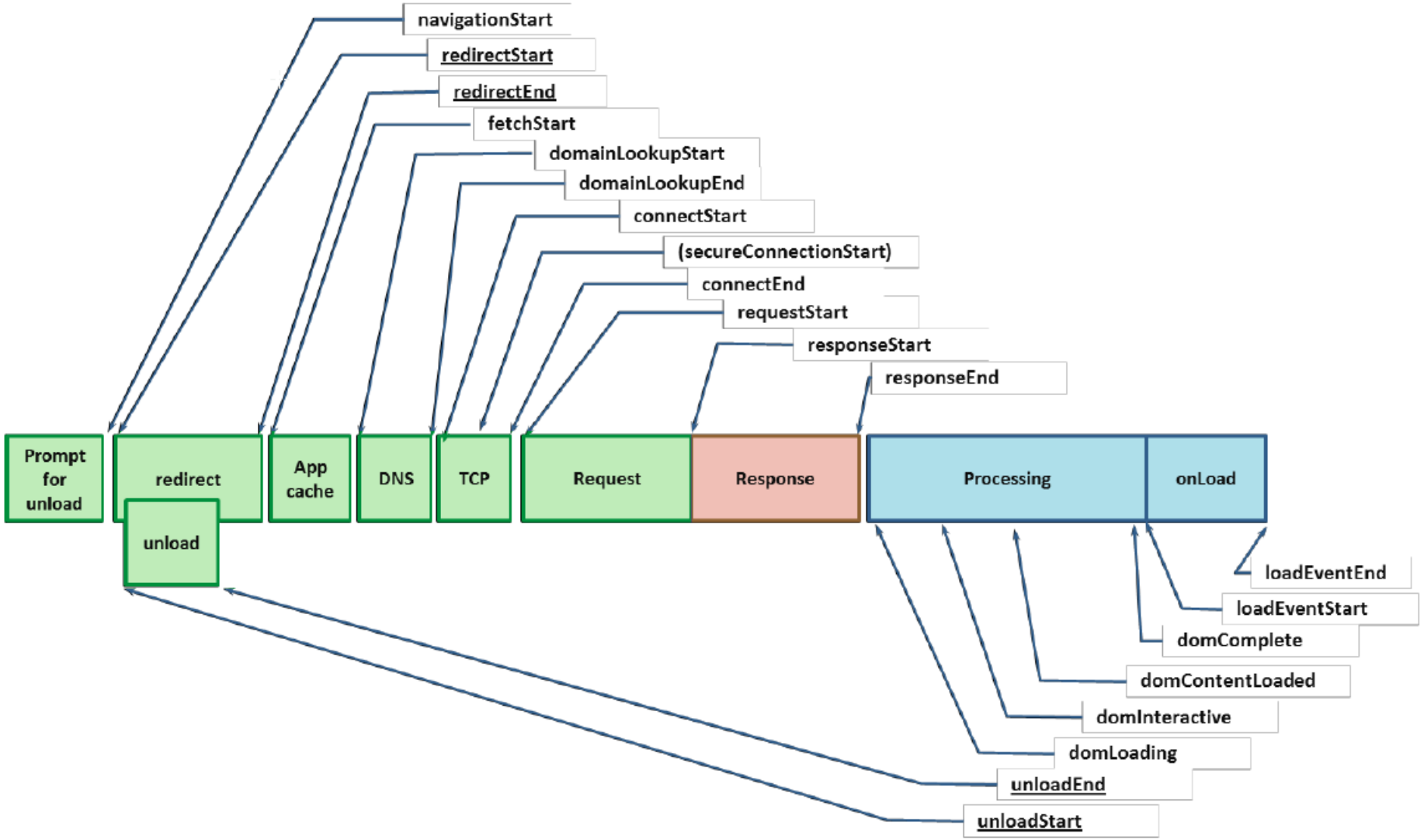
Web Page Loading

Web Server Loading

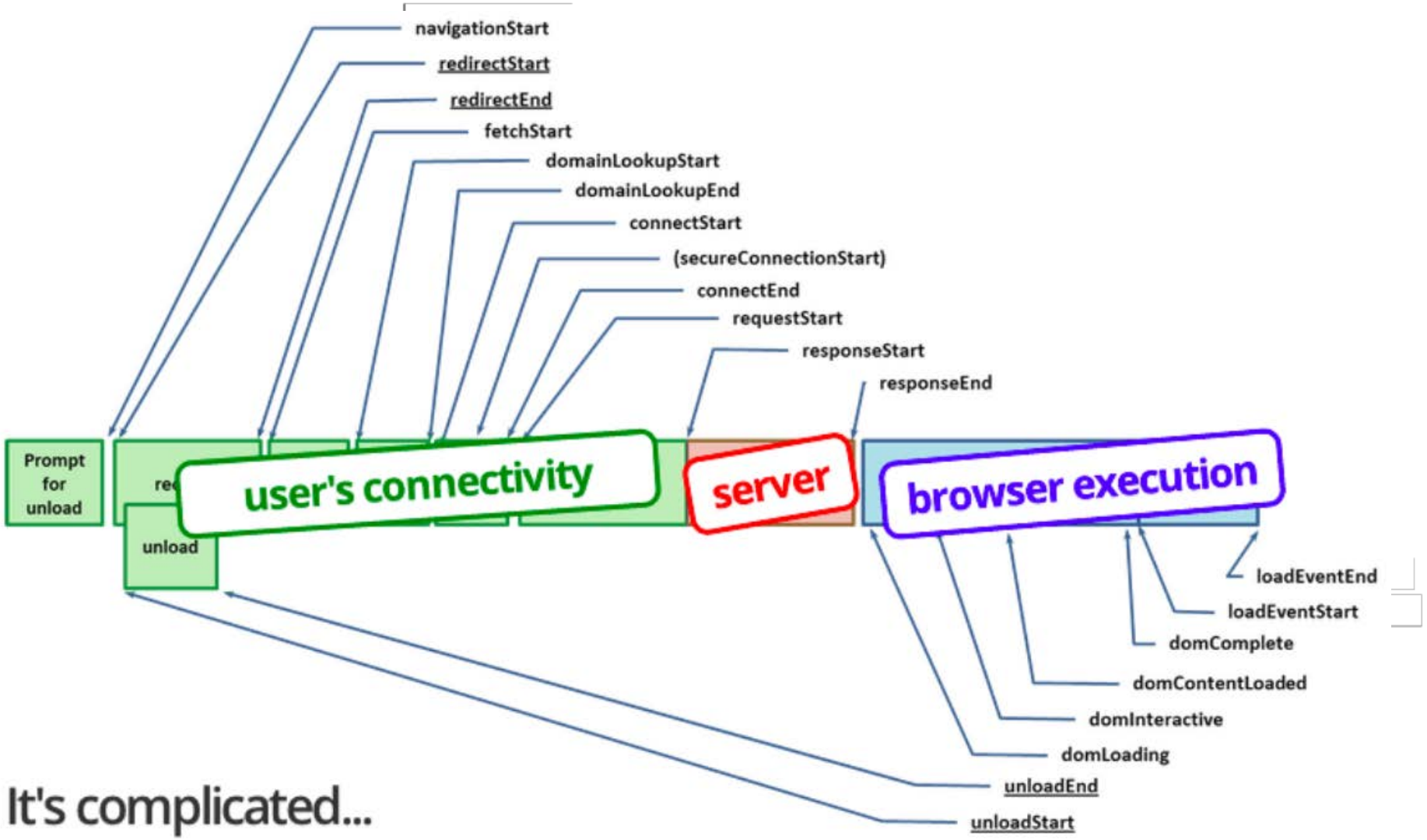
Web Performance



Navigation Timing (W3C)

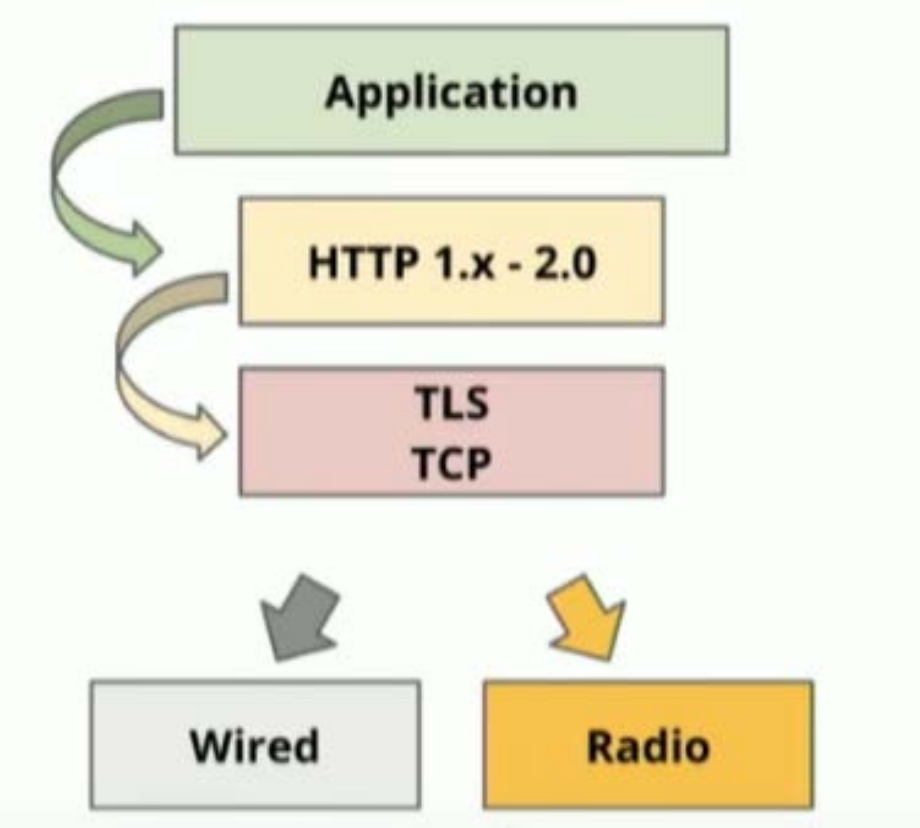


