01

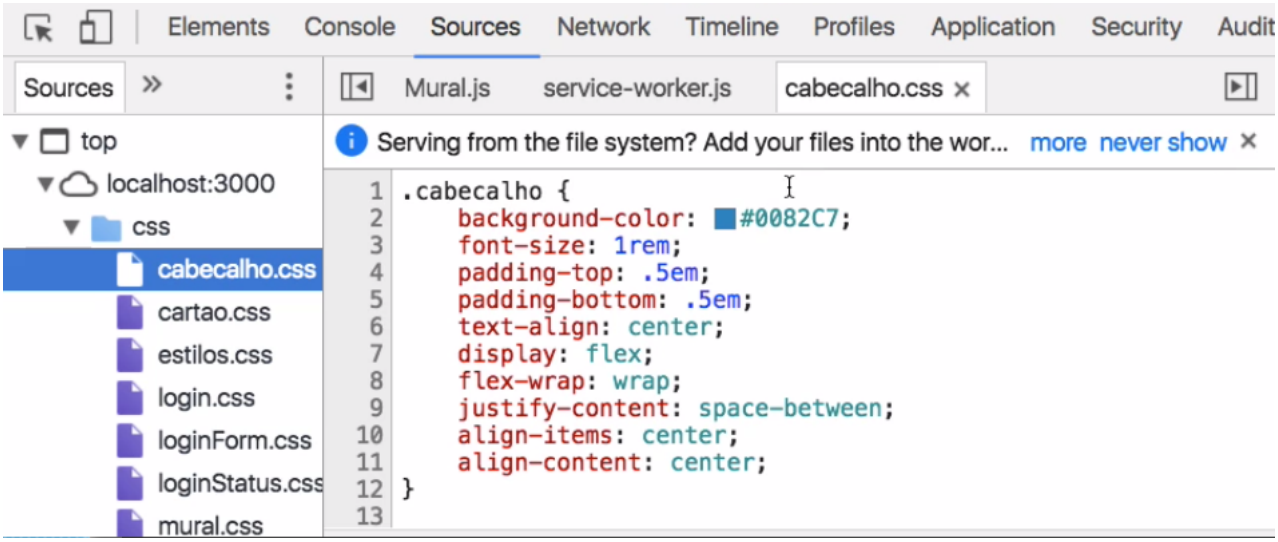
Atualizando nossa Web App - Parte 1

Transcrição

A **PWA** que desenvolvemos está totalmente funcional. Para comemorar, faremos uma alteração nela, trocando a cor do cabeçalho de azul para vermelho. Abrirei o arquivo `cabecalho.css`, alterando o `background-color` para "red":

```
.cabecalho {
  background-color: red;
}
```

Recarregarei a página e verificarei que a alteração não foi feita. Ao abrir a aba "Sources", tentaremos entender o porquê do `cabecalho.css` não ter sido atualizado:



Todos os arquivos são acessados via `service-worker.js` e, mais especificamente, vêm do Cache Storage ou do servidor. Se houver um arquivo no cache, por exemplo o próprio `cabecalho.css`, e caso consultemos "Cache > Cache Storage", ele aparece listado no cache `ceep-arquivos`. Por esta lógica, o arquivo nunca será acessado pelo servidor, e sim pelo cache.

O Service Worker precisa ser avisado de que o arquivo foi atualizado. Para isto, teremos que implementar um pouco de código. Podemos avisar o navegador de que é preciso atualizar o cache. Os arquivos vieram parar no Cache Storage pois foram listados e, na instalação do Service Worker, os adicionei ali (através de `caches.open("ceep-arquivos")`):

```
self.addEventListener("install", function(){
  console.log("Instalou")
  caches.open("ceep-arquivos").then(cache => {
    cache.addAll(arquivos)
    })
  })
```

Este código faz com que os arquivos venham do servidor e sejam colocados no cache. Para atualizar o Cache Storage neste instante, precisaremos executar este mesmo código. Repetir esta função implicaria em instalarmos o Service Worker novamente, o que só ocorre quando alguma alteração é feita no arquivo `service-worker.js`. Utilizaremos este comportamento a nosso favor.

