Esse é o **CC50.**
O Curso de Harvard, no Brasil 2012/01

Pset 8: Ônibus do CC50

a ser entregue até: 19:00, sex 04/05

Objetivos.

- Preparar você para o Projeto Final.
- Introduzir você a JavaScript e a APIs terceirizados.
- Ensiná-lo a se ensinar novas linguagens.

Leitura recomendada.

- `http://www.w3schools.com/js/`
- `https://developer.mozilla.org/en/JavaScript/Guide`
- `http://code.google.com/apis/maps/documentation/javascript/tutorial.html`
- `http://code.google.com/apis/earth/documentation/`

NOTIFICAÇÃO.

Para este Pset, você é convidado e incentivado a consultar "recursos externos", incluindo livros, a Web e amigos, para aprender mais sobre HTML, CSS, PHP e SQL, desde que o seu trabalho em geral seja seu próprio. Em outras palavras, ainda continua a haver uma linha, mesmo que não precisamente definida, entre aprender com os outros e falar que o trabalho de outras pessoas é seu.

Você pode adotar ou adaptar trechos de código escritos por outras pessoas (encontrados em algum livro, online, ou em outro local), contanto que você cite (na forma de comentários) a origem dos mesmos.

E você pode aprender dos seus colegas, contanto que pedidos de ajuda não se transformem em "me mostre o seu código" ou "escreva isso para mim". Você não pode, para ficar claro, examinar o código dos colegas. Em caso de dúvida quanto à esses quesitos, manda um e-mail para ajuda@cc50.com.br.

Honestidade Acadêmica.

Todo o trabalho feito no sentido do cumprimento das expectativas deste curso deve ser exclusivamente seu, a não ser que a colaboração seja expressamente permitida por escrito pelo instrutor do curso. A colaboração na realização de Psets não é permitida, salvo indicação contrária definida na especificação do Set.

Ver ou copiar o trabalho de outro indivíduo do curso ou retirar material de um livro, site ou outra fonte, mesmo em parte e apresentá-lo como seu próprio constitui desonestidade acadêmica, assim como mostrar ou dar a sua obra, mesmo em parte, a um outro estudante. Da mesma forma é desonestidade acadêmica apresentação dupla: você não poderá submeter o mesmo trabalho ou similar a este curso que você enviou ou vai enviar para outro. Nem poderá fornecer ou tornar as soluções disponíveis para os Psets para os indivíduos que fazem ou poderão fazer este curso no futuro.

Você está convidado a discutir o material do curso com os outros, a fim de melhor compreendê-lo. Você pode até discutir sobre os Psets com os colegas, mas você não pode compartilhar o código. Em outras palavras, você poderá se comunicar com os colegas em Português, mas você não pode comunicar-se em, digamos, C. Em caso de dúvida quanto à adequação de algumas discussões, entre em contato com o instrutor.

Você pode e deve recorrer à Web para obter referências na busca de soluções para os Psets, mas não por soluções definitivas para os problemas. No entanto, deve-se citar (como comentários) a origem de qualquer código ou técnica que você descubra fora do curso.

Todas as formas de desonestidade acadêmica são tratadas com rigor.

Licença.

Notas:

Seu trabalho neste Pset será avaliado em três quesitos principais:

Exatidão. Até que ponto o seu código é consistente com as nossas especificações e livre de bugs?
Design. Até que ponto o seu código é bem escrito*(*escrito claramente, funcionando de forma eficiente, elegante, e / ou lógica)?
Estilo. Até que ponto o seu código é legível (comentado e indentado, com nomes de variáveis apropriadas)?

Começando.

☐ OMG, último Pset.

