TheBi
BRAND, BLOG & BUSINESS

PWA

Realizza una Semplice **Progressive Web Application** e Trasforma il tuo Sito in una **PWA**

bibibi.it

TheBi
BRAND, BLOG & BUSINESS

INDICE EBOOK PWA

COSA SONO LE PWA

INTRODUZIONE ALLE PROGRESSIVE WEB APPLICATION

Una Progressive Web App o **PWA** (termine coniato da Google) è un sito web che utilizza moderne tecnologie Web per offrire agli utenti un'esperienza mobile migliore rispetto ad un'app nativa iOS o Android.

Le PWA sono essenzialmente delle **Applicazioni Ibride**, cioè sono una via di mezzo tra un **sito web** e un'**applicazione per device mobile**.

Una PWA tende a comportarsi come un'applicazione per dispositivi mobile. **Sviluppare una PWA** non richiede la conoscenza di linguaggi di programmazione come Objective C o Java (linguaggi specifici per iOS e Android) perché il suo sviluppo richiede solo la conoscenza di linguaggi di scripting client-side.

Le PWA, quindi, sono multipiattaforma proprio perché vengono sviluppate con tecnologie web come Html, Javascript e Json.

Requisiti di una PWA

- Protocollo SSL Attivo
- Sito Responsive
- Presenza del Service Worker e Funzionamento Offline
- Presenza del File Manifest
- Prestazioni e Velocità del Sito su Reti lente al primo avvio (3g)
- Sito Cross Browser (Chrome, Safari, Mozilla, Opera, Edge)
- URL Puliti e Raggiungibili

Realizza una PWA con 2 semplici macro interventi:

- Creazione del file Manifest.json
- Creazione e Installazione del Service Worker

IL FILE MANIFEST.JSON

SVILUPPIAMO IL FILE MANIFEST.JSON

Il file Manifest.json conterrà tutte le informazioni di base della Progressive Web Application.
Ecco un Esempio:


```
{
  "manifest_version": 1,
  "version": "1.0.0",
  "short_name": "PWA Esempio",
  "name": "sitoweb.it",
  "description": "La mia prima Progressive Web Application.",
  "icons": [
      {
        "src": "/PWA/images/512x512.png",
        "sizes": "512x512",
        "type": "image/png"
      },
      {
        "src": "/PWA/images/384x384.png",
        "sizes": "384x384",
        "type": "image/png"
      },
      {
        "src": "/PWA/images/192x192.png",
        "sizes": "192x192",
        "type": "image/png"
      },
      {
        "src": "/PWA/images/152x152.png",
        "sizes": "152x152",
        "type": "image/png"
      },
      {
        "src": "/PWA/images/144x144.png",
        "sizes": "144x144",
        "type": "image/png"
      },
      {
        "src": "/PWA/images/128x128.png",
        "sizes": "128x128",
        "type": "image/png"
      },
      {
        "src": "/PWA/images/96x96.png",
        "sizes": "96x96",
        "type": "image/png"
      },
      {
        "src": "/PWA/images/72x72.png",
        "sizes": "72x72",
        "type": "image/png"
      }
    ],
"start_url": ".",
"lang": "it-IT",
"background_color": "#424242",
"theme_color": "#cc194b",
"display": "standalone",
"orientation": "portrait-primary"
}
```

IL FILE MANIFEST.JSON

SVILUPPIAMO IL FILE MANIFEST.JSON ED I META TAG SPECIFICI

Cosa abbiamo inserito nel file Manifest.json?

- Versione del manifest
- Nome abbreviato
- Descrizione
- Tutti i formati delle icone
- La prima pagina della pwa
- La lingua parlata
- I colori da utilizzare per personalizzare il browser
- La tipologia del display
- L'orientamento

Per incorporare il nostro **manifest.json** e iniziare a realizzare la nostra progressive web app, dobbiamo inserire, tra i tag <head> e </head> del sito da trasformare, le seguenti stringhe:

<link **rel="manifest" href="/manifest.json"**>
<meta **name="theme-color" content="#cc194b"**/>

(Questo ultimo meta tag si inserisce per una questione di compatibilità con alcuni browser che non riuscirebbero a interpretare i colori di base della pwa contenuti nel manifest.json.)

