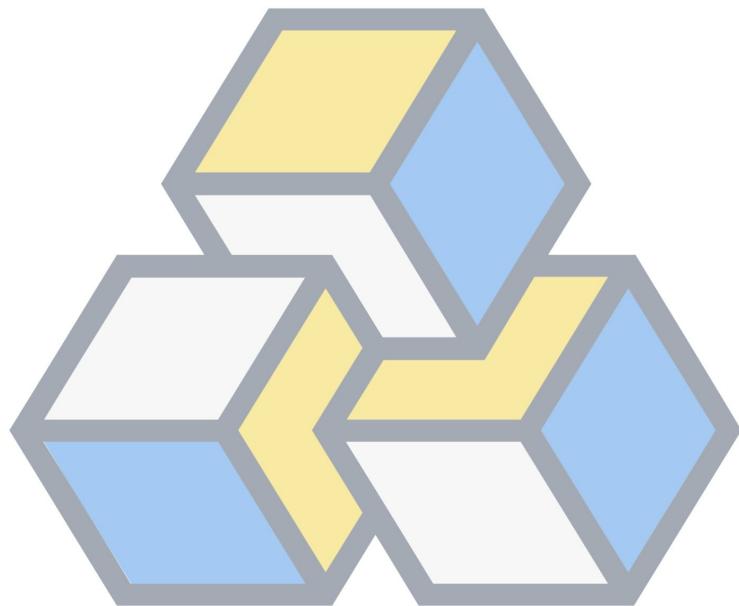


JavaScript Moderno para front-end e back-end

Curso JS-12



Sumário

1 Exercício: Começando pelo visual do navegador	1
1.1 Objetivo	1
1.2 Passo a passo com código	2
2 Exercício: Página inicial – hoisting, var, let, const e o ECMAScript	3
2.1 Objetivo	3
2.2 Passo a passo com código	3
3 Exercício: Aceitando os termos de uso – JavaScript e as APIs do Browser que não são o JavaScript	5
3.1 Objetivo	5
3.2 Passo a passo com código	7
4 Exercício: Aceitando os termos de uso só uma vez – valores null e undefined.	9
4.1 Objetivo	9
4.2 Passo a passo com código	9
5 Exercício: Escolhendo uma página inicial	11
5.1 Objetivo	11
5.2 Passo a passo com código	11
6 Exercício: Impedindo página inicial inválida – Type Coercion e os valores truthy e falsy	13
6.1 Objetivo	13
6.2 Passo a passo com código	13
7 Exercício: Salvando a página inicial	15
7.1 Objetivo	15
7.2 Passo a passo com código	15
8 Exercício: Pedindo permissão para salvar informações – compartilhamento de código entre scripts e o escopo global	16
8.1 Objetivo	16
8.2 Passo a passo com código	17

9 Exercício: Encapsulando código com módulos – o começo de tudo com as IIFEs	19
9.1 Objetivo	19
9.2 Passo a passo com código	19
10 Exercício: Encapsulando código com módulos – os ESModules e a história até aqui	21
10.1 Objetivo	21
10.2 Passo a passo com código	21
11 Exercício: Centralizando módulos – um único ponto de entrada	23
11.1 Objetivo	23
11.2 Passo a passo com código	23
12 Exercício: Início da página de configuração – novos módulos storage	24
12.1 Objetivo	24
12.2 Passo a passo com código	25
13 Exercício: Módulos storage alterando o local storage	28
13.1 Objetivo	28
13.2 Passo a passo com código	28
14 Exercício: ESModules versus compiladores e editores – caminhos absolutos e módulos agregadores	30
14.1 Objetivo	30
14.2 Passo a passo com código	30
15 Exercício: Módulos Storage - alterando valores com funções closure	32
15.1 Objetivo	32
15.2 Passo a passo com código	33
16 Exercício: Página de configuração – salvando informações	35
16.1 Objetivo	35
16.2 Passo a passo com código	35
17 Exercício: Protegendo o storage – closures para encapsulamento e funções como valores	37
17.1 Objetivo	37
17.2 Passo a passo com código	38
18 Exercício: Um novo módulo para tratar endereços	40
18.1 Objetivo	40
18.2 Passo a passo com código	40
19 Exercício: Inciando a navegação – eventos do iframe para atualizar a barra de endereços	43
19.1 Objetivo	43
19.2 Passo a passo com código	43

20 Exercício: Inciando a navegação: mais eventos para melhorar a barra de endereços	45
20.1 Objetivo	45
20.2 Passo a passo com código	45
21 Exercício: Arrumando o reload, recarregando a página atual - os Event Emitters	47
21.1 Objetivo	47
21.2 Passo a passo com código	47
22 Exercício: Botão home e separação de responsabilidades com a função carregar	50
22.1 Objetivo	50
22.2 Passo a passo com código	50
23 Exercício: Digitando o endereço na barra de endereços – event emitters e informações sobre o evento	53
23.1 Objetivo	53
23.2 Passo a passo com código	53
24 Exercício: Botão limpa tudo - Object.keys e iteração em listas	55
24.1 Objetivo	55
24.2 Passo a passo com código	55
25 Exercício: Botão limpa tudo – manipulação e acesso a listas com funções de Array.prototype	57
25.1 Objetivo	57
25.2 Passo a passo com código	57
26 Exercício: Botões avançar e voltar – listas, constantes são mutáveis e mais encapsulamento.	59
26.1 Objetivo	59
26.2 Passo a passo com código	59
27 Exercício: Executando menos lógica nos emitters: objetos literais e funções factory	62
27.1 Objetivo	62
27.2 Passo a passo com código	62
28 Exercício: Atualizando objeto endereço ao digitar	65
28.1 Objetivo	65
28.2 Passo a passo com código	65
29 Exercício: Validando tipos de parâmetros com instanceof e typeof	67
29.1 Objetivo	67
29.2 Passo a passo com código	67
30 Exercício: Criando o tipo Endereco - funções construtoras	69
30.1 Objetivo	69

30.2 Passo a passo com código	69
31 Exercício: Tratamento de erros com erros customizados - mais funções construtoras	72
31.1 Objetivo	72
31.2 Passo a passo com código	72
32 Exercício: Sobrescrevendo métodos padrão – o prototype	75
32.1 Objetivo	75
32.2 Passo a passo com código	75
33 Exercício: Erros customizados extendendo Error – mais prototype e as classes do ECMAScript	78
33.1 Objetivo	78
33.2 Passo a passo com código	79
34 Exercício: Adicionando sites na barra de favoritos – as classes como syntax sugar e o bind	83
34.1 Objetivo	83
34.2 Passo a passo com código	84

Versão: 23.4.25

EXERCÍCIO: COMEÇANDO PELO VISUAL DO NAVEGADOR

1.1 OBJETIVO

Nesse exercício ainda não faremos nenhum código JavaScript. Começaremos alterando nosso código html para exibir a barra de navegação e a janela principal do navegador já carregada na página `blank`.



Página Inicial

Figura 1.1: Visual do navegador

Para recarregar o Cake e ver as alterações você pode apertar `Ctrl+R` ou clicar nos corações da janela inicial:

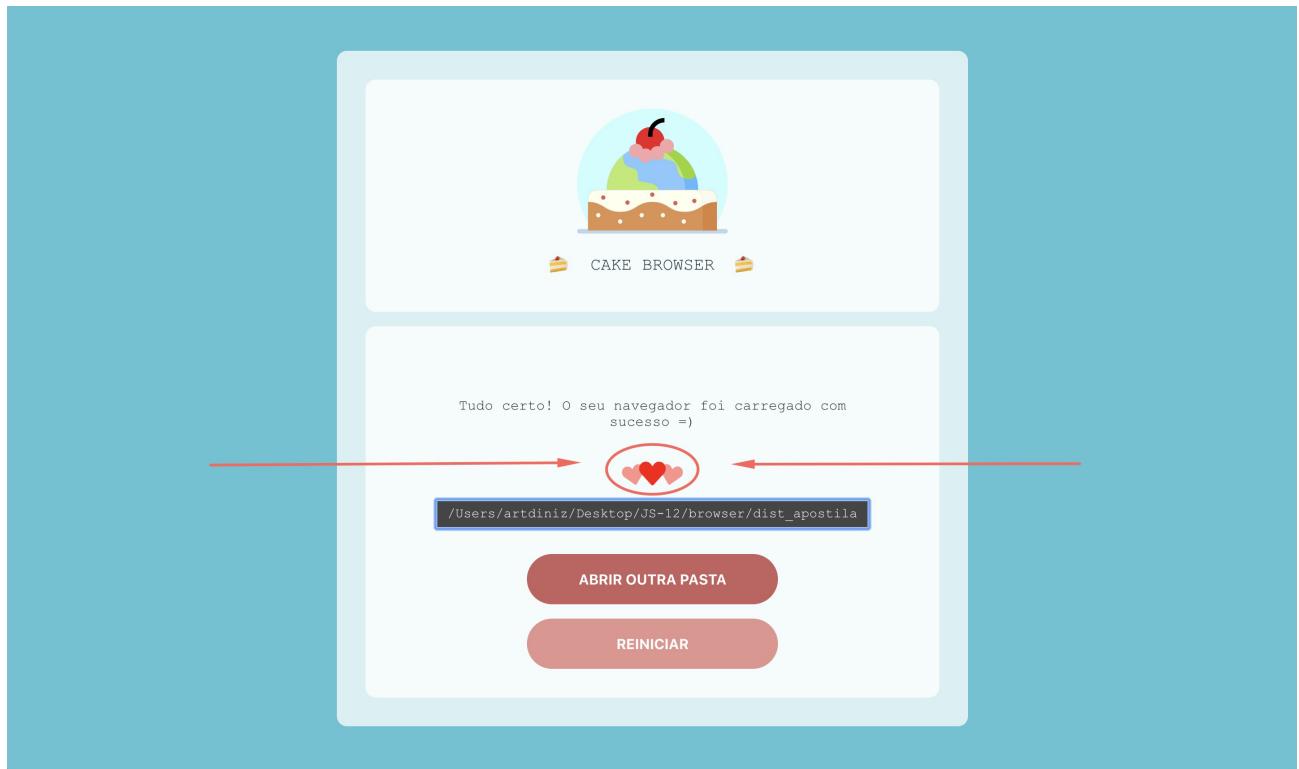


Figura 1.2: Corações da janela inicial

1.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `index.html` na pasta **raíz do projeto** faça as seguintes alterações:

```
# index.html

<meta charset="utf-8">

<link rel="stylesheet" href="styles/cake.css">

+<header>
+  <input type="image" src="images/libCake/recarregar.svg">
+  <input type="text" value="blank">
+</header>
+
+<iframe src="blank"></iframe>
+
<script src="scripts/cake.js"></script>
```

CAPÍTULO 2

EXERCÍCIO: PÁGINA INICIAL – HOISTING, VAR, LET, CONST E O ECMASCRIPT

2.1 OBJETIVO

Nesses próximos exercícios, começaremos a implementar tudo aquilo que acontece quando alguém abre nosso navegador pela primeira vez.

Uma das funcionalidades é que pediremos para a pessoa escolher uma página inicial.

Nesse exercício, especificamente, criaremos nosso primeiro código JavaScript que altera a barra de endereço e a janela principal carregada. Por enquanto a página inicial pode ser fixa.

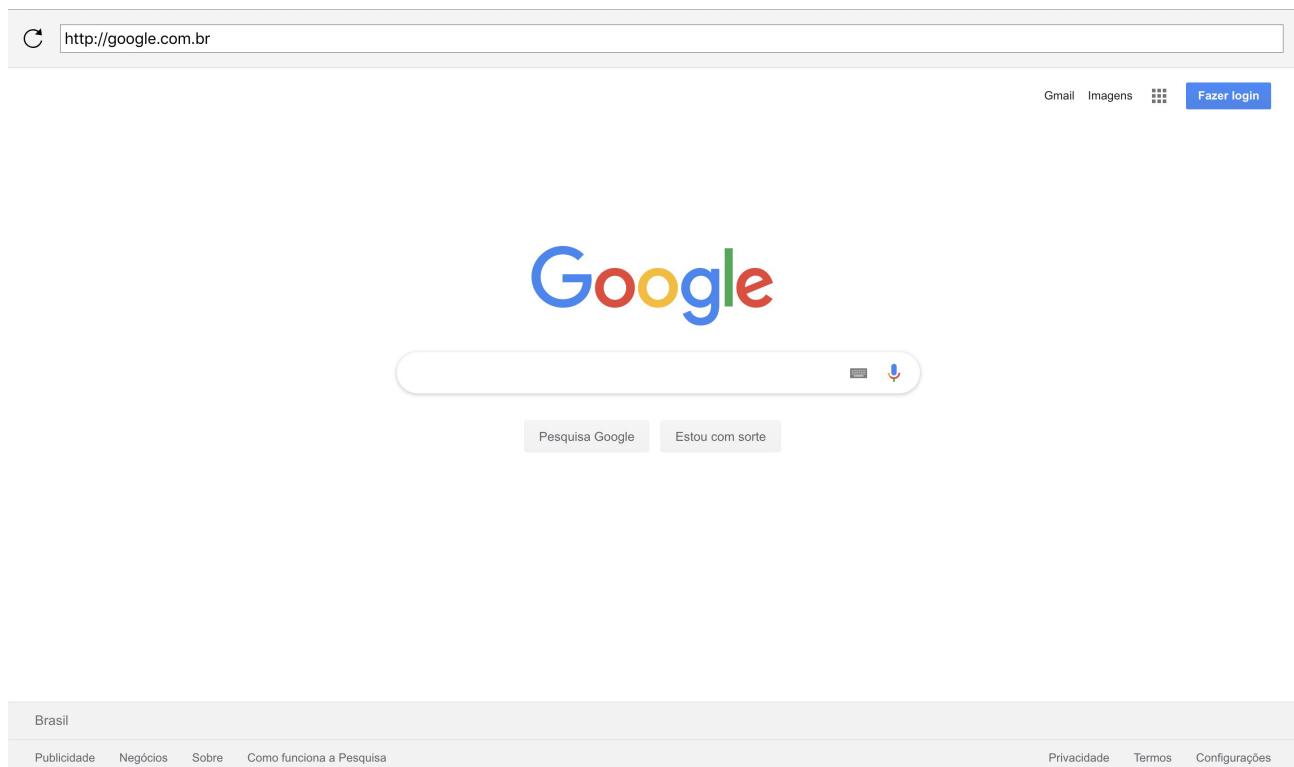


Figura 2.1: Navegador com página inicial já carregada

2.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `index.html` na pasta **raíz do projeto** faça as seguintes alterações:

```
# index.html
```

```
<meta charset="utf-8">

<link rel="stylesheet" href="styles/cake.css">

<header>
-   <input type="image" src="images/libCake/recarregar.svg">
-   <input type="text" value="blank">
+   <input type="image" src="images/libCake/recarregar.svg" onclick="window.location.reload()">
+   <input type="text" id="$inputEndereco" value="blank">
</header>

-<iframe src="blank"></iframe>
+<iframe src="blank" id="$janelaPrincipal"></iframe>

<script src="scripts/cake.js"></script>
+<!-- endbuild -->
+
+<script src="scripts/pedeInfosIniciais/pedePaginaInicial.js"></script>
```

2. Crie o arquivo `pedePaginaInicial.js` na pasta `scripts/pedeInfosIniciais` com o seguinte código:

```
# scripts/pedeInfosIniciais/pedePaginaInicial.js
```

```
+const paginaInicial = 'http://google.com.br'
+
+$janelaPrincipal.src = paginaInicial
+$inputEndereco.value = paginaInicial
```

CAPÍTULO 3

EXERCÍCIO: ACEITANDO OS TERMOS DE USO – JAVASCRIPT E AS APIs DO BROWSER QUE NÃO SÃO O JAVASCRIPT

3.1 OBJETIVO

Se a pessoa não aceitar os termos de uso do nosso navegador, precisamos impedir que ela continue navegando. Para isso, fecharemos a janela caso ela não aceite os termos.

