Progressive Web Apps

Conceitos e introdução

Semana Acadêmica IFET Barbacena 2018

O que são PWAs?

“Esses [web] aplicativos não são entregues através da App Store, eles são apenas websites que tomaram a dose certa de vitaminas.” (Fonte)

PWA x apps nativos

Porque desenvolver um web app

Aplicativos de e-commerce têm uma taxa de conversão

3x

maiores que as versões de sites móveis

Fonte: https://jmango360.com/wiki/mobile-app-vs-mobile-website-statistics/

Apenas

13%

do tempo é usado na web

87%

do tempo é utilizado em aplicativos nativos

Fonte: comScore Mobile Metrix, US, 2015 (PWA - The Complete Guide - Maximilian Schwarzmüller)

80%

Do **tempo** utilizado em smartphones em aplicativos nativos

Apenas os

3

aplicativos favoritos são utilizados com frequência

Fonte: comScore Mobile Metrix, US, 2015 (PWA - The Complete Guide - Maximilian Schwarzmüller)

Dos **1000** aplicativos mais populares

apps **nativos** têm

3.3

milhões de acessos

versões web dos mesmos têm

8.9

milhões de acessos

Fonte: comScore Mobile Metrix, US, 2015 (PWA - The Complete Guide - Maximilian Schwarzmüller)

Apps nativos

- Notificações push
- Atalhos na home screen
- Funcionamento offline
- Acesso a recursos nativos de hardware

Sites mobile

- Grande alcance
- Atualizações instantâneas

PWAs

- Notificações push
- Atalhos na home screen
- Funcionamento offline
- Acesso a recursos nativos de hardware
- Grande alcance
- Atualizações instantâneas

Partes das aplicações PWA

- Web app manifest
- Service Workers
  - Cache
  - Sincronização em segundo plano
  - Notificações push

https://pwa.rocks

O que precisaremos

- Bons navegadores para desenvolvedores a.k.a Chrome
- Node js e npm
- IDE: Notepad, Atom, VS Code, etc.

Manifest

Adicionando na home screen

Propriedades do arquivo

https://developer.mozilla.org/en-US/docs/Web/Manifest

Propriedades

```
{
  "name": "HackerWeb",   // Nome longo da aplicação - será usado na splashscreen

  "short_name": "HackerWeb", // Nome curto - será mostrado abaixo do ícone

  "start_url": "/index.html", // Uma vez clicado o atalho na homescreen, define qual página irá abrir

  "scope": ".", // Escopo do PWA. Define que a pasta atual inteira será utilizada por este manifest

  "display": "standalone", // Define se o app será aberto junto ao navegador ou de forma separada

  "background_color": "#fff", // Define a cor de fundo ao carregar o app

  "theme_color": "#FFFFFF", // Cor do tema - define a cor da barra de título do app - ex. task switcher
```

Propriedades

```
15
16    "description": "A simply readable Hacker News app.", // Pode ser usado quando um usuário favoritar o site
17
18    "dir": "ltr", // Define o sentido de leitura do texto. LTR = da esquerda para a direita
19
20    "lang": "en-US", // Idioma utilizado nos campos name e short_name
21
22    "orientation": "portrait-primary", // Configura (ou obriga) a orientação padrão - retrato ou paisagem
23
```

Propriedades

```
23
24    "icons": [ // Lista dos ícones do app
25      {
26        "src": "images/touch/homescreen48.png", // Caminho da imagem
27        "sizes": "48x48", // Dimensões da imagem
28        "type": "image/png" // Tipo da imagem
29      },
30      {
31        "src": "images/touch/homescreen72.png",
32        "sizes": "72x72",
33        "type": "image/png"
34      }
35    ],
36
```

Propriedades

```
36
37    "related_applications": [ // Lista de apps nativos relacionados ao web app
38      {
39        "platform": "play", // Define a plataforma do app - play = Google Play
40        "url": "https://play.google.com/store/apps/details?id=cheeaun.hackerweb" // Url na loja
41      }
42    ]
43  }
44
```

Adicionando manifesto

- Adicionar arquivo de manifesto
- Testar no computador e smartphone
- Aba “Application” do Chrome Dev Tools

App install banners

Colocando o web app app na home screen

Requisitos necessários - Out 2018

- O web app não deve estar instalado
- O usuário deve ter interagido com o app recentemente
- Deve incluir um manifest com o seguinte:
  - Short_name ou name
  - Ícones: deve incluir pelo menos um 192x192 e um 512px
  - Start_url
  - Display deve ser: fullscreen, standalone ou minimal-ui
  - O site deve ser HTTPS
- O deve possuir um service worker com um evento fetch