Il file manifest.json dovrà essere posizionato nella root principale del sito web, ad esempio https://www.sitoweb.it/manifest.json

IL FILE MANIFEST.JSON

SVILUPPIAMO IL FILE MANIFEST.JSON ED I META TAG SPECIFICI

Completiamo le operazioni nell'<head> inserendo alcuni meta tag importanti e specifici:

Favicon del Sito Web

<link rel="icon" href="favicon.ico" type="image/x-icon" />

Meta Tag Specifici per iOS

<link rel="apple-touch-icon" href="images/152x152.png">
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-status-bar-style"
content="black">
<meta name="apple-mobile-web-app-title" content="PWA Esempio">

Meta Tag Specifici per OS Windows

<meta name="application-name" content="PWA Esempio" />
<meta name="msapplication-TileImage"
content="images/144x144.png">
<meta name="msapplication-TileColor" content="#cc194b">

LE ICONE DELLA PWA

REALIZZA LE ICONE IN DIVERSE DIMENSIONI

Ora dovrai realizzare le icone che rappresentano la PWA. Esse verranno utilizzate come icona di installazione nel device così come una normalissima App Mobile.

Le icone della PWA dovranno avere differenti dimensioni per essere maggiormente compatibili con i diversi supporti mobile.

Ecco l'elenco delle icone che ho inserito nel manifest.json:

- 512x512.png
- 384x384.png
- 192x192.png
- 152x152.png
- 144x144.png
- 128x128.png
- 98x98.png
- 72x72.png

Nel mio esempio ho inserito soltanto l'estensione PNG ma può essere inserito anche un set di icone in SVG.

Ricordati di impostare il percorso corretto alle icone.

```
{
        "src": "/PWA/images/512x512.png",
        "sizes": "144x144",
        "type": "image/png"
},
```

IL FILE SERVICE WORKER

SVILUPPIAMO IL FILE SERVICE-WORKER.JS

Il Service Worker è il Core delle PWA. Il Service Worker lavora in background e si pone tra il sito web e la rete, un pò come un Server Proxy. Esso ci consentirà di offrire i contenuti in assenza di rete.

Ma cosa fa un Service Worker? Include funzionalità come le notifiche push, la gestione della cache, la navigazione offline e la sincronizzazione in background.

Iniziamo a Sviluppare un Semplice Service Worker:
Aprite il vostro editor di Codice preferito, incollate la porzione di codice sottostante e salvate il documento “service-worker.js”

```
var cacheName = 'pwa-nomesito';
var filesToCache = [
  '/',
  '/index.html',
  '/css/stylesheet.css',
  '/images/pwa-logo.svg',
  '/js/core.js'
];
/* Avvia il Service Worker e Memorizza il contenuto nella cache */
self.addEventListener('install', function(e) {
  e.waitUntil(
    caches.open(cacheName).then(function(cache) {
      return cache.addAll(filesToCache);
    })
  );
});
/* Serve i Contenuti Memorizzati quando sei Offline */
self.addEventListener('fetch', function(e) {
  e.respondWith(
    caches.match(e.request).then(function(response) {
      return response || fetch(e.request);
    })
  );
});
```

IL FILE SERVICE WORKER

SVILUPPIAMO IL FILE SERVICE-WORKER.JS

Ora che abbiamo realizzato il Service Worker non ci rimane che richiamare lo script all'interno del sito web da trasformare in una PWA.

Il Service Worker va posizionato nella root principale del sito, così come il manifest.

https://www.sitoweb.it/service-worker.js

Questo passaggio è abbastanza semplice e si tratta di inserire nel footer (prima della chiusura del </body>) lo script che abbiamo preparato.