Navigation Timing (W3C)



It's complicated...

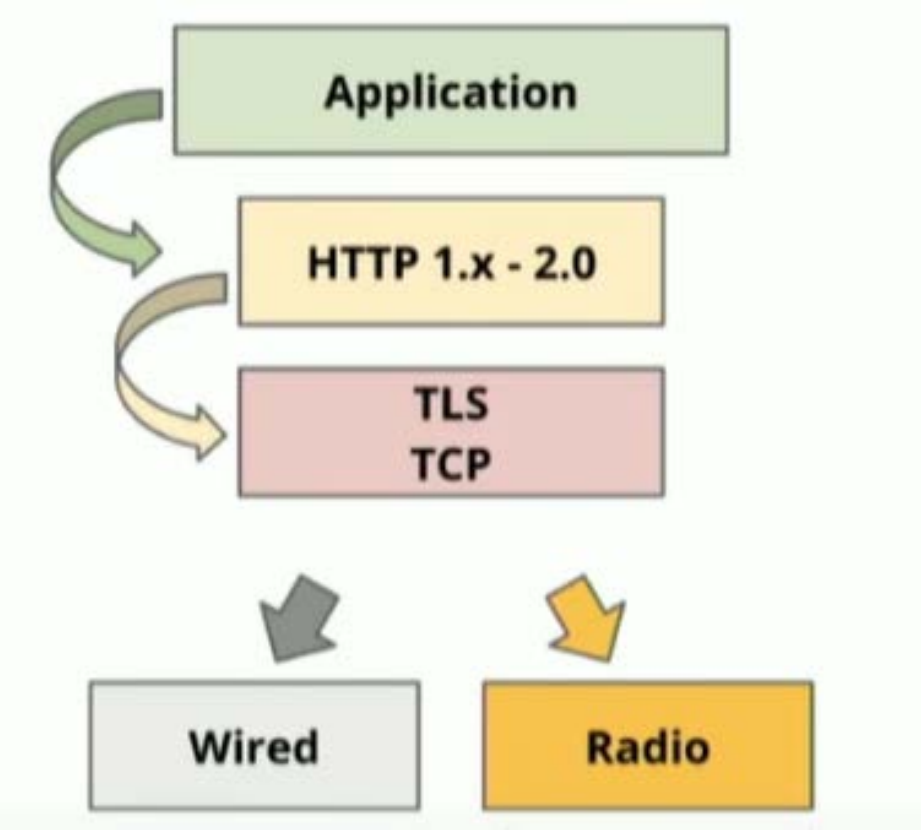
Web Performance



- 有線、無線網路的效能
- 有線使用技術的效能、頻寬
  - ADSL
  - Ethernet
  - ATM…etc.,
- 無線
  - 3/4/5 G
  - WiFi
  - 傳輸範圍…etc.,