O `offline.manifest` vem do Application Cache, cujos arquivos são baixados novamente apenas mediante atualização da "versão", inventada por nós (`#v3` por exemplo). No caso, a versão em uso é simplesmente um comentário que colocamos, com números que vão sendo alterados para notificação da versão atual.

Agora que utilizamos o Service Worker, podemos criar um código em JAVASCRIPT que, quando executado, mostra qual a versão atual em uso. Como esse valor colocado ali não é usado para nada, vamos colocá-lo em forma de comentário no arquivo `offline.manifest` :

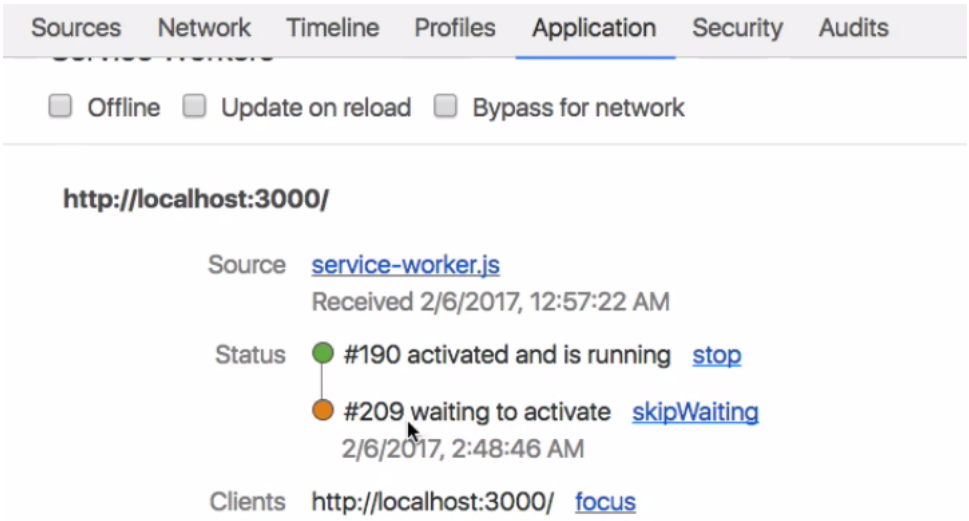
```
//versao 2
```

Alteramos o Service Worker, o que significa que ele será recarregado. Vamos verificar se o novo Service Worker foi instalado e os arquivos acessados são da atualização mais recente. O Cache Storage foi atualizado, temos um cabeçalho novo. Para atualização de arquivos, o comentário em JAVASCRIPT é o suficiente. O que acontece, porém, quando altero um arquivo pré-existente no cache? Se quiser renomear um arquivo que já existia antes?

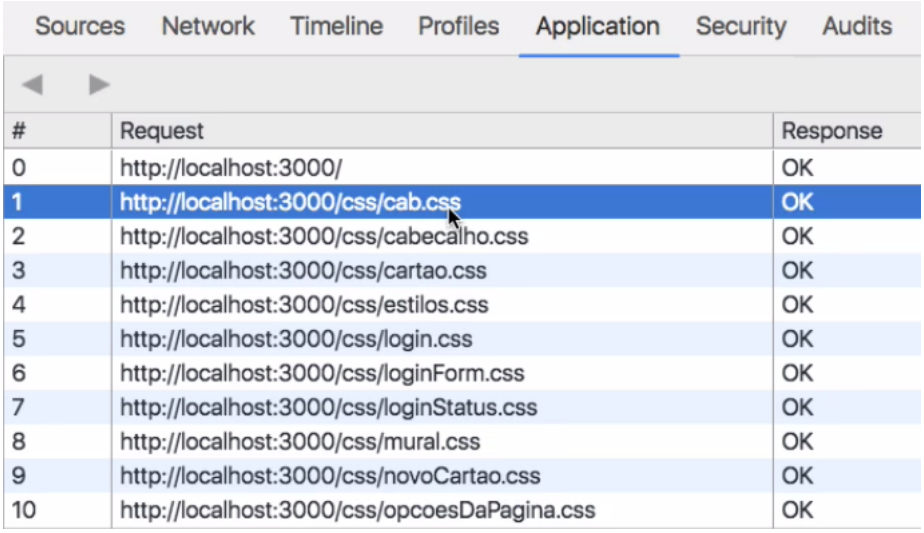
Se quisermos alterar `cabecalho.css` para `cab.css` , por exemplo, precisaremos modificar outros arquivos também. Primeiramente, a tag `<link>` em `index.html` passa a ser:

```
<link rel="stylesheet" href="cab.css">
```

