Projeto 10. Checklist

Projeto rejeitado sem revisão

- Partes da funcionalidade não foram implementadas: não há nenhuma classe `Card` (Cartão) ou `FormValidador`.
- Há perguntas direcionadas aos revisores do projeto.

Requisitos do projeto

Geral

- Seu projeto contém:
  - `index.html` e `index.css`
  - Um diretório `blocks`
  - Uma pasta `images`
  - Arquivos de script `Card.js`, `FormValidator.js`, `index.js` e `utils.js`
  - Um arquivo `README.md`
- As folhas de estilo estão conectados em um arquivo separado.
- A formatação do código é consistente e tabulada de acordo as melhores práticas de elementos aninhados.
- O arquivo `README.md` contém:
  1. O nome do projeto
  2. Uma descrição do projeto e sua funcionalidade
  3. Um descrição das tecnologias e técnicas utilizadas
  4. O link para as páginas do GitHub
- A estrutura de arquivos BEM é utilizada.

O projeto está em conformidade com os seguintes requisitos de estilo de código:

- O projeto atende aos princípios da metodologia BEM.
- camelCase é usado para os nomes de funções e variáveis.
- Somente substantivos são usado como nomes de variáveis.
- Os nomes de variáveis descrevem o que está armazenado nelas. Se o projeto tiver diversas variáveis com dados parecidos, então estas variáveis terão nomes exclusivos, mas descritivos.
- O plural dos substantivos é usado para as coleções NodeList.
- Os nomes descritivos são usados para as funções, refletindo o que elas são.
- Os nomes da funções começam com um verbo.
- Os nomes não devem incluir abreviações impróprias ou pouco claras.

HTML/CSS

- `normalize.css` foi importado para `index.css` antes de outros arquivos CSS.
- `viewport` está configurado corretamente, `title` e `lang` estão sendo usados.

Todos os recursos listados no resumo foram implementados e estão funcionando corretamente:

- Todas as seções do projeto foram programadas.
- 6 cartões foram criados através de JavaScript.
- O formulário para adicionar cartões foi corretamente programado em HTML e CSS, o formulário pode ser aberto e o envio deste formulário adiciona um cartão.
- O botão curtir é funcional.
- O recurso para remover cartões foi implementado corretamente.
- O cartão pode ser adicionado pressionando Enter enquanto o campo de texto estiver ativo.

JavaScript + HTML/CSS

Janelas modais:

- Os pop-ups são criado com HTML e CSS. Eles não devem ser criados dinamicamente com JS.
- As caixas pop-up podem ser fechadas em qualquer resolução de tela.
- A caixa pop-up com a imagem abre corretamente, e as imagens são exibidas corretamente com suas proporções mantidas.
- O pop-up pode ser fechado clicando em qualquer lugar fora de suas bordas ou pressionando a tecla Esc.
- O pop-up não fechará ao ser clicado dentro de suas bordas, ou seja, no próprio formulário, não na sobreposição.

Validação:

- Todos os campos de entrada são validados por uma função universal.
- Os atributos HTML 5 e a propriedade JS `ValidityState` são utilizadas para validar os dados de entrada.
- Uma função separada controla o status do botão `Submit`.
- A função `enableValidation()` é usada para habilitar a validação.
- O botão `Submit` fica inativo caso algum dos campos não sejam validados.

POO:

- As classes ES6 são usadas.
- Cada classe (`Card` e `FormValidator`) é descrita em um arquivo JS separado e são importadas para o `index.js`.
- Para cada cartão é criada uma instância de classe `Card`. A classe `Card` deve cumprir os seguintes requisitos:
    1. O construtor deve usar os dados do cartão (texto e um link para a imagem).

2. O construtor deve usar o seletor de seu template contendo o código HTML.
3. Possui métodos privados para adicionar ouvintes de eventos, lidar com eventos de clique e preparar o cartão para exibição.
4. Possui um método público que devolve o código HTML finalizado com os ouvintes do evento anexados.

- Para cada formulário é criada uma instância de classe `FormValidator`. A classe `FormValidator` deve cumprir os seguintes requisitos:
  - O construtor deve receber um objeto de configuração que armazena seletores e classes de formulário.
  - O construtor deve receber o elemento HTML do formulário para ser validado.
  - Possui métodos privados para processar o formulário. Em cada método, é necessário fazer menção ao campo de classe, e não passá-lo para cada método, como foi implementado anteriormente.
  - Possui um método público `enableValidation()`. Chamar depois de criar uma instância de classe.
- Cada classe realiza uma única tarefa. Tudo relacionado a essa tarefa deve constar dentro de sua classe.

O código é otimizado:

- Quaisquer dados inseridos pelo usuário não devem ser atribuídos à propriedade `innerHTML`.
- Não há código duplicado. Se uma linha do código tiver que ser repetida, ela deve ser escrita como uma função separada.
- Se uma variável for declarada usando `let`, seu valor deve mudar em algum lugar.
- Se um valor de variável não mudar, será declarado usando `const`.
- Todos os valores numéricos são atribuídos às variáveis. Os valores únicos que não possuem suas próprias variáveis são chamados "números mágicos" e são considerados uma prática ruim na programação.

- Operações nos elementos DOM são executadas antes de serem inseridos no layout.
- Uma função desempenha uma única tarefa, por exemplo, retornar o código HTML do cartão.
- Um ouvinte de evento é adicionado para fechar o pop-up pressionando `Esc` quando o pop-up for aberto e é removido quando o pop-up for fechado.

Acessibilidade da Interface

- Todos os links e elementos interativos têm um estado `:hover`.
- Todos os elementos `<img>` têm um atributo `alt` contendo uma descrição no idioma da página.