Critical rendering path - Crash course on web performance (Fluent 2013)

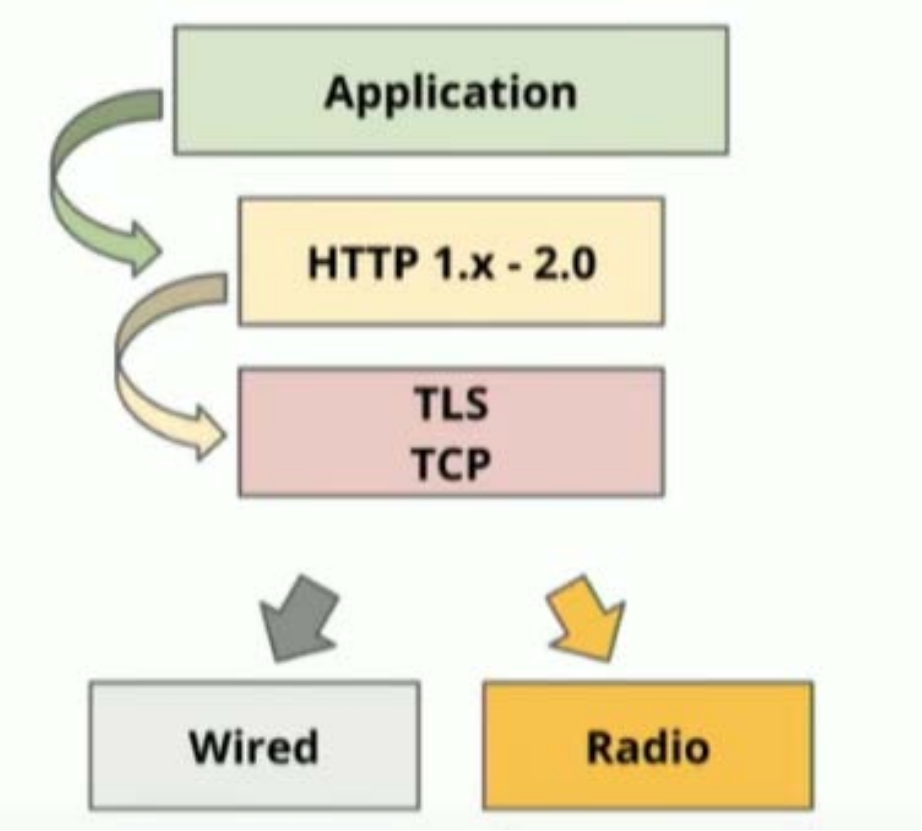
TCP



- OS Kernel 特性
- Slow start after idle
- TCP Windows scaling
- Server位置
- TCP Connections 建立
- 壓縮傳送資料效率
- ….

Critical rendering path - Crash course on web performance (Fluent 2013)

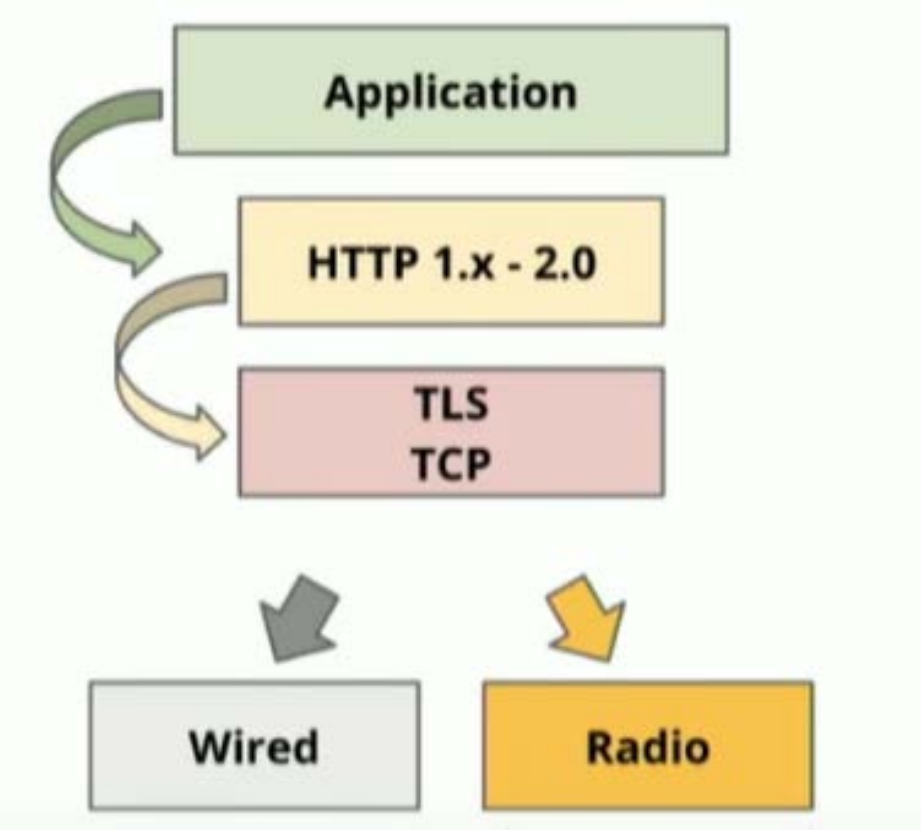
TLS



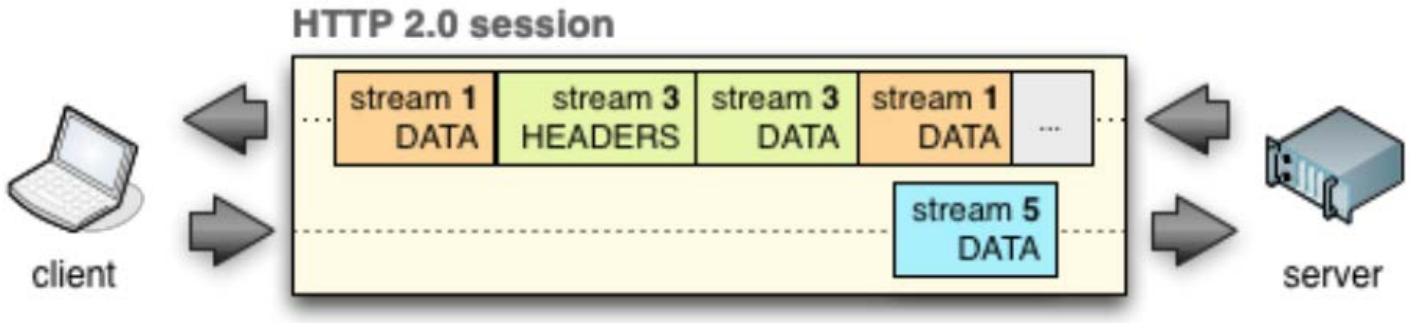
- TLS libraries
- Session Caching/Session Tickets
- Early TLS termination(CDN)
- Optimize TLS record size
- Optimize Certificate size
- Optimize TLS compression
- 伺服器名稱指示（Server Name Indication，縮寫：SNI）
- HTTP Strict Transport Security
- …