☐ Se você ainda não tiver feito isso, considere fazer o download e instalar o **Firefox** e o **Firebug** (nessa ordem), sendo que ambos estão disponíveis no site do curso em **Software.** Uma alternativa para esses programas também é a simples utilização do **Google Chrome.** Uma vez instalado, o Firebug aparecerá como uma opção no menu **Ferramentas** do Firefox. Toda vez que você quiser imprimir alguma informação na janela do seu navegador, para debugar a la `printf` em C, você pode incluir uma linha como

```
console.log("hello, world");
```

no seu código JavaScript. Enquanto a janela do Firebug estiver aberta, você verá esse texto nela. Certifique-se de remover qualquer dessas linhas antes de enviar o seu trabalho. Alternativamente, você pode incluir uma linha como

```
alert("hello, world");
```

no seu código JavaScript, mas o pop-up que vai aparecer tende a ser mais irritante, especialmente se você chamar `alert` acidentalmente dentro de um loop!

☐ Para esse Pset, o seu trabalho deve se comportar da mesma em pelo menos dois desses principais navegadores:

- ☐ Chrome 5.x
- ☐ Firefox 3.x
- ☐ Internet Explorer 8.x
- ☐ Opera 10.x
- ☐ Safari 5.x

Certifique-se, então, de testar o seu trabalho com pelo menos dois navegadores. Está tudo bem, no entanto, utilizar apenas um sistema operacional. E está tudo bem se você notar pequenas diferenças estéticas entre os dois navegadores. Certifique-se de registrar quais navegadores você utilizou na forma de comentários dentro do seu código.

☐ **Saia do seu Ubuntu! Eu sei que é triste mas o plugin do Google Earth, parte essencial desse Pset, não funciona no Linux. Todo o trabalho desse Pset será realizado dentro do Windows.**

☐ Já no Windows vá para a URL abaixo.

```
http://earth.google.com/plugin/
```

Se você for solicitado a baixar o plugin do Google Earth, vá em frente e instale-o. Com o plugin instalado, você pode precisar recarregar essa página ou reiniciar o seu navegador, mas no final você deve ver a Terra em 3D embutida na página.[1] Feche a página assim que isso funcionar.

Aliás, é melhor minimizar o número de programas e janelas que você abre enquanto trabalha neste conjunto de problemas, o plugin do Google Earth gosta de consumir ciclos de CPU e memória RAM. De fato, se você sentir que o seu computador está ficando lento, pare alguns programas ou até mesmo reinicie-o (depois de salvar o seu trabalho).

☐ Siga para a pasta `pset8`, onde você extraiu os arquivos desse Pset. Lá você deve ver uma pasta com fotos, todos os arquivos JavaScript que nós vamos usar, um arquivo CSS e o arquivo `index.html`, abra-o com o seu browser (e não com um editor de texto).

Você deverá se encontrar no campus da Universidade de Harvard.

Ônibus.

☐ Sua missão para este conjunto de problemas é implementar o seu próprio serviço de transporte que pega passageiros por todo o campus e os coloca em suas casas. O seu ônibus já está equipado com um motor, um volante, e 35 assentos. Vamos dar uma volta? Aqui está como dirigir com o seu teclado:[2]

Para frente: W
Para trás: S
Vire à esquerda: seta para a esquerda
Vire à direita: seta para a direita
Deslize para a esquerda: A
Deslize para a direita: D
Olhe para baixo: seta para baixo
Olhe para cima: seta para cima

Note que o mouse não é utilizado nesse mundo 3D. Na verdade, você não deveria nem clicar com ele, se não você vai tirar o "foco" do nosso (e, em breve, seu) código JavaScript, fazendo com que ônibus pare de responder às suas teclas. Se isso acontecer, basta clicar no mapa 2D ou em qualquer coisa acima dele (dentro da mesma janela) para dar o foco de volta ao código, fazendo com que o ônibus responda às suas teclas novamente.

De qualquer forma, vá dar um passeio! (Bicicletas são proibidas na grama, mas ônibus não tem problema) Quando você se aproxima de edifícios, você pode perceber que eles ficam mais bonitos a medida que mais imagens de satélite são baixadas automaticamente. Dê umas voltas (tente não cortar caminho através dos edifícios). Ao longo do caminho, você provavelmente verá alguns rostos de cientistas da computação. Aqueles em breve serão seus passageiros!

