Roteiro

JavaScript

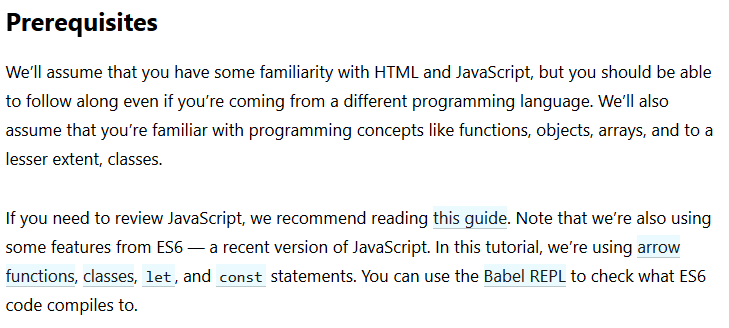
Tópicos

Opção 1

* DOM
* Tipos de dados
* Variáveis
* Operadores
* Estrutura de controles
* Objetos
* Arrays
* Funções
* Objetos customizados
* Closures
* Arrow functions
* Classes (propriedades, métodos e herança e entender o "this")
* let e const

Pré-Requisitos – React

<https://reactjs.org/tutorial/tutorial.html>



**Links**

<https://www.w3schools.com/js/default.asp>

<https://flaviocopes.com/javascript-introduction/>

<https://medium.com/trainingcenter/afinal-javascript-e-ecmascript-s%C3%A3o-a-mesma-coisa-498374abbc47>

<https://medium.freecodecamp.org/javascript-naming-conventions-dos-and-don-ts-99c0e2fdd78a>

**O que é JavaScript?**

Linguagem de programação



**Onde?**

Back-end, Mobile, Front-End, Banco de Dados

**Como?**

HTML -> Estrutura

CSS -> Estilização

JavaScript -> Comportamento

[https://cdn-images-1.medium.com/max/1600/0\*ILZZpnli\_R1FU3p4.gif](https://cdn-images-1.medium.com/max/1600/0*ILZZpnli_R1FU3p4.gif)

http-server

**Início**

*login.html*



**DOM**

Representa o documento HTML da página.

Pasta: svigufo-js-base

Imprimir o documento inteiro da página

<script>

console.log(document);

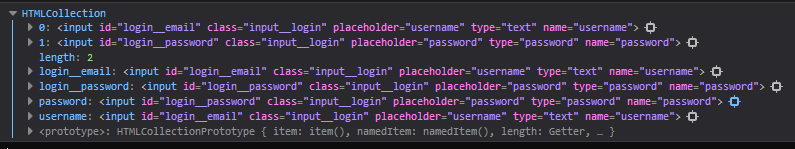
</script>

Buscar o input pela classe

<script>

console.log(document.getElementsByClassName("input\_\_login"));

</script>



Buscar o input pelo id

<script>

console.log(document.getElementById("login\_\_email"));

</script>

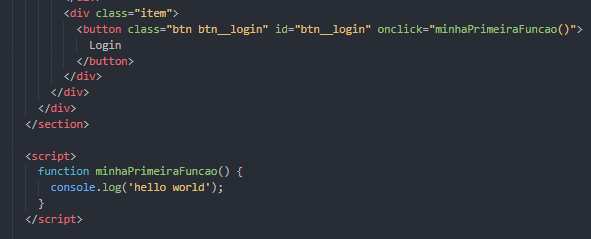
Utilizando o querySelector

<script>

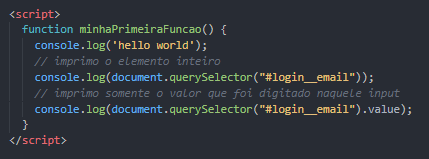
console.log(document.querySelector("#login\_\_email"));