Critical rendering path - Crash course on web performance (Fluent 2013)

HTTP 1.x -> 2.0

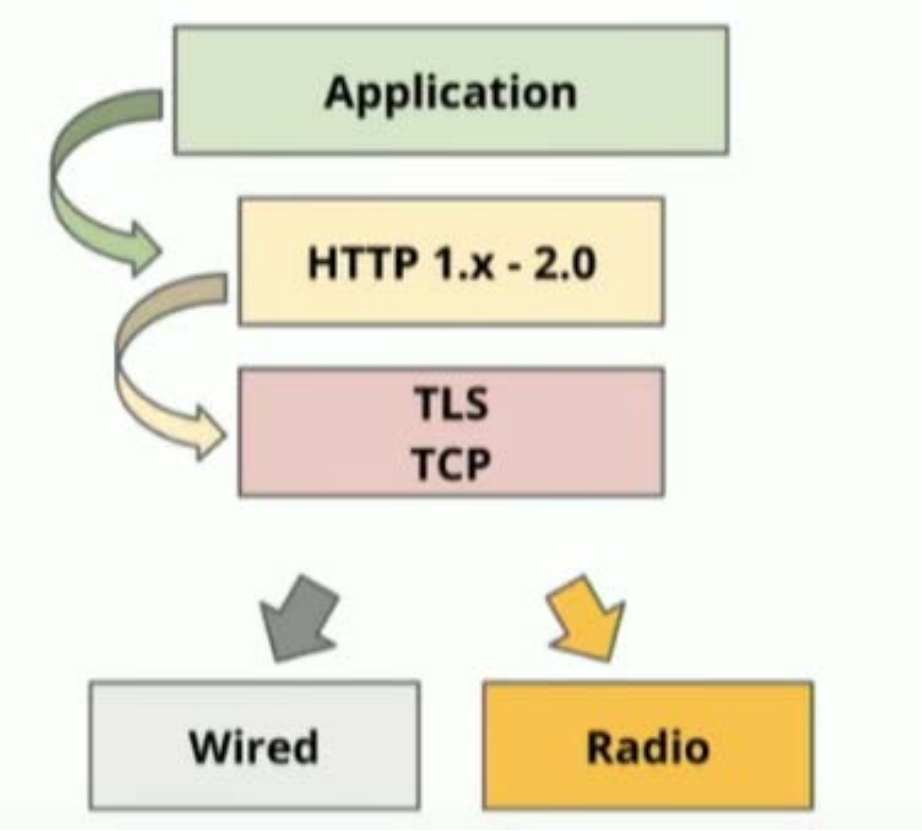


- 鏈結/使用的檔案 (CSS, JS)
- 最佳化小圖片
- 分流資源檔案
- 最小化協定的影響
- 壓縮資源檔案
- 使用快取
- …



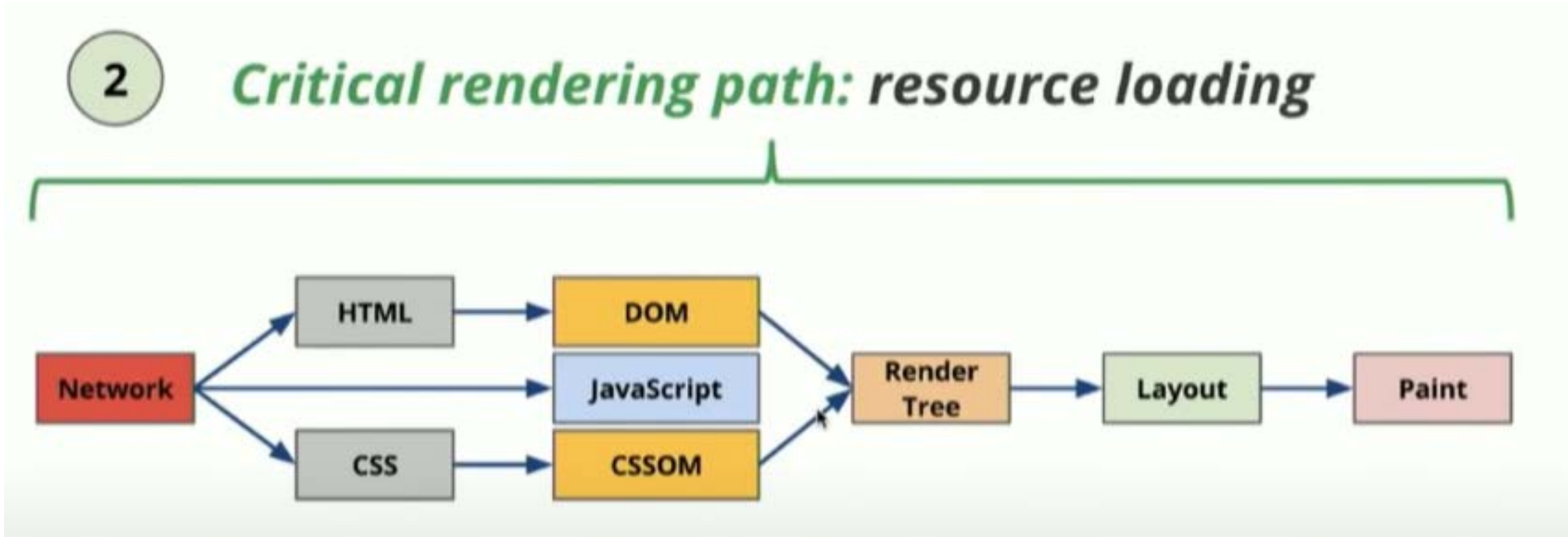
Critical rendering path - Crash course on web performance (Fluent 2013)

Application



- XMLHttpRequest
- Server-Sent Event
- WebSocket
- WebRTC
  - DataChannel – UDP in the Browser!

