

An illustration of a desk setup. On the left, a dog with a striped hat sits on a patterned rug. Behind it is a shelf with several books and a parrot. In the center, a laptop displays the title 'Security Training Game Results' and a bar chart with three bars of increasing height. A hand is shown typing on the laptop keyboard. To the right of the keyboard is a small notepad with the text 'FEED ERNEST!'.

Security Training Game Results

FEED
ERNEST!



THE EXPLORER'S
ENCYCLOPEDIA
The Guide to Every Adventure

Home page
Contents
Featured discoveries
Famous explorers
Historical expeditions

The House of Volkov's Security Team

From the Explorer's Encyclopedia, the guide to every adventure

This is a private entry with restricted access. To view the public entry for the House of Volkov click here.

The **House of Volkov's Security Team** is responsible for protecting the highly valuable jewels on display in the House of Volkov's private collection. Since the theft of the Monk Diamond, the jewels on display in the shop are kept in locked cases. The reinforced glass is 200 times stronger than normal glass.



The security Training Game Rules

- ◆ Every second, 6 people will be shown on-screen
- ◆ 5 will be guests
1 will be a thief
- ◆ If you click on the thief,
you gain 1 point
- ◆ If you click on a guest,
you lose 2 points
- ◆ This happens 6 times in total
- ◆ The aim of the game is to spot
all 6 thieves and score 6 points

In light of the theft, the owner Viktor Volkov has trained his security team to be on the alert for any customers behaving suspiciously. In the last year several petty crooks have been caught trying to unlock the display cases with skeleton keys.

The special exhibition Mr Volkov is hosting for the Monk Diamond will be the most high-profile event ever held by the House of Volkov. Mr Volkov decided to consult Professor Bairstone about the security strategy for the opening night of the exhibition.

The Bond Brothers have used disguises in past robberies and Mr Volkov and Professor Bairstone are worried that members of the gang, or their associates, might try to infiltrate the exhibition by posing as guests.

The agreed strategy is to use the method developed for museums in London by Professor Li. A computer game with a simple set of rules has proved to be an effective way of sharpening the reaction times of members of a security team.

The game needs to be built urgently so the team are properly trained before the opening of the special exhibition.

BUILDING A GAME

Now you've read the brief for Mission 5 you're ready to start building your game. This mission works in a slightly different way to the missions you have completed so far. You are going to build the game as you work through the mission. Follow the step-by-step instructions and copy the code and you'll have the game ready for Professor Bairstone in no time.

It's going to be a rush to get the game built in time, so start coding quickly!



1. create a HTML file

Just like in all the other missions, the first thing we need to do is create a new HTML file. Call your new HTML file **securitygame.html**. Copy this code into your text-editing program:

```
<!DOCTYPE html>
<html>
<head>
  <title>Security Game</title>
</head>
<body>
</body>
</html>
```



Save your HTML file in your **Coding** folder on your desktop.

2. Build the game board

Now we need to build the basic structure of our game. We need to code a game board in our web page. The game board will be the area in our browser where the game will work. When the security team play the game, the guests and thief will appear on the game board.

Add your game board by coding an empty `<div>` in the `<body>` of your page. Add an id attribute to your `<div>`. Your `<body>` code will look like this:

```
<body>
  <div id="board">
  </div>
</body>
```

Id attribute

Then in your `<head>` add a CSS class that will change the look of your `<div>`. Use a CSS selector, called the id selector, to find your `<div>` by its id attribute. CSS selectors are an easy way to select various groups of elements to style. Using the CSS id selector is simple. All you have to do is create a class name using a hash (#) and the id attribute of the HTML element you want to change.

Create a CSS class called `board` that sets the CSS properties and values of your `<div>` to these things. The code for the CSS class in your `<head>` needs to look like this:

```
<head>
<title>Security Game</title>
<style>
  #board {
    border: 1px solid black;
    background-color: gray;
    height: 350px;
    width: 650px;
  }
</style>
</head>
```

CSS id
selector

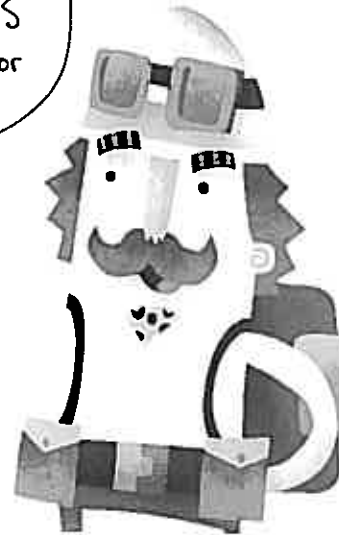


Save your HTML file and open it in your browser. You will see your empty game board displayed on-screen.

