**Javascript WebCam Challenge**

**Summary**

Imagine, that you have access to a list of PTZ cameras around the globe and you can control them remotely via a simple JS API. The first thing what you would do is implementing a nice UI to let other folks control them as well, wouldn’t you?

**Your challenge**

Create remote controller web-application for cameras with your favorite JavaScript framework (Angular, Backbone, Ember, React, etc.) using the attached WebCam API (*webcam.js*).

Please build a web app with 2 tabs:

1. **Sources panel**: The panel will present the list of the available cameras. By selecting an item from the list, the camera simulator's source needs to be changed (using the setSource service of *WebCam* simulator).
2. **Control panel**: Here, the user can control the camera by moving and holding the mouse inside of a circle. Please draw a circle, and based on the mouse position, update the camera offset using the move service of *WebCam*. The *WebCam* has multiple statuses (e.g. position). The user can update the x,y offset by only moving his/her mouse. E.g: Where the user touches and holds the mouse inside the circle that is a *reference point*. If the user starts moving (and holding) the mouse, the offset is the difference between the new mouse position and the *reference point*, which is the first mouse positon. So, if the user touches and holds the mouse somewhere inside the circle without moving it, the offset is zero.

Cameras list can be requested from:  
http://runningios.com/screamingbox/cameras.json

**Requirements**

* Make sure to create a good software architecture (be modular, flexible, maintainable).
* Use the attached WebCam API to simulate a webcamera
* Use the best practices of the selected JS framework.
* Bonus points: App is mobile optimized and the camera can be controlled by dragging as well.

The task can be solved in 3 hours ideally, but the required time can be vary based on the chosen JS framework. So, the time is not a critical assessment criteria, the quality of the provided solution is much more important.

**Example Solution**

[Video](http://runningios.com/screamingbox/webcam_example.mov)

**Code Example**



**Webcam.js API Reference**

* **getCameraNode**. It returns a node of WebCam view and allows you to render it anywhere.

var cameraView = WebCam.getCameraNode();  
document.body.appendChild(cameraView);

* **setSource**. It sets camera source to WebCam. ([Available camera sources in JSON here](runningios.com/screamingbox/cameras.json))



* **move**. It moves the cameras based on the input offset. The camera has statuses, like position, so the input offset will be added to the current position (and will update it).
* Take 2 arguments:
  + **X** - X axis offset and
  + **Y** - Y axis offset.

// Moved the camera view down 30px and right 30px

WebCam.move(30, 30);