Os termos são os seguintes:

Olá --nome da pessoa aqui--!

Antes de usar o Cake, precisamos que você aceite nossos termos de uso:

- Você aceita que este software foi feito por pessoas que participaram do curso de JavaScript.
- Você aceita que o código dessas pessoas pode acessar tudo o que você digitar aqui.
- Você aceita que tudo aqui está em desenvolvimento e por isso não recomendamos que você troque de navegador agora

Lembre-se que nesse momento, usaremos diversas API's do navegador que nos permitem controlá-lo com JavaScript:

- `prompt`
- `confirm`
- `alert`
- `window.close`

Apesar de escrevermos esse código com JavaScript, essas API's não fazem parte da especificação *ECMAScript*.

Já os valores retornados por essas API's e os valores que passamos como parâmetros para elas, são todos especificados pela ECMA:

- `template strings` para strings multi-linha e interpolação de variáveis com \ Um texto aqui

- ```
`${nomeDaVariavel}` ``
```
- booleanos valores *true* e *false*
  - null representa algo vazio



Figura 3.1: Passo 1 - Pedir o nome da pessoa

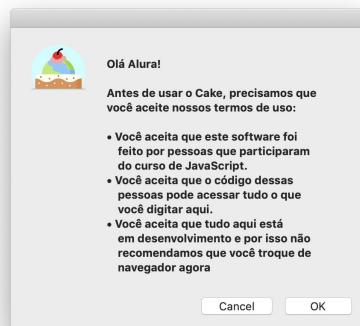


Figura 3.2: Passo 2 - Exibir os termos e perguntar se aceita ou não

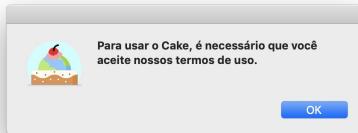


Figura 3.3: Passo 3 - Se não aceitar, avisar que é obrigatório e fechar a janela

## 3.2 PASSO A PASSO COM CÓDIGO

1. No arquivo **index.html** na pasta **raiz do projeto** faça as seguintes alterações:

```
index.html
```

```
<meta charset="utf-8">

<link rel="stylesheet" href="styles/cake.css">

<header>
 <input type="image" src="images/libCake/recarregar.svg" onclick="window.location.reload()">
 <input type="text" id="$inputEndereco" value="blank">
</header>

<iframe src="blank" id="$janelaPrincipal"></iframe>

<script src="scripts/cake.js"></script>

+<script src="scripts/pedeInfosIniciais/termosDeUso.js"></script>
<script src="scripts/pedeInfosIniciais/pedePaginaInicial.js"></script>
```

2. Crie o arquivo **termosDeUso.js** na pasta **scripts/pedeInfosIniciais** com o seguinte código:

```
scripts/pedeInfosIniciais/termosDeUso.js
```

```
+const nome = prompt('Olá, qual o seu nome?')
+const aceitouOsTermos = confirm(`

+ Olá ${nome}!
+
```

```
+ Antes de usar o Cake, precisamos que
+ você aceite nossos termos de uso:
+
+ • Você aceita que este software foi
+ feito por pessoas que participaram
+ do curso de JavaScript.
+ • Você aceita que o código dessas
+ pessoas pode acessar tudo o que
+ você digitar aqui.
+ • Você aceita que tudo aqui está
+ em desenvolvimento e por isso não
+ recomendamos que você troque de
+ navegador agora
+`)
+
+if (!aceitouOsTermos) {
+ alert(
+ nome + ', para continuar é necessário que você aceite os termos de uso.'
+)
+ window.close()
+}
```

# EXERCÍCIO: ACEITANDO OS TERMOS DE USO SÓ UMA VEZ – VALORES NULL E UNDEFINED.

## 4.1 OBJETIVO

Por enquanto, toda vez que a pessoa acessa o navegador pedimos que ela aceite os termos de uso. Nesse exercício, passaremos a lembrar que a pessoa já aceitou anteriormente e não pediremos mais.

Para armazenar informações, podemos utilizar uma API do *Browser* chamada *Local Storage*

## 4.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `termosDeUso.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/termosDeUso.js

+const aceitouOsTermosAnteriormente = localStorage.getItem('aceitouOsTermos')

+if (aceitouOsTermosAnteriormente === null) {
 const nome = prompt('Olá, qual o seu nome?')
 const aceitouOsTermos = confirm(`

 Olá ${nome}!

 Antes de usar o Cake, precisamos que
 você aceite nossos termos de uso:

 • Você aceita que este software foi
 feito por pessoas que participaram
 do curso de JavaScript.
 • Você aceita que o código dessas
 pessoas pode acessar tudo o que
 você digitar aqui.
 • Você aceita que tudo aqui está
 em desenvolvimento e por isso não
 recomendamos que você troque de
 navegador agora
 `)

 if (!aceitouOsTermos) {
 alert(
 nome + ', para continuar é necessário que você aceite os termos de uso.'
)
 window.close()
 }
}
```

```
- }
+ } else {
+ localStorage.setItem('aceitouOsTermos', true)
+ }
+}
```

## CAPÍTULO 5

# EXERCÍCIO: ESCOLHENDO UMA PÁGINA INICIAL

## 5.1 OBJETIVO

Ao invés de fixar a página inicial, passaremos a pedí-la às pessoas.

Um detalhe importante é que a tag `<iframe>` só entende um endereço como externo caso ele seja iniciado com algum protocolo como `http://`:

```
<iframe src="http://caelum.com.br"></iframe>
```

Caso a pessoa não digite nenhum protocolo, adicionaremos um `http://` ao início do endereço digitado.

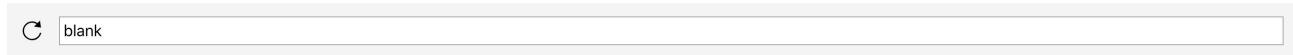


Figura 5.1: Pedindo endereço da página inicial

## 5.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `pedePaginaInicial.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedePaginaInicial.js

-const paginaInicial = 'http://google.com.br'
+let paginaInicial = prompt('Escolha a página inicial')
+
+if (
+ paginaInicial.substring(0, 7) !== 'http://' &&
+ paginaInicial.substring(0, 8) !== 'https://'
+) {
+ paginaInicial = 'http://' + paginaInicial
+}

$janelaPrincipal.src = paginaInicial
$inputEndereco.value = paginaInicial
```

# EXERCÍCIO: IMPEDINDO PÁGINA INICIAL INVÁLIDA – TYPE COERCION E OS VALORES TRUTHY E FALSEY

## 6.1 OBJETIVO

Nem sempre as pessoas vão digitar algo no campo para inserir o endereço da página inicial. Nesses casos o endereço final será "http://" .

Caso a pessoa cancele o *popup* para inserir o endereço, o valor retornado será `undefined` . Nesse casos o endereço final será "http://`undefined`" .

Nesse exercício, impediremos que a página inicial seja definida com algum desses valores inválidos.

## 6.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `pedePaginaInicial.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedePaginaInicial.js

let paginaInicial = prompt('Escolha a página inicial')

-if (
- paginaInicial.substring(0, 7) !== 'http://' &&
- paginaInicial.substring(0, 8) !== 'https://'
)-{
- paginaInicial = 'http://' + paginaInicial
}
+if (paginaInicial) {
+ if (
+ paginaInicial.substring(0, 7) !== 'http://' &&
+ paginaInicial.substring(0, 8) !== 'https://'
+) {
+ paginaInicial = 'http://' + paginaInicial
+ }
-$janelaPrincipal.src = paginaInicial
-$inputEndereco.value = paginaInicial
+$janelaPrincipal.src = paginaInicial
+$inputEndereco.value = paginaInicial
+}
```



# EXERCÍCIO: SALVANDO A PÁGINA INICIAL

## 7.1 OBJETIVO

Assim como fizemos com os termos de uso, podemos salvar o endereço da página inicial e apenas pedí-lo se não houver endereço salvo.

Nessa verificação de existência de um endereço salvo, podemos usar o mecanismo de *Type Coercion* novamente.

## 7.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `pedePaginaInicial.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedePaginaInicial.js

-let paginaInicial = prompt('Escolha a página inicial')
+let paginaInicial = localStorage.getItem('paginaInicial')
+
+if (!paginaInicial) {
+ paginaInicial = prompt('Escolha a página inicial')
+}

if (paginaInicial) {
 if (
 paginaInicial.substring(0, 7) !== 'http://'
 && paginaInicial.substring(0, 8) !== 'https://'
) {
 paginaInicial = 'http://' + paginaInicial
 }

 $janelaPrincipal.src = paginaInicial
 $inputEndereco.value = paginaInicial
+
+ localStorage.setItem('paginaInicial', paginaInicial)
}
```

# EXERCÍCIO: PEDINDO PERMISSÃO PARA SALVAR INFORMAÇÕES – COMPARTILHAMENTO DE CÓDIGO ENTRE SCRIPTS E O ESCOPO GLOBAL

## 8.1 OBJETIVO

Acabamos de salvar o endereço da página inicial da pessoa. Mas antes de sair salvando todas as informações da pessoa, precisamos pedir permissão para isso. A pessoa deve poder optar se quer ou não que a gente armazene as informações dela.

Caso ela opte por não salvar as informações, não será preciso pedir nem armazenar o endereço da página inicial.

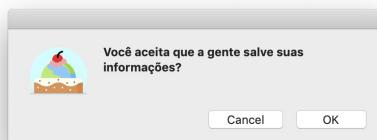


Figura 8.1: Perguntando se a pessoa quer salvemos suas informações

## 8.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `index.html` na pasta **raiz do projeto** faça as seguintes alterações:

```
index.html

<meta charset="utf-8">

<link rel="stylesheet" href="styles/cake.css">

<header>
 <input type="image" src="images/libCake/recarregar.svg" onclick="window.location.reload()">
 <input type="text" id="$inputEndereco" value="blank">
</header>

<iframe src="blank" id="$janelaPrincipal"></iframe>

<script src="scripts/cake.js"></script>

<script src="scripts/pedeInfosIniciais/termosDeUso.js"></script>
+<script src="scripts/pedeInfosIniciais/pedeAceitouSalvar.js"></script>
<script src="scripts/pedeInfosIniciais/pedePaginaInicial.js"></script>
```

2. Crie o arquivo `pedeAceitouSalvar.js` na pasta **scripts/pedeInfosIniciais** com o seguinte código:

```
scripts/pedeInfosIniciais/pedeAceitouSalvar.js

+let aceitouSalvar = JSON.parse(localStorage.getItem('aceitouSalvar'))
+
+if (aceitouSalvar === null) {
+ aceitouSalvar = confirm('Você aceita que a gente salve suas informações?')
+
+ if (!aceitouSalvar) {
+ alert('Você pode mudar isso na página de configurações')
+ }
+
+ localStorage.setItem('aceitouSalvar', aceitouSalvar)
+}
```

3. No arquivo `pedePaginaInicial.js` na pasta **scripts/pedeInfosIniciais** faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedePaginaInicial.js

-del paginaInicial = localStorage.getItem('paginaInicial')
+if (aceitouSalvar === true) {
+ let paginaInicial = localStorage.getItem('paginaInicial')

-if (!paginaInicial) {
- paginaInicial = prompt('Escolha a página inicial')
-}
-
-if (paginaInicial) {
- if (
- paginaInicial.substring(0, 7) !== 'http://'
- || paginaInicial.substring(0, 8) !== 'https://'
-) {
```

```
- paginaInicial = 'http://' + paginaInicial
+ if (!paginaInicial) {
+ paginaInicial = prompt('Escolha a página inicial')
}

- $janelaPrincipal.src = paginaInicial
- $inputEndereco.value = paginaInicial
+ if (paginaInicial) {
+ if (
+ paginaInicial.substring(0, 7) !== 'http://' &&
+ paginaInicial.substring(0, 8) !== 'https://'
+) {
+ paginaInicial = 'http://' + paginaInicial
+ }
}

- localStorage.setItem('paginaInicial', paginaInicial)
+ $janelaPrincipal.src = paginaInicial
+ $inputEndereco.value = paginaInicial
+
+ localStorage.setItem('paginaInicial', paginaInicial)
+ }
```

# EXERCÍCIO: ENCAPSULANDO CÓDIGO COM MÓDULOS – O COMEÇO DE TUDO COM AS IIFES

## 9.1 OBJETIVO

Nesse exercício, limitaremos o acesso e a manipulação das variáveis dos nossos scripts por outros scripts. Faremos isso criando um **módulo** com uma IIFE e ativando o modo *strict* do JavaScript.

Lembrando que **IIFE** é um acrônimo de:

Immediately Invoked Function Expression

Traduzindo: Função Anônima Imediatamente Invocada

Faremos isso pois os navegadores carregam os arquivos nas tags `<script>` e fazem com que todos rodem no mesmo escopo. Toda variável que esteja 'solta' num script fica global! Vimos que por vários motivos isso pode ser indesejável.

## 9.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `pedeAceitouSalvar.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedeAceitouSalvar.js
```

```
-let aceitouSalvar = JSON.parse(localStorage.getItem('aceitouSalvar'))
+const aceitouSalvarGlobal = (function() {
+ 'use strict'

-if (aceitouSalvar === null) {
- aceitouSalvar = confirm('Você aceita que a gente salve suas informações?')
+ let aceitouSalvar = JSON.parse(localStorage.getItem('aceitouSalvar'))

- if (!aceitouSalvar) {
- alert('Você pode mudar isso na página de configurações')
+ if (aceitouSalvar === null) {
+ aceitouSalvar = confirm('Você aceita que a gente salve suas informações?')
+
+ if (!aceitouSalvar) {
+ alert('Você pode mudar isso na página de configurações')
```

```

+
+ }
+
+ localStorage.setItem('aceitouSalvar', aceitouSalvar)
 }

- localStorage.setItem('aceitouSalvar', aceitouSalvar)
-}
+
+ return aceitouSalvar
+})()

```

2. No arquivo `pedePaginaInicial.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedePaginaInicial.js

-if (aceitouSalvar === true) {
+if (aceitouSalvarGlobal === true) {
 let paginaInicial = localStorage.getItem('paginaInicial')

 if (!paginaInicial) {
 paginaInicial = prompt('Escolha a página inicial')
 }

 if (paginaInicial) {
 if (
 paginaInicial.substring(0, 7) !== 'http://' &&
 paginaInicial.substring(0, 8) !== 'https://'
) {
 paginaInicial = 'http://' + paginaInicial
 }
 }

 $janelaPrincipal.src = paginaInicial
 $inputEndereco.value = paginaInicial

 localStorage.setItem('paginaInicial', paginaInicial)
}
}
```

# EXERCÍCIO: ENCAPSULANDO CÓDIGO COM MÓDULOS – OS ESMODULES E A HISTÓRIA ATÉ AQUI

## 10.1 OBJETIVO

Apóieitando que estamos criando nosso navegador em cima de uma plataforma atualizada, já temos a possibilidade de usar os ESMODULES para a criação de módulos no nosso código.

Nesse exercício, deixaremos de ter scripts com IIFE's e passaremos a criar módulos com a sintaxe dos ESMODULES. A partir desse exercício, será possível importar e renomear valores de qualquer arquivo, sem conflitos de nomes e sem dependência de uma ordem de inclusão no `html`.

## 10.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `index.html` na pasta `raiz do projeto` faça as seguintes alterações:

```
index.html
```

```
<meta charset="utf-8">

<link rel="stylesheet" href="styles/cake.css">

<header>
 <input type="image" src="images/libCake/recarregar.svg" onclick="window.location.reload()">
 <input type="text" id="$inputEndereco" value="blank">
</header>