https://developers.google.com/web/fundamentals/app-install-banners

Service Workers

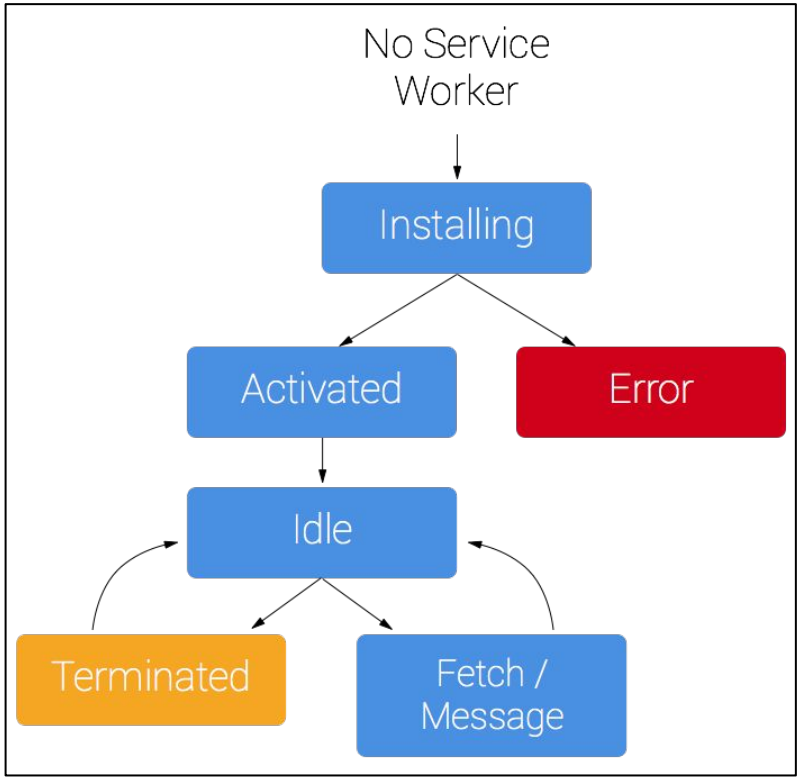
O que são Service Workers

- Códigos Javascript que podem ser executados em segundo plano
  - Não dependem de uma aba ou do browser aberto
  - Não possuem acesso ao DOM
- São especialistas em escutar eventos.
- Um único SW pode gerenciar várias páginas

Eventos gerenciados pelo SW

- Fetch: Qualquer requisição HTTP gera um evento fetch.
- Notificações Push: o SW é o responsável por receber notificações recebidas e exibi-la ao usuário
- Interação com notificações: Reagir com as notificações mostradas ao usuário.
- Sincronização em segundo plano: Se uma ação não pode ser executada no momento corrente, esta é salva e executada assim que a conexão for restabelecida
- Eventos do ciclo de vida do SW: instalação, ativação, remoção, etc

Ciclo de vida do SW



Fonte: https://developers.google.com/web/fundamentals/primers/service-workers/

Registrando um SW

```
1
2  if('serviceWorker' in navigator){
3    navigator.serviceWorker
4      .register('/sw.js')
5      .then(function(event){
6        console.log('SW registrado',event)
7      })
8  }
9
```

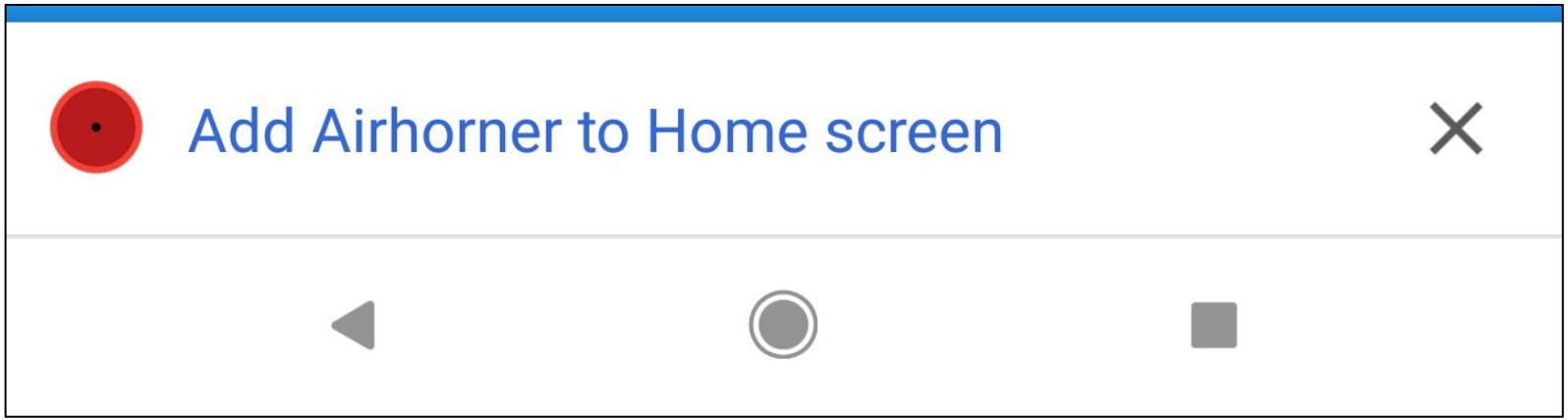
Escutando eventos no SW

- Instalação
- Ativação
- Fetch

App install banners

Continuação

Colocando o web app app na home screen



A “mini info bar”