Esempio 1 - Inserire lo Script direttamente nel foglio HTML

```
<script async>
window.onload = () => {
  'use strict';

  if ('serviceWorker' in navigator) {
    navigator.serviceWorker
             .register('./service-worker.js');
  }
}
</script>
```

IL FILE SERVICE WORKER

SVILUPPIAMO IL FILE SERVICE-WORKER.JS

Esempio 2 - Richiamare un file contenente lo script
(nell'esempio core.js)

```
<script src="js/core.js" async></script>
```

È consigliabile inserirlo nel footer per non bloccare il parser della pagina (il rendering).

In entrambi gli esempi ho inserito **l'attributo "async"** per caricare la risorsa javascript asincronicamente.

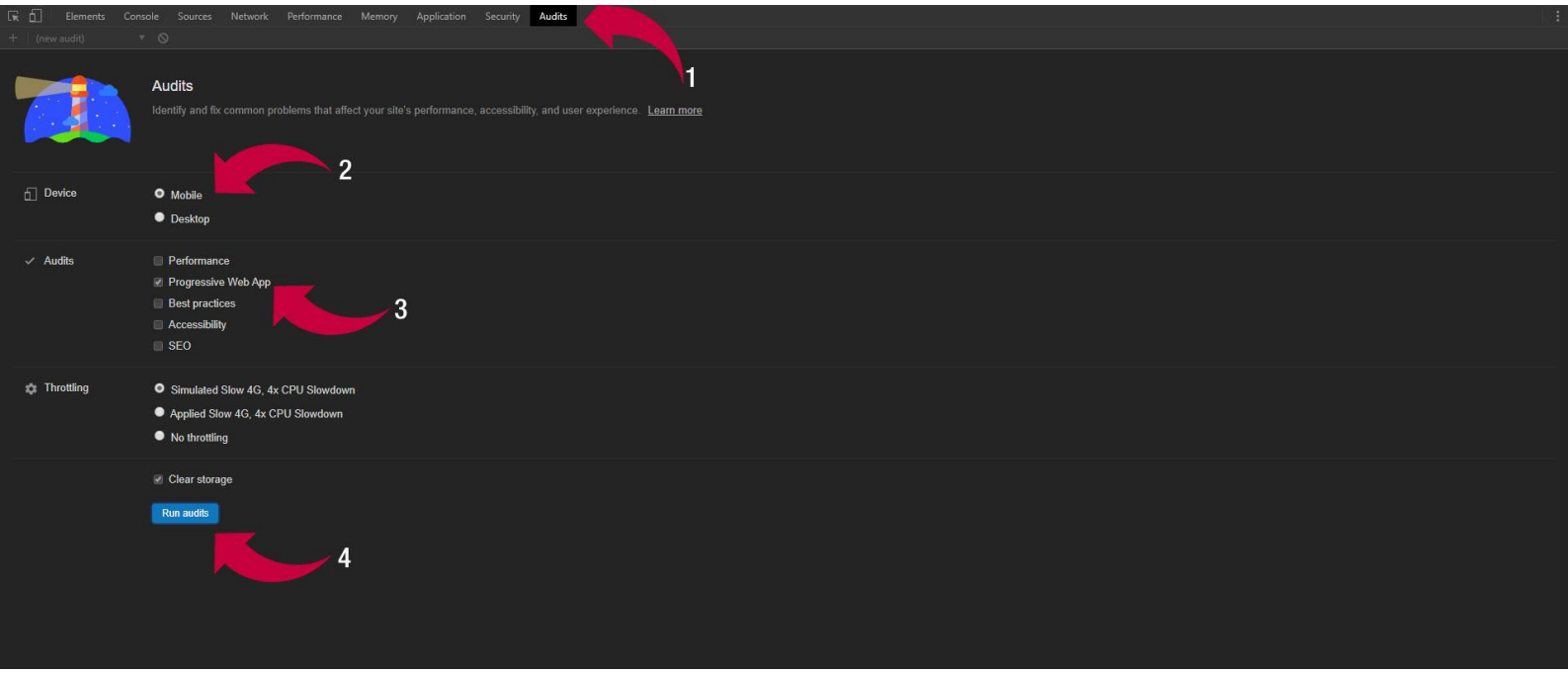
Prima di passare al Test di Conformità e testare, appunto, che la nostra PWA non abbia errori, è giusto specificare che l'esempio riportato sopra si riferisce allo sviluppo di una semplice PWA su un sito di microscopiche dimensioni.