<iframe src="blank" id="$janelaPrincipal"></iframe>

<script src="scripts/cake.js"></script>

<script src="scripts/pedeInfosIniciais/termosDeUso.js"></script>
-<script src="scripts/pedeInfosIniciais/pedeAceitouSalvar.js"></script>
-<script src="scripts/pedeInfosIniciais/pedePaginaInicial.js"></script>
+<script type="module" src="scripts/pedeInfosIniciais/pedePaginaInicial.js"></script>
```

2. No arquivo `pedeAceitouSalvar.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedeAceitouSalvar.js
```

```

-const aceitouSalvarGlobal = (function() {
- 'use strict'
+let aceitouSalvar = JSON.parse(localStorage.getItem('aceitouSalvar'))

- let aceitouSalvar = JSON.parse(localStorage.getItem('aceitouSalvar'))
+if (aceitouSalvar === null) {
+ aceitouSalvar = confirm('Você aceita que a gente salve suas informações?')

- if (aceitouSalvar === null) {
- aceitouSalvar = confirm('Você aceita que a gente salve suas informações?')
-
- if (!aceitouSalvar) {
- alert('Você pode mudar isso na página de configurações')
 }
-
- localStorage.setItem('aceitouSalvar', aceitouSalvar)
+ if (!aceitouSalvar) {
+ alert('Você pode mudar isso na página de configurações')
 }

- return aceitouSalvar
})()
+ localStorage.setItem('aceitouSalvar', aceitouSalvar)
}
+
+export default aceitouSalvar

```

3. No arquivo `pedePaginaInicial.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedePaginaInicial.js
```

```

-if (aceitouSalvarGlobal === true) {
+import aceitouSalvar from './pedeAceitouSalvar.js'
+
+if (aceitouSalvar === true) {
 let paginaInicial = localStorage.getItem('paginaInicial')

 if (!paginaInicial) {
 paginaInicial = prompt('Escolha a página inicial')
 }

 if (paginaInicial) {
 if (
 paginaInicial.substring(0, 7) !== 'http://' &&
 paginaInicial.substring(0, 8) !== 'https://'
) {
 paginaInicial = 'http://' + paginaInicial
 }

 $janelaPrincipal.src = paginaInicial
 $inputEndereco.value = paginaInicial

 localStorage.setItem('paginaInicial', paginaInicial)
 }
}

```

# EXERCÍCIO: CENTRALIZANDO MÓDULOS – UM ÚNICO PONTO DE ENTRADA

## 11.1 OBJETIVO

Neste exercício não adicionaremos nenhuma nova funcionalidade. Reorganizaremos nosso jeito de importar todo nosso código JavaScript. Centralizaremos nossos código JS no módulo `main.js` que fará todos os `import` necessários para que nossa app rode.

## 11.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `index.html` na pasta **raiz do projeto** faça as seguintes alterações:

```
index.html
```

```
<meta charset="utf-8">

<link rel="stylesheet" href="styles/cake.css">

<header>
 <input type="image" src="images/libCake/recarregar.svg" onclick="window.location.reload()">
 <input type="text" id="$inputEndereco" value="blank">
</header>

<iframe src="blank" id="$janelaPrincipal"></iframe>

<script src="scripts/cake.js"></script>

-<script src="scripts/pedeInfosIniciais/termosDeUso.js"></script>
-<script type="module" src="scripts/pedeInfosIniciais/pedePaginaInicial.js"></script>
+<script type="module" src="scripts/main.js"></script>
```

2. Crie o arquivo `main.js` na pasta `scripts` com o seguinte código:

```
scripts/main.js
```

```
+import './pedeInfosIniciais/termosDeUso.js'
+import './pedeInfosIniciais/pedeAceitouSalvar.js'
+import './pedeInfosIniciais/pedePaginaInicial.js'
```

# EXERCÍCIO: INÍCIO DA PÁGINA DE CONFIGURAÇÃO – NOVOS MÓDULOS STORAGE

## 12.1 OBJETIVO

Criamos uma nova página de configuração que será acessada quando a pessoa clicar no ícone da engrenagem ao lado da barra de endereço.

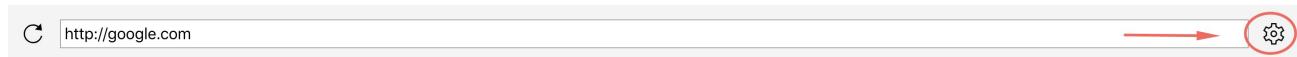


Figura 12.1: Link para a página de configuração

Nessa página de configuração, visualizaremos as informações salvas no `localStorage`, mas não acessaremos ele diretamente. Para evitar duplicação de código, criaremos nossos novos módulos dentro de `storage/`. Esses módulos serão responsáveis por todo acesso ao `localStorage`.

## Configurações

Página inicial:

Permissão de compartilhamento:

[Voltar](#)

Figura 12.2: Página de configuração exibindo as informações salvas

## 12.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `index.html` na pasta **raiz do projeto** faça as seguintes alterações:

```
index.html
```

```
<meta charset="utf-8">

<link rel="stylesheet" href="styles/cake.css">

<header>
 <input type="image" src="images/libCake/recarregar.svg" onclick="window.location.reload()">
 <input type="text" id="$inputEndereco" value="blank">
+
+
+
</header>

<iframe src="blank" id="$janelaPrincipal"></iframe>

<script src="scripts/cake.js"></script>

<script type="module" src="scripts/main.js"></script>
```

2. Crie o arquivo `paginaConfiguracao.js` na pasta **scripts** com o seguinte código:

```
scripts/paginaConfiguracao.js
```

```
+import paginaInicial from '/scripts/storage/paginaInicial.js'
```

```
+import aceitouSalvar from '/scripts/storage/aceitouSalvar.js'
+
+$inputPaginaInicial.value = paginaInicial
+$inputPermitiuSalvar.checked = aceitouSalvar
```

3. No arquivo `pedeAceitouSalvar.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedeAceitouSalvar.js
```

```
-let aceitouSalvar = JSON.parse(localStorage.getItem('aceitouSalvar'))
+import aceitouSalvar from '/scripts/storage/aceitouSalvar.js'

if (aceitouSalvar === null) {
- aceitouSalvar = confirm('Você aceita que a gente salve suas informações?')
+ const aceitou = confirm('Você aceita que a gente salve suas informações?')

- if (!aceitouSalvar) {
+ if (!aceitou) {
 alert('Você pode mudar isso na página de configurações')
 }

- localStorage.setItem('aceitouSalvar', aceitouSalvar)
+ localStorage.setItem('aceitouSalvar', aceitou)
}

-
-export default aceitouSalvar
```

4. No arquivo `pedePaginaInicial.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedePaginaInicial.js
```

```
-import aceitouSalvar from './pedeAceitouSalvar.js'
+import aceitouSalvar from '/scripts/storage/aceitouSalvar.js'
+import paginaInicial from '/scripts/storage/paginaInicial.js'

if (aceitouSalvar === true) {
- let paginaInicial = localStorage.getItem('paginaInicial')
+ let paginaInicialPadrao = paginaInicial

- if (!paginaInicial) {
- paginaInicial = prompt('Escolha a página inicial')
+ if (!paginaInicialPadrao) {
+ paginaInicialPadrao = prompt('Escolha a página inicial')
 }

- if (paginaInicial) {
+ if (paginaInicialPadrao) {
 if (
 - paginaInicial.substring(0, 7) !== 'http://'
 - paginaInicial.substring(0, 8) !== 'https://'
+ paginaInicialPadrao.substring(0, 7) !== 'http://'
+ paginaInicialPadrao.substring(0, 8) !== 'https://'
) {
 - paginaInicial = 'http://' + paginaInicial
+ paginaInicialPadrao = 'http://' + paginaInicialPadrao
 }
 }
}
```

```

- $janelaPrincipal.src = paginaInicial
- $inputEndereco.value = paginaInicial
+ $janelaPrincipal.src = paginaInicialPadrao
+ $inputEndereco.value = paginaInicialPadrao

- localStorage.setItem('paginaInicial', paginaInicial)
+ localStorage.setItem('paginaInicial', paginaInicialPadrao)
}
}

```

5. Crie o arquivo **aceitouSalvar.js** na pasta **scripts/storage** com o seguinte código:

```
scripts/storage/aceitouSalvar.js

+export default JSON.parse(localStorage.getItem('aceitouSalvar'))
```

6. Crie o arquivo **paginaInicial.js** na pasta **scripts/storage** com o seguinte código:

```
scripts/storage/paginaInicial.js

+export default localStorage.getItem('paginaInicial')
```

7. No arquivo **config.html** na pasta **raiz do projeto** faça as seguintes alterações:

```
config.html

<link rel="stylesheet" href="styles/cake-config.css">

<h1>Configurações</h1>

<label>
 Página inicial:
- <input type="text">
+ <input type="text" id="$inputPaginaInicial">
</label>

<label>
 Permissão de compartilhamento:
- <input type="checkbox">
+ <input type="checkbox" id="$inputPermitiuSalvar">
</label>

<button>
 Salvar configurações
</button>
Voltar
+
+<script type="module" src="scripts/paginaConfiguracao.js"></script>
```

# EXERCÍCIO: MÓDULOS STORAGE ALTERANDO O LOCAL STORAGE

## 13.1 OBJETIVO

Para evitar código duplicado que altera o `localStorage`, criamos funções `setter` em nossos módulos de `/scripts/storage/` que serão responsáveis por alterar o `localStorage`.

Refatoramos nosso código anterior para que não haja mais acesso ao `localStorage` fora desses módulos.

## 13.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `pedeAceitouSalvar.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedeAceitouSalvar.js

-import aceitouSalvar from '/scripts/storage/aceitouSalvar.js'
+import aceitouSalvar, {
+ setAceitouSalvar
+} from '/scripts/storage/aceitouSalvar.js'

if (aceitouSalvar === null) {
 const aceitou = confirm('Você aceita que a gente salve suas informações?')

 if (!aceitou) {
 alert('Você pode mudar isso na página de configurações')
 }

- localStorage.setItem('aceitouSalvar', aceitou)
+ setAceitouSalvar(aceitou)
}
```

2. No arquivo `pedePaginaInicial.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedePaginaInicial.js

import aceitouSalvar from '/scripts/storage/aceitouSalvar.js'
-import paginaInicial from '/scripts/storage/paginaInicial.js'
+import paginaInicial, {
+ setPaginaInicial
```

```

+} from '/scripts/storage/paginaInicial.js'

if (aceitouSalvar === true) {
 let paginaInicialPadrao = paginaInicial

 if (!paginaInicialPadrao) {
 paginaInicialPadrao = prompt('Escolha a página inicial')
 }

 if (paginaInicialPadrao) {
 if (
 paginaInicialPadrao.substring(0, 7) !== 'http://' &&
 paginaInicialPadrao.substring(0, 8) !== 'https://'
) {
 paginaInicialPadrao = 'http://' + paginaInicialPadrao
 }
 }

 $janelaPrincipal.src = paginaInicialPadrao
 $inputEndereco.value = paginaInicialPadrao

- localStorage.setItem('paginaInicial', paginaInicialPadrao)
+ setPaginaInicial(paginaInicialPadrao)
 }
}

```

3. No arquivo `aceitouSalvar.js` na pasta `scripts/storage` faça as seguintes alterações:

```
scripts/storage/aceitouSalvar.js
```

```

export default JSON.parse(localStorage.getItem('aceitouSalvar'))
+
+export function setAceitouSalvar(aceitouSalvar) {
+ localStorage.setItem('aceitouSalvar', aceitouSalvar)
+}

```

4. No arquivo `paginaInicial.js` na pasta `scripts/storage` faça as seguintes alterações:

```
scripts/storage/paginaInicial.js
```

```

export default localStorage.getItem('paginaInicial')
+
+export function setPaginaInicial(valor) {
+ localStorage.setItem('paginaInicial', valor)
+}

```

# EXERCÍCIO: ESMODULES VERSUS COMPILADORES E EDITORES – CAMINHOS ABSOLUTOS E MÓDULOS AGREGADORES

## 14.1 OBJETIVO

Nos nossos códigos, passamos a importar nossos módulos com um caminho absoluto:

```
import { aceitouSalvar } from '/scripts/storage/aceitouSalvar.js'
```

O VSCode por padrão acha que endereços iniciados com / se referem à pasta raiz do seu computador, e não do nosso projeto. Isso se deve pelo fato de que ESMODULES ainda não são amplamente adotados em todas as plataformas.

Em projetos que usam `import` e `export`, na maioria das vezes, o código é passado para algum compilador que traduz esses módulos para módulos de padrões mais antigos, como *CommonJS* ou até mesmo *IIFEs*, que comentamos anteriormente. Esse processo de compilação geralmente ocorre na máquina de quem está desenvolvendo, e assim, caminhos iniciados em / são considerados também como a partir da pasta raiz do computador.

Com ESMODULES nos browsers, esses nossos imports são URLs relativas sempre ao endereço em que a aplicação está rodando. No nosso caso, relativas à pasta raiz do nosso projeto.

A configuração que faremos diz que a pasta raiz do nosso projeto deve ser considerada sempre que algum endereço de `import` iniciado com / for encontrado no nosso código.

Com essa configuração, ganhamos diversas facilidades do editor na hora de acessar, navegar e até mesmo mudar códigos entre nossos módulos.

## 14.2 PASSO A PASSO COM CÓDIGO

1. Crie o arquivo `jsconfig.json` na pasta **raiz do projeto** com o seguinte código:

```
jsconfig.json
```

```
+{
+ "compilerOptions": {
+ "baseUrl": ".",
+ "module": "esnext",
+ "paths": {
+ "//*": [".//*"]
+ }
+ }
+}
```

2. No arquivo `main.js` na pasta `scripts` faça as seguintes alterações:

```
scripts/main.js

-import './pedeInfosIniciais/termosDeUso.js'
-import './pedeInfosIniciais/pedeAceitouSalvar.js'
-import './pedeInfosIniciais/pedePaginaInicial.js'
+import '/scripts/pedeInfosIniciais/index.js'
```

3. Crie o arquivo `index.js` na pasta `scripts/pedeInfosIniciais` com o seguinte código:

```
scripts/pedeInfosIniciais/index.js

+import './termosDeUso.js'
+import './pedeAceitouSalvar.js'
+import './pedePaginaInicial.js'
```

# EXERCÍCIO: MÓDULOS STORAGE - ALTERANDO VALORES COM FUNÇÕES CLOSURE

## 15.1 OBJETIVO

Chegamos num ponto da aplicação no qual precisamos alterar o valor de `aceitouSalvar` e `paginaInicial`. Porém, até a gora elas eram os valores padrões exportados pelos nossos módulos de `/scripts/storage/`. Vimos que esses valores exportados em nossos módulos são constantes e não podem ter seus valores alterados. Assim, transformarmos esses valores em variáveis `let`:

```
-export default JSON.parse(localStorage.getItem('aceitouSalvar'))
+export let aceitouSalvar = JSON.parse(localStorage.getItem('aceitouSalvar'))
```

Porém, mesmo declarados como `let` vimos que outros módulos que importam essas variáveis não conseguem alterá-las. Qualquer propriedade exportada por um módulo fica constante para qualquer outro módulo que o importe.

Esse comportamento permite que a gente controle como e quem acessará e modificará essas variáveis.

Para permitir que essas variáveis possam ter seu valor alterado, alteramos nossas funções `setter` para que alterassem o valor dos nossos `let`:

```
export let aceitouSalvar = JSON.parse(localStorage.getItem('aceitouSalvar'))

export function setAceitouSalvar(valor) {
+ aceitouSalvar = valor
 localStorage.setItem("aceitouSalvar", valor)
}
```

Aproveitamos o comportamento das funções no JS que têm acesso a todas as variáveis dentro do escopo em que foram criadas.

Se alguém chamar `setAceitouSalvar`, conseguirá alterar o valor da variável `aceitouSalvar`, por mais que ela esteja encapsulada no módulo.

Podemos chamar `setAceitouSalvar` de *Closure*. Um tipo de função que tem acesso a um ambiente/escopo privado onde foi criada.

As *closures* permitem que a gente encapsule código em nossos módulos e defina de que modo outros códigos terão ou não acesso a variáveis e outras funções.

## 15.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `pedeAceitouSalvar.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedeAceitouSalvar.js

-import aceitouSalvar, {
- setAceitouSalvar
-} from '/scripts/storage/aceitouSalvar.js'
+import * as storage from '/scripts/storage/aceitouSalvar.js'

-if (aceitouSalvar === null) {
+if (storage.aceitouSalvar === null) {
 const aceitou = confirm('Você aceita que a gente salve suas informações?')

 if (!aceitou) {
 alert('Você pode mudar isso na página de configurações')
 }

- setAceitouSalvar(aceitou)
+ storage.setAceitouSalvar(aceitou)
}
```

2. No arquivo `pedePaginaInicial.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedePaginaInicial.js

-import aceitouSalvar from '/scripts/storage/aceitouSalvar.js'
-import paginaInicial, {
+import { aceitouSalvar as storageAceitouSalvar } from '/scripts/storage/aceitouSalvar.js'
+import {
+ paginaInicial,
 setpaginaInicial
} from '/scripts/storage/paginaInicial.js'

-if (aceitouSalvar === true) {
+if (storageAceitouSalvar === true) {
 let paginaInicialPadrao = paginaInicial

 if (!paginaInicialPadrao) {
 paginaInicialPadrao = prompt('Escolha a página inicial')
 }

 if (paginaInicialPadrao) {
 if (
 paginaInicialPadrao.substring(0, 7) !== 'http://' &&
 paginaInicialPadrao.substring(0, 8) !== 'https://'
) {
 paginaInicialPadrao = 'http://' + paginaInicialPadrao
 }
 }
}

$janelaPrincipal.src = paginaInicialPadrao
```

```
$inputEndereco.value = paginaInicialPadrao
 setPaginaInicial(paginaInicialPadrao)
}
}
```

3. No arquivo **aceitouSalvar.js** na pasta **scripts/storage** faça as seguintes alterações:

```
scripts/storage/aceitouSalvar.js

-export default JSON.parse(localStorage.getItem('aceitouSalvar'))
+export let aceitouSalvar = JSON.parse(localStorage.getItem('aceitouSalvar'))

-export function setAceitouSalvar(aceitouSalvar) {
- localStorage.setItem('aceitouSalvar', aceitouSalvar)
+export function setAceitouSalvar(valor) {
+ aceitouSalvar = valor
+ localStorage.setItem('aceitouSalvar', valor)
}
```

4. No arquivo **paginaInicial.js** na pasta **scripts/storage** faça as seguintes alterações:

```
scripts/storage/paginaInicial.js

-export default localStorage.getItem('paginaInicial')
+export let paginaInicial = localStorage.getItem('paginaInicial')

export function setpaginaInicial(valor) {
+ paginaInicial = valor
 localStorage.setItem('paginaInicial', valor)
}
```

# EXERCÍCIO: PÁGINA DE CONFIGURAÇÃO – SALVANDO INFORMAÇÕES

## 16.1 OBJETIVO

Nesse código, implementaremos a lógica do botão de salvar as configurações na página de configurações.

Para adicionar comportamento ao botão de salvar criaremos uma função chamada `salvar`. Nós criamos essa função para guardar e poder apontar para o pedaço de código que deve ser executado quando o evento de click acontecer no botão. Não seremos nós que executaremos essa função, mas sim o navegador, por isso passamos apenas o nome dela na propriedade `onclick` do botão:

```
$botaoSalvar.onclick = salvar
```

Chamamos esse tipo de função de função de callback.

Como passaremos a importar todas as propriedades dos dois módulos de storage. Criaremos um módulo agregador `/scripts/storage/index.js` que diferentemente dos outros módulos agregadores que fizemos, vai exportar todas as propriedades dos módulos na pasta `/scripts/storage/`.

## 16.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `paginaConfiguracao.js` na pasta `scripts` faça as seguintes alterações:

```
scripts/paginaConfiguracao.js

-import paginaInicial from './scripts/storage/paginaInicial.js'
-import aceitouSalvar from './scripts/storage/aceitouSalvar.js'
+import * as storage from './scripts/storage/storage.js'

-$inputPaginaInicial.value = paginaInicial
-$inputPermitiuSalvar.checked = aceitouSalvar
+$inputPaginaInicial.value = storage.paginaInicial
+$inputPermitiuSalvar.checked = storage.aceitouSalvar
+
+$botaoSalvar.onclick = salvar
+
+function salvar() {
+ storage.setAceitouSalvar($inputPermitiuSalvar.checked)
+ storage.setPaginaInicial($inputPaginaInicial.value)
+}
```

2. Crie o arquivo **storage.js** na pasta **scripts/storage** com o seguinte código:

```
scripts/storage/storage.js

+export * from '/scripts/storage/aceitouSalvar.js'
+export * from '/scripts/storage/paginaInicial.js'
```

3. No arquivo **config.html** na pasta **raiz do projeto** faça as seguintes alterações:

```
config.html

<link rel="stylesheet" href="styles/cake-config.css">

<h1>Configurações</h1>

<label>
 Página inicial:
 <input type="text" id="$inputPaginaInicial">
</label>

<label>
 Permissão de compartilhamento:
 <input type="checkbox" id="$inputPermitiuSalvar">
</label>

-<button>
+<button id="$botaoSalvar">
 Salvar configurações
</button>
Voltar

<script type="module" src="scripts/paginaConfiguracao.js"></script>
```

# EXERCÍCIO: PROTEGENDO O STORAGE – CLOSURES PARA ENCAPSULAMENTO E FUNÇÕES COMO VALORES

## 17.1 OBJETIVO

Nos exercícios anteriores, vimos como funções podem permitir o acesso à variáveis encapsuladas dentro dos nossos módulos.

Porém, criamos uma função `setAceitouSalvar` que recebe qualquer tipo de valor e atribui isso à variável `aceitouSalvar`. Essa variável deveria ser `null` ou `boolean`, sempre, e não deveríamos colocar qualquer valor nela! Podemos usar as *closures* para limitar melhor o que pode ser feito com essa nossa variável. Ao invés de exportar uma função que aceita qualquer valor, exportaremos 2 funções que ou colocam o valor como `true`, ou como `false`.

Os códigos que forem alterar nossa variável devem antes escolher qual das funções devem ser chamadas para isso.

Alteraremos nosso código anterior para isso! Note que nós escolheremos a função que deve ser executada usando um `if` ternário, salvando a função escolhida numa variável:

```
const setAceitouOuNao = $inputPermitiuSalvar.checked === true
 ? storage.setSimAceitouSalvar
 : storage.setNaoAceitouSalvar
```

Salvar uma função numa variável é possível pois no JavaScript, funções são um tipo de dado/variável.

Isso significa que podemos declarar e criar nossas funções de duas maneiras no JavaScript.

Como uma ***Function expression***:

```
const nomeDaFuncao = function () {
```

}

Como uma ***Function declaration***

```
function nomeDaFuncao() {
```

}

Lembrando que as ***Function expressions*** estão sujeitas aos mesmos comportamentos de *hoisting* das variáveis, ou seja: caso tenha declarado com `const` e `let`, só poderá usar a variável após a declaração dela; já no caso do `var`, é possível usar a variável antes do ponto onde declaramos ela, porém ela terá valor `undefined` no código enquanto não for feita a atribuição.

## 17.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `paginaConfiguracao.js` na pasta `scripts` faça as seguintes alterações:

```
scripts/paginaConfiguracao.js

import * as storage from '/scripts/storage/storage.js'

$inputPaginaInicial.value = storage.paginaInicial
$inputPermitiuSalvar.checked = storage.aceitouSalvar

$botaoSalvar.onclick = salvar

function salvar() {
- storage.setAceitouSalvar($inputPermitiuSalvar.checked)
+ const setAceitouOuNao =
+ $inputPermitiuSalvar.checked === true
+ ? storage.setSimAceitouSalvar
+ : storage.setNaoAceitouSalvar
+
+ setAceitouOuNao()
+
+ storage.setPaginaInicial($inputPaginaInicial.value)
}
```

2. No arquivo `pedeAceitouSalvar.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedeAceitouSalvar.js

import * as storage from '/scripts/storage/aceitouSalvar.js'

if (storage.aceitouSalvar === null) {
 const aceitou = confirm('Você aceita que a gente salve suas informações?')

 if (!aceitou) {
 alert('Você pode mudar isso na página de configurações')
 }

- storage.setAceitouSalvar(aceitou)
+ const setAceitouOuNao =
+ aceitou === true ? storage.setSimAceitouSalvar : storage.setNaoAceitouSalvar
+
+ setAceitouOuNao()
}
```

3. No arquivo `aceitouSalvar.js` na pasta `scripts/storage` faça as seguintes alterações:

```
scripts/storage/aceitouSalvar.js
```

```
export let aceitouSalvar = JSON.parse(localStorage.getItem('aceitouSalvar'))

-export function setAceitouSalvar(valor) {
+function setAceitouSalvar(valor) {
 aceitouSalvar = valor
 localStorage.setItem('aceitouSalvar', valor)
}
+
+export function setSimAceitouSalvar() {
+ setAceitouSalvar(true)
}
+
+export function setNaoAceitouSalvar() {
+ setAceitouSalvar(false)
}
```

# EXERCÍCIO: UM NOVO MÓDULO PARA TRATAR ENDEREÇOS

## 18.1 OBJETIVO

Transformar um endereço em um endereço completo iniciado em 'http://' é algo que precisaremos fazer a todo momento.

Inclusive, já estamos precisando fazer isso na página de configuração e no momento que a pessoa escolhe sua página inicial.

Para evitar duplicação de código, vamos criar um módulo que exporta uma função `formataEndereco` que faz exatamente o que o seu nome diz.

Nosso módulo exportará apenas uma função por enquanto, porém, seguiremos o padrão de exportar apenas propriedades nomeadas e não um `export default`. Assim, quem importar nosso módulo não será obrigado a escolher um nome qualquer para a função e por padrão usará o nome que já demos a ela:

```
import { formataEndereco } from '/scripts/endereco/formataEndereco.js'
```

Lembrando que damos o nome de *destructuring* para essa sintaxe `import { formataEndereco }`, que acessa e escolhe importar apenas algumas propriedades de um módulo.

## 18.2 PASSO A PASSO COM CÓDIGO

- Crie o arquivo `formataEndereco.js` na pasta `scripts/endereco` com o seguinte código:

```
scripts/endereco/formataEndereco.js

+function formataEndereco(enderecoPraFormatar) {
+ if (
+ enderecoPraFormatar.substring(0, 7) !== 'http://' &&
+ enderecoPraFormatar.substring(0, 8) !== 'https://'
+) {
+ // Assignment Atribuição
+ enderecoPraFormatar = 'http://' + enderecoPraFormatar
+ }
+
+ return enderecoPraFormatar
+}
+
+export { formataEndereco }
```

2. No arquivo `paginaConfiguracao.js` na pasta `scripts` faça as seguintes alterações:

```
scripts/paginaConfiguracao.js

import * as storage from '/scripts/storage/storage.js'
+import { formataEndereco } from '/scripts/endereco/formataEndereco.js'

$inputPaginaInicial.value = storage.paginaInicial
$inputPermitiuSalvar.checked = storage.aceitouSalvar

$botaoSalvar.onclick = salvar

function salvar() {
 const setAceitouOuNao =
 $inputPermitiuSalvar.checked === true
 ? storage.setSimAceitouSalvar
 : storage.setNaoAceitouSalvar

 setAceitouOuNao()

- storage.setPaginaInicial($inputPaginaInicial.value)
+ const enderecoCompleto = formataEndereco($inputPaginaInicial.value)
+ $inputPaginaInicial.value = enderecoCompleto
+
+ storage.setPaginaInicial(enderecoCompleto)
}
```

3. No arquivo `pedePaginaInicial.js` na pasta `scripts/pedeInfosIniciais` faça as seguintes alterações:

```
scripts/pedeInfosIniciais/pedePaginaInicial.js

import { aceitouSalvar as storageAceitouSalvar } from '/scripts/storage/aceitouSalvar.js'
import {
 paginaInicial,
 setPaginaInicial
} from '/scripts/storage/paginaInicial.js'
+import { formataEndereco } from '/scripts/endereco/formataEndereco.js'

if (storageAceitouSalvar === true) {
 let paginaInicialPadrao = paginaInicial

 if (!paginaInicialPadrao) {
 paginaInicialPadrao = prompt('Escolha a página inicial')
 }

 if (paginaInicialPadrao) {
- if (
- paginaInicialPadrao.substring(0, 7) !== 'http://' &&
- paginaInicialPadrao.substring(0, 8) !== 'https://'
-) {
- paginaInicialPadrao = 'http://' + paginaInicialPadrao
- }
+ const enderecoCompleto = formataEndereco(paginaInicialPadrao)

- $janelaPrincipal.src = paginaInicialPadrao
- $inputEndereco.value = paginaInicialPadrao
+ $janelaPrincipal.src = enderecoCompleto
+ $inputEndereco.value = enderecoCompleto
```

```
- setPaginaInicial(paginaInicialPadrao)
+ setPaginaInicial(enderecoCompleto)
}
}
```

# EXERCÍCIO: INCIANDO A NAVEGAÇÃO – EVENTOS DO IFRAME PARA ATUALIZAR A BARRA DE ENDEREÇOS

## 19.1 OBJETIVO

Nossa barra de endereços fica estática enquanto a pessoa navega usando o cake. Para que a barra de endereço seja atualizada com o endereço da página atual, criaremos uma função de `callback` que será chamada pelo navegador sempre que o evento de `load` for disparado no `<iframe>`.

Funções de `callback` são aquelas funções que criamos, mas não executamos. Quem executará elas será o próprio navegador, dado algum evento que ocorra.

O conceito de evento de `load` numa tag `<iframe>` está estritamente relacionado a JavaScript que roda em navegadores. Em qualquer outra plataforma que rode JavaScript, mas que não esteja relacionada com navegadores (como o `Node.js`) esse tipo de evento não existe. Por isso, é importante reforçar que o nome dos eventos e onde eles serão disparados não fazem parte do que chamamos de JavaScript e dependem totalmente da plataforma na qual estamos desenvolvendo.

O mais importante é notar que a ideia de eventos e funções de `callback` que executam quando esses eventos ocorrem sempre vai existir. Isso sim faz parte do JavaScript. Seja um evento de `load` num `<iframe>` do navegador, ou seja um evento `data` num `httpserver` do `Node.js`.

## 19.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `main.js` na pasta `scripts` faça as seguintes alterações:

```
scripts/main.js

import '/scripts/pedeInfosIniciais/index.js'
+import '/scripts/navegacao/barraEndereco.js'
```

2. Crie o arquivo `barraEndereco.js` na pasta `scripts/navegacao` com o seguinte código:

```
scripts/navegacao/barraEndereco.js

+$janelaPrincipal.onload = exibeEndereco
```

```
+
+function exibeEndereco(){
+ $inputEndereco.value = $janelaPrincipal.contentWindow.location.href
+}
```

# EXERCÍCIO: INCIANDO A NAVEGAÇÃO: MAIS EVENTOS PARA MELHORAR A BARRA DE ENDEREÇOS

## 20.1 OBJETIVO

Enquanto navegamos usando o cake, a barra de endereços sempre está com o endereço completo da página atual. Em buscas do google, a url fica gigantesca. Agora que conhecemos melhor a ideia de eventos e callbacks, podemos melhorar isso e replicar o comportamento da barra de endereços de outros navegadores.

Mostraremos apenas um endereço resumido enquanto a pessoa navega e quando a barra de endereço é focada, mostramos o endereço completo para que possa ser copiado, por exemplo.

