UDACITY

Classic Arcade Game Clone

REVIEW

CODE REVIEW 4

Meets Specifications

Hey , Congratulations ! You have passed after doing such a great work , Keep learning and stay Udacian

Game Functions

✓

The game functions correctly and runs error free

- **Player can not move off screen**
- **Vehicles cross the screen**
- **Vehicle-player collisions happen logically (not too early or too late)**
- **Vehicle-player collision resets the game**
- **Something happens when player wins**

Good job here!
To exceed specs you can implement any additional functionality:
multiple vehicle types
timed games
etc
The player can not move off screen ✅
Vehicles cross the screen ✅
Vehicle-player collisions happen logically (not too early or too late) ✅
Vehicle-player collision resets the game ✅
Something happens when player wins ✅
Nice work! 💪

Object-Oriented Code

✓

Game objects (player and vehicles) are implemented using JavaScript object-oriented programming features, and should encapsulate all properties and methods necessary for game functions.
For example, Player objects require `x`, `y`, and `sprite` properties. Enemy objects should have those three properties in addition to a `speed` property.

Excellent work using prototype methods and following object oriented principles.
I would suggest you to do some research on ES6 classes .
It would be great if you use Prototypal inheritance or normal inheritance to inherit your Enemy and Player class from a common superclass.

Documentation

✓

A `README` file is included detailing all steps required to successfully run the application.

✓

Comments are present and effectively explain longer code procedures. As a rule of thumb: describe what all custom functions and object methods do.

Good job! Some notes about comments.
Remember that there are always three places where you should write concise comments:
File header comment
Function header comment
Inline or above line comment

File header comment
You should write at least what the code in particular file should do. You can also write your name, the date and why you wrote the code.
Function header comment
This comment should provide information about the purpose of the function. You should include at least the required parameters (if any), the transformations, and the expected output
Inline (above line) comment
You should write this type of comment in any part of your code you feel that someone will not get what you are trying to achieve with commented code.
In term of comments style, you can use
JSDoc
YUIDoc
Docco
ESDoc
Dokker
or some other javascript documentation tool
Using comments containing JSDoc, programmers can add documentation describing the application programming interface of the code they're creating.
This is then processed, by various tools, to produce documentation in accessible formats like HTML and Rich Text Format.
Source
Finally, if you may require some inspiration or advice about what or how to write better comments, you can check the following blog post: http://www.hongkiat.com/blog/source-code-comment-styling-tips/
References
http://www.cs.utah.edu/~germain/PPS/Topics/commenting.html
http://stackoverflow.com/questions/6815903/what-is-the-correct-way-of-code-comments-in-javascript
https://www.thinkful.com/learn/javascript-best-practices-1/#Comment-as-Much-as-Needed-but-Not-More
https://github.com/airbnb/javascript#comments

Code is formatted with consistent, logical, and easy-to-read formatting as described in the Udacity JavaScript Style Guide.

Good job!
I recommend to use special services to validate your code on errors and warnings:
http://jshint.com/ (I prefer this one)
http://jslint.com/
http://eslint.org/
The best if you integrate them right into your text editor (e.g. sublime or atom) or use in your grunt/gulp workflow (installing instructions).
Also don't forget about code validation.
My favorite tools are:
HTML Validation: https://validator.w3.org/
CSS Validation: https://jigsaw.w3.org/css-validator/