6.MITx: Week 2 Capstone Project

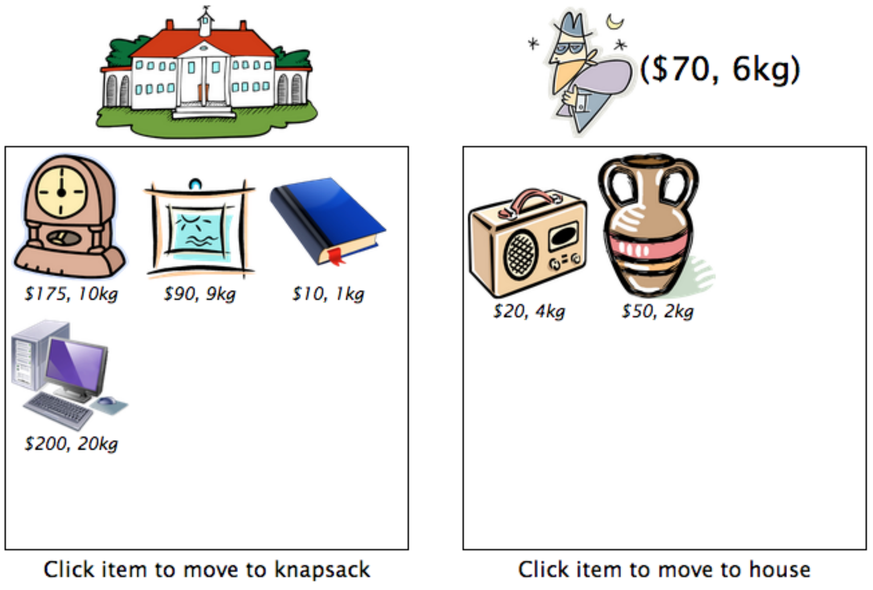
Deadline: Friday, 3p demo session

Whew! We're finally at the end of our two-week long bootcamp that gave you a taste of how to get things done in the browser environment. To wrap things up here's a small capstone project, which you can implement using any and all of the tools, libraries, techniques, etc. that we've discussed (plus any others that you've found on your own!). The goal is for <u>everyone</u> to have a demo ready to go at our demo session on Friday at 3p. You can work alone, or feel free to team up with others -- each team will make a single presentation on Friday.

We'd like you to build an interaction that demonstrates the knapsack problem: Given a set of items, each with a weight and a value, determine whether to include each item in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. Starting with the following DOM element in an HTML file

```
<div class="knapsack" data-maxweight="20">
  <img src="clock.png" data-value="175" data-weight="10" data-name="clock">
  <img src="painting.png" data-value="90" data-weight="9" data-name="painting">
  <img src="radio.png" data-value="20" data-weight="4" data-name="radio">
  <img src="vase.png" data-value="50" data-weight="2" data-name="vase">
  <img src="book.png" data-value="10" data-weight="1" data-name="book">
  <img src="computer.png" data-value="200" data-weight="20" data-name="computer">
</div>
```

create an interaction involving a burglar and a house full of items. It could look something like



Initially all the items are in the house. The user indicates which items should be moved to the knapsack (e.g., by clicking on them) and sees running totals of the knapsack's value and weight. Attempts to exceed the knapsack's capacity don't succeed, instead producing a tasteful alert explaining the problem. Note that the knapsack capacity and the available items (along with their weights and values) are specified in div.knapsack DOM element.

You can find a running version of the interaction at

http://web.mit.edu/6.mitx/www/demo/knapsack

Feel free to build a similar demo or develop your own scenario. You can grab the images from the demo if you want, but not the .js or .css -- these you should figure out yourself. Your demo should have at least the same functionality as our version; hopefully it will have more!

To-do list:

- have your code reviewed by someone else. We'll ask them to report on the results during the demo.

- use our basic pattern for knapsack.js: locate all the div.knapsack DOM elements and process them to add the appropriate interaction components. As always, JQuery will be very useful. We recommend a model-view-controller software architecture for your interaction code.

- use a knapsack.css file to control the layout of your visualization components.

- The visualizations for the items should include an image and a constructed caption that indicates their values and weights (provided by the initial dom.knapsack HTML).

- Figure how the user will select an item to include in the knapsack (clicking, dragging, …) and how selection will appear visually: an item "jumps" to their its location, or maybe there's a nicer, more complicated animation where it slides from its old location to its new one.

- Moving items into/out of the knapsack changes the running total of the knapsack's value and weight -- the visualization should display these to the user and keep them updated as the knapsack's contents change.

- A selection shouldn't be allowed to exceed the knapsack's capacity. The change in location should be disallowed and a tasteful alert shown to the user. Using the built-in Javascript alert() function would NOT be tasteful...

Extensions for fame and glory:

- capture the student's answers to the values achieved with the various selection strategies described in the staff's demo and send them to a server for grading. Indicate to the user whether the answers were right or wrong.

- use the bootstrap.js library to create an elegant look and feel.

- use the backbone.js library to build your model-view-controller code and to pass events around.

- use D3 to show a pie-chart visualization of the used and available knapsack capacity and/or a graph of knapsack value over time as items are added and removed.

- experiment with various ways to animate the selection/deslection of items for the knapsack. Can make it so that clicking on an item makes it move smoothly to its new location, e.g., by using JQuery's .animate() function?

- add a sound effect when the knapsack's capacity is exceeded, e.g., http://www.youtube.com/watch?v=iMpXAknykeg

- use localStorage() or parse.com to keep track of the interaction state and restore it when revisiting the page.

- wow us!