Para resumir o endereço, criaremos um objeto URL que contém cada parte da url separada como uma propriedade:

```
{
 protocol: 'http://',
 hostname: 'google.com',
 search: '/?q=teste&prop=234tghjkl'
}
```

Esse objeto será construído ao chamarmos a função construtora `URL`. Essa função já existe e está presente em quase todas as plataformas que rodam JavaScript. Para criar esse objeto, basta chamarmos a função, porém, antecedido de um novo operador, o `new`:

```
const url = new URL($janelaPrincipal.contentWindow.location.href)
```

## 20.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `barraEndereco.js` na pasta `scripts/navegacao` faça as seguintes alterações:

```
scripts/navegacao/barraEndereco.js

+$inputEndereco.onfocus = exibeEnderecoCompleto

+$inputEndereco.onblur = exibeEnderecoResumido
-$janelaPrincipal.onload = exibeEndereco
+$janelaPrincipal.onload = exibeEnderecoResumido
```

```
+
-function exibeEndereco(){
+function exibeEnderecoCompleto(){
 $inputEndereco.value = $janelaPrincipal.contentWindow.location.href
}
+
+function exibeEnderecoResumido() {
+ const url = new URL($janelaPrincipal.contentWindow.location.href)
+ const enderecoResumido = url.hostname
+
+ $inputEndereco.value = enderecoResumido
+}
```

# EXERCÍCIO: ARRUMANDO O RELOAD, RECARREGANDO A PÁGINA ATUAL - OS EVENT EMITTERS

## 21.1 OBJETIVO

No momento, quando apertamos o botão de recarregar voltamos para a página inicial. Para decidir qual página será carregada quando o navegador inicia ou é recarregado, precisaremos ficar atentos aos eventos de 'load' que o `<iframe>` emite enquanto navegamos e salvar a página atual temporariamente. Como a página atual só deve ser armazenada temporariamente enquanto o cake está aberto, utilizaremos um novo tipo de storage que é apagado automaticamente quando o Cake é fechado: o `sessionStorage`.

Quanto ao evento `load`, já estamos ouvindo ele lá no código da barra de endereço. Porém, esse código que mantém a página atual no recarregamento não é relacionado e não deve ficar junto com o código que exibe o endereço resumido na barra. Para permitir essa separação passaremos a cadastrar múltiplos *Event Listeners* no nosso *Event Emmitter*, o `<iframe>`.

Isso significa que ao invés de mudar uma propriedade num objeto como antes, executaremos um método que recebe como parâmetro o nome do evento e a função de callback a ser executada. No caso dos *EventEmitters* do navegador, o nome desse método é `addEventListener`.

## 21.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `main.js` na pasta `scripts` faça as seguintes alterações:

```
scripts/main.js

import '/scripts/pedeInfosIniciais/index.js'
import '/scripts/navegacao/barraEndereco.js'
+import '/scripts/navegacao/paginaAtual.js'
```

2. No arquivo `barraEndereco.js` na pasta `scripts/navegacao` faça as seguintes alterações:

```
scripts/navegacao/barraEndereco.js

-$inputEndereco.onfocus = exibeEnderecoCompleto
```

```

-$inputEndereco.onblur = exibeEnderecoResumido
-$janelaPrincipal.onload = exibeEnderecoResumido
+$inputEndereco.addEventListener('focus', exibeEnderecoCompleto)
+
+$inputEndereco.addEventListener('blur', exibeEnderecoResumido)
+$janelaPrincipal.addEventListener('load', exibeEnderecoResumido)

function exibeEnderecoCompleto(){
 $inputEndereco.value = $janelaPrincipal.contentWindow.location.href
}

function exibeEnderecoResumido() {
 const url = new URL($janelaPrincipal.contentWindow.location.href)
 const enderecoResumido = url.hostname

 $inputEndereco.value = enderecoResumido
}

```

3. Crie o arquivo **paginaAtual.js** na pasta **scripts/navegacao** com o seguinte código:

```

scripts/navegacao/paginaAtual.js

+import * as storagePaginaInicial from '/scripts/storage/paginaInicial.js'
+import { formataEndereco } from '/scripts/endereco/formataEndereco.js'
+
+const paginaAtual = sessionStorage.getItem('paginaAtual')
+
+const paginaPraCarregar = paginaAtual !== null
+ ? paginaAtual
+ : storagePaginaInicial.paginaInicial
+
+const enderecoCompleto = formataEndereco(paginaPraCarregar)
+$janelaPrincipal.src = enderecoCompleto
+$inputEndereco.value = enderecoCompleto
+
+$janelaPrincipal.addEventListener('load', salvaPaginaAtual)
+
+function salvaPaginaAtual(){
+ const endereco = $janelaPrincipal.contentWindow.location.href
+ sessionStorage.setItem('paginaAtual', endereco)
+}

```

4. No arquivo **pedePaginaInicial.js** na pasta **scripts/pedeInfosIniciais** faça as seguintes alterações:

```

scripts/pedeInfosIniciais/pedePaginaInicial.js

import { aceitouSalvar as storageAceitouSalvar } from '/scripts/storage/aceitouSalvar.js'
import { paginaInicial, setPaginaInicial } from '/scripts/storage/paginaInicial.js'

import { formataEndereco } from '/scripts/endereco/formataEndereco.js'

if(storageAceitouSalvar === null || storageAceitouSalvar === true){
 let paginaInicialPadrao = paginaInicial

 if(!paginaInicialPadrao) {
 paginaInicialPadrao = prompt("Escolha a página inicial")
 }
}
```

```
if(paginaInicialPadrao) {
 const enderecoCompleto = formataEndereco(paginaInicialPadrao)
 - $janelaPrincipal.src = enderecoCompleto
 - $inputEndereco.value = enderecoCompleto
 - setPaginaInicial(enderecoCompleto)
}
}
```

# EXERCÍCIO: BOTÃO HOME E SEPARAÇÃO DE RESPONSABILIDADES COM A FUNÇÃO CARREGAR

## 22.1 OBJETIVO

Adicionaremos um novo botão que quando clicado nos redirecionará para a página inicial.

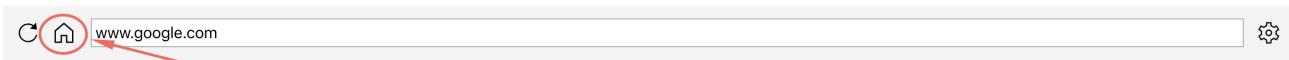


Figura 22.1: Novo botão para voltar à página inicial

Para evitar duplicação do código que altera a página carregada na janela principal, criaremos uma novo módulo que exporta uma função `carregar`, responsável por isso. Para organizar melhor o código, criaremos também um módulo agregador em `/scripts/navegacao`.

Agora que estamos lidando com *Event Listeners*, vale lembrar também que as funções de callback nem sempre serão reaproveitadas em algum outro lugar. Muitas vezes só criamos essas funções para adicionar *Event Listeners* e nesses casos, elas podem ser pasadas direto como parâmetro do `addEventListener`:

```
$botaoHome.addEventListener('click', function vaiParaHome() {
 carregar(paginaInicial)
})
```

## 22.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `index.html` na pasta **raiz do projeto** faça as seguintes alterações:

```
index.html

<meta charset="utf-8">

<link rel="stylesheet" href="styles/cake-2afdf04e92.css"/>

<header>
 <input type="image" src="images/libCake/recarregar.svg" onclick="window.location.reload()">
+ <input type="image" src="images/libCake/casa.svg" id="$botaoHome">
+
```

```

<input type="text" id="$inputEndereco">

</header>

<iframe frameborder="0" id="$janelaPrincipal"></iframe>

<script src="scripts/cake-8709f3abc4.js"></script>

<script type="module" src="scripts/main.js"></script>

```

2. No arquivo **main.js** na pasta **scripts** faça as seguintes alterações:

```
scripts/main.js
```

```

import '/scripts/pedeInfosIniciais/index.js'
- import '/scripts/navegacao/barraEndereco.js'
- import '/scripts/navegacao/paginaAtual.js'
+ import '/scripts/navegacao/index.js'

```

3. Crie o arquivo **botaoHome.js** na pasta **scripts/navegacao** com o seguinte código:

```
scripts/navegacao/botaoHome.js
```

```

+import { paginaInicial } from '/scripts/storage/paginaInicial.js'
+import { carregar } from '/scripts/navegacao/carregar.js'
+
+$botaoHome.addEventListener('click', function vaiParaHome() {
+ carregar(paginaInicial)
+})

```

4. Crie o arquivo **carregar.js** na pasta **scripts/navegacao** com o seguinte código:

```
scripts/navegacao/carregar.js
```

```

+export function carregar(enderecoCompleto) {
+ $janelaPrincipal.src = enderecoCompleto
+ $inputEndereco.value = enderecoCompleto
+}

```

5. Crie o arquivo **index.js** na pasta **scripts/navegacao** com o seguinte código:

```
scripts/navegacao/index.js
```

```

+import '/scripts/navegacao/barraEndereco.js'
+import '/scripts/navegacao/paginaAtual.js'
+import '/scripts/navegacao/botaoHome.js'

```

6. No arquivo **paginaAtual.js** na pasta **scripts/navegacao** faça as seguintes alterações:

```
scripts/navegacao/paginaAtual.js
```

```

import * as storagePaginaInicial from '/scripts/storage/paginaInicial.js'
import { formataEndereco } from '/scripts/endereco/formataEndereco.js'
+import { carregar } from '/scripts/navegacao/carregar.js'

```

```
const paginaAtual = sessionStorage.getItem('paginaAtual')

const paginaPraCarregar = paginaAtual !== null
? paginaAtual
: storagePaginaInicial.paginaInicial

const enderecoCompleto = formataEndereco(paginaPraCarregar)
-$janelaPrincipal.src = enderecoCompleto
-$inputEndereco.value = enderecoCompleto
+carregar(enderecoCompleto)

-$janelaPrincipal.addEventListener('load', salvaPaginaAtual)
-
-function salvaPaginaAtual(){
+$janelaPrincipal.addEventListener('load', function salvaPaginaAtual(){
 const endereco = $janelaPrincipal.contentWindow.location.href
 sessionStorage.setItem('paginaAtual', endereco)
})
})
```

# EXERCÍCIO: DIGITANDO O ENDEREÇO NA BARRA DE ENDEREÇOS – EVENT EMITTERS E INFORMAÇÕES SOBRE O EVENTO

## 23.1 OBJETIVO

Nesse exercício, implementaremos a navegação pela barra de endereços.

No navegador, todo `<input>` é um *Event Emitter* que emite, entre muitos eventos, o evento de `keyup` quando está selecionado e alguém aperta e solta uma tecla do teclado.

Porém, dessa vez não basta saber o nome do evento, já que precisamos de informações sobre a tecla que foi pressionada. Só podemos carregar a página quando a tecla apertada for `Enter`. Para isso, podemos contar com uma característica dos *Event Emitters*: as informações que só o *event emitter* tem como saber – e que você precisa acessar de alguma maneira - geralmente são passadas para você como parâmetro da função de callback. No caso do evento `keyup`, o primeiro parâmetro do callback é um objeto que representa o evento e contém uma propriedade chamada `key` com a informação que precisamos.

## 23.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `barraEndereco.js` na pasta `scripts/navegacao` faça as seguintes alterações:

```
scripts/navegacao/barraEndereco.js

+import { carregar } from '/scripts/navegacao/carregar.js'
+import { formataEndereco } from '/scripts/endereco/formataEndereco.js'

$inputEndereco.addEventListener('focus', exibeEnderecoCompleto)

$inputEndereco.addEventListener('blur', exibeEnderecoResumido)
$janelaPrincipal.addEventListener('load', exibeEnderecoResumido)

function exibeEnderecoCompleto(){
 $inputEndereco.value = $janelaPrincipal.contentWindow.location.href
}
```

```
function exibeEnderecoResumido() {
 const url = new URL($janelaPrincipal.contentWindow.location.href)
 const enderecoResumido = url.hostname

 $inputEndereco.value = enderecoResumido
}

+
// Todo parâmetro é opcional
+$inputEndereco.addEventListener('keyup', function(evento) {
+ // o navegador executou o callback
+ // as informações
+ const apertouEnter = evento.key === 'Enter'
+ if(apertouEnter) {
+ const enderecoCompleto = formataEndereco($inputEndereco.value)
+ carregar(enderecoCompleto)
+ }
+})
```

# EXERCÍCIO: BOTÃO LIMPA TUDO - OBJECT.KEYS E ITERAÇÃO EM LISTAS

## 24.1 OBJETIVO

Antes de seguir incrementando nossa navegação armazenando o histórico e tudo mais. Seria legal testarmos a aplicação completa e ver se tudo está funcionando. Para isso, temos que apagar todo o storage tanto `local` como `session`.

Pensando nisso, nesse exercício vamos iniciar a implementação de um botão que apaga as informações armazenadas nos *storages*. Por enquanto apagaremos todas as informações salvas, mas num próximo exercício evoluíremos o código para manter algumas informações como `aceitouSalvar` e `aceitouOsTermos`, que não devem ser apagadas.

## 24.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `config.html` na pasta `raiz do projeto` faça as seguintes alterações:

```
config.html

<link rel="stylesheet" href="styles/cake-config-33d6f649ee.css"/>

<h1>Configurações</h1>

<label>
 Página inicial:
 <input type="text" id="$inputPaginaInicial">
</label>

<label>
 Permissão para armazenamento:
 <input type="checkbox" id="$inputPermitiuSalvar">
</label>

<button id="$botaoSalvar">
 Salvar configurações
</button>
+<button id="$botaoLimpaTudo">
+ Limpar tudo
+</button>
Voltar

<script type="module" src="scripts/paginaConfiguracao.js"></script>
```

2. No arquivo `paginaConfiguracao.js` na pasta `scripts` faça as seguintes alterações:

```
scripts/paginaConfiguracao.js

import * as storage from '/scripts/storage/storage.js'
import { formataEndereco } from '/scripts/endereco/formataEndereco.js'

$inputPaginaInicial.value = storage.paginaInicial
$inputPermitiuSalvar.checked = storage.aceitouSalvar

$botaoSalvar.onclick = salvar

function salvar() {
 const setAceitouOuNao =
 $inputPermitiuSalvar.checked === true
 ? storage.setSimAceitouSalvar
 : storage.setNaoAceitouSalvar

 setAceitouOuNao()

 const enderecoCompleto = formataEndereco($inputPaginaInicial.value)
 $inputPaginaInicial.value = enderecoCompleto

 storage.setPaginaInicial(enderecoCompleto)
}

+$botaoLimpaTudo.addEventListener('click', function () {
+ const listaChavesLocalStorage = Object.keys(localStorage)
+ for(let i = 0; i < listaChavesLocalStorage.length; i++) {
+ const chave = listaChavesLocalStorage[i]
+ localStorage.removeItem(chave)
+ }
+
+ const listaChavesSessionStorage = Object.keys(sessionStorage)
+ for(let chave of listaChavesSessionStorage) {
+ sessionStorage.removeItem(chave)
+ }
+})
```

# EXERCÍCIO: BOTÃO LIMPA TUDO – MANIPULAÇÃO E ACESSO A LISTAS COM FUNÇÕES DE ARRAY.PROTOTYPE

## 25.1 OBJETIVO

Agora que já conseguimos apagar todas as informações dos *storages*, podemos filtrar apenas as informações que queremos remover.

No nosso caso, não temos como saber quais serão todas as informações que podemos remover, já que muitas informações podem ser adicionadas no futuro. Porém, sabemos quais informações não queremos apagar: `aceitouSalvar` e `aceitouOsTermos`. Com isso, podemos fazer uma lista de propriedades permanentes e nos basear nela para decidir se removemos ou não uma certa informação.

Para enxugar o código, lembre de usar os métodos "escondidos" em todo `Array` no JavaScript.

