

Required test items

1. HTML requirements

HTML requirements : html/body/head element requirements, the charset must be utf-8

Tips : HTML requirements

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <!-- do something -->
</head>
<body>
  <!-- do something -->
</body>
</html>
```

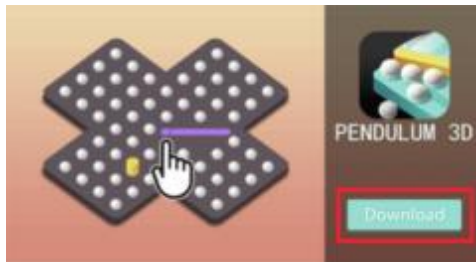
Tips : Viewpoint recommended

```
<meta name="viewport" content="width=device-width,user-scalable=no,initial-scale=1.0, minimum-scale=1.0,maximum-scale=1.0"/>
```

2. Called download button method

A CTA button must remain visible throughout the entire playable. All features that direct to App Store or any webpage must call API [window.install](#) && [window.install\(\)](#);
(Please Note: The playable should not initiate its own redirection to the App Store or webpage. All redirections must go through our interface.)

Creatives must support PC browser environments. Ensure basic interactions (e.g., swipe, click) can be triggered via mouse in PC simulation mode. (The detection tool runs in a browser—if your interaction only works on mobile (e.g., mouse dragging isn't supported), the process may not complete properly.)



Sample : Called download button method

Tips : Use the SDK, install

```
var downloadBtn = document.getElementById('download-btn');
downloadBtn.addEventListener('click', function (e) {
  window.install && window.install();
});
```

3. Called gameEnd method

At the end of the playable (when the playable wins or fails), you should call the API [window.gameEnd](#) && [window.gameEnd\(\)](#);

Creatives must support PC browser environments. Ensure basic interactions (e.g., swipe, click) can be triggered via mouse in PC simulation mode.(The detection tool runs in a browser—if your interaction only works on mobile (e.g., mouse dragging isn't supported), the process may not complete properly.)

4. Called gameReady method

All resources need to be loaded while playable initializing, once the loading completes must call API [window.gameReady](#) && [window.gameReady\(\)](#);

5. Called gameStart method

By exposing a global public function name "gameStart" in the playable, we will automatically call this function at the beginning of the playable,so that developers can handle some logic at the beginning of the playable, such as starting the countdown, starting the background music, etc.

Tips: Use the SDK, gameStart

```
function gameStart() { // do something }
```

6. Called gameRetry method

If the playable is designed to play again, please call "window.gameRetry()" method when play again is initiated. If not, you needn't add the method in palayable.
[window.gameRetry](#) && [window.gameRetry\(\)](#);

7. Called gameClose method

By Exposing a global public function name " gameClose" in the playable, we will automatically call this function at the end of the playable, so that developers can handle some logic at the end of the playable, such as turn off this background music, etc.

Tips: Use the SDK, gameClose