Acionando o prompt manualmente

```
9
10  let deferredPrompt;
11  window.addEventListener('beforeinstallprompt',function(e){
12    // Previne o Chrome de mostrar o prompt automaticamente
13    e.preventDefault();
14    // Guarda o evento para uso futuro
15    deferredPrompt = e;
16  })
17
```

Adiando o prompt para o usuário

```
21
22          // Mostra o prompt
23          deferredPrompt.prompt();
24
25          // Aguarda a resposta do usuário
26          deferredPrompt.userChoice
27            .then(function(choiceResult){
28
29              if(choiceResult.outcome === 'accepted')
30                console.log('Usuário aceitou o prompt');
31              else
32                console.log('Usuário negou o prompt');
33
34              deferredPrompt = null;
35            })
```

Mostrando o prompt em um ponto específico

Promises
e
API Fetch

Promises

O que são Promises?

- O objeto Promise representa a conclusão (ou falha) de uma operação assíncrona e seu valor resultante.
- Promises é uma nova tecnologia que permite a execução de tarefas assíncronas em Javascript de forma melhorada em comparação com o uso de callbacks.
- O uso de promises permite o encadeamento de chamadas e melhor organização do código.

```
var img1 = document.querySelector('.img-1');

img1.addEventListener('load', function() {
  // woo yey image loaded
});

img1.addEventListener('error', function() {
  // argh everything's broken
});
```

Códigos com callbacks

```
1
2  if('serviceWorker' in navigator){
3    navigator.serviceWorker
4      .register('/sw.js')
5      .then(function(event){
6        console.log('SW registrado',event)
7      })
8  }
9
```

Códigos com Promises - registrando um SW

```
getJSON((data) => {
  logData(data);
  getJSON((data) => {
    logData(data);
    getJSON((data) => {
      logData(data);
      getJSON((data) => {
        logData(data);
        getJSON((data) => {
          logData(data);
        });
      });
    });
  });
});
```

```
function callLogDataPromise(data) {
  logData(data);
  return getJSONPromise();
}

getJSONPromise()
  .then(callLogDataPromise)
  .then(callLogDataPromise)
  .then(callLogDataPromise)
  .then(callLogDataPromise)
  .then(callLogDataPromise);

// Or with a loop:
let callPromise = getJSONPromise();

for(let i = 0; i < 5; i++) {
  callPromise.then(callLogDataPromise);
}
```

O fim do callback hell

Fetch

O que é Fetch

- Permite fazer requisições similares ao XMLHttpRequest
- Como são utilizadas Promises, a legibilidade do código é melhorada, evitando callback hell
- Para utilizar service workers as requisições devem ser feitas com código assíncrono, devem utilizar fetch

```
function reqListener() {
  var data = JSON.parse(this.responseText);
  console.log(data);
}

function reqError(err) {
  console.log('Fetch Error :-S', err);
}

var oReq = new XMLHttpRequest();
oReq.onload = reqListener;
oReq.onerror = reqError;
oReq.open('get', './api/some.json', true);
oReq.send();
```

Requisições com XHR

```
fetch('./api/some.json')
  .then(
    function(response) {
      if (response.status !== 200) {
        console.log('Looks like there was a problem. Status Code: ' +
          response.status);
        return;
      }

      // Examine the text in the response
      response.json().then(function(data) {
        console.log(data);
      });
    }
  )
  .catch(function(err) {
    console.log('Fetch Error :-S', err);
  });
```

Requisições com fetch

Polyfills

- https://github.com/github/fetch
- https://github.com/stefanpenner/es6-promise

API Cache

Características

- Evolução do Application Cache
- A API Cache permite armazenar pares de chave e valor, sendo que as chaves são requisições e as respostas obtidas serão armazenadas no cache.
- O cache pode ser acessado pelo service worker e também pelos Javascripts “normais” das páginas, tornando a implementação mais simplista.
- Em resumo, dados do cache podem ser retornados ao invés de uma requisição de rede. Assim, quando a conexão é perdida o usuário ainda consegue utilizar a aplicação.

Métodos principais

- Open: Abre ou cria um novo cache
- Match: verifica se tem uma correspondência no cache, se sim, retorna.
- Add: Recebe uma requisição como argumento. Guarda a resposta no cache
- AddAll: Tem o funcionamento do Add, mas com várias requisições
- Put: Recebe como argumento uma requisição e resposta para armazenamento

Adicionando caches

- Cache de todos os recursos do site
- Cache sob demanda
- Estratégias de cache
  - Network then cache
  - Cache then network
  - Cache only
  - Network only

Próximos passos

- IndexedDB
- Sincronização em segundo plano
- Notificações web
- Gerenciamento de service worker com Workbox e outros

Links úteis

- MDN - Rede de Desenvolvedores da Mozilla - https://developer.mozilla.org/en-US/
- Google Developers - PWA - https://developers.google.com/web/progressive-web-apps/
- Exemplos de Service Workers - https://github.com/GoogleChrome/samples/tree/gh-pages/service-worker
- Google Codelabs - https://codelabs.developers.google.com/
- Tutorial Cache API - https://flaviocopes.com/cache-api/

FIM