## 25.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `paginaConfiguracao.js` na pasta `scripts` faça as seguintes alterações:

```
scripts/paginaConfiguracao.js

import * as storage from '/scripts/storage/storage.js'
import { formataEndereco } from '/scripts/endereco/formataEndereco.js'

$inputPaginaInicial.value = storage.paginaInicial
$inputPermitiuSalvar.checked = storage.aceitouSalvar

$botaoSalvar.onclick = salvar

function salvar() {
 const setAceitouOuNao =
 $inputPermitiuSalvar.checked === true
 ? storage.setSimAceitouSalvar
 : storage.setNaoAceitouSalvar

 setAceitouOuNao()

 const enderecoCompleto = formataEndereco($inputPaginaInicial.value)
 $inputPaginaInicial.value = enderecoCompleto

 storage.setPaginaInicial(enderecoCompleto)
```

```
}

$botaoLimpaTudo.addEventListener('click', function () {
+ const chavesPermanentes = [
+ 'aceitouSalvar',
+ 'aceitouTermos'
+]
+
+ const listaChavesLocalStorage = Object.keys(localStorage)
- for(let i = 0; i < listaChavesLocalStorage.length; i++) {
- const chave = listaChavesLocalStorage[i]
- localStorage.removeItem(chave)
-
- }
+ for(let chave of listaChavesLocalStorage) {
+ const isChavePermanente = chavesPermanentes.includes(chave)
+
+ if(!isChavePermanente){
+ localStorage.removeItem(chave)
+ }
+ }
+
const listaChavesSessionStorage = Object.keys(sessionStorage)
for(let chave of listaChavesSessionStorage) {
 sessionStorage.removeItem(chave)
}
+
+ window.location.reload()
})
```

# EXERCÍCIO: BOTÕES AVANÇAR E VOLTAR – LISTAS, CONSTANTES SÃO MUTÁVEIS E MAIS ENCAPSULAMENTO.

## 26.1 OBJETIVO

Para complementar a navegação precisamos dos botões de avançar e voltar no histórico.

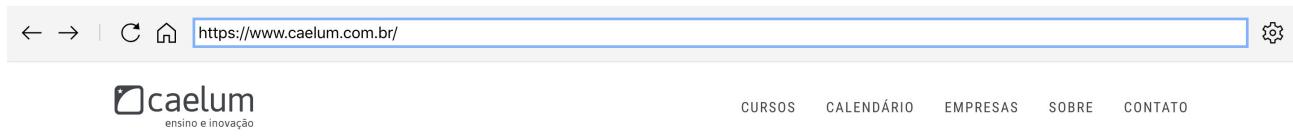


Figura 26.1: Botões avançar e voltar

Quando clicados, os botões devem obter o endereço anterior ou seguinte, de acordo com a ordem em que foram carregados. Criaremos uma lista que será atualizada a todo momento que uma página for carregada no `<iframe>`. Essa lista com os endereços do histórico não precisa nem deve ser acessada e manipulada diretamente; nem para acessarmos esses endereços, nem para adicionarmos um endereço quando alguma página nova é carregada.

Lembre que se a lista estiver exposta, mesmo sendo uma constante, ela pode ser manipulada, já que constantes não são imutáveis. No nosso caso vamos encapsular essa lista num módulo e permitir apenas alguns tipos de operações através desse módulo: `adicionarEndereco`, `voltar` e `avancar`.

## 26.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `index.html` na pasta **raiz do projeto** faça as seguintes alterações:

```
index.html

<meta charset="utf-8">

<link rel="stylesheet" href="styles/cake.css"/>

<header>
+ <input type="image" src="images/libCake/seta-voltar.svg" id="$botaoVoltar">
+ <input type="image" src="images/libCake/seta-avancar.svg" id="$botaoAvancar">
+

```

```

<input type="image" src="images/libCake/recarregar.svg" onclick="window.location.reload()">
<input type="image" src="images/libCake/casa.svg" id="$botaoHome">

<input type="text" id="$inputEndereco">

</header>

<iframe frameborder="0" id="$janelaPrincipal"></iframe>

<script src="scripts/cake-8709f3abc4.js"></script>

<script type="module" src="scripts/main.js"></script>

```

2. Crie o arquivo **historico.js** na pasta **scripts/navegacao** com o seguinte código:

```
scripts/navegacao/historico.js
```

```

+const listaSites = JSON.parse(sessionStorage.getItem('historico')) || []
+let posicao = JSON.parse(sessionStorage.getItem('posicaoHistorico')) || -1
+
+export function adiciona(endereco) {
+ if(endereco !== listaSites[posicao]) {
+ listaSites.splice(posicao + 1)
+ listaSites.push(endereco)
+ posicao++
+ sessionStorage.setItem('historico', JSON.stringify(listaSites))
+ sessionStorage.setItem('posicaoHistorico', posicao)
+ }
+}
+
+export function volta() {
+ const isPrimeiraPosicao = posicao === 0
+ if(listaSites.length !== 1 && !isPrimeiraPosicao) {
+ posicao = posicao - 1
+ sessionStorage.setItem('posicaoHistorico', posicao)
+ return listaSites[posicao]
+ }
+}
+
+export function avanca() {
+ const isUltimaPosicao = posicao === listaSites.length - 1
+ if(listaSites.length !== 1 && !isUltimaPosicao) {
+ posicao = posicao + 1
+ sessionStorage.setItem('posicaoHistorico', posicao)
+ return listaSites[posicao]
+ }
+}

```

3. Crie o arquivo **botaoAvancarVoltar.js** na pasta **scripts/navegacao** com o seguinte código:

```
scripts/navegacao/botaoAvancarVoltar.js
```

```

+import * as historico from '/scripts/navegacao/historico.js'
+
+$janelaPrincipal.addEventListener('load', function salvaPaginaAtual(){
+ const endereco = $janelaPrincipal.contentWindow.location.href
+ historico.adiciona(endereco)
+})
+
```

```
+$botaoVoltar.addEventListener('click', function() {
+ const enderecoVolta = historico.volta()
+ if(enderecoVolta !== undefined)
+ carregar(enderecoVolta)
+})
+
+$botaoAvancar.addEventListener('click', function() {
+ const enderecoAvanca = historico.avanca()
+ if(enderecoAvanca !== undefined)
+ carregar(enderecoAvanca)
+})
```

4. No arquivo `index.js` na pasta `scripts/navegacao` faça as seguintes alterações:

```
scripts/navegacao/index.js
```

```
import '/scripts/navegacao/barraEndereco.js'
import '/scripts/navegacao/paginaAtual.js'
import '/scripts/navegacao/botaoHome.js'
+import '/scripts/navegacao/historico.js'
+import '/scripts/navegacao/botaoAvancarVoltar.js'
```

# EXERCÍCIO: EXECUTANDO MENOS LÓGICA NOS EMITTERS: OBJETOS LITERAIS E FUNÇÕES FACTORY

## 27.1 OBJETIVO

Nesse exercício, diminuiremos a quantidade e a frequência de código escrito e executado para obter o **endereço resumido** e o **endereço completo** exibidos na barra de endereço.

A todo momento que focamos e desfocamos a barra de endereço, processamos o endereço atual da janela principal para obter o **endereço resumido** e o **endereço completo**.

Já que queremos em vários momentos diferentes essas duas informações sobre o endereço atual, podemos processar essas informações apenas uma vez e armazenar em variáveis.

Poderíamos criar duas variáveis, uma para cada informação, porém, se quisermos separar nossa lógica de negócio dos *Event Emitters*, provavelmente vamos acabar criando duas funções, uma para cada informação. Com duas funções, o código para obter tanto o **endereço resumido** quanto o **endereço completo** é muito parecido e parte dele seria duplicado. Para reaproveitar o código de criação do **endereço resumido** e do **endereço completo**, podemos ter apenas uma função que retorna as duas informações para a gente.

Com a possibilidade de apenas um valor sendo retornado por essa função, esse valor descreveria nosso endereço como a combinação de dois valores, a **urlResumida** e sua **urlCompleta**. Um tipo que serve exatamente para isso no JavaScript é o objeto:

```
{
 urlResumida: 'google.com',
 urlCompleta: 'http://google.com/?q=teste'
}
```

Uma função que retorna um objeto como esse é o que chamamos de uma **Função Factory**

## 27.2 PASSO A PASSO COM CÓDIGO

1. Crie o arquivo **criaEndereco.js** na pasta **scripts/endereco** com o seguinte código:

```
scripts/endereco/criaEndereco.js
```

```
+function criaEndereco(endereco) {
+ const url = new URL(endereco)
+
+ let enderecoCompleto
+ let enderecoResumido
+
+ if(url.hostname === 'localhost') {
+ const paginaLocal = url.pathname.replace("/", "")
+ enderecoCompleto = paginaLocal
+ enderecoResumido = paginaLocal
+ } else {
+ enderecoCompleto = url.toString()
+ enderecoResumido = url.hostname
+ }
+
+ return {
+ urlCompleta: enderecoCompleto,
+ urlResumida: enderecoResumido
+ }
+}
+
+export { criaEndereco }
```

2. No arquivo **formataEndereco.js** na pasta **scripts/endereco** faça as seguintes alterações:

```
scripts/endereco/formataEndereco.js
```

```
function formataEndereco(enderecoPraFormatar) {
+ if(!enderecoPraFormatar || enderecoPraFormatar ==='blank') {
+ return 'blank'
+ }
+
+ if (
+ enderecoPraFormatar.substring(0, 7) !== 'http://' &&
+ enderecoPraFormatar.substring(0,8) !== 'https://'
+) {
+ // Assignment Atribuição
+ enderecoPraFormatar = 'http://' + enderecoPraFormatar
+ }
+
+ return enderecoPraFormatar
+}

export { formataEndereco }
```

3. No arquivo **barraEndereco.js** na pasta **scripts/navegacao** faça as seguintes alterações:

```
scripts/navegacao/barraEndereco.js
```

```
import { carregar } from '/scripts/navegacao/carregar.js'
+
+import { formataEndereco } from '/scripts/endereco/formataEndereco.js'
+import { criaEndereco } from '/scripts/endereco/criaEndereco.js'
+
+
+let endereco
```

```

+$janelaPrincipal.addEventListener('load', function() {
+ endereco = criaEndereco(
+ $janelaPrincipal.contentWindow.location.href
+)
+})
+
+
$inputEndereco.addEventListener('focus', exibeEnderecoCompleto)

$inputEndereco.addEventListener('blur', exibeEnderecoResumido)
$janelaPrincipal.addEventListener('load', exibeEnderecoResumido)

-function exibeEnderecoCompleto(){
- $inputEndereco.value = $janelaPrincipal.contentWindow.location.href
-}
+
+function exibeEnderecoCompleto(){
+ $inputEndereco.value = endereco.urlCompleta
+}

function exibeEnderecoResumido() {
- const url = new URL($janelaPrincipal.contentWindow.location.href)
- const enderecoResumido = url.hostname
-
- $inputEndereco.value = enderecoResumido
+ $inputEndereco.value = endereco.urlResumida
}

$inputEndereco.addEventListener('keyup', function(evento) {
 const apertouEnter = evento.key === 'Enter'
 if(apertouEnter) {
 const enderecoCompleto = formataEndereco($inputEndereco.value)
 carregar(enderecoCompleto)
 }
})

```

# EXERCÍCIO: ATUALIZANDO OBJETO ENDEREÇO AO DIGITAR

## 28.1 OBJETIVO

É normal que um site demore um certo tempo para carregar. Nesses casos, quando digitamos o endereço e apertamos `Enter` na barra de endereço, caso a gente desfoque ou foque na barra de endereço enquanto a página está carregando, perderemos o endereço digitado. O endereço da barra de endereços só será atualizado quando a página for carregada.

Isso acontece porque nós não atualizamos o objeto endereço quando a pessoa aperta `Enter` após digitar o endereço na barra de endereços. Nesse exercício implementaremos esse comportamento. Lembrando que esse objeto de endereço é utilizado para definir o endereço atual que será exibido na barra.

Para resolver isso, nossa **função factory** que cria esse objeto passará a lidar com endereços vindos da pessoa que está usando o navegador. Por isso, precisamos tratar os casos de endereços em branco e sem o prefixo `http://`, como já fizemos anteriormente em outra função.

## 28.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `criaEndereco.js` na pasta `scripts/endereco` faça as seguintes alterações:

```
scripts/endereco/criaEndereco.js

function criaEndereco(endereco) {
- const url = new URL(endereco)
-
+ let enderecoCompleto
+ let enderecoResumido
+
+ if(!endereco || endereco === 'blank') {
+ enderecoCompleto = 'blank'
+ enderecoResumido = 'blank'
+ } else {
+ const url = new URL(endereco)

 if(url.hostname === 'localhost') {
 const paginaLocal = url.pathname.replace("/", "")
 enderecoCompleto = paginaLocal
 enderecoResumido = paginaLocal
 }
 }
}
```

```

 } else {
 enderecoCompleto = url.toString()
 enderecoResumido = url.hostname
 }
 }

 return {
 urlCompleta: enderecoCompleto,
 urlResumida: enderecoResumido
 }
}

export { criaEndereco }

```

2. No arquivo **barraEndereco.js** na pasta **scripts/navegacao** faça as seguintes alterações:

```

scripts/navegacao/barraEndereco.js

import { carregar } from '/scripts/navegacao/carregar.js'

import { formataEndereco } from '/scripts/endereco/formataEndereco.js'
import { criaEndereco } from '/scripts/endereco/criaEndereco.js'

let endereco

$janelaPrincipal.addEventListener('load', function() {
 endereco = new criaEndereco(
 $janelaPrincipal.contentWindow.location.href
)
})

$inputEndereco.addEventListener('focus', exibeEnderecoCompleto)

$inputEndereco.addEventListener('blur', exibeEnderecoResumido)
$janelaPrincipal.addEventListener('load', exibeEnderecoResumido)

function exibeEnderecoCompleto(){
 $inputEndereco.value = endereco.urlCompleta
}

function exibeEnderecoResumido() {
 $inputEndereco.value = endereco.urlResumida
}

// Todo parâmetro é opcional
$inputEndereco.addEventListener('keyup', function(evento) {
 // o navegador executou o callback
 // as informações
 const apertouEnter = evento.key === 'Enter'
 if(apertouEnter) {
 const enderecoCompleto = formataEndereco($inputEndereco.value)
 endereco = criaEndereco(enderecoCompleto)
 carregar(enderecoCompleto)
 }
})

```

# EXERCÍCIO: VALIDANDO TIPOS DE PARÂMETROS COM INSTANCEOF E TYPEOF

## 29.1 OBJETIVO

No momento, estamos executando duas funções que tratam o endereço digitado pela pessoa; uma para formatar a string do o endereço que será passado para a função `carregar` e outra **função factory** que é usada para criar o objeto endereço, que define o conteúdo exibido na barra de endereço. No momento as duas funções fazem praticamente o mesmo tratamento do endereço e para evitar a duplicação do código passaremos a usar apenas nossa **função factory**. Por enquanto alteraremos apenas a função `carregar`, passando a aceitar um objeto de endereço como argumento.

Para que essas alterações não impactem os códigos que usam a função `carregar`, precisamos que ela continue aceitando strings de endereço assim como nossos novos objetos de endereço. Para isso, precisaremos verificar o tipo dos argumentos recebidos para decidir como caregaremos o endereço.

## 29.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `carregar.js` na pasta `scripts/navegacao` faça as seguintes alterações:

```
scripts/navegacao/carregar.js

+import { criaEndereco } from "/scripts/endereco/criaEndereco.js";
+
+export function carregar(seiLah) {
- $janelaPrincipal.src = enderecoCompleto
- $inputEndereco.value = enderecoCompleto
+ let endereco
+
+ if(typeof seiLah === "string" || !seiLah) {
+ endereco = criaEndereco(seiLah)
+ } else if(seiLah instanceof criaEndereco) {
+ endereco = seiLah
+ } else {
+ throw new Error(`
+ Você passou um endereço que não é nem string nem de criaEndereco:
+ Valor: ${JSON.stringify(seiLah)}
+ Tipo: ${typeof seiLah}
+ `)
+ }
}
```

```
$janelaPrincipal.src = endereco.urlCompleta
$inputEndereco.value = endereco.urlCompleta
}
```

# EXERCÍCIO: CRIANDO O TIPO ENDEREÇO - FUNÇÕES CONSTRUTORAS

## 30.1 OBJETIVO

No exercício anterior passamos a utilizar os operadores `typeof` para verificar tipos primitivos e o `instanceof` para verificar o tipo de objetos criados por funções construtoras.