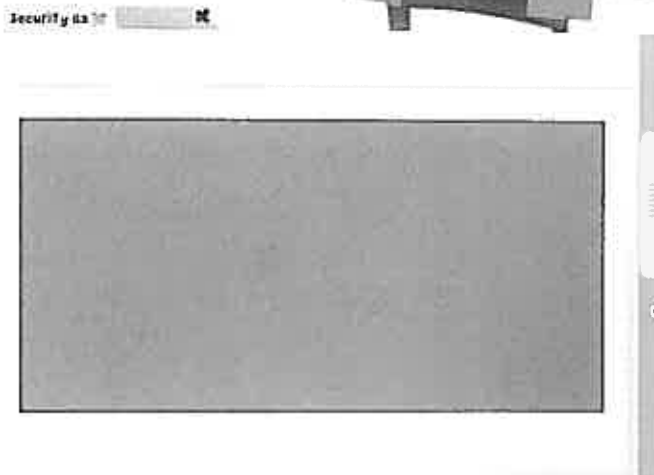
The game board needs:

- ♦ A 1px solid black border
- ♦ A grey background
- ♦ A height of 350px
- ♦ A width of 650px

This is like the CSS element selector in Mission 1 or the CSS type attribute selector in Mission 3.



Great work!



3. Add a button

Now we have a game board, we should add a button to our page. When the player clicks the button, the code for our game will run and the game will start.

Add your button above the `<div>` in your `<body>`. Create the button like you did in Mission 3, using the `<input/>` tag and type and value attributes. Set the `onclick` attribute of your `<input/>` tag to call a JavaScript function called `startGame` that will start the game. The code for your button will look like this:

```
<input type="button" value="Play" onclick="startGame()"/>
```

Now create the `startGame` function that your button will call when it's clicked. Add the function underneath the `<div>` in your `<body>`. Your `<script>` block will look like this:

```
<script>
  function startGame() {
  }
</script>
```

The complete code block in your text-editing program will now look like this:

```
<!DOCTYPE html>
<html>
<head>
  <title>Security Game</title>
  <style>
    #board {
      border: 1px solid black;
      background-color: gray;
      height: 350px;
      width: 650px;
    }
  </style>
</head>
<body>
  <input type="button" value="Play" onclick="startGame()"/>
  <div id="board">
  </div>
  <script>
    function startGame() {
    }
  </script>
</body>
</html>
```

Button

Function call

JavaScript function

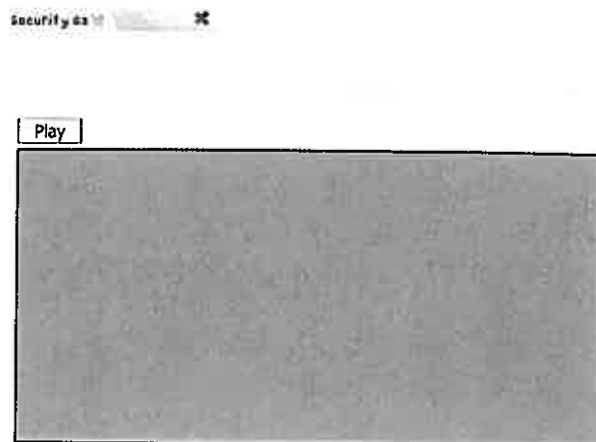
You're becoming
a real coding
whiz!





Save your code and open it in your browser. You will see your button on-screen. At the moment when you click it, nothing will happen. We need to add some code to our startGame function.

Let's get that button working!



4. create a JavaScript timer

Our game is going to test the reaction times of the security team at the House of Volkov. For our game to work, we need to learn how we can use JavaScript to run a piece of code again and again, after a certain period of time has passed. This is called a timer.