```
function gameClose() { // do something }
```

8. Close button handling method

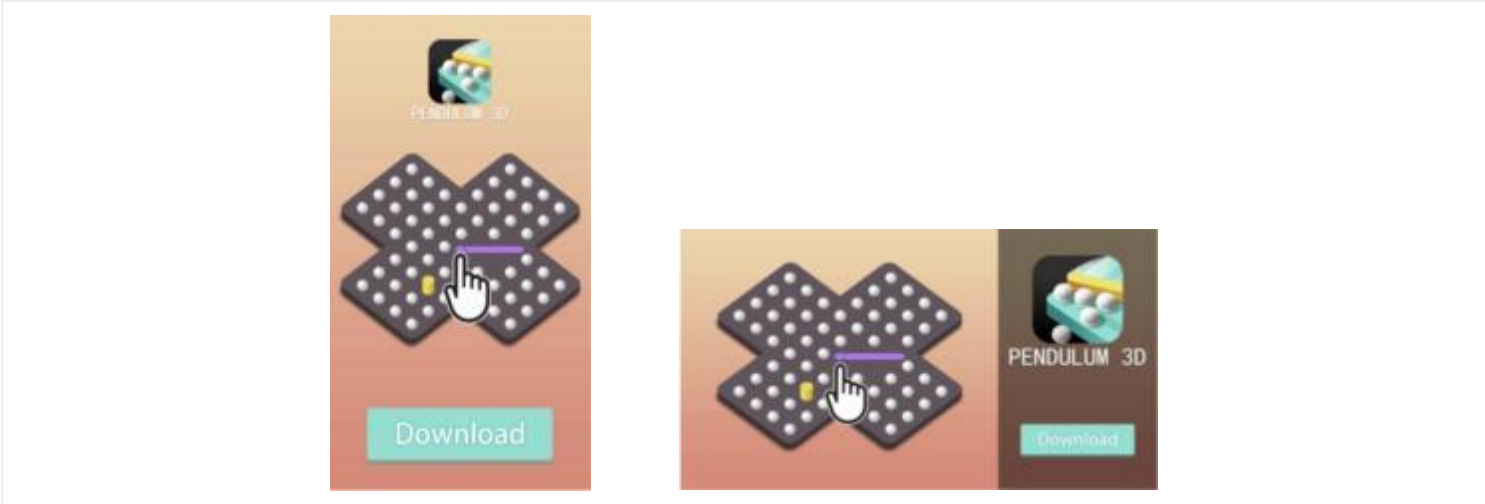
We will handle the close button of the playable, and please do not add it yourself.

9. Loading page handling method

We will add a LOADING page, please do not add it yourself.

10. Playable adaptation handling method

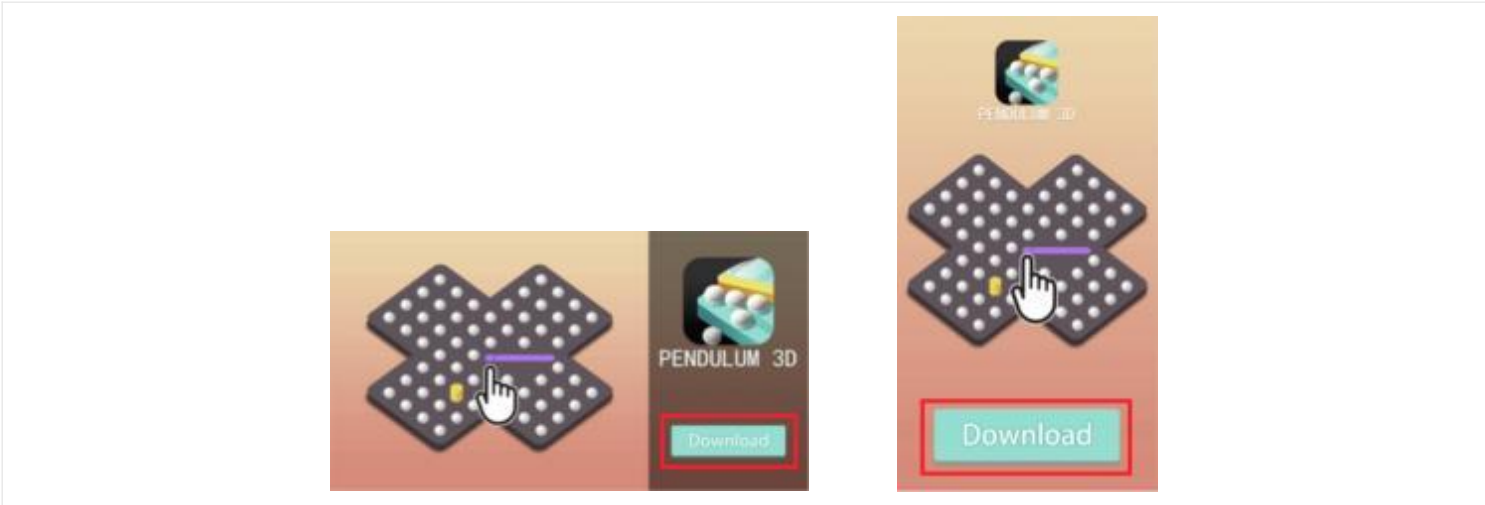
The playable must can be adapted to both landscape and portrait mode, and supported for free rotation.



Sample: Screen adaptation of different devices, horizontal or vertical screen views

11 . Interactions needed when playable is won or lost

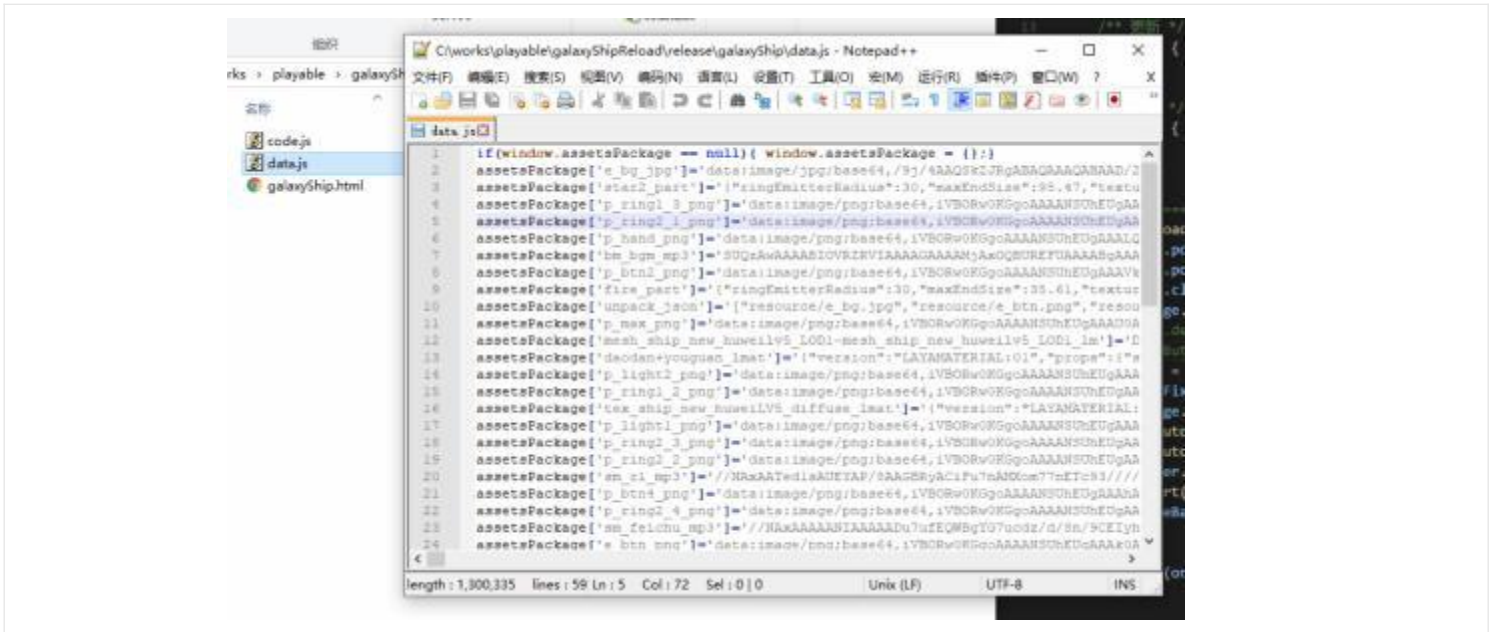
The download button of the playable winning or losing page must have the prompt for the user to click it by some kinds of effect.



Sample: Playable End

12. File handling method

All files besides JS and HTML should be processed into base64.Before export, please remove all unused or redundant libraries and references (e.g., debugging tools or dev plugins from the engine). Invalid references may cause loading issues or compatibility problems that affect detection.



Sample : base64

13. File specification

Zip file of the Playable should not be larger than 5M in size. All file names are only allowed letters, numbers, and underscores. Name of Zip File, Assets Folder, HTML, must be the same. The HTML file inside the ZIP needs to be openable locally.

14. Storage requirements for assets dependent on codes

Please put the resources which the playable depends on locally in the folder, and make sure that there is no dynamical request for that. Before export, please remove all unused or redundant libraries and references (e.g., debugging tools or dev plugins from the engine). Invalid references may cause loading issues or compatibility problems that affect detection.

15. Do not override global console method

This may cause runtime issues, breaking core features like redirect and gameClose.

16. Avoid auto-redirects without user actions

Mintegral blocks auto-redirects (e.g., redirecting right after load or with a delayed timer).

Note: This behavior is not currently detectable by the testing tool.

Optional

17. We have a method to report the buried point of the playable. The usage is as follows:

Make a description file named action.json at first, placed in the root directory, the content is a description of each buried point, we can use 5 buried points at most, as

shown in Figure below.

Then call the method by `window.HttpAPI()` when we need to report, such as `window.HttpAPI && window.HttpAPI.sendPoint("action&action="`

`+ action");` The first parameter action is a number, which could be 1 2 3 4 5, indicating the corresponding buried point.

action.json	
1	"action1": "description"
2	"action2": "description"
3	"action3": "description"
4	"action4": "description"

Specification

QA process

The QA process normally takes 3-5 business days. Plz refer to Mintegral AM if there's any question about the process.