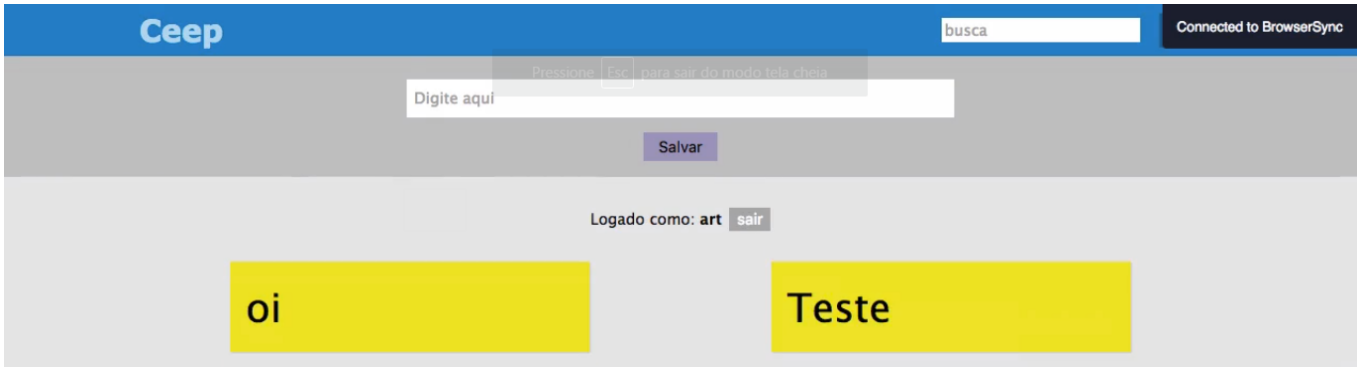
02

Removendo e Editando cartões no localStorage

Transcrição

Nós já conseguimos adicionar cartões novos e recarregar a página sem perdê-los. Mas, quando mudamos o `tipo` (ou seja, a cor) e recarregamos, o `tipo` volta para o anterior. E se mudarmos o conteúdo? Acrescentaremos texto ao cartão `Teste`.

E ao recarregarmos, temos:



O cartão voltou ao conteúdo anterior. Só estamos salvando (`salvaCartoes`) os cartões na função `adiciona`, não a cada alteração. Temos que encontrar um ponto do código no qual possamos chamar a `salvaCartoes` quando os cartões sofrerem alterações.

```
Const Mural = (function(_render, Filtro){
  "use strict"
  let cartoes = JSON.parse(localStorage.getItem("cartoes")) || []
  const render = () => _render({cartoes: cartoes, filtro: Filtro.tagsETexto});

  render()

  Filtro.on("filtrado", render)

  function salvaCartoes (){
    localStorage.setItem("cartoes", JSON.stringify(
      cartoes.map(cartao => ({conteudo: cartao.conteudo, tipo: cartao.tipo})
      ))
  }

  function adiciona(cartao){
    if(logado){
```

```
        cartoes.push(cartao)
        salvaCartoes()
        cartao.on("mudanca.**", render)
        cartao.on("remocao", ()=>{
          cartoes = cartoes.slice(0)
          cartoes.splice(cartoes.index0f(cartao),1)
        render()
    )}
    render()
        return true
      } else {
        alert("Você não está logado")

      }

  }
```

Atualmente, a linha `cartao.on("mudanca.**", render)` nos mostra que estamos renderizando imediatamente a cada modificação feita. Dentro da função adiciona, sempre que recebemos um cartão novo, adicionamos o cartão à lista e os adicionamos ao `localStorage` . Cadastramos instruções para o que deve acontecer quando o cartão for removido e quando ele for alterado. Esse cartão é o que chamamos de **emissor de eventos**.

Quando ele é removido do painel, também sai da lista de cartões e é preciso renderizar novamente. O mesmo acontece para qualquer mudança de propriedade, mudanças tanto no conteúdo quanto no tipo provocam renderização. É por isso que a remoção continua a funcionar. Isso tudo já funciona para todo cartão que passa pela função `adiciona` .

Se recarregarmos a página da maneira que ela está agora, esses cartões não estarão voltando pela função `adiciona` , mas pelo `JSON.parse` que os mapeia. Eles não passam novamente pela função `adiciona` , então não conseguimos removê-los, e quando mudamos sua cor ou conteúdo, essa mudança some quando a página é recarregada.

```
...

function adiciona(cartao){
  if(logado){
    cartoes.push(cartao)
    salvaCartoes()
    cartao.on("mudanca.**", render)
    cartao.on("remocao", ()=>{
      cartoes = cartoes.slice(0)
      cartoes.splice(cartoes.index0f(cartao),1)
    render()
)}
render()
    return true
  } else {
    alert("Você não está logado")

  }

}
```

Precisamos dar uma jeito de fazer o que a `adiciona()` faz com os *listeners* ou *renders* de eventos, mas com os cartões armazenados. Uma alternativa é pegar cada cartão da nossa lista (`forEach`) e aplicar todos esse listeners, os copiando da

`adiciona()`.

```
Const Mural = (function(_render, Filtro){
  "use strict"
  let cartoes = JSON.parse(localStorage.getItem("cartoes")) || []
  cartoes.forEach(cartao => {
    cartao.on("mudanca.**", render)
    cartao.on("remocao", ()=>{
      cartoes = cartoes.slice(0)
      cartoes.splice(cartoes.index0f(cartao),1)
    render()
    })

  const render = () => _render({cartoes: cartoes, filtro: Filtro.tagsETexto});

  render()

  Filtro.on("filtrado", render)

  function salvaCartoes (){
    localStorage.setItem("cartoes", JSON.stringify(
      cartoes.map(cartao => ({conteudo: cartao.conteudo, tipo: cartao.tipo})
      ))
  }
```