Porém, nossa **função factory** `criaEndereco` não é uma **função construtora** e a expressão `objeto instanceof criaEndereco` sempre resultará em `false` para objetos criados a partir dela. Nesse exercício transformaremos nossa função factory numa função construtora, inclusive a renomeando para `Endereco`.

Passaremos também a chamar apenas a função construtora na navegação pela barra de endereço. Removeremos o código que formata o que foi digitado para uma string de endereço válida e passaremos o objeto endereço como parâmetro para a função `carregar`.

## 30.2 PASSO A PASSO COM CÓDIGO

- Na pasta `scripts/endereco`, renomeie o arquivo `criaEndereco.js` para `Endereco.js`. Faça também as seguintes modificações no código:

```
scripts/endereco/Endereco.js

-function criaEndereco(endereco) {
+function Endereco(endereco) {
+
+ if(
+ !(this instanceof Endereco))
+) {
+ return new Endereco(endereco)
+ }
+
let enderecoCompleto
let enderecoResumido

if(!endereco || endereco ==='blank') {
 enderecoCompleto = 'blank'
 enderecoResumido = 'blank'
} else {
 const url = new URL(endereco)
```

```

 if(url.hostname === 'localhost') {
 const paginaLocal = url.pathname.replace("/", "")
 enderecoCompleto = paginaLocal
 enderecoResumido = paginaLocal
 } else {
 enderecoCompleto = url.toString()
 enderecoResumido = url.hostname
 }
 }
}

+
+
+ this.urlCompleta = enderecoCompleto
+ this.urlResumida = enderecoResumido

- return {
- urlCompleta: enderecoCompleto,
- urlResumida: enderecoResumido
- }
}

-export { criaEndereco }
+export { Endereco }

```

2. No arquivo **barraEndereco.js** na pasta **scripts/navegacao** faça as seguintes alterações:

```

scripts/navegacao/barraEndereco.js

import { carregar } from '/scripts/navegacao/carregar.js'
-
-
- import { formataEndereco } from '/scripts/endereco/formataEndereco.js'
- import { criaEndereco } from '/scripts/endereco/criaEndereco.js'
+import { Endereco } from '/scripts/endereco/Endereco.js'

let endereco

$janelaPrincipal.addEventListener('load', function() {
- endereco = new criaEndereco(
+ endereco = new Endereco(
- $janelaPrincipal.contentWindow.location.href
+ this.contentWindow.location.href
)
})

$inputEndereco.addEventListener('focus', exibeEnderecoCompleto)

$inputEndereco.addEventListener('blur', exibeEnderecoResumido)
$janelaPrincipal.addEventListener('load', exibeEnderecoResumido)

function exibeEnderecoCompleto(){
 $inputEndereco.value = endereco.urlCompleta
}

function exibeEnderecoResumido() {
 $inputEndereco.value = endereco.urlResumida
}

$inputEndereco.addEventListener('keyup', function(evento) {
 const apertouEnter = evento.key === 'Enter'
 if(apertouEnter) {
-
 const enderecoCompleto = formataEndereco($inputEndereco.value)
-
 endereco = criaEndereco(enderecoCompleto)
-
 carregar(enderecoCompleto)
 }
})

```

```

+ endereco = Endereco($inputEndereco.value)
+ carregar(endereco)
}
})

```

3. No arquivo **carregar.js** na pasta **scripts/navegacao** faça as seguintes alterações:

```
scripts/navegacao/carregar.js
```

```

-import { criaEndereco } from "/scripts/endereco/criaEndereco.js";
+import { Endereco } from "/scripts/endereco/Endereco.js";

export function carregar(seiLah) {
 let endereco

 if(typeof seiLah === "string" || !seiLah) {
- endereco = criaEndereco(seiLah)
+ endereco = new Endereco(seiLah)
- } else if(seiLah instanceof criaEndereco) {
+ } else if(seiLah instanceof Endereco) {
 endereco = seiLah
 } else {
 throw new Error(`

- Você passou um endereço que não é nem string nem de criaEndereco:
+ Você passou um endereço que não é nem string nem Endereco:
 Valor: ${JSON.stringify(seiLah)}
 Tipo: ${typeof seiLah}
 `)
 }

 $janelaPrincipal.src = endereco.urlCompleta
 $inputEndereco.value = endereco.urlCompleta
}

```

# EXERCÍCIO: TRATAMENTO DE ERROS COM ERROS CUSTOMIZADOS - MAIS FUNÇÕES CONSTRUTORAS

## 31.1 OBJETIVO

Ao digitarmos endereços inválidos, nossa função construtora está disparando um erro que só é visível no console. Nada acontece no navegador e a pessoa que digitou o endereço inválido fica sem saber o que aconteceu.

Para melhorar a experiência de uso do navegador, passaremos a capturar esses erros e alertar a pessoa caso o endereço digitado seja inválido:



Figura 31.1: Alerta em casos de endereço inválido

No momento da captura dos erros precisamos ter cuidado para não capturar outros tipos de erro, como erros decorrentes de falhas no código. Não queremos que um erro de programação faça com que a pessoa ache que digitou o endereço errado. Para capturar apenas erros específicos de endereço inválido, precisaremos lançar um erro customizado, ou seja, criaremos um novo tipo de objeto de erro.

Os outros tipos de erro, que não sabemos ao certo quais são, não devem ser exibidos para a pessoa; devem parar a execução e aparecer apenas no console.

Um endereço inválido de exemplo para usar nos seus testes é "invalido:invalido"

## 31.2 PASSO A PASSO COM CÓDIGO

1. No arquivo **Endereco.js** na pasta **scripts/endereco** faça as seguintes alterações:

```
scripts/endereco/Endereco.js

+import {CakeEnderecoInvalidoError} from '/scripts/erros/CakeEnderecoInvalidoError.js'
+
function Endereco(endereco) {

 if(
 this === undefined ||
 (this !== undefined && !(this instanceof Endereco))
) {
 return new Endereco(endereco)
 }

 let enderecoCompleto
 let enderecoResumido

 if(!endereco || endereco ==='blank') {
 enderecoCompleto = 'blank'
 enderecoResumido = 'blank'
 } else {
 const url = new URL(endereco)
 if (
 endereco.substring(0, 7) !== 'http://' &&
 endereco.substring(0,8) !== 'https://'
) {
 // Assignement Atribuição
 endereco = 'http://' + endereco
 }
 +
 let url
 try {
 url = new URL(endereco)
 } catch(error) {
 const erroCustomizado = new CakeEnderecoInvalidoError(endereco)
 throw erroCustomizado
 }
 }

 if(url.hostname === 'localhost') {
 const paginaLocal = url.pathname.replace("/", "")
 enderecoCompleto = paginaLocal
 enderecoResumido = paginaLocal
 } else {
 enderecoCompleto = url.toString()
 enderecoResumido = url.hostname
 }
}

this.urlCompleta = enderecoCompleto
this.urlResumida = enderecoResumido
}

export { Endereco }
```

2. Crie o arquivo **CakeEnderecoInvalidoError.js** na pasta **scripts/erros** com o seguinte código:

```
scripts/erros/CakeEnderecoInvalidoError.js
```

```
+function CakeEnderecoInvalidoError(endereco){
+ this.endereco = endereco
+ this.message = `Não consegui entender o endereço: \n\n${endereco} `
+}
+
+export { CakeEnderecoInvalidoError }
```

3. No arquivo **barraEndereco.js** na pasta **scripts/navegacao** faça as seguintes alterações:

```
scripts/navegacao/barraEndereco.js

import { carregar } from '/scripts/navegacao/carregar.js'
import { Endereco } from '/scripts/endereco/Endereco.js'

+import { CakeEnderecoInvalidoError } from '/scripts/erros/CakeEnderecoInvalido.js'
+
let endereco

$janelaPrincipal.addEventListener('load', function() {
 endereco = new Endereco()
 this.contentWindow.location.href
})
})

$inputEndereco.addEventListener('focus', exibeEnderecoCompleto)

$inputEndereco.addEventListener('blur', exibeEnderecoResumido)
$janelaPrincipal.addEventListener('load', exibeEnderecoResumido)

function exibeEnderecoCompleto(){
 $inputEndereco.value = endereco.urlCompleta
}

function exibeEnderecoResumido() {
 $inputEndereco.value = endereco.urlResumida
}

$inputEndereco.addEventListener('keyup', function(evento) {
 const apertouEnter = evento.key === 'Enter'
 if(apertouEnter) {
 - endereco = Endereco($inputEndereco.value)
+
+ try {
+ endereco = Endereco($inputEndereco.value)
+ } catch (error) {
+ if(error instanceof CakeEnderecoInvalidoError){
+ alert(error.message)
+ } else {
+ throw error
+ }
+
+ carregar(endereco)
 }
 }
})
```

# EXERCÍCIO: SOBRESCREVENDO MÉTODOS PADRÃO – O PROTOTYPE

## 32.1 OBJETIVO

Voltaremos para a página de configuração, onde ainda utilizamos nossa função que formata strings de endereço. Chamamos essa função para formatar o endereço da página inicial quando a pessoa aperta o botão de salvar, porém, essa função não lida com erros de endereço inválido. Para mudar isso, passaremos a usar nossa função construtora de endereços que já faz isso para a gente.

O código da página de configuração, que salva o endereço da página inicial e exibe ele no campo, espera que esse endereço seja uma string. Agora que nosso endereço é um objeto, teríamos que alterar todo o código da página de configuração para acessar as propriedades certas do nosso objeto.

Para evitar a modificação de todo o código da página de configuração que espera que o endereço seja uma string, podemos aproveitar o comportamento de *Type Coercion* do JavaScript (quando um valor é convertido de um tipo para outro). Para strings, geralmente essa conversão se dá chamando o método `.toString()`, presente em todos os objetos do JavaScript.

Nesse exercício, implementaremos a função `toString` dos nossos objetos endereço que deve retornar o endereço completo. Um detalhe importante é que queremos apenas **uma** função `toString` sendo reaproveitada para todos os objetos de `Endereco`.

## 32.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `Endereco.js` na pasta `scripts/endereco` faça as seguintes alterações:

```
scripts/endereco/Endereco.js

import {CakeEnderecoInvalidoError} from '/scripts/erros/CakeEnderecoInvalidoError.js'

function Endereco(endereco) {

 if(
 this === undefined ||
 (this !== undefined && !(this instanceof Endereco))
) {
 return new Endereco(endereco)
 }
}
```

```

let enderecoCompleto
let enderecoResumido

if(!endereco || endereco === 'blank') {
 enderecoCompleto = 'blank'
 enderecoResumido = 'blank'
} else {
 if (
 endereco.substring(0, 7) !== 'http://' &&
 endereco.substring(0,8) !== 'https://'
) {
 // Assignement Atribuição
 endereco = 'http://' + endereco
 }

 let url
 try {
 url = new URL(endereco)
 } catch(error) {
 const erroCustomizado = new CakeEnderecoInvalidoError(endereco)
 throw erroCustomizado
 }

 if(url.hostname === 'localhost') {
 const paginaLocal = url.pathname.replace("/", "")
 enderecoCompleto = paginaLocal
 enderecoResumido = paginaLocal
 } else {
 enderecoCompleto = url.toString()
 enderecoResumido = url.hostname
 }
}

this.urlCompleta = enderecoCompleto
this.urlResumida = enderecoResumido
}

+
+function toString() {
+ return this.urlCompleta
+}
+
+Endereco.prototype.toString = toString

export { Endereco }

```

2. No arquivo `paginaConfiguracao.js` na pasta `scripts` faça as seguintes alterações:

```

scripts/paginaConfiguracao.js

import * as storagePaginaInicial from '/scripts/storage/paginaInicial.js'
import * as storageAceitouSalvar from '/scripts/storage/aceitouSalvar.js'

-import { formataEndereco } from '/scripts/endereco/formataEndereco.js'
+import { Endereco } from '/scripts/endereco/Endereco.js'

$inputPaginaInicial.value = storagePaginaInicial.paginaInicial
$inputPermitiuSalvar.checked = storageAceitouSalvar.aceitouSalvar

$botaoSalvar.onclick = salvar

function salvar(){
 const funcaoEscolhida = $inputPermitiuSalvar.checked === true

```

---

```

 ? storageAceitouSalvar.setAceitou
 : storageAceitouSalvar.setNaoAceitou

 funcaoEscolhida()

- const enderecoCompleto = formataEndereco($inputPaginaInicial.value)
+ const enderecoCompleto = new Endereco($inputPaginaInicial.value)
$inputPaginaInicial.value = enderecoCompleto

storagePaginaInicial.setPaginaInicial(enderecoCompleto)
}

$botaolimpaTudo.addEventListener('click', function () {
 const chavesPermanentes = [
 'aceitouSalvar',
 'aceitouTermos'
]

 const listaChavesLocalStorage = Object.keys(localStorage)
 for(let chave of listaChavesLocalStorage) {
 const isChavePermanente = chavesPermanentes.includes(chave)

 if(!isChavePermanente){
 localStorage.removeItem(chave)
 }
 }

 const listaChavesSessionStorage = Object.keys(sessionStorage)
 for(let chave of listaChavesSessionStorage) {
 sessionStorage.removeItem(chave)
 }

 window.location.reload()
})

```

# EXERCÍCIO: ERROS CUSTOMIZADOS EXTENDENDO ERROR – MAIS PROTOTYPE E AS CLASSES DO ECMASCIPT

## 33.1 OBJETIVO

Agora que já usamos nossa função construtora na página de configuração, passaremos a deixar o campo de endereço da página inicial em branco e alertaremos a pessoa caso o endereço digitado seja inválido:



Figura 33.1: Alerta de endereço inválido

Também exibiremos algumas mensagens e o erro no console, como um aviso ou *warning*, utilizando a função `console.warn`. No caso do Cake, ele fica com uma cor de fundo amarela ao invés de vermelha. Ao exibir esse aviso no console, queremos que o caminho até o código que disparou o erro também seja exibido – muito parecido com o que é exibido em erros:

## Configurações

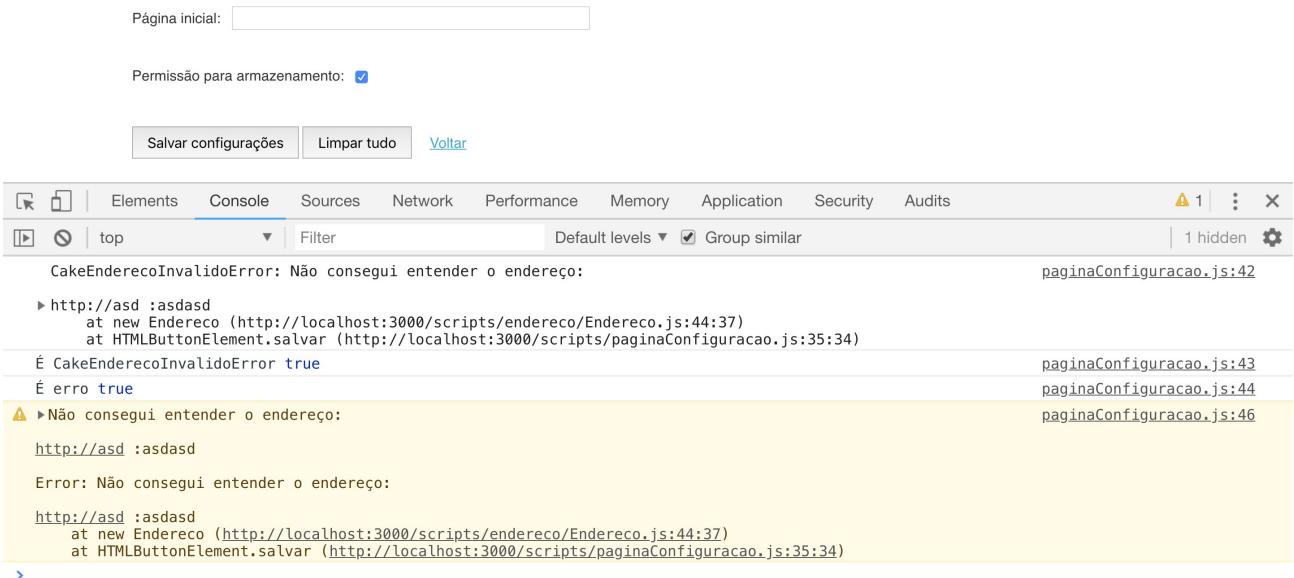


Figura 33.2: Mensagens e avisos no console

Para isso, transformaremos nosso erro de endereço inválido num tipo `Error` do Javascript. No passo a passo com código, mostraremos duas abordagens equivalentes para transformar nosso erro de endereço inválido num `Error` do JavaScript. Uma das abordagens usa a função construtora que já temos e a outra usa a sintaxe de `class` que tenta simplificar a sintaxe para criação de funções construtoras mais complexas.