Critical rendering path - Crash course on web performance (Fluent 2013)

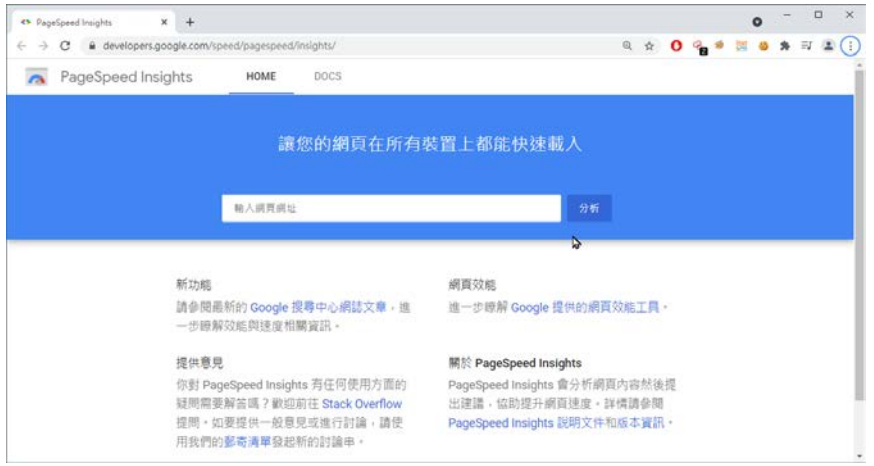

2
Critical rendering path: resource loading
Network
HTML
CSS
DOM
JavaScript
CSSOM
Render Tree
Layout
Paint

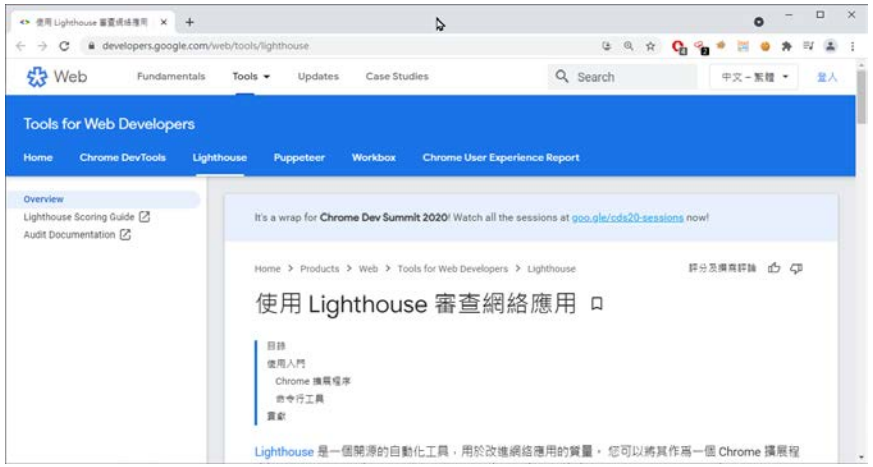
關鍵性能指標：
加載時間(user's perceived load time.)

- 瞬時：低於100 毫秒.
- 100 毫秒到 300 毫秒的延遲是可以察覺的
- 1秒鐘是使用者不分心的極限
- 使用者希望網站在 2 秒內加載完成
- 3 秒後，40% 的使用者可能會放棄存取網站
- 10 秒是使用者保持注意力的極限

Web Page Loading

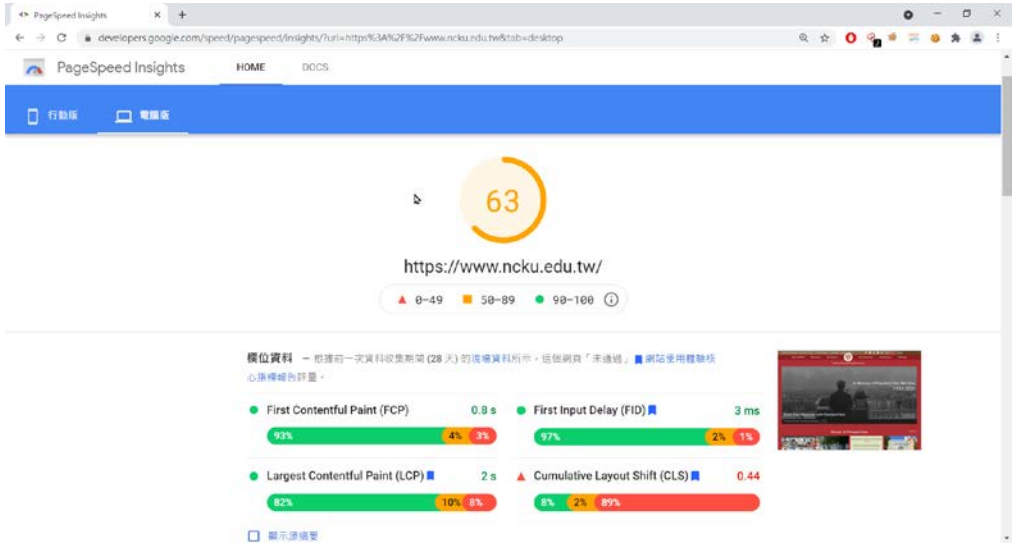
- 了解客戶端性能的兩個最佳工具
  - Google PageSpeed Insights
    - 分析網頁內容並產出建議以加快網頁加載速度的服務。減少頁面加載時間會降低跳出率並提高轉化率。
    - https://developers.google.com/speed/pagespeed/insights/
  - Google Lighthouse
    - 用於提高網頁質量的open-sources自動化工具。前端開發人員可用Google Chrome 工具中提供的 Lighthouse 指標。
    - https://developers.google.com/web/tools/lighthouse
  - Web.dev https://web.dev/measure/





