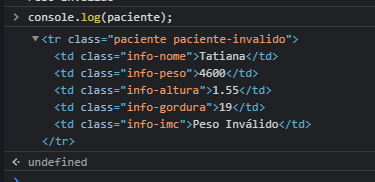
JavaScript

Introdução ao JS:

-No JS pode rodar comandos no console para testar comportamentos do código ex:



- Classe Documente

* É a representação do HTML para o JS
* Através dele se faz a manipulação de dados.
* .querySelector() = serve para buscar um único registro através de ID, classe, ou nome da variável.
* .querySelectorAll() = serve para buscar todos os registros daquela chave.
* .textContent = acessa o conteúdo da linha somente ex = 

- .classList

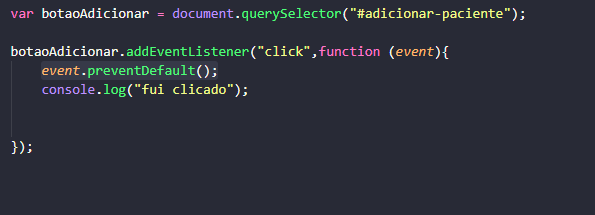
Mostra todas as **classes** que aquele HTML tem ex. 

Exemplo de adicionar css para todos h2.



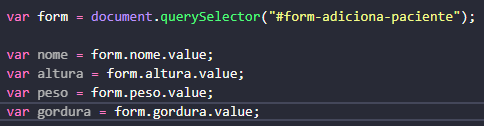
Classe event.preventDefault()

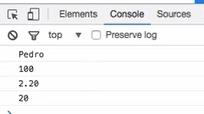
Remove o comportamento padrão exemplo do botão ao colocar isto ele não recarrega a pagina e também não apaga os campos do formulário ex.



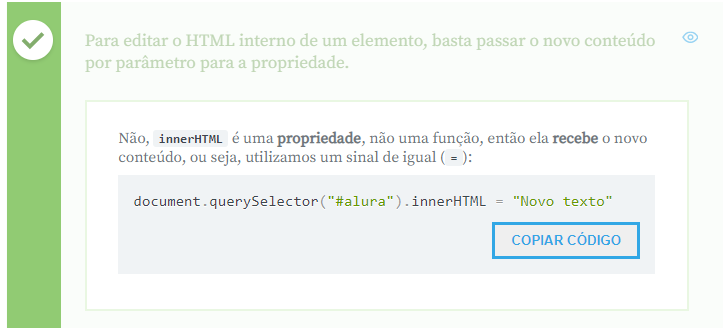
Value

Em JS o .value pega o valor de um comando HTML exemplo um input

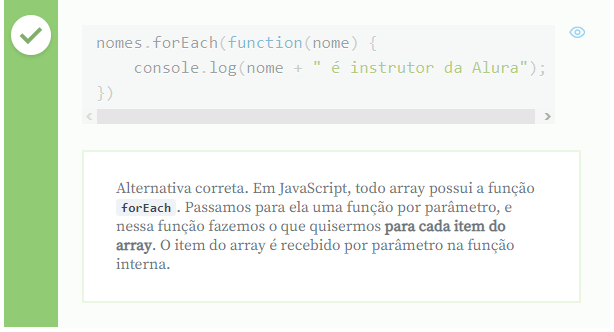




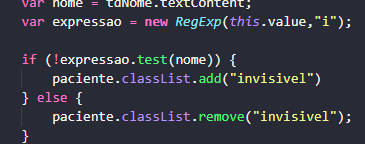
InnerHTML



ForEach

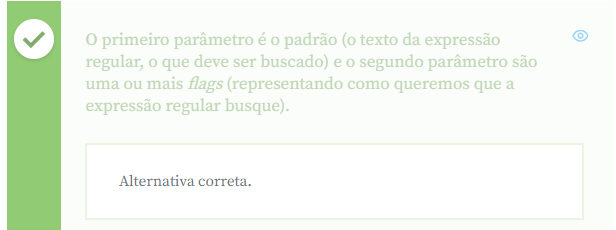


Regex em JS



Exercício

**regex**, utilizando **new RegExp(..)**. Quais são os parâmetros que podemos passar para o construtor?



O primeiro parâmetro que devemos passar para o construtor é o padrão (o texto da expressão regular, o que deve ser buscado) e o segundo parâmetro são uma ou mais flags (representando como queremos que a expressão regular busque). Por exemplo, podemos definir que não queremos que haja distinção entre letras maiúsculas e minúsculas, através da flag **i**.