## 33.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `Endereco.js` na pasta `scripts/endereco` faça as seguintes alterações:

```
scripts/endereco/Endereco.js

-import {CakeEnderecoInvalidoError} from '/scripts/erros/CakeEnderecoInvalidoError.js'
+import {CakeEnderecoInvalidoError} from '/scripts/erros/CakeEnderecoInvalidoErrorClasse.js'

function toString() {
 return this.urlCompleta
}

function Endereco(endereco) {

 if(
 this === undefined ||
 (this !== undefined && !(this instanceof Endereco))
) {
 return new Endereco(endereco)
 }
}
```

```

let enderecoCompleto
let enderecoResumido

if(!endereco || endereco ==='blank') {
 enderecoCompleto = 'blank'
 enderecoResumido = 'blank'
} else {
 if (
 endereco.substring(0, 7) !== 'http://' &&
 endereco.substring(0,8) !== 'https://'
) {
 // Assignement Atribuição
 endereco = 'http://' + endereco
 }

 let url
 try {
 url = new URL(endereco)
 } catch(error) {
 const erroCustomizado = new CakeEnderecoInvalidoError(endereco)
 throw erroCustomizado
 }

 if(url.hostname === 'localhost') {
 const paginaLocal = url.pathname.replace("/", "")
 enderecoCompleto = paginaLocal
 enderecoResumido = paginaLocal
 } else {
 enderecoCompleto = url.toString()
 enderecoResumido = url.hostname
 }
}

this.urlCompleta = enderecoCompleto
this.urlResumida = enderecoResumido
}

Endereco.prototype = {
 toString: toString
}

export { Endereco }

```

2. No arquivo **CakeEnderecoInvalidoError.js** na pasta **scripts/erros** faça as seguintes alterações:

```

scripts/erros/CakeEnderecoInvalidoError.js

function CakeEnderecoInvalidoError(endereco){
+ if(
+ !(this instanceof CakeEnderecoInvalidoError)
+) {
+ return new CakeEnderecoInvalidoError(endereco)
+ }
+
+
+ const erro = new Error('CakeEnderecoInvalidoError')
+ this.stack = erro.stack
 this.endereco = endereco
 this.message = `Não consegui entender o endereço: \n\n${endereco}`
}

```

```
+CakeEnderecoInvalidoError.prototype = Error.prototype
+CakeEnderecoInvalidoError.prototype.toString = function() {
+ return `${this.message}\n\n${this.stack}`
+}
+
+export { CakeEnderecoInvalidoError }
```

3. Crie o arquivo **CakeEnderecoInvalidoErrorClasse.js** na pasta **scripts/erros** com o seguinte código:

```
scripts/erros/CakeEnderecoInvalidoErrorClasse.js
```

```
+class CakeEnderecoInvalidoError extends Error {
+ constructor(endereco){
+ super('CakeEnderecoInvalidoError')
+ this.endereco = endereco
+ this.message = `Não consegui entender o endereço: \n\n${endereco} `
+ }
+
+ toString() {
+ return `${this.message}\n\n${this.stack}`
+ }
+}
+
+export { CakeEnderecoInvalidoError }
```

4. No arquivo **paginaConfiguracao.js** na pasta **scripts** faça as seguintes alterações:

```
scripts/paginaConfiguracao.js
```

```
import * as storagePaginaInicial from '/scripts/storage/paginaInicial.js'
import * as storageAceitouSalvar from '/scripts/storage/aceitouSalvar.js'

-import { formataEndereco } from '/scripts/endereco/formataEndereco.js'
+import { Endereco } from '/scripts/endereco/Endereco.js'
+import { CakeEnderecoInvalidoError } from './erros/CakeEnderecoInvalidoErrorClasse.js';

$inputPaginaInicial.value = storagePaginaInicial.paginaInicial
$inputPermitiuSalvar.checked = storageAceitouSalvar.aceitouSalvar

// o que vai ser executado quando clicar
// o que vai ser executado quando o evento de click acontecer
$botaoSalvar.onclick = salvar

// função de callback
// hoisting
// função é um tipo de dado
// executada em um outro momento do tempo
// Declaração de função
// Function declaration
function salvar(){

 // Expressão de função
 // Function expression
 const funcaoEscolhida = $inputPermitiuSalvar.checked === true
 ? storageAceitouSalvar.setAceitou
 : storageAceitouSalvar.setNaoAceitou

 funcaoEscolhida()
```

```

- const enderecoCompleto = new Endereco($inputPaginaInicial.value)
- $inputPaginaInicial.value = enderecoCompleto
-
- storagePaginaInicial.setPaginaInicial(enderecoCompleto)
+ try {
+ const enderecoCompleto = new Endereco($inputPaginaInicial.value)
+
+ $inputPaginaInicial.value = enderecoCompleto.toString()
+ storagePaginaInicial.setPaginaInicial(enderecoCompleto)
+ } catch (error){
+ if(error instanceof CakeEnderecoInvalidoError){
+ console.dir(error)
+ console.log('É CakeEnderecoInvalidoError', error instanceof CakeEnderecoInvalidoErro
r)
+ console.log('É erro', error instanceof Error)
+ $inputPaginaInicial.value = ''
+ console.warn(error.toString())
+ alert(error.message)
+ } else {
+ throw error
+ }
+ }
}

$botaoLimpaTudo.addEventListener('click', function () {
 const chavesPermanentes = [
 'aceitouSalvar',
 'aceitouTermos'
]

 const listaChavesLocalStorage = Object.keys(localStorage)
 for(let chave of listaChavesLocalStorage) {
 const isChavePermanente = chavesPermanentes.includes(chave)

 if(!isChavePermanente){
 localStorage.removeItem(chave)
 }
 }

 const listaChavesSessionStorage = Object.keys(sessionStorage)
 for(let chave of listaChavesSessionStorage) {
 sessionStorage.removeItem(chave)
 }

 window.location.reload()
})

```

# EXERCÍCIO: ADICIONANDO SITES NA BARRA DE FAVORITOS – AS CLASSES COMO SYNTAX SUGAR E O BIND

## 34.1 OBJETIVO

Nesse exercício criaremos a funcionalidade de adicionar favoritos na barra de favoritos.

Criaremos o botão de favoritos à frente do campo de endereço:

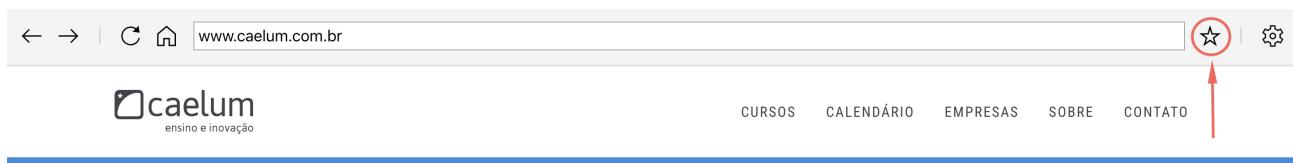


Figura 34.1: Botão de favoritos na barra de endereço

Quando a pessoa clicar no botão de favoritos, devemos perguntar o nome do favorito que será exibido na barra e após a confirmação, um novo favorito deve ser adicionado na barra de favoritos:

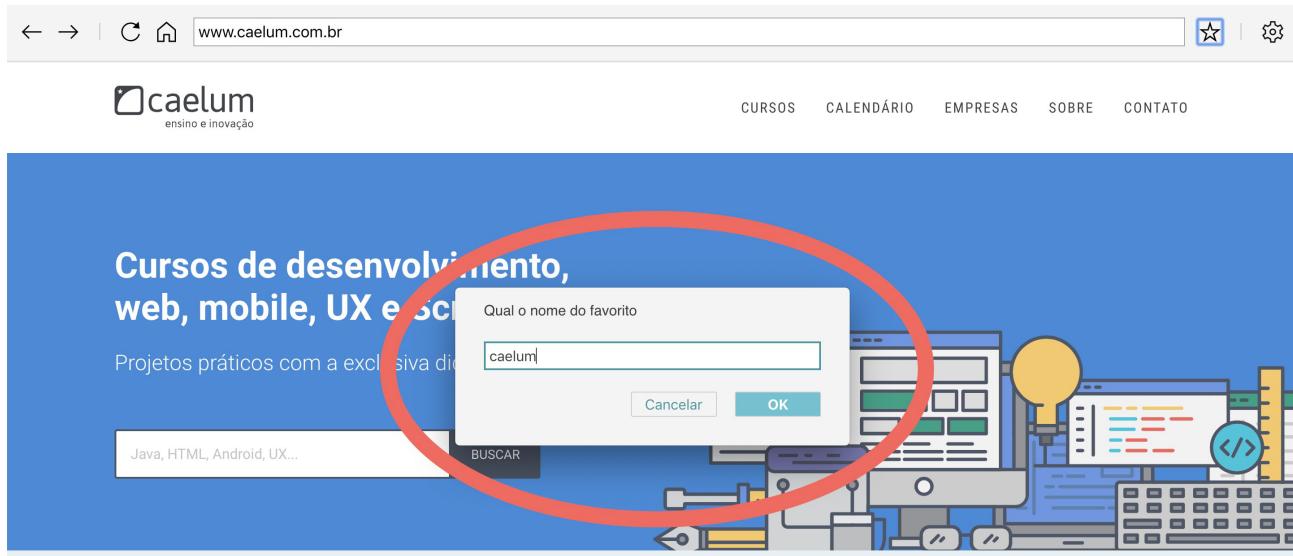


Figura 34.2: Perguntando o nome do favorito



Figura 34.3: Favorito sendo exibido na barra de favoritos

Caso a pessoa não digite um nome para o favorito, o endereço resumido dever ser exibido:

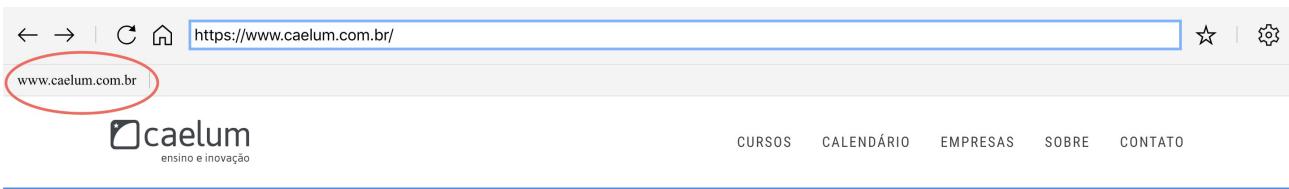


Figura 34.4: Endereço resumido sendo exibido

A parte visual da barra de endereços não é o foco desse exercício. Assim, o código que faz essa parte já está parcialmente pronto. Para essa parte visual será preciso apenas criar uma tag `<nav>` vazia abaixo da tag `<header>`. Com isso pronto, sempre que quisermos adicionar um favorito na barra chamaremos uma função que já está pronta chamada `$Cake.addFavorite`.

Essa função espera como parâmetro um objeto, assim:

```
$Cake.addFavorite({
 nome: 'Teste',
 descricao: 'teste.com',
 onclick: function() { alert('oi') }
})
```

No exemplo acima, quando um favorito é clicado na barra de favoritos, ele exibirá um popup com o texto `'oi'`. Para nossa funcionalidade, precisamos que um favorito, quando selecionado, execute a função `carregar` passando como parâmetro o endereço do favorito.

Todo o código da criação do favorito deve ser executado no evento de `click` do botão de favoritos. Para que a lógica no *event listener* se mantenha o mais simples possível, isolaremos a criação do objeto favorito numa função construtora. Como precisaremos que nosso objeto favorito tenha acesso às propriedades de um objeto do tipo `Endereco` para decidir o valor de `nome` e para definir o que acontece no `onclick`, nós faremos com que nosso objeto favorito seja também um `Endereco`.

## 34.2 PASSO A PASSO COM CÓDIGO

1. No arquivo `index.html` na pasta `raíz do projeto` faça as seguintes alterações:

```
index.html
```

```
<meta charset="utf-8">
```

```

<link rel="stylesheet" href="styles/cake-2afdf04e92.css"/>

<header>
 <input type="image" src="images/libCake/seta-voltar.svg" id="$botaoVoltar">
 <input type="image" src="images/libCake/seta-avancar.svg" id="$botaoAvancar">

 <input type="image" src="images/libCake/recarregar.svg" onclick="window.location.reload()">
 <input type="image" src="images/libCake/casa.svg" id="$botaoHome">
-
+
 <input type="text" id="$inputEndereco">
+
+ <input type="image" src="images/libCake/estrela-vazia.svg" id="$botaoFavoritos">
+

+
+
+
</header>
+<nav></nav>

<iframe frameborder="0" id="$janelaPrincipal"></iframe>

<script src="scripts/cake-8709f3abc4.js"></script>

<script type="module" src="scripts/main.js"></script>

```

2. Crie o arquivo **botaoFavoritos.js** na pasta **scripts/favoritos** com o seguinte código:

```

scripts/favoritos/botaoFavoritos.js

+import { carregar } from '/scripts/navegacao/carregar.js'
+import { Endereco } from "/scripts/endereco/Endereco.js";
+
+class Favorito extends Endereco{
+
+ constructor(nome, endereco) {
+ super(endereco)
+ this.nome = nome || this.urlResumida
+ this.descricao = this.urlResumida
+
+ // `bind` fixa o valor do `this` quando `onclick` é executado
+ this.onclick = this.onclick.bind(this)
+
+ /*
+ Opção ao bind é criar uma outra função como uma *arrow function*:
+ this.onclick = () => this.__proto__.onclick()
+
+ O `this` em *arrow functions* tem escopo léxico.
+ Ou seja, o `this` vai sempre apontar para o this do lugar onde a função foi criada.
+ */
+ }
+
+ onclick () {
+ carregar(this)
+ }
+}
+$botaoFavoritos.addEventListener('click', function(){
+ const nome = prompt("Qual o nome do favorito")
+
+ const favorito = new Favorito(nome, $janelaPrincipal.contentWindow.location.href)

```

```
+
+
+ $Cake.addFavorite(favorito)
+})
```

3. Crie o arquivo **index.js** na pasta **scripts/favoritos** com o seguinte código:

```
scripts/favoritos/index.js

+import '/scripts/favoritos/botaoFavoritos.js'
```

4. No arquivo **main.js** na pasta **scripts** faça as seguintes alterações:

```
scripts/main.js

import '/scripts/pedeInfosIniciais/index.js'
import '/scripts/navegacao/index.js'
+import '/scripts/favoritos/index.js'
```