</script>

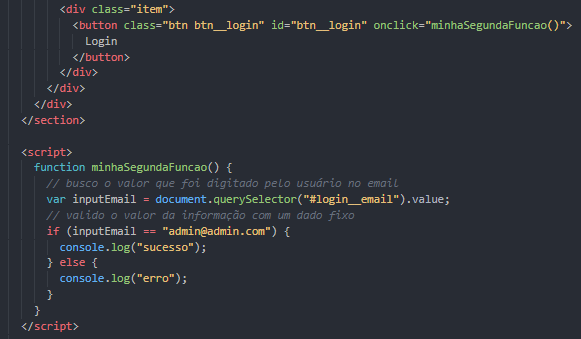
Colocando eventos no botão



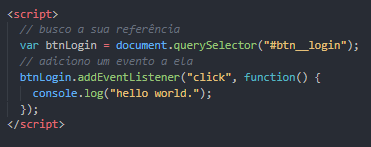
Função para imprimir os valores referentes ao input.



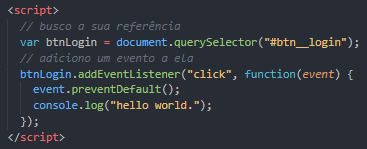
Função para validar as informações que foram digitadas pelo usuário.



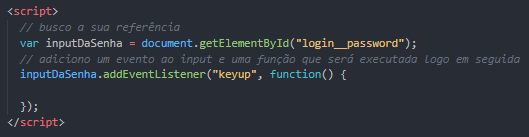
Adicionando comportamento ao botão de outra maneira (Obs.: remover o onclick do botão).

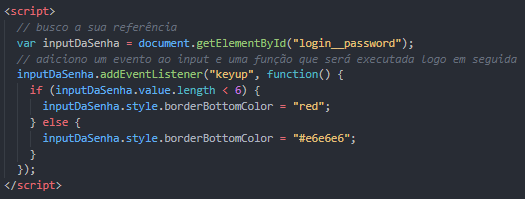


Evitando o comportamento padrão da página.



Adicionando um comportamento de keyup no input de password.





**Exercícios**

Pasta – 1.n*ews-js-base*

Dado um formulário para envio de e-mail, cujo botão de enviar está desabilitado e vermelho. Você deverá habilitar somente o botão caso o tamanho do valor do input seja maior do que 5. Troque a cor em caso positivo para “#ECDDF3”. Caso contrário, deixe vermelho.

E, após o botão ter sido habilitado, imprima uma mensagem de sucesso no console.

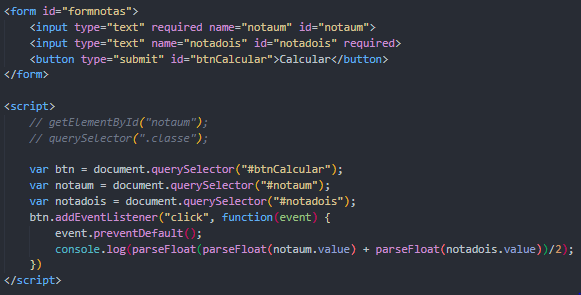


Desafio Extra:

Habilite somente o botão quando além da condição anterior, o input contenha um ‘@’ (arroba) e pelo menos um ‘.’ (ponto).

Pasta – 1.1.notas-js-base

Colocado dois campos de input para o usuário e um botão, ao clicar no botão, realizar o seguinte cálculo: (notaum + notadois) / 2. Além disso, imprimir no console o valor da média que foi calculada.



*login.html*

Buscando elementos no DOM e alterando sua informação.

<script>

*// guardo em uma variável*

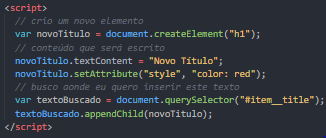
var textoDeExemplo = document.querySelector("#item\_\_description");

*// altero o seu valor*

textoDeExemplo.textContent = "Texto Alterado";

</script>

Além de alterar os valores dos elementos, eu posso criá-los.



*tipos-eventos.html*

Mostrar o início de arrays para que eles vejam o mesmo de outras linguagens mostrando de maneira bem objetiva o for e o foreach.

Arrays







PUSH/POP



Funções pré-definidas



Ajustar esta parte para criar os elementos na tela e fazer referência aos tipos de eventos e eventos.

