What good a game is without the use of rules of physics? So this tutorial will be focused on giving a basic introduction to physics API in *phaser.js*.

We'll start with the directory structure in our code.

```
|
|_ assets
|  |_ images
|      |_ sky.png
|      |_ bubble.png
|
|_ js
|  |_ phaser.min.js
|
|
|_ index.html
```

Same as our previous "Hello World" example, we start by pasting the following code inside *index.html* file.

```
<!doctype html>
<html lang="en">
        <head>
                <meta charset="UTF-8" />
                <title>Phaser Tutorial</title>
                <script type="text/javascript"
src="js/phaser.min.js"></script>
        </head>
        <body>

                <script type="text/javascript">

                var HEIGHT = 400;
                var WIDTH = 400;

                var game = new Phaser.Game(WIDTH, HEIGHT, Phaser.AUTO,
'hello-world-example', { preload: preload, create: create, update:
update });
```

```
                function preload() {
                        game.load.image('background',
'assets/images/sky.png');
                }

                function create() {
                        background = game.add.sprite(0, 0,
'background');
                        background.scale.setTo(4,4);
                }

                function update() {
                }

                </script>

        </body>
</html>
```

We just initialized our game object with a background image.

Now we'll add a 'bubble' object in our game and add gravity to it. So replace the *preload* and *create* function with the following piece of code.

```
function preload() {
        game.load.image('background', 'assets/images/sky.png');
        game.load.image('bubble-img', 'assets/images/bubble.png');
}

function create() {
        game.physics.startSystem(Phaser.Physics.ARCADE);
        background = game.add.sprite(0, 0, 'background');
        background.scale.setTo(4,4);
        bubble = game.add.sprite(WIDTH / 2, 0, 'bubble-img');
        bubble.scale.setTo(0.3, 0.3);
        game.physics.enable(bubble, Phaser.Physics.ARCADE);
        bubble.body.gravity.y = 200;
}
```

Line #3 loads an image resource for our 'bubble' object.

Line #7 enables the ARCADE physics in our game.

Line #12-#13 adds the 'bubble' object inside our game screen and do necessary scaling.

Line #15-#16 enables the physics rule on our 'bubble' object and adds gravity to it.

Now if we run the code we'll see an animation of a dropping bubble which will pass through our game screen from up to down.

Add the following code to make the bubble stop when it hits our game world boundary.

```
bubble.body.collideWorldBounds = true;
```

Now add bounce property to our bubble object and see the output. Also modify the parameter value of *set()* function and see the difference. Try with value bigger than *1* and smaller than *1*.

```
bubble.body.bounce.set(1);
```

Add velocity to both horizontal and vertical direction to our bubble object and check out the effect.

```
bubble.body.velocity.setTo(200,200);
```

Here we are added a horizontal velocity of 200 and vertical velocity 200.

So this is all for this tutorial. Try it out and play with it, change different values and try to understand how they work. The code is available in github.