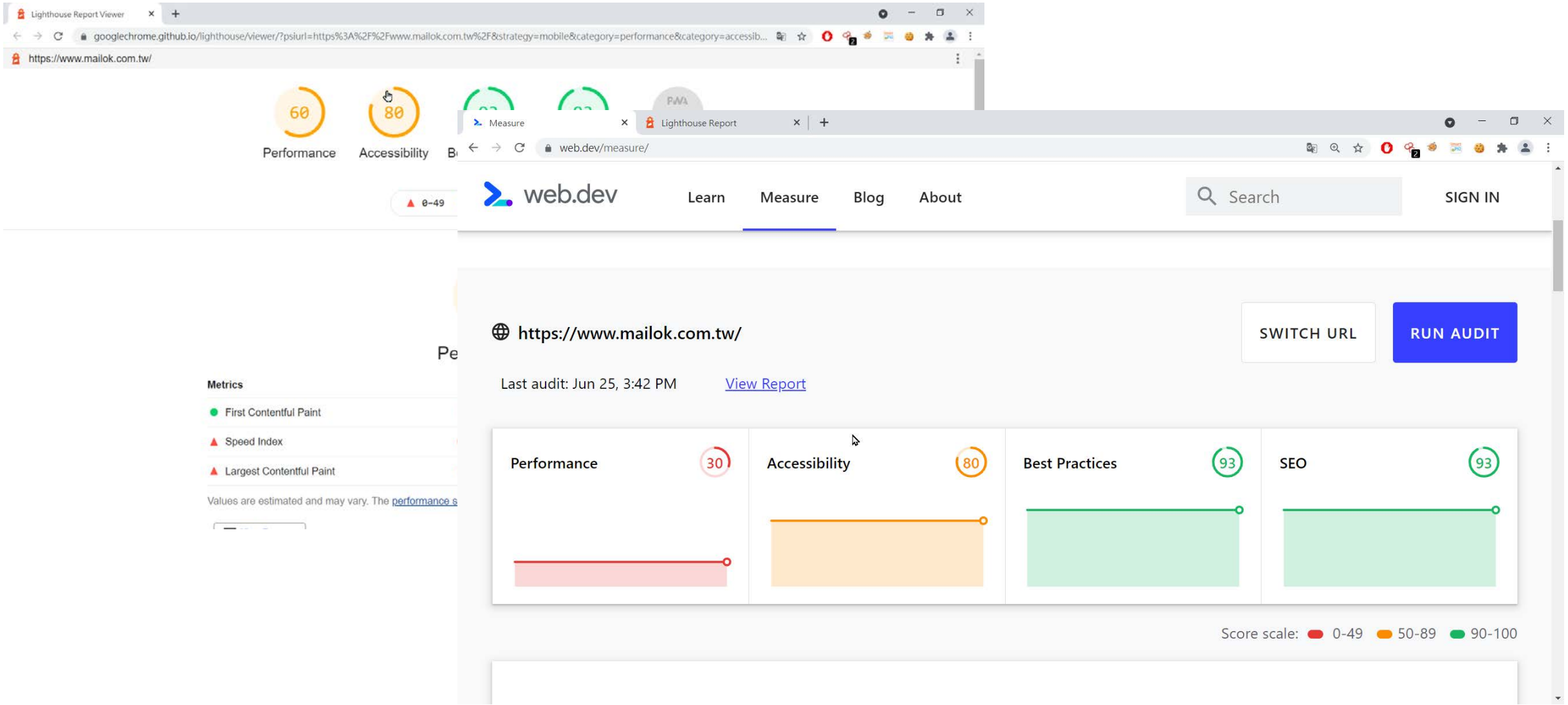
Google PageSpeed Insights

- 可檢測行動裝置與電腦瀏覽器載入情況
- 使用 Google 網站使用體驗核心指標報告作為指標
  - First Contentful Paint (FCP) ：以用戶為中心的衡量感知加載速度的指標
  - Largest Contentful Paint (LCP)：衡量加載性能。為了提供良好的用戶體驗，LCP 應在頁面首次開始加載後的2.5 秒內發生。
  - 首次輸入延遲 (FID)：測量交互性。為了提供良好的用戶體驗，頁面的 FID 應為 100 毫秒或更短。
  - 累積佈局偏移 (CLS)：測量視覺穩定性。為了提供良好的用戶體驗，頁面應保持0.1的 CLS或更少。



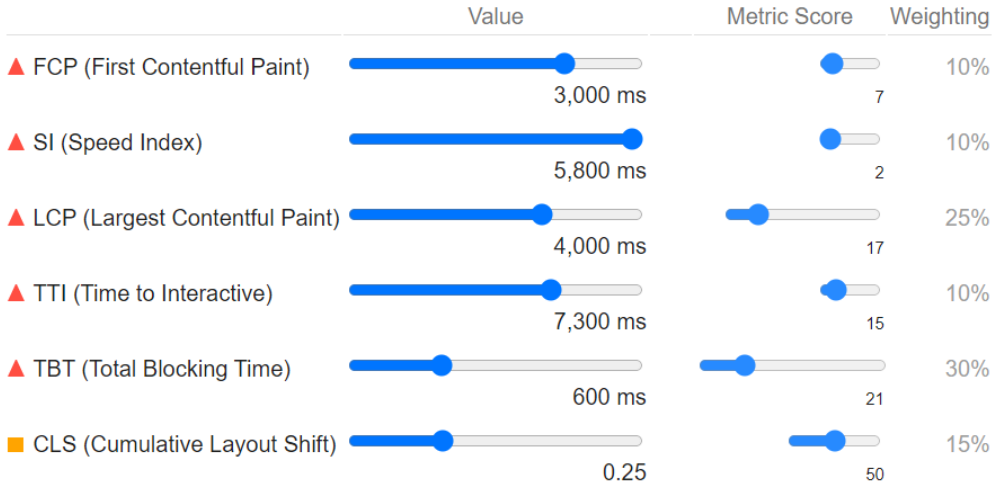
	Good	Needs Improvement	Poor
FCP	[0, 1800ms]	(1800ms, 3000ms]	over 3000ms
FID	[0, 100ms]	(100ms, 300ms]	over 300ms
LCP	[0, 2500ms]	(2500ms, 4000ms]	over 4000ms
CLS	[0, 0.1]	(0.1, 0.25]	over 0.25