JavaScript has a built-in timer function called `setTimeout`, which allows you to call a function after a given amount of time has passed. All you have to do is give the `setTimeout` function the name of the function you want to call and the amount of time as arguments. As you know from Mission 2, you pass a function an argument by putting the argument in brackets. This time we are passing the function two arguments. Let's take a look at how you would use `setTimeout` to call a function after a certain time.

In this example the `gameTimer` function will be called after 1 second, or 1,000 milliseconds.

`setTimeout`
function

Function to call

Amount of time to
call it after

`setTimeout(gameTimer, 1000);`

Did you notice?

You have to use milliseconds with the `setTimeout` function. There are 1,000 milliseconds in a second. To work out the number of milliseconds you need, multiply the number of seconds by 1,000. So if you wanted your function to be called after 3 seconds, you would multiply 3 by 1,000.

Turn over to
practise using
the `setTimeout`
function!



CODE SKILLS

► USING THE SETTIMEOUT FUNCTION

JavaScript timers, such as the `setTimeout` function, are very useful for building games. Let's have a go at using the `setTimeout` function, so you know how it works and can add it to your game later in the mission. Let's code a program that counts up a number every second.

1. Open up your text-editing program.

Create a new HTML file called **timers.html**. Type out this code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Timers</title>
</head>
<body>
  <div id="number">
  </div>
</body>
</html>
```

2. Now add the `<script>` tag to your `<head>`. Inside your `<script>` block create a variable and set its value to 0 using the assignment (`=`) operator, like this:

```
<head>
  <title>Timers</title>
  <script>
    var count = 0;
  </script>
</head>
```

3. Now create a function called `updateCount`. Every time the function is called it will use the addition operator (`+`) to add 1 to the value of your variable. Then use `getElementById` and `innerHTML` to find your empty `<div>` and set the contents to the variable. Your code will look like this:

```
<script>
  var count = 0;
  function updateCount() {
    count = count + 1;
    document.getElementById("number").innerHTML = count;
  }
</script>
```

Add 1

Update screen

Find <div>

4. Now we need to add a function call to our `<body>`. Add the function call like this:

```
<body>
  <div id="number">
  </div>
  <script>
    updateCount();
  </script>
</body>
```

5. Finally we have to add a timer that will call our function every second to our `<script>` block. We need to use the `setTimeout` function. We have to pass the function name and the number of milliseconds to the `setTimeout` function as arguments. Our `<script>` block will look like this:

```
<script>
  var count = 0;
  function updateCount() {
    count = count + 1;
    document.getElementById("number").innerHTML = count;
    setTimeout(updateCount, 1000);
  }
</script>
```

Call updateCount
in 1 second



Save your HTML file and open it in your browser. You will see your timer in action. The `setTimeout` function will call your `updateCount` function every second. The function will run and every second 1 will be added to the value of your variable. The number in your browser will update automatically.

Timers

20

That's rather clever.
I'm impressed!

How are we going
to use this skill to
build the game?

Timers

21

5. create a game loop

Games are one of the most difficult things you can code and there are many different ways you can build them. A popular way is to use a game loop. You used a loop in Mission 3 and we are going to need to learn more about them to build our game.

A game loop is a JavaScript function that gets called over and over again while your game is running. You can use a game loop to check if a

player has done something, draw a HTML element on your screen and run the code for the game.

We can use the `setTimeout` built-in function to create a game loop. We need to add a new function to our `<script>` block that pops up an alert every 3 seconds. Let's call this function `gameLoop`. This new function will be called when our button is clicked. Let's take a look at our code block:

```
<!DOCTYPE html>
<html>
<head>
  <title>Security Game</title>
  <style>
    #board {
      border: 1px solid black;
      background-color: gray;
      height: 350px;
      width: 650px;
    }
  </style>
</head>
<body>
  <input type="button" value="Play" onclick="startGame()"/>
  <div id="board">
  </div>
  <script>
    function startGame() {
      gameLoop();
    }
    function gameLoop() {
      alert("Game over!");
      setTimeout(gameLoop, 3000);
    }
  </script>
</body>
</html>
```

Function

Function call

Alert

Timer

Number of
milliseconds

I wonder if I could sniff
out the Bond Brothers?
I've got a very good nose.



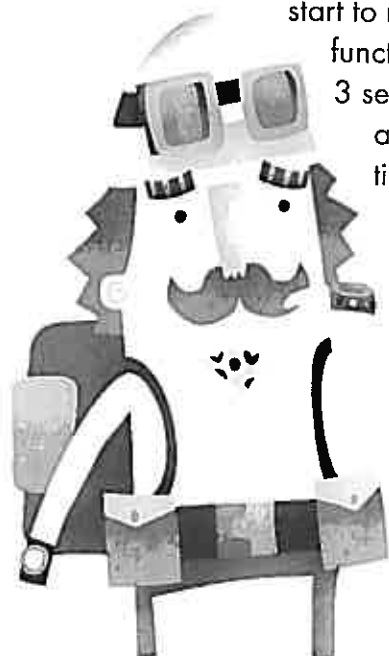
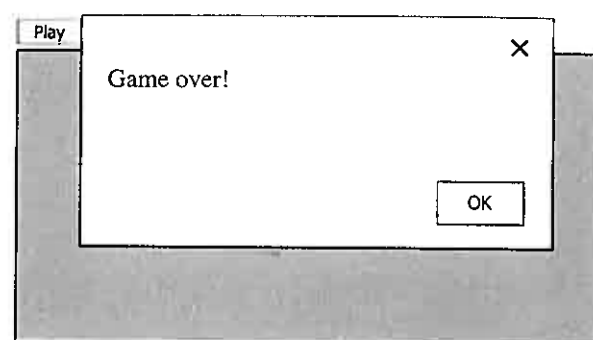


Once you have made these changes, save your code and open it in your browser.

When you click your button and call your startGame function, the game loop will

start to run. Your gameLoop function will be called every 3 seconds and every time an alert will pop up. Every time it pops up, press OK so it keeps popping up.

Security 55



Great start to the mission! But where's the thief?



6. Add the guests and thief to the game board

Now you've coded your game board, button and game loop, we need to add the characters to our game. We want six different characters to flash on-screen in different places every second. Five of the characters will be guests; one character will be a thief. The security team will test their reaction times by trying to click on the thief when he flashes on-screen.

Let's start creating the characters on our game board. Each character needs its own `<div>`. Create six `<div>` tags nested inside your game board `<div>` and number them 1 to 6, like this:

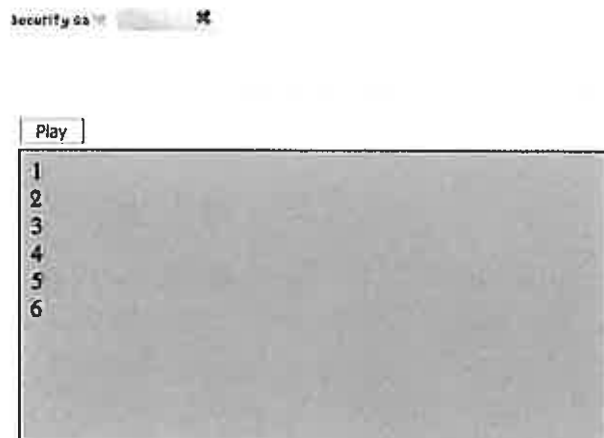
Let's get these characters coded!



```
<div id="board">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>
```



Save your code and refresh your page. You will see your six <div> tags on your game board, like this:



Now let's use CSS to change the design and layout of our <div> tags. Let's create a CSS class called character that will make our <div> tags square blue boxes. In your text-editing program, add this new CSS class to the <style> tag in your <head>. Your complete <style> block will now look like this:

```
<!DOCTYPE html>
<html>
<head>
  <title>Security Game</title>
  <style>
    #board {
      border: 1px solid black;
      background-color: gray;
      height: 350px;
      width: 650px;
    }
    .character {
      background-color: lightblue;
      width: 120px;
      height: 120px;
      padding: 10px;
      margin: 10px;
      float: left;
    }
  </style>
</head>
```

New CSS
class

Did you notice?

We have used the float CSS property. This will make our <div> tags align with one another.



Then apply your new character CSS class to each of your six `<div>` tags using the class attribute, like you did in Mission 1. Your `<body>` will look like this:

```
<body>
  <input type="button" value="Play" onclick="startGame()"/>
  <div id="board">
    <div class="character">1</div>
    <div class="character">2</div>
    <div class="character">3</div>
    <div class="character">4</div>
    <div class="character">5</div>
    <div class="character">6</div>
  </div>
  <script>
    function startGame() {
      gameLoop();
    }
    function gameLoop() {
      alert("Game over!");
      setTimeout(gameLoop, 3000);
    }
  </script>
</body>
</html>
```

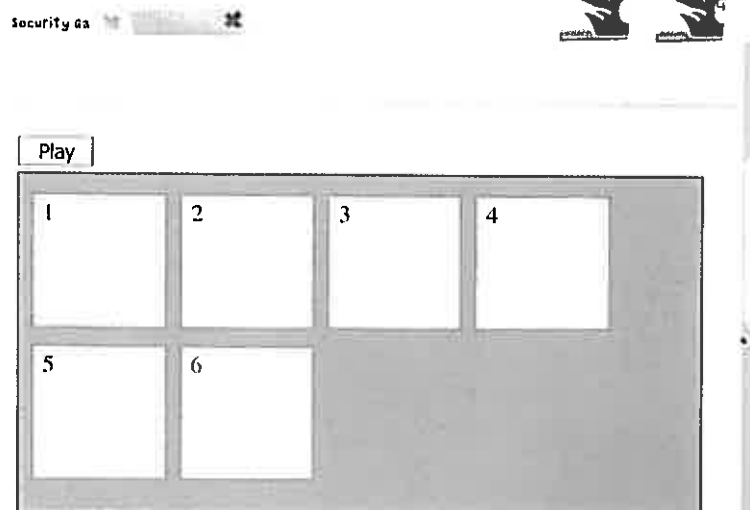
Class attribute



Save your code and refresh your page. You will see the CSS properties applied to your `<div>` tags.



Something's making my nose itch.



7. Use the game loop to stop the game

Now we've added our characters to the game board, we need to make some changes to our game loop. At the moment it pops up an alert every .3 seconds. We need to change our loop so it will stop after a certain period of time. We're going to do this by keeping count of how many times we loop. When our loop has run a certain number of times, our game will end.

To keep count of our loops, let's create a variable that will increase in value every time it's called. To do that we should code a variable called `loops` that is set to the value 0. Add this variable to your `<script>` block before your `gameLoop` function, like this:

```
<script>
  function startGame() {
    gameLoop();
  }
  var loops = 0;
  function gameLoop() {
    alert("Game over!");
    setTimeout(gameLoop, 3000);
  }
</script>
```

Variable

We need to use this variable in our `gameLoop` function. We want to add 1 to its value every time the function is called. To do this, we could code like this:

```
loops = loops + 1;
```

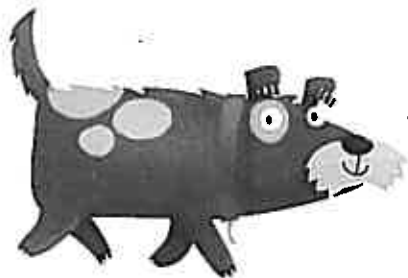
However we can actually use a new JavaScript operator to write the same instruction in a simpler and shorter way. It's called the increment operator (`++`) and it's just like the operators we learnt about in Mission 2. We can use the increment operator (`++`) to add 1 to the value of our variable. All we have to write is this:

Increment
operator

```
loops++;
```

We can use the increment operator (`++`) in our `gameLoop` function to count how many times our loop has been called. Remove the alert from your `gameLoop` function and replace it with your new piece of code, like this:

```
function gameLoop() {
  loops++;
  setTimeout(gameLoop, 3000);
}
```



This gives me paws
for thought!



Now we want to make sure that our game loops for a fixed number of times before the game ends. We do this by adding an if statement and an else statement to our gameLoop function, which will check how many times our loop has run.

We need to make sure that when the loop has run for a certain number of times, the setTimeout function calling the loop will stop and an alert will pop up. We want our loop to run 12 times. It will run every 3 seconds until the twelfth loop, when the game will end. Our game will last for a total of 33 seconds (11 multiplied by 3) before it ends on the twelfth loop. Let's have a look at our whole <script> block with the new variable, if statement and else statement:

```
<script>
  function startGame() {
    gameLoop();
  }
  var loops = 0;
  function gameLoop() {
    loops++;
    if(loops < 12) {
      setTimeout(gameLoop, 3000);
    }
    else {
      alert("Game over!");
    }
  }
}</script>
```

Labels in the original image: "If statement" points to the if block, "Else statement" points to the else block, and "Timer" points