Webdev - Módulo 2
Roteiro de aula

Aula 8 - Hard: Aquela herança

Tópicos da aula:

- This
- Bind e Arrow Function
- Controllers
- Herança

Objetivos da aula:

1. Entender o que são Boas Práticas em programação
2. Conceituar e empregar This, Bind and Arrow Function
3. Empregar o conceito de Controllers na prática
4. Empregar herança como meio de evitar repetição de códigos
5. Compreender o que é polimorfismo

Atividades da Aula

➔ **Atividade mão da massa: Quem tem o controle?**
- ◆ abra o projeto "Minhas Anotações" iniciado na aula 11, no VSCode (link para o repositório de referência (https://github.com/resilia-br/anotacoes);
- ◆ Caso ainda não tenha feito, crie , dentro da pasta src/front, as pastas "models", "views" e "controllers";
- ◆ Dentro da pasta "controllers" crie o arquivo "anotacoes-controllers.js"
- ◆ Dentro do arquivo:
  - Declare uma classe chamada "AnotacoesController";
  - A classe deve ter um constructor que possui como atributo um array onde serão armazenados as anotações, o mesmo deve ser inicializado vazio;
  - A classe deve ter um método chamada "addAnotacoes" que:
    - manipula o DOM para obter os valores dos campos de anotação de um formulário (a ser criado no index.html) e cria uma instância da classe Model "Anotacoes", que está

declarada dentro da pasta "models";
    - insere a instância da anotação criada dentro do array que foi inicializado no constructor;
- No arquivo index.html, importe o controller criado e crie o formulário que irá possuir um campo de input para receber o texto da anotação e um botão de submit que irá chamar o método "addAnotacoes" do controller.
- Dica: crie um arquivo index.js para inserir o eventHandler do botão de submit. Dentro da callback do evento, crie uma instância do "AnotacoesController" para poder chamar o método "addAnotacoes".

Momento Aprendizagem em Pares

➔ Esse momento é dedicado para vocês desenvolverem suas demandas e entregas para o curso.

➔ Utilize esse tempo da maneira que preferir, mas atente-se às aulas que você deve realizar as entregas.
- **Dica:** Nas Propostas dos projetos, vocês encontram uma sugestão de organização para a realização das atividades.
- **Lembre-se:** O momento de Aprendizagem em Pares é justamente para fazer trocas e aprender em comunidade, aproveite seus colegas e se desenvolvam juntos!

➔ **Entregas:**
- Projeto individual: aula 5 - TECH
- Projeto em grupo: aula 11 - TECH
- Apresentação do projeto: aula 11 - TECH

Revisão da aula

➔ As **arrow functions** não vinculam seu próprio this; em vez disso, herdam o do escopo pai, o que é chamado de "escopo léxico". Isso torna as arrow functions uma ótima escolha em alguns cenários, mas muito ruim em outros.

➔ Nós reutilizamos o construtor e os métodos minHeight e minWidth de TextCell. Um RTextCell é agora basicamente equivalente a TextCell, exceto que seu método draw contém uma função diferente. Este padrão é chamado **herança**. Isso nos permite construir tipos de dados levemente diferentes a partir de tipos de dados existentes, com relativamente pouco esforço. Típicamente, o novo construtor vai chamar o antigo construtor (usando o método call para ser capaz de dar a ele o novo objeto assim como o seu valor this). Uma vez que esse construtor tenha sido chamado, nós podemos assumir que todos os campos que o tipo do antigo objeto supostamente contém foram adicionados. Nós

organizamos para que o protótipo do construtor derive do antigo protótipo, então as instâncias deste tipo também vão ter acesso às propriedades deste protótipo. Finalmente, nós podemos sobrescrever algumas das propriedades adicionando-as ao nosso novo protótipo.

➔ A principal razão para isso é que este tópico é geralmente confundido com **polimorfismo**, vendido como uma ferramenta mais poderosa do que realmente é, e subsequentemente usado em excesso de diversas formas horríveis. Portanto, encapsulamento e polimorfismo podem ser usados para separar pedaços de código de cada um, reduzindo o emaranhamento de todo o programa, enquanto herança é fundamentalmente vinculada aos tipos, criando mais emaranhados.

Para ajudar

Links interessantes:

- .bind(), Arrow functions e o this, Claudio Júnior <https://claudiojunior.me/posts/bind-arrow-function-e-o-this/> Acesso em ago. 2022
- Quando (e por que) usar as arrow functions da ES6 — e quando não as usar, free code camp <https://www.freecodecamp.org/portuguese/news/quando-e-por-que-usar-as-arrow-functions-da-es6-e-quando-nao-as-usar/> Acesso em ago. 2022
- Function.prototype.bind(), MDN <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/bind> Acesso em ago. 2022
- JavaScript: Herança e métodos estáticos, DevMedia <devmedia.com.br/javascript-heranca-e-metodos-estaticos/41199> Acesso em ago. 2022
- Herança em Javascript, MDN <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Objects/Classes_in_JavaScript> Acesso em ago. 2022
- Conceitos e Exemplos – Polimorfismo: Programação Orientada a Objetos, DevMedia <https://www.devmedia.com.br/conceitos-e-exemplos-polimorfismo-programacao-orientada-a-objetos/18701> Acesso em ago. 2022
- Don't repeat yourself, wikipedia <https://pt.wikipedia.org/wiki/Don%27t_repeat_yourself> Acesso em ago. 2022.