# Candidate assignment
## Senior HTML5 game developer

The purpose of this test is to assess your technical suitability for the role of senior html5 game developer. You should aim to complete the test to the same level of quality that you would expect to achieve if you were working as a developer with us.

The test should take no more than four hours or an evening to complete. If you run out of time then please summarise what else you would have done had you had more time available.

You are free to use any open source libraries you think might be useful. You should ensure that your code is written in such a way that it can be easily maintained in the future. We value code which is readable, comprehensible, well-structured and modular. In short, be: DRY, KISS, SOLID. TDD would be a plus but it’s not a requirement.

Please note down which browser version(s) your code supports.

## Slot game

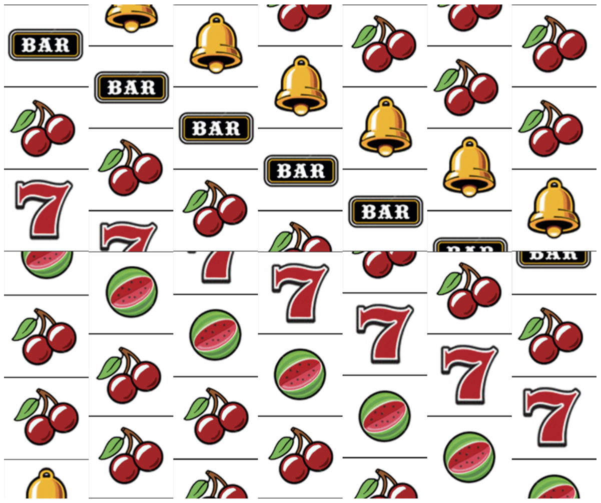
1. **Create a game that consists of two different scenes**
    a. The first one should be a welcome splash page with a “play” link/button.
    b. The second one should be a slot game with 3 reels. This slot machine should request each roll to the server via a defined REST-JSON API web service (use mock-ups to simulate the API response).

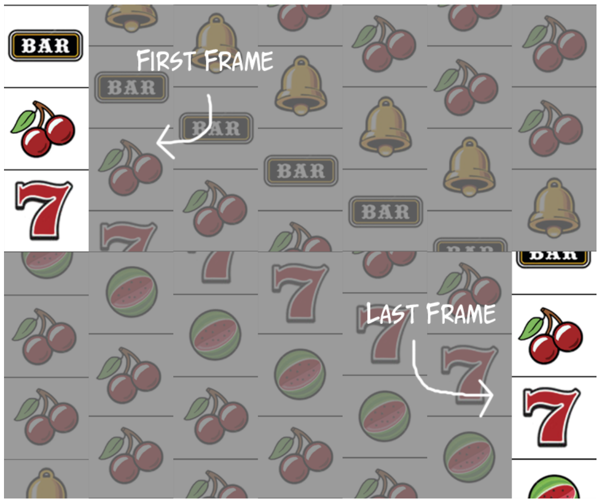
2. You are free to use any framework that you deem suitable for the job, but we are currently using CreateJS to develop HTML5 canvas based games.
3. We will value good code structure and following best practices according to platform over putting gradients on a button or some fancy images. We are looking for a very talented JavaScript developer. Act accordingly.
4. Generally, you are encouraged to use relevant frameworks to reduce the amount of code required. You are also encouraged to script the building of your software. Work smart, not hard.
5. Add a Readme.txt to your assignment with a short explanation and any further action

you would take to make this code production-ready performance-wise.

The reels are provided as a single sprite (with a help addendum). All the reels are the same and the elements are:

| | |
|---|---|
| Element 1 = SEVEN | (frame 13) |
| Element 2 = CHERRY | (frame 1) |
| Element 3 = BAR | (frame 3) |
| Element 4 = BELL | (frame 5) |
| Element 5 = CHERRY | (frame 7) |
| Element 6 = CHERRY | (frame 9) |
| Element 7 = WATERMELON | (frame 11) |





The API specifications are as follows (use mock-ups to simulate a response):

- Parameters:

    - **bets**: the number of bets per roll. Unless you are doing the bonus feature, this should be always 1.
- Response:
    - **reels**: the central position of each reel.
    - **prize**: the prize won.

Response examples:

```
{reels: [1,7,2], prize: 0}
```

This would mean that the central position of the first reel would be SEVEN (element 1, frame 13), the central position of the second reel would be WATERMELON (element 7, frame 11), and the central position of the third reel would be CHERRY (element 2, frame 1).

```
{reels: [1,1,1], prize: 500}
```

This would mean that the central position of the first reel would be SEVEN (element 1, frame 13), as would the second and third reels central position.

## Bonus features

Feel free to add features that you think increase the usability or performance significantly. You should assume that the application will run on a device with a 2.5G-like Internet connection.

Add a mechanism to bet more than 1 line per roll.

Show the prize accumulated.