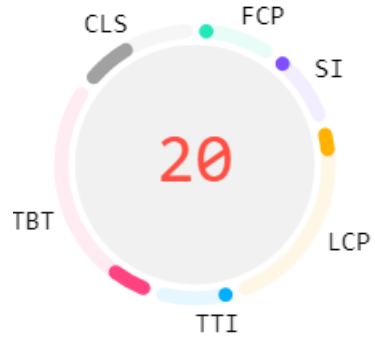
Google Lighthouse & Wed.dev



Lighthouse Scoring Calculator

- FCP (First Contentful Paint)：瀏覽器呈現第一段 DOM 內容所需的時間。
- SI (Speed Index)：視覺顯示速度速度指數
- LCP (Largest Contentful Paint) ：以用戶為中心的衡量感知加載速度的指標，為可見的最大圖像或文本塊的渲染時間
- TTI (Time to Interactive)：衡量一個頁面需要多長時間才能載入與提供互動，如圖片輪動等
- TBT (Total Blocking Time)：衡量頁面被阻止回應使用者輸入（例如鼠標點擊、屏幕點擊或鍵盤按下）的總時間
- CLS (Cumulative Layout Shift)：累積佈局偏移 (CLS) 是衡量視覺穩定性指標，衡量頁面整個生命週期內發生的每個意外佈局偏移的最大*佈局偏移分數*的度量





https://googlechrome.github.io/lighthouse/scorecalc/

自動化 Open-Scource loading 測試工具

- Cypress, https://www.cypress.io/
  - Npm based
  - 捕獲 Google Chrome 的性能數據以進行測試運行
- Selenium, https://www.selenium.dev
  - 用於自動化Web 瀏覽器的工具
  - 使用 HAR 文件。這是使用BrowserMob 代理(BMP) 完成的。BMP 允許您操作 HTTP 請求和響應、捕獲 HTTP 內容以及將性能數據導出為 HAR 文件。
- **JMeter's WebDriver Sampler,** https://www.blazemeter.com/blog/jmeter-webdriver-sampler
  - JMeter 與 Selenium Webdriver 一起使用
  - 對於測試 AJAX、基於 GWT 的 Web 應用程序和模擬用戶操作的性能非常有用。

Server Side Loading

- 從單個端點進行快速負載測試，了解每秒請求能力：
  - Apache Bench, https://httpd.apache.org/docs/2.4/programs/ab.html ,用於對 Apache HTTP 服務器進行基準測試
  - Siege, https://www.joedog.org/siege-home/ ， HTTP 負載測試和基準測試
- 負載測試框架
  - Locust.io, http://locust.io/ , 非常適合了解服務器端的性能。
  - Bees with Machine Guns, https://github.com/newsapps/beeswithmachineguns, 用於武裝（創建）許多蜜蜂（微型 EC2 實例）以攻擊（負載測試）目標（Web 應用程序）的實用程序
  - Multi-Mechanize, https://github.com/cgoldberg/multi-mechanize , 是一個用於性能和負載測試的開源框架 (Python)
  - Httperf, https://code.google.com/p/httperf/ , 為生成各種 HTTP 工作負載和測量 Web 服務器性能的靈活工具
  - Apache Jmeter, https://jmeter.apache.org/ , 用於測試靜態和動態資源（文件、servlet、Perl 腳本、Java 對象、數據庫和查詢、FTP 服務器等）的性能
  - Element, https://element.flood.io/, 是一個開源 Puppeteer 節點庫，它使用基於瀏覽器的負載測試工具。

Apache performance

- Multi-Processing Modules (MPMs)
  - **MPM Prefork:** multiple child processes (More Memory)
  - **MPM Worker:** multiple child processes with many threads each
  - **MPM Event:** multiple child processes with many threads each (keep-alive mgmt.)
    - event MPM需要Linux系統（Linux 2.6+）對EPoll的支持，才能啟用。

Key Tuning parameters

- Timeout
- KeepAlive
- MaxKeepAliveRequests
- keepAliveTimeout
- StartServers
- ThreadsPerchild
- ThreadLimit
- MaxClient
- ServerLimit
- MaxrequestsRperChild

Live Performance Monitor of HTTP Server

Module mod_status

Key Tuning parameters

- Timeout
  - 連接建立後，不傳資料多久要中斷連接
  - Server等待多久client沒有request，中斷連線
- KeepAlive
  - 當你的伺服器需要提供許多靜態檔案供使用者存取，建議使用KeepAlive來增進效能。但是若你的伺服器記憶體相當稀少，建議可以將此功能關閉，以節省記憶體被佔用的情況發生。
  - **KeepAliveTimeout :**此設定值可以在 connection 關閉之前給這些請求多長的等待時間。
  - **MaxKeepAliveRequests**：限制每個 connection 允許發送的請求數量上限，此設定主要可以觀察你的網站或服務開啟的當下同時間會送出多少個請求到伺服器作為依據，再來進行設定。

Key Tuning parameters

- 是否使用keepalive？下列3種情況：
  1. Client 存取網站，除了HTML外，還用很多Javascript、CSS、圖片，並且每個檔案都在HTTP Server上。
  2. Client 存取網站，除了HTML外，只用了一個 Javascript。
  3. Client 存取網站， 使用動態網頁，由程式產生即時內容，不引用Javascript、CSS、圖片。

1. 適合打開KeepAlive
2. 都可以
3. 應該關閉keepAlive

Key Tuning parameters

HttpdProcessNumber = KeepAliveTimeout * TotalRequestPerSecond / Average(KeepAliveRequests)
HttpdUsedMemory = HttpdProcessNumber * MemoryPerHttpdProcess

Apache Process 數量 = KeepAliveTimeout * 每秒HTTP Request 總數 / 平均 KeepAlive Requests
Apache使用記憶體 = Apache Process 總數 * 每個HTTP Process使用的記憶體大小

- 平均 KeepAlive Requests ：指每個client連上http server後持續發出的HTTP Request
- KeepAliveTimeout = 0 ,keepalive off, 表示每個連線完成即中斷。
  - 若每秒client request多，KeepAliveTimeout 值大、平均 Keepalive Request值小都會造成 Apache process 數量多與使用記憶體多

Key Tuning parameters

- Keepalive
  - 如果HTTP Server 記憶體非常大，不管是否使用keepalive 都沒差。
  - HTTP Server不大並有大量對於檔案的存取、或主要處理動態頁面，關閉KeepAlive會節省記憶體，並可將記憶體拿來做cache，提升存取效率與讓系統更加穩定。
  - 在一個連接中有多個Request，應該開啟；討論區、動態產生內容，則建議關閉。
  - 前端有proxy等 KeepAlive應該開啟。

Apache performance

- StartServers 伺服器啟動時建立的Process數量
- ServerLimit 系統配置的最大Process數量
- MinSpareServers 空閒Process的最小數量(若小於，每秒生成)
- MaxRequestWorkers 限定伺服器同一時間內Client端最大接入的請求數量，與ServerLimit連動。
- MaxConnectionsPerChild 每個Process在其生命周期內允許最大的請求數量，建議設置為非零能夠防止(偶然的)耗盡記憶體，並給Process一個有限壽命，從而有助於當伺服器負載減輕的時候減少活動Process的數量(重生的機會)。