MAP



FILTER



REDUCE



**Exercícios**

Pasta: ‘2.*pets-js’*

Dada uma lista de pets, contendo nome, idade e tipo de pet, realize os exercícios abaixo:

1. Liste somente os animais cujo tipo seja igual a gato;
2. Liste somente os animais cuja idade seja superior a 7;
3. Some a idade de todos os pets;
4. Traga somente os nomes dos pets;
5. Adicione mais um pet chamado doguinho, cuja idade é igual a 13 e o tipo seja cachorro;
6. Mostre a quantidade de cachorros que possuem na lista;



Desafio: além de imprimir no console, criar os elementos na tela (em uma tabela, por exemplo), cada iteração do item.

**API – Início**

*tipos-eventos.html*

Mostrar uma lista fixa na tabela – criando elementos e mostrando na tela.



Comunicando com a API.

<https://medium.com/beginners-guide-to-mobile-web-development/the-fetch-api-2c962591f5c>



[**https://developers.google.com/web/fundamentals/primers/promises**](https://developers.google.com/web/fundamentals/primers/promises)

**XMLHttpRequest**



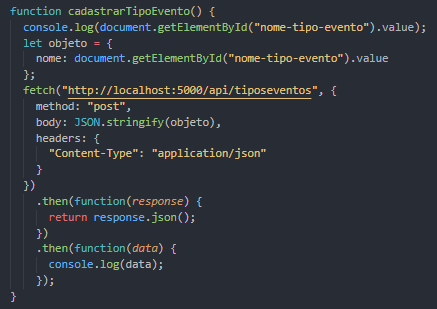


**FETCH**

Listando tipos de eventos utilizando a Fetch API



Realize o cadastro de um tipo de evento



<https://braziljs.org/blog/fetch-api-e-o-javascript/>

**Exercícios**

Pasta: 3.ui-personagens-base

API de personagens

<https://salty-atoll-57329.herokuapp.com/personagens>

<https://salty-atoll-57329.herokuapp.com/>

Com a API disponibilizada acima, construa um layout (podendo utilizar bibliotecas/frameworks, como semantic ui, materialize, bootstrap) para que a lista de personagens seja mostrada para o usuário.

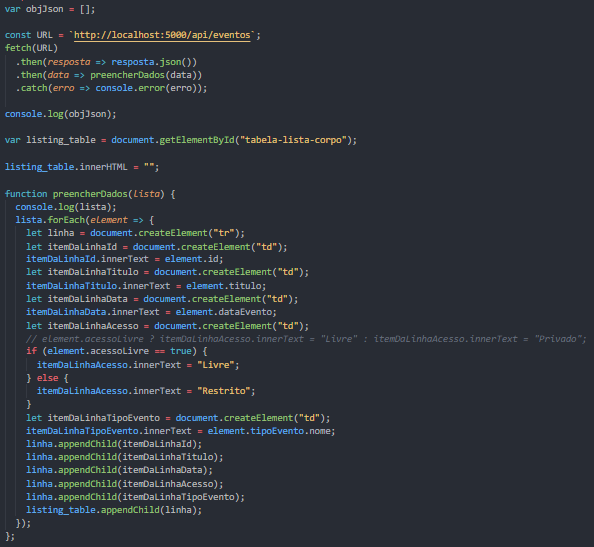
Não há restrições quanto a utilização do *XMLHttpRequest ou* *Fetch API*.

Um exemplo de sua saída, deverá ser a mostrada abaixo, conforme imagem:

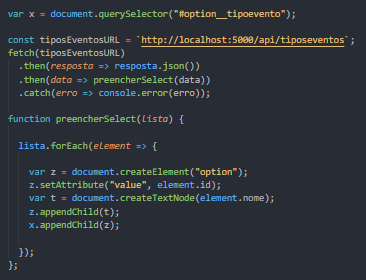


*eventos.html*

Mostrando a lista de eventos



Preenchendo o select de tipos de eventos



Cadastrando um novo evento





**Exercícios**

Realize o cadastro de um novo usuário.