L'esempio prevede il salvataggio in cache solo di una pagina html, un foglio di stile, il logo SVG e un file javascript.

Si possono implementare anche font, immagini e ulteriori pagine statiche modificando il service-worker dell'esempio che scaricherai più avanti.

TESTA LA PWA

TESTA IL FUNZIONAMENTO DELLA PWA CON LIGHTHOUSE



In questa fase analizzeremo il sito web dove ci verrà restituito solo il punteggio della PWA.

1 - Raggiungiamo l'url del sito da trasformare
2 - Apriamo la console sviluppatori del Browser Chrome
3 - Dal menu della console, spostiamoci sulla voce AUDITS
4 - Clicchiamo il pulsante RUN AUDITS
5 - Inizierà l'analisi del sito (rimaniamo sulla pagina del sito da analizzare)
6 - Finita la valutazione, riceveremo una schermata con 4 voci, Performance / Accessibilità / Sicurezza / PWA. Se cliccate la voce PWA, vi sarà mostrato il report.

TESTA LA PWA

TESTA IL FUNZIONAMENTO DELLA PWA CON LIGHTHOUSE

Se avete seguito alla lettera la procedura di **trasformazione da Sito a PWA**, dovreste poter visualizzare una schermata con tutti “**pallini verdi**”, il che significa che siete riusciti nell’intento.

Se invece notate alcuni “**pallini rossi**”, allora c’è qualcosa che non va.

LightHouse vi fornirà ogni spiegazione, individuando i punti da correggere e le soluzioni da applicare. Ora, invece, controlleremo se il manifest e il service worker funzionano correttamente e vengono interpretati dal browser.

Andiamo per step:
1 - Spostiamoci nella console sviluppatori alla voce Application (immagine)

2 - Sulla sinistra troveremo un menu addizionale che contiene le voci Manifest, Service Worker e Clear cache (e altre che non ci interessano in questa fase). Queste 3 voci ci aiuteranno a capire il corretto funzionamento della PWA.

3 - Spostiamoci sulla voce Manifest e controlliamo che i dati inseriti attraverso l’editor di codice siano correttamente decodificati dalla console (Name, Short Name, Start URL, Background color, Orientation, Display e il Set di Icone che abbiamo precedentemente preparato)

TESTA LA PWA

TESTA IL FUNZIONAMENTO DELLA PWA CON LIGHTHOUSE

4 - Ora spostiamoci nella voce Service Worker e controlliamo che stia operando bene. Possiamo capirlo perché, se il service worker sta lavorando correttamente, avremo un pallino verde alla voce status con le relative “activated e running”.

5 - Se invece abbiamo ottenuto degli errori sulla console, consiglio di revisionare punto per punto gli interventi proposti in questo ebook. È probabile che per un errore di sintassi lo script non funzioni correttamente.

Trasformare un Sito Web in un Sito Progressivo, nella mia visione delle cose, è un importante mattoncino da aggiungere all’evoluzione delle nostre competenze e delle tecnologie da adottare per la “scalata”.

Quando un utente ci naviga da dispositivo mobile (oggi come oggi, è praticamente l’80% del traffico), deve poter vivere un’esperienza soddisfacente in termini sia di performance sia di usabilità, pena l’abbandono della pagina e delle sue conversioni.

SCARICA UNA PWA DI ESEMPIO

DOWNLOAD GRATUITO DI UNA SEMPLICE PWA

Puoi **Scaricare Gratuitamente la PWA** utilizzata nell'eBook.

Spero che il mio eBook ti sia Piaciuto!

Luca Carchesio - Web Designer, Esperto SEO & Trainer
Seguimi sul mio Sito Personale **bibibi.it**
Lascia un Like sulla pagina Fan su Facebook
Aggiungimi ai tuoi **Collegamenti su Linkedin**