Prefork
StartServers 5
Server Limit 8
MinSpareServers 5
MaxSpareServers 10
MaxRequestWorkers 250
MaxConnectionsPerChild 0

Apache performance

- StartServers 伺服器啟動時建立的Process數量
- ServerLimit 系統配置的最大Process數量
- MaxSpareThreads 空閒Threads的最大數量
- MinSpareThreads 空閒Threads的最小數量
- ThreadsPerChild每個Process產生的Threads數量
- MaxRequestWorkers /MaxClients限定伺服器同一時間內客戶端最大接入的請求數量
- MaxConnectionsPerChild每個子進程在其生命周期內允許最大的請求數量

Worker
StartServers 3
ServerLimit 16
MinSpareThreads 75
MaxSpareThreads 250
ThreadsPerChild 25
MaxRequestWorkers 400
MaxConnectionsPerChild 1000

Apache performance

- **Apache Performance Tuning**
  - https://httpd.apache.org/docs/2.4/misc/perf-tuning.html
- Apache Performance Tuning on Linux
  - https://www.alibabacloud.com/blog/apache-performance-tuning-on-linux_595181

Linux Performance

- 單體的效能(如 CPU , MEMORY , DISK , IO …),
- 針對整個系統(單一伺服器 server , 單一儲存裝置 Storage 或是機櫃 Rack),

Linux Performance

- **CPU**
  - **CPU Burn-in**
    只能針對單核心在做運算,不推薦使用
  - **Geekbench**
    針對 整數 (Integer performance) ,浮點數 (Floating point performance) 與 記憶體 (Memory performance)的運算(主要涵蓋了 加密,圖像處理,信號處理與物理模擬)

Linux Performance

- **Memory**
  - **Intel® Memory Latency Checker**
  - **stream memory benchmark**
  - **java performance testing tools**

Linux Performance

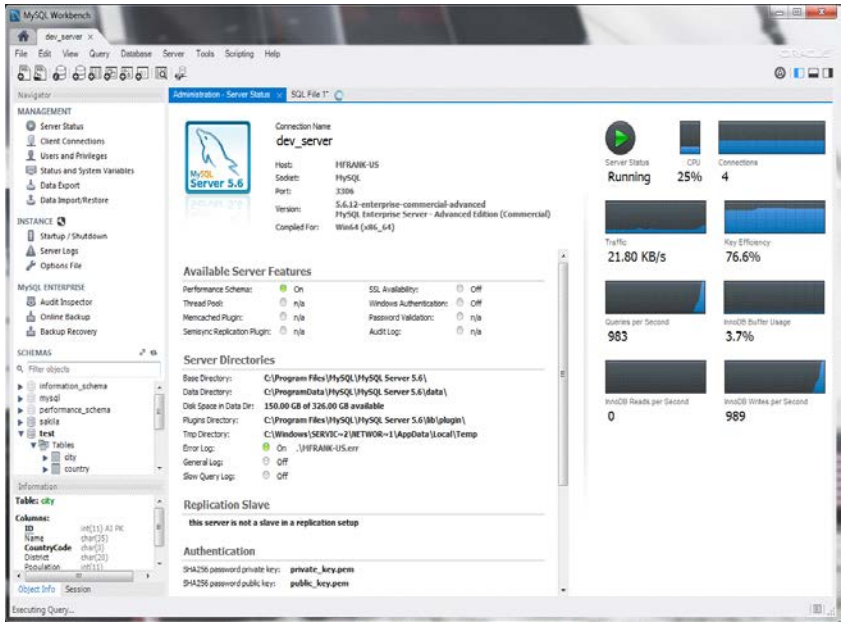
- **Disk**
  - 測試磁碟 I/O 的工具 – **iometer – Quick start** , **iometer**
  - 除了 iometer 可以來測試硬碟的速度,你可以透過 **hdparm** 或是 **dd** 這是一個比較簡單的方式來看目前硬碟傳送的速度
  - 以前用過 dd , hdparm 或是 Iometer 來測試 Disk I/O 的是速度,但 dd, hdparm 太過簡單,Iometer 太過老舊 ,目前看到一套 **Fio – Flexible I/O Tester** 測試工具.
  - VDBench
  - diskspd
  - ATTO Disk Benchmark

Linux Performance

- **Network**
  - 網路效能測試工具 – **iperf**
  - 網路效能測試工具 – **qperf** 功能比 iperf 要強大的多.
  - **Pktgen** – Packet-Generation 它是一個 Linux 核心的模組,它可以以非常快的速度產生封包
  - 另外一套專業級的網路效能測試工具 –**IxChariot**
    IXIA系列的 **IxAutomate**, **IxExplorer**, **IxLoad**
  - **其他 nttcp , netperf , vdbench , Finisar Medusa Test Tool**
  - **smartbits** / **TestCenter** 是一套專門用來檢視網路設備的硬體檢測裝置,是由 SPIRENT 這一家公司所提供主要可以用來檢視網卡或是 switch 等網路裝置.
    乙太網路的發展已經大大超過當年的儲存貴族 Fiber Channel ,40/100G Network 都已經要問世了,測試治具也有了 Spirent 40/100G Ethernet test solutions
- 如果透過這些工具測試出來,發現效能不好該怎麼辦,這時候可以透過 Network 效能測試與調整

Application: DB

- 資料庫的效能評比
  - **TPC Benchmark™ E (TPC-E)** – is a new On-Line Transaction Processing (OLTP) workload
  - **The TPC Benchmark™H (TPC-H)** – is a decision support benchmark
- MySQL workbench
  - https://www.mysql.com/products/workbench/



Top 10 performance engineering techniques that work

- 1. 識別不同層次的交易工作/成本
- 2. 定義監控的 KPI
- 3. 減少您分析的交易數量
- 4.等待測試完成再分析
- 5. 確保可重複的結果
- 6. 增加你的負擔(測試)
- 7. 使用可視化來發現異常
- 8. 尋找 KPI 趨勢和高峰以識別瓶頸
- 9. 不要忽視交易工作/成本
- 10.增加監測的精細度(粒度)以獲得更好的清晰度

- 1. Identify tier-based engineering transactions
- 2. Monitored KPIs
- 3. Reduce the number of transactions you analyze
- 4. Wait for the test to complete before analyzing
- 5. Ensure reproducible results
- 6. Ramp up your load
- 7. Use visualization to spot anomalies
- 8. Look for KPI trends and plateaus to identify bottlenecks
- 9. Don’t lose sight of engineering transactions
- 10. Increase granularity for better clarity