Para saber mais sobre expressões regulares, há um curso bem interessante aqui na Alura, o curso [**Expressões regulares: Capturando textos de forma mágica**](https://cursos.alura.com.br/course/expressoes-regulares).

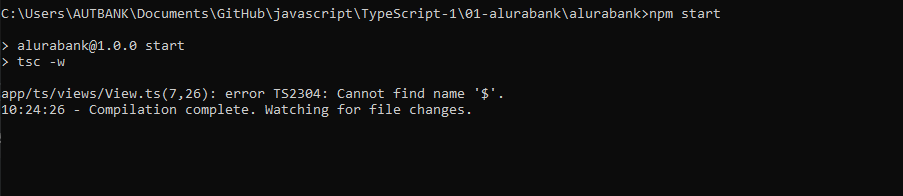
HTTP – AJAX

O XMLHtttpRequest é objeto responsável por fazer requisições HTTP com o Javascript.

Olhando as afirmativas, sabemos sim que ele deve ser configurado anteriormente com a função .open(), que ele pode trafegar diversos tipos de dados e não somente o XML ( ele tem esse nome graças a um vestígio histórico), e a requisição só é enviada mesmo após chamarmos o método .send().

TypeScript

Ao for rodar TS deve ir até a pasta do projeto e rodar o comando “npm start” para poder converter o arquivo ts em js.



Criando uma Classe

class Negociacao{

    constructor(*data*, *quantidade*, *valor*){

*this*.\_data = *data*;

*this*.\_quantidade = *quantidade*;

*this*.\_valor = *valor*;

    }

    get data(){

        return *this*.\_data ;

    }

    get quantidade(){

        return *this*.\_quantidade;

    }

    get valor(){

        return *this*.\_valor;

    }

    get volume (){

        return *this*.\_quantidade \* *this*.\_valor;

    }

}

Sobre TypeScript

O uso do Node.js não é opcional, pois o compilador depende desta plataforma para funcionar, inclusive é baixado pelo seu gerenciador de pacotes npm. Além disso, o código TypeScript precisa ser traduzido/convertido para um código em ECMASCRIPT para que seja entendido pelo navegador. Lembre-se que apenas ECMASCRIPT é suportado pelo navegador.

Sobre casting explícito

Correto! Realizamos o casting explícito de Element para HTMLTableElement. Inclusive, devido ao casting, o TypeScript infere que o tipo de tabela será HTMLTableElement.

Criando elementos do DOM dinamicamente

Através da propriedade innerHTML que recebe uma string que é convertida para elementos do DOM.

TSD

Um arquivo que mapeia para tua biblioteca ou jquery quais são os tipos que o js deve considerar

o repositório @types

No npm, existe uma série de TypeScript definitons files para as mais diversas bibliotecas e frameworks do mercado. Por exemplo, se quisermos instalar o tsd do jQuery, acessamos

<https://www.npmjs.com/package/@types/jquery>

Se quisermos do lodash ou underscore acessamos

https://www.npmjs.com/package/@types/lodash

https://www.npmjs.com/package/@types/underscoreCOPIAR CÓDIGO

Dessa forma, antes de sair buscando pela internet os arquivos tsd que precisamos, podemos tentar a sorte executando o comando:

npm install @types/nomeDaLibOuFramework --save-devCOPIAR CÓDIGO

Nesse sentido, se quisermos instalar os tsd das três bibliotecas que foram citadas, fazemos:

npm install @types/jquery --save-dev

npm install @types/lodash --save-dev

npm install @types/underscore --save-devCOPIAR CÓDIGO

Qualquer tsd files que esteja dentro de node\_modules/@types será lidado automaticamente pelo compilador do TypeScript.

É preciso se conformar quando não houver do Typing para sua biblioteca preferida, neste caso, a estratégia do declare var que vimos neste treinamento é uma saída, não muito ideal, mas que permitirá seu código compilar até que você encontre seu tsd.

Namespace

Na classe

namespace *Views* {

    export class MensagemView extends *Views*.View<*string*> {

        template(*model*: *string*): *string* {

            return `<p class="alert alert-info">${*model*}</p>`

        }

    }

}

No uso da classe

    private \_mensageView = **new** *Views*.MensagemView('#mensagemView');