E, em `offline.manifest` , na listagem dos arquivos, faço a mesma alteração, substituindo "cabecalho" por "cab", assim como em `service-worker.js` . Dessa forma, atualizei o nome do arquivo em todos os lugares em que ele aparece. Como o conteúdo do arquivo `service-worker.js` sofreu alteração, quando a página for recarregada, é esperado que a aplicação perceba isto, e é o que acontece, pois vemos que o Service Worker foi modificado:



Verificaremos, então, no "Cache Storage > ceep-arquivos", e vemos que ambos os arquivos, o antigo e o novo, estão presentes ali:



O código continuará funcionando, porém o arquivo `cabecalho.css` não precisaria mais existir, uma vez que a informação armazenada nele é a mesma que consta em `cab.css` . Por ter renomeado o arquivo, o cache acha que precisa manter ambos os arquivos, porque simplesmente adicionamos todos os arquivos da lista, não fazendo nada com aqueles que tínhamos anteriormente; não removemos o que deixamos de usar no Cache Storage. Poderíamos começar a fazer isto agora, ou seja, um código que verifica se o arquivo está ou não no cache e, se este for o caso, removê-lo.

No entanto, não precisamos verificar cada arquivo. Se estou na versão 2 do Service Worker, por exemplo, o cache também precisa ser atualizado, seus arquivos recarregados ou removidos, nomes sendo alterados... O cache precisa ser atualizado junto com a versão, ele precisa funcionar e, além disso, ser refeito.

Na hora de instalarmos o Service Worker, não vamos simplesmente adicionar todos os arquivos. O que faremos é, para além disso, remover o que tínhamos antes através do acesso aos caches. Em `service-worker.js` , portanto, deletaremos tudo que estiver em `ceep-arquivos` :

```
self.addEventListener("install", function(){
  console.log("Instalou")

  caches.delete("ceep-arquivos")
  caches.open("ceep-arquivos").then(cache => {
    cache.addAll(arquivos)
    })


  })
```

Apaguei tudo o que tinha ali, e quando for adicionar os arquivos, só farei isto com aqueles que estiverem listados no código atual. É importante que `caches.delete()` tenha sido completado antes de adicionarmos todos os arquivos. Ou seja, executamos a função `caches.open()` , adicionando os arquivos nesse cache somente quando a `caches.delete()` for finalizada:

```
self.addEventListener("install", function(){
  console.log("Instalou")

  caches.delete("ceep-arquivos")
    .then(function(){
      caches.open("ceep-arquivos").then(cache => {
        cache.addAll(arquivos)
```

```
    })
  })
```

Quando a app for instalada, o cache anterior é apagado e o novo é carregado. Testaremos isto no navegador, carregando a página novamente. O Service Worker novo foi instalado com sucesso, porém ainda não foi ativado. Isto quer dizer que, ao carregarmos "ceep-arquivos" do Cache Storage, não encontraremos mais o arquivo `cabecalho.css`, apenas `cab.css`, pois avisamos o Service Worker de que ele havia sido atualizado, uma vez que seu código foi modificado, e o cache antigo deletado. Aparentemente, tudo funciona perfeitamente desta maneira.

Temos um problema: no momento em que deletamos o cache e entramos no callback `caches.open()`, não temos mais cache algum. Significa que para que ele funcione offline, é preciso re-adicionar todos os arquivos, o que só ocorre após a limpeza do cache. Era o que tinha que acontecer, mas no meio do processo de inclusão de todos os arquivos, algo pode dar errado.

O usuário pode ter a rede comprometida, por exemplo, ficando sem acesso à internet. Se um dos arquivos for impossibilitado de ser adicionado à listagem do cache a partir da função `cache.addAll`, acontece o que já vimos antes: seu comportamento é "matar" tudo, eliminando todos os arquivos. Qualquer erro implica na **não** criação do cache, e também na inexistência de quaisquer arquivos. Além disso, tudo aquilo que existia anteriormente estará deletado! Isso tudo é considerado "má prática". Não deveríamos deletar nada antes de nos assegurarmos de que tudo foi baixado novamente.

Este código recém-criado teria que ser invertido, mas se adicionasse os arquivos ao Cache Storage para posteriormente deletar os antigos, estaria fazendo-o no mesmo cache que está sendo deletado, e perderíamos as informações necessárias:

```
self.addEventListener("install", function(){
  console.log("Instalou")

  caches.open("ceep-arquivos").then(cache => {
      cache.addAll(arquivos)
      caches.delete("ceep-arquivos")
  })

})
```

Isto é um problema, pois se estamos querendo atualizar o cache, não estamos utilizando o mesmo de antes. Começaremos a adicionar arquivos em outros caches, com nomes distintos. Assim, quando o cache for acessado, adicionaremos os arquivos no cache novo. Depois disso, pode-se deletar o que havia antes. Para termos certeza de que a função `caches.delete()` é executada somente após adição dos arquivos, reescrevo o código da seguinte maneira:

```
self.addEventListener("install", function(){
  console.log("Instalou")

  caches.open("ceep-arquivos-2").then(cache => {
      cache.addAll(arquivos)
      .then(function(){
        caches.delete("ceep-arquivos")
      })


  })

})
```

Sendo que **"ceep-arquivos-2"** é o cache novo, deletaremos o antigo recarregando a página. O Service Worker será reinstalado e posteriormente ativado. Ao abrirmos "Cache Storage" e recarregarmos a página, verificaremos o que aconteceu: existe apenas o "ceep-arquivos-2", o que comprova que o cache foi criado, substituindo o antigo corretamente.

A cada atualização do Service Worker, efetivada por conta de qualquer modificação feita em um dos arquivos, por exemplo, teremos que alterar tanto o nome do cache `ceep-arquivos` (para "ceep-arquivos-3", "ceep-arquivos-4"...), quanto o comentário em `service-worker.js` (`//versao 3` , `//versao 4` e variantes). Para evitar este trabalho em dobro, utilizaremos uma variável `versao` que aponta o número da versão atual:

```
let versao = 3
```

Em vez de colocarmos o nome do arquivo como texto, vamos concatená-lo com a versão da app que estiver sendo utilizada, tratando-se portanto de uma **versão dinâmica**:

```
self.addEventListener("install", function(){
  console.log("Instalou")

  caches.open("ceep-arquivos-" + versao).then(cache => {
      cache.addAll(arquivos)
      .then(function(){
        caches.delete("ceep-arquivos")
      })


  })

})
```

Precisamos agora apagar a versão anterior para os usuários. Acrescentarei outra função `caches.delete()` , contatenando-a com a `versao - 1` , mantendo a função que já existia pois alguns usuários podem não passar pela versão anterior, e estejam baixando a versão atual (por exemplo: alguém está baixando a versão 3 sem ter usado a 2).

```
self.addEventListener("install", function(){
  console.log("Instalou")

  caches.open("ceep-arquivos-" + versao).then(cache => {
      cache.addAll(arquivos)
      .then(function(){
        caches.delete("ceep-arquivos-" + (versao -1))
        caches.delete("ceep-arquivos")
      })


  })

})
```

Tudo o que fazemos é cumulativo, e a app precisa ser atualizada para todos os usuários, então mantivemos a função `caches.delete("ceep-arquivos")` . As versões anteriores são deletadas automaticamente, e executaremos o código para ver se o Service Worker será atualizado. Depois disto, recarregaremos a página, abriremos o "Cache Storage". Carregaremos a

página mais uma vez e abriremos o `"ceep-arquivos-3"` com os arquivos correspondentes e atualizados. O próprio código "decide" qual cache manter a partir da versão em uso.