Agora temos um código que é aplicado tanto em cartões que são adicionados, como nos armazenados. Mas não queremos renderizar o mural a cada cartão. Queremos salvar o cartão, com a função adequada, `salvaCartoes`. Não é preciso executar a função aqui, pois estamos fazendo uma referência a ela. Não estaremos salvando o cartão imediatamente, apenas quando ele for alterado.

```
Const Mural = (function(_render, Filtro){
  "use strict"
  let cartoes = JSON.parse(localStorage.getItem("cartoes")) || []
  cartoes.forEach(cartao => {
    cartao.on("mudanca.**", salvaCartoes)
    cartao.on("remocao", ()=>{
      cartoes = cartoes.slice(0)
      cartoes.splice(cartoes.index0f(cartao),1)
    render()
    })

  const render = () => _render({cartoes: cartoes, filtro: Filtro.tagsETexto});

  render()

  Filtro.on("filtrado", render)

  function salvaCartoes (){
    localStorage.setItem("cartoes", JSON.stringify(
      cartoes.map(cartao => ({conteudo: cartao.conteudo, tipo: cartao.tipo})
      ))
  }
```

Com isso, devemos poder excluir qualquer cartão sem problemas.

De fato, conseguimos deletar um dos cartões sem problemas. E alterar sua cor?

Ao atualizar, as mudanças permaneceram, graças aos listeners. Para melhorar nosso código, poderíamos colocá-los em um só lugar. Assim, podemos criar algo como `preparaCartao`, que já reúna os listeners de que precisamos ao passarmos o `cartao` como parâmetro.

```
Const Mural = (function(_render, Filtro){
  "use strict"
  let cartoes = JSON.parse(localStorage.getItem("cartoes")) || []
  const render = () => _render({cartoes: cartoes, filtro: Filtro.tagsETexto});

  render()

  Filtro.on("filtrado", render)

  function preparaCartao(cartao){
    cartao.on("mudanca.**", salvaCartoes)
    cartao.on("remocao", ()=>{
      cartoes = cartoes.slice(0)
      cartoes.splice(cartoes.index0f(cartao),1)
      render()
    })
  }


  function salvaCartoes (){
    localStorage.setItem("cartoes", JSON.stringify(
      cartoes.map(cartao => ({conteudo: cartao.conteudo, tipo: cartao.tipo})
      ))
  }

  function adiciona(cartao){
    if(logado){
      cartoes.push(cartao)
      salvaCartoes()
      cartao.on("mudanca.**", render)
      cartao.on("remocao", ()=>{
        cartoes = cartoes.slice(0)
        cartoes.splice(cartoes.index0f(cartao),1)
      render()
  )}
  render()
      return true
    } else {
      alert("Você não está logado")

    }

}
```

Então podemos parar de ouvir os eventos manualmente na função `adiciona` , e chamar a `preparaCartao` .

```
Const Mural = (function(_render, Filtro){
  "use strict"
  let cartoes = JSON.parse(localStorage.getItem("cartoes")) || []
  const render = () => _render({cartoes: cartoes, filtro: Filtro.tagsETexto});

  render()

  Filtro.on("filtrado", render)
```

```
    function preparaCartao(cartao){
      cartao.on("mudanca.**", salvaCartoes)
      cartao.on("remocao", ()=>{
        cartoes = cartoes.slice(0)
        cartoes.splice(cartoes.index0f(cartao),1)
        render()
      })
    }


    function salvaCartoes (){
      localStorage.setItem("cartoes", JSON.stringify(
        cartoes.map(cartao => ({conteudo: cartao.conteudo, tipo: cartao.tipo})
        ))
    }

    function adiciona(cartao){
      if(logado){
        cartoes.push(cartao)
        salvaCartoes()
        cartao.on("mudanca.**", render)
        preparaCartao(cartao)
    render()
        return true
      } else {
        alert("Você não está logado")

      }

  }
```

Se alguma mudança for detectada, os nossos cartões serão salvos, pois chamamos a `preparaCartao` . Vamos recarregar a página e testar uma mudança de cor.

Mudamos o cartão azul para vermelho, e ao atualizar, ele permaneceu vermelho. O mesmo deve valer para remoção. Vamos excluir esse cartão vermelho.

Mas, quando recarregamos a página:

Ele voltou a aparecer. O que está acontecendo? Voltaremos à função `preparaCartao`.

```
function preparaCartao(cartao){
  cartao.on("mudanca.**", salvaCartoes)
  cartao.on("remocao", ()=>{
    cartoes = cartoes.slice(0)
    cartoes.splice(cartoes.index0f(cartao),1)
    render()
  })
}
```

Sempre que um cartão é deletado, executamos o listener da `remocao`. Nele, clonamos a lista com o `slice`, e tiramos o cartão da lista com o `splice`. Mas só estamos usando a variável `cartoes`. Não estamos salvando a nova lista sem o cartão que foi

removido. Assim como precisamos chamar a `salvaCartoes` para adicionar um cartão à lista, precisaremos chamá-la também quando ele for removido.

```
function preparaCartao(cartao){
  cartao.on("mudanca.**", salvaCartoes)
  cartao.on("remocao", ()=>{
    cartoes = cartoes.slice(0)
    cartoes.splice(cartoes.index0f(cartao),1)
    salvaCartoes
    render()
  })
}
```

Dessa maneira, quando recarregamos a aplicação e removemos o mesmo cartão, veremos:

E, ao recarregar novamente:

Ele não voltou. Ficou realmente removido do mural. A funcionalidade de salvar os cartões no `localStorage` parece estar funcionando totalmente. Vamos mudar o conteúdo do cartão para nos certificarmos disso.