Projeto Final

Dicas

- Recomendamos que comece trabalhando primeiro em uma simples grade de cartas. Não se preocupe com estilização, simplesmente faça algo clicável na página
- Desenvolva como cada carta é estruturada. Lembre-se, você deve representar os dois lados da carta. Você terá dois elementos separados, empilhados um sobre o outro?
- Quando tiver uma grade, adicione a lógica de cliques para revelar o lado oculto de cada carta
- Em seguida, trabalhe na lógica de correspondência de pares. Como o seu jogo "sabe" se um jogador dá um palpite correto ou incorreto?
- Recomendamos que deixe a estilização para o final. Permita que a lógica e a funcionalidade de seu jogo ditem a estilização.

Diretrizes de estilo

Diretrizes de CSS

- Consulte o Guia de Estilo CSS da Udacity.

Diretrizes de HTML

- Consulte o Guia de Estilo HTML da Udacity

Diretrizes JavaScript

- Consulte o Udacity JavaScript Style Guide.

Diretrizes Git

- Consulte o Guia de estilo de Git

Desenvolvimento de estrátegia

É muito importante que planeje seu projeto antes de começar a escrever qualquer código. Quebre o seu projeto em pequenos pedaços de trabalho e planeje sua abordagem em cada um. É muito mais fácil fazer o debug e consertar um problema se você fez apenas uma pequena mudança. As coisas ficam mais difíceis se você demorar muito pra testar seu código. Casas não são construídas tudo de uma vez, mas de tijolo por tijolo.

Fique à vontade de implementar sua própria ideia de fluxo de trabalho, mas se você dar uma travada -- aqui estão algumas dicas rápidas pra começar!

Comece construindo um grid dos cartões

Depois de tudo, o resto das funcionalidades do seu jogo depende do grid.

- Quantos pares de cartões você terá?
- Que estrutura de dados você pode usar pra armazenar os símbolos dos cartões? Como você vai iterar (i.e., loop) sob essa estrutura de dados?
- Pense sobre como você pode criar, digamos, uma lista não ordenada em HTML por meio dessa estrutura. Lembre-se de algumas ferramentas e métodos que você aprendeu:
    - `createElement()`
    - `querySelector()`
    - `getElementbyId()`
    - `appendChild()`
    - `Document`
- Seus cartões são colocados aleatoriamente no grid?
- Desenvolve o HTML necessário para representar um cartão. Lembre-se, você precisa representar os dois lados do cartão, e os símbolos ficam de virados "pro chão"
- Como você pode usar propriedades de CSS como transform ou opacity para representar os lados de um cartão?

Adicione a funcionalidade de lidar com os cliques

Os cliques devem revelar a parte "oculta" de cada cartão. Clicar em um primeiro cartão deve fazê-lo virar, mostrar o símbolo, e permanecer virado. Clicar em um cartão diferente também deve fazê-lo virar e mostrar o símbolo.

- Quais evento(s) você deve escutar?
- Como cada evento afeta o CSS?
- Como você pode prevenir que o usuário selecione o mesmo cartão duas vezes?

Trabalhe na lógica de acerto. Como o seu jogo "sabe" se um jogador combinou ou não dois cartões de mesmo símbolo?

- Pense sobre como você pode armazenar temporariamente em algum lugar o primeiro cartão virado. Depois, este cartão precisa ser comparado com o próximo a ser virado.
- Se dois cartões combinam, eles devem permanecer virados
- Se dois cartões não combinam, eles devem voltar a esconder seus símbolos

Trabalhe na condição de jogo ganho

Como seu jogo "sabe" se um jogador ganhou?

- Seu usuário deve ver um modal quando o jogo termina

Implemente funcionalidades adicionais:

- Botão de reiniciar

- Avaliação por estrelas
- Temporizador (como o setTimeout() pode entrar nisso?)
- Contador de movimentos do jogo

Recomendamos implementar a maioria dos estilos e design do jogo só no final de tudo

Permita que a lógica e funcionalidade do seu jogo determinem a estilização, design, etc.

Controle de versão

Recomendamos que você use o Git desde o começo. Certifique-se de se comprometer com frequência e de usar mensagens de confirmação bem-formatadas, que estejam em conformidade com o nosso Guia de estilo de Git.