[1] Se informado de que "você não tem permissão para usar este serviço através de SSL" é provável que você tem um plugin que força TLS ou HTTPS sempre. Você vai querer desativar qualquer desses plugins, temporariamente, pelo menos para `www.google.com`, que é onde o API do Google Earth mora.

[2] O seu ônibus pode de fato deslizar para a esquerda e para a direita, como se todas as quatro rodas pudessem girar 90 graus.

Agora vá em frente e clique algumas vezes no sinal de menos (-) no canto superior esquerdo do mapa 2D. Você deverá ver cada vez mais marcadores vermelhos a medida que você diminui o zoom. Cada um deles representa um passageiro à sua espera. Se você passar o mouse sobre cada marcador, verá o nome e a origem de cada passageiro. O ônibus azul representa, lógico, você! Você é bem vindo a clicar e arrastar o mapa 2D para ver mais do campus; logo que você começar a dirigir novamente, o mapa irá se recentralizar em torno de você.

Acima do mapa 2D está uma lista das suas 35 cadeiras, todas elas vazias. Acima disso, no canto superior direito do aplicativo, estão dois botões: **Pegar passageiros** e **Deixar passageiros**. Nenhum deles funciona ainda, mas ambos funcionarão daqui a pouco!

Se você se perder, basta recarregar a página, e você vai ser devolvido à entrada da Universidade. Os seus passageiros também serão reposicionadas pseudoaleatoriamente por todo o campus.

☐ Tudo bem, vamos dar um passeio pelo código que você recebeu nesse Pset. Abra index.html e leia as linhas de HTML. Primeiro note como esse arquivo faz referência a vários outros no seu cabeçalho: `services.css`, `math3d.js`, `buildings.js`, `passengers.js`, `shuttle.js` e `service.js` bem como a URL do Google API. Nota agora como a tag body tem alguns parâmetros de evento, que chamam funções declaradas em service.js. Perceba como cada um dos elementos div tem um id, para que possamos estilizá-los com CSS ou acessá-los via JavaScript.

No momento, o HTML dentro de dois desses elementos `div` (`#announcements` e `#seats`) ainda não é o final. Eventualmente, haverão alguns anúncios, e com certeza haverão passageiros nos assentos!

Em seguida abra `service.css`, que estiliza esse HTML. Você não precisa entender todo o CSS aqui dentro, mas saiba que, em geral, `div#foo`, se refere ao elemento `div` cujo `id` é `foo`.

Agora, dê uma olhada em `buildings.js`. `BUILDINGS` é um array declarado nesse arquivo que contém vários objetos, cada um dos quais representa um prédio no campus. Cada edifício tem um número "root" associado a ele (Harvardês para ID de um edifício), um nome e um endereço, e um par de coordenadas que representa um dos vértices do prédio. Nós poderíamos ter declarado esse array em `service.js`, mas já que ele é muito grande, decidimos isolá-lo em seu próprio arquivo.

Da mesma forma, os passageiros têm o seu próprio arquivo. Em `passengers.js` você encontra `PASSANGERS`, outro array de objetos, cada um dos quais representa um passageiro no campus. Cada passageiro tem um nome de usuário (um identificador único), um nome, e uma casa. Aliás, cada um desses nomes de usuário mapeia para um JPEG de nome semelhante em `pset8/passengers`/. Note, porém, que algumas casas compreendem vários prédios, e, por isso, mesmo que "Mather House" apareça tanto em `passengers.js` quanto em `buildings.js`, "Quincy House" aparece apenas em `passengers.js`. Em `buildings.js`, entretanto, você encontra objetos para “Quincy House New Residence Hall”, "Quincy House Library", e (de forma legal e confusa) "Mather Hall, Quincy House". E assim você vai encontrar em `houses.js` uma terceira (e última!) estrutura de dados. Esse é um array associativo cujas chaves são nomes de

casas e cujos valores são objetos que representam coordenadas de casas. E assim você pode acessar as coordenadas de uma casa usando algo como:

```
var lat = HOUSES["Mather House"].lat;
var lng = HOUSES["Mather House"].lng;
```

Você pode assumir que as coordenadas em `houses.js` são as coordenadas oficiais de destino dos passageiros. Na verdade, nós já colocamos marcadores amarelos nessas coordenadas no mapa 2D para ajudar o motorista. Nós não nos preocupamos em colocar marcadores na Terra 3D, já que eles não são tão fáceis de detectar.

Agora, dê uma olhada em `shuttle.js`. Esse arquivo é um pouco extravagante, mas ele implementa efetivamente uma estrutura de dados (struct) chamada `Shuttle`.[1] Você não precisa entender tudo nesse arquivo, mas saiba que dentro de um `Shuttle` (Ônibus) você encontra duas propriedades: `position`, que guarda a posição do ônibus (incluindo a sua longitude e latitude), e `states`, que guarda todos os vários estados de um ônibus.

Ok, agora volte sua atenção para `service.js`, onde você vai fazer a maior parte do seu trabalho, mas você está livre para modificar qualquer um dos arquivos desse Pset, exceto `math3d.js` (que é apenas uma biblioteca), `buildings.js`, `houses.js` e `passengers.js`. Na parte de cima de `Service.js` você vai ver um monte de constantes e variáveis globais. Logo abaixo dessas linhas estão duas chamadas de `google.load`, que carrega os dois APIs dos quais esse aplicativo depende:

Google Maps JavaScript API V3
`http://code.google.com/apis/maps/documentation/javascript/tutorial.html`

Google Earth API
`http://code.google.com/apis/earth/documentation/`

Calma! Você não precisa ler toda essa documentação; é bem mais provável que você queira explorá-la quando necessário. Mas dê uma lida na primeira página de cada um deles para ter uma noção do que cada API faz.

Em `service.js` você encontra todas aquelas funções chamadas pelos eventos de `index.html`. Desça até a implementação de `load` primeiro. É essa função que incorpora os mapas 2D e 3D no seu browser. Em seguida vá até a implementação de `initCB;` essa função é chamada assim que o Plugin do Google Earth foi carregado, e, por isso, é nessa função que nós terminamos a inicialização do aplicativo (se algo der errado, `failureCB` é chamada em seu lugar). Observe, em particular, que `initCB` "instancia" um objeto, do tipo `Shuttle` (o ônibus em si).

Em seguida dê uma olhada na função chamada `keystroke`. É essa função que lida com as teclas que você aperta. É provável que essa função traga de volta algumas memórias do `sudoku`, embora dessa vez nós usamos `if` e `else`, em vez de um `switch`. Quando você aperta uma

[1] Embora Shuttle comporte-se como uma "classe", JavaScript é na verdade uma linguagem "baseada em protótipos": `http://en.wikipedia.org/wiki/Prototype-based`

tecla, essa função altera o estado do ônibus simplesmente atualizando a propriedade adequada em `shuttle.states` com um valor de `true` ou `false`.

Finalmente, dê uma olhada na função chamada `populate`. É essa função que coloca caras amigáveis por todo o campus. Cada pessoa é implementada como um "marcador" na Terra 3D e como um "marcador" no mapa 2D.

Não há necessidade de qualquer PHP ou SQL para esse Pset, apenas CSS, HTML e JavaScript. Certifique-se, então, de recarregar o seu navegador (ou limpar o cache), muitas vezes, para que você tenha sempre a última versão do seu código.[1]

☐ Tudo bem, está na hora de começar a pegar os passageiros. Lembre-se `index.html` onde, quando o botão **Pegar passageiros** é clicado, a função `pickup` em `service.js` é acionada. Implemente `pickup` da seguinte forma.

Se o botão for clicado quando o ônibus estiver a menos de 15 metros de distância de um passageiro e pelo menos um assento estiver vazio, o passageiro deve ganhar um assento no ônibus e ser removido tanto do mapa em 2D quanto da Terra 3D (se vários passageiros estão dentro de um raio de 15 metros do ônibus, você deve pegar o maior número deles, enquanto houver lugares vazios). Se não há nenhum passageiro em um raio de 15 metros do ônibus, anuncie esse fato. Se alguns passageiros estiverem dentro do alcance do ônibus, mas o ônibus não tiver nenhum assento livre, anuncie isso também (e não pegue o passageiro). Todos os anúncios devem sumir (ou serem substituídos por um texto padrão) logo que o ônibus começar a se mover.

Não sabe como proceder? Isso é parte do desafio! É bem provável que você encontre tais incertezas quando mergulhar no seu projeto final. Sempre que você estiver em dúvida, leia a documentação dos APIs, e pergunte para o Google (faça uma pesquisa) para obter dicas. E você deve postar em `cc50.com.br/forum` toda vez que tiver alguma dúvida. Mas nós vamos lhe dar algumas dicas para que você consiga começar a implementar a função `pickup`.

Para calcular a distância `d` entre o ônibus e algum ponto `(lat, lng)`, você provavelmente vai querer usar algo como:

```
var d = shuttle.distance(lat, lng);
```

Para remover um marcador `p` da Terra 3D, você provavelmente vai querer usar algo como:

```
var features = earth.getFeatures();
features.removeChild(p);
```

Para remover um marcador `m` do mapa 2D, você provavelmente vai querer usar algo como:

```
m.setMap(null);
```

[1] Em particular, você pode querer forçar o recarregamente do seu navegador, segurando Shift quando for recarregar.

Infelizmente, `populate` não se lembra, por enquanto, das posições das pessoas que são colocadas no mapa, por isso você pode precisar ajustar essa função também. Talvez você possa armazenar essas posições em algum(ns) array(s) ou objeto(s)?

Vamos pensar agora em como dar um assento a um passageiro? É bem provável que você vai querer usar um array de tamanho 35 para representar todos os lugares. E também é bem provável que você queira atualizar `div#seats` toda vez que esse array for atualizado. Lembre-se que você pode atualizar o HTML dentro de um elemento usando algo como:

```
document.getElementById("seats").innerHTML = "hello, world";
```

Embora `index.html` use atualmente uma lista ordenada para mostrar os lugares no ônibus, você é bem-vindo a apresentá-los da forma que você achar melhor. Certifique-se, porém, de listar os nomes dos passageiros e as casas de destino, para que você saiba quem eles são e onde você deve deixá-los.

Para fazer anúncios, um código como o seguinte dará conta do trabalho:

```
document.getElementById("announcements").innerHTML = "hello, world";
```

☐ Tudo bem, agora é a hora de implementar o botão de **Deixar passageiros**! Lembre-se que você viu em `buildings.js` que cada passageiro vive em uma casa. E lembre-se de `index.html` onde, assim que esse botão é clicado, a função chamada `dropoff` que mora em `service.js` é chamada. Você deve implementar esse recurso da seguinte forma.

Se o botão é clicado quando o ônibus está a menos de 30 metros da casa de um dos passageiros a bordo, esse passageiro deve ser deixado lá basicamente fazendo com que o seu assento fique vázio (se houver vários passageiros a bordo que vivem nessa mesma casa, todos devem ser deixados lá da mesma maneira). Não há a necessidade de recolocar um marcador para os passageiros deixados; assuma que eles estão indo direto para dentro de casa. Se o ônibus não está perto o suficiente de alguma casa de algum passageiro, faça um anúncio avisando isso. Qualquer anúncio deve sumir (ou ser substituído por um texto padrão), logo que o ônibus começar a se mover.

☐ Bom trabalho até agora! Basta dizer que esse serviço de ônibus está começando a se parecer com um jogo. É hora de deixar você usar um pouco da sua criatividade (pense nos seus dias de Scratch!). Implemente qualquer um (1) dos recursos abaixo.

- ☐ Implemente um sistema de pontuação, no qual o motorista ganha um ponto por cada passageiro deixado em casa. Certifique-se de anunciar a pontuação do motorista sempre que ela mudar. E certifique-se de anunciar quando todos os passageiros tiverem sido deixados em suas respectivas casas.

- ☐ Implemente um temporizador, segundo o qual o motorista só tem um certo número de minutos ou segundos, para deixar um certo número de passageiros em casa. É provável que você ache as funções `window.setInterval` e/ou `window.setTimeout` interessantes.[1,2]

- ☐ Ao invés de apenas listar os nomes e as casas dos passageiros sentados, agrupe-os de alguma forma por casa de modo que fique fácil de ver quantos dos seus passageiros estão destinados a uma casa qualquer.

- ☐ Implemente um botão que, ao ser apertado, dá ao ônibus a capacidade de "voar". Mas ele só pode pegar e deixar passageiros com as rodas no chão. Deixamos para você decidir que combinações de teclas usar para o vôo.

- ☐ Substitua o ônibus azul no mapa 2D por uma seta que aponta na direção que o ônibus está de fato apontando (na terra 3D). É provável que você queira girar um elemento `canvas` ou usar 360 ícones diferentes (um para cada grau). Se você for implementar essa última abordagem, você pode fazer isso com menos ícones, um a cada alguns poucos graus.

- ☐ Implemente a capacidade de teletransportar o ônibus instantaneamente para qualquer prédio no campus, através da seleção do nome da construção em um menu.

- ☐ Implemente a capacidade de aumentar ou diminuir a velocidade do ônibus.

- ☐ Faça com que os passageiros nunca sejam posicionados pseudoaleatoriamente por `populate` em suas próprias casas, para que o motorista não se irrite que por que alguns passageiros só querem viajar alguns metros.

- ☐ Implemente a capacidade fazer passeios de ônibus em um campus de uma outra universidade ou em sua cidade natal (então convide os seus amigos e sua família para passear com você!).

- ☐ Implemente a capacidade de informar aos passageiros, através de um anúncio, a posição atual do ônibus. Você deve achar a seguinte URL bem interessante:

  `http://code.google.com/apis/maps/documentation/geocoding/#ReverseGeocoding`

- ☐ Implemente um sistema de piloto automático. Quando um botão ou tecla for clicado, o piloto automático dirige (sem teletransportar) até um passageiro ou uma casa. Está tudo bem se o seu piloto automático gostar de dirigir através de prédios.

- ☐ Implemente um sistema de combustível, segundo o qual o ônibus começa com uma quantidade finita de combustível e só pode viajar um certo número de metros antes que ele se esgote. Construa um ou mais postos de gasolina no campus nos quais o ônibus pode

[1] `https://developer.mozilla.org/en/DOM/window.setInterval`
[2] `https://developer.mozilla.org/en/DOM/window.setTimeout`

reabastecer (clicando em um botão ou tecla quando estiver próximo, ou dirigindo através do posto).

☐ Invente o seu próprio recurso! Está tudo bem se você quiser implementar um recurso não listado aqui, mas que envolve um esforço similar. Seja criativo!

Embora você só precise implementar um dos recursos listados acima, você pode nos impressionar (e os seus amigos) com mais do que um "just for fun"! E você está convidado a alterar a estética da sua interface de usuário, incluindo os ícones do mapa 2D.[1]

Checando...

Antes de considerar esse Pset terminado, melhor perguntar a si mesmo algumas coisas e depois voltar e melhorar o seu código conforme o necessário! Não considere essa lista como uma lista de todas as nossas expectativas, mas apenas como alguns lembretes úteis. As caixas que vieram antes representam a lista completa! Para ser claro, considere as perguntas abaixo retóricas. Não há a necessidade de respondê-las por escrito para nós, pois todas as suas respostas devem ser "sim!"

☐ Você só pega passageiros que estão em um raio de 15 metros do ônibus?
☐ Você só deixa um passageiro se o ônibus está em um raio de 30 metros da sua casa?
☐ Se vários passageiros estão dentro de um desses raios falados, você lida com eles adequadamente?
☐ Você implementou pelo menos um recurso adicional?

Como sempre, se você não respondeu "sim" a um ou mais dessas perguntas porque você está tendo alguma dificuldade, mande um email para `ajuda@cc50.com.br`!

☐ Esse foi o Pset 8, o último!

[1] Dê uma olhada em `http://code.google.com/p/google-maps-icons/` que você deve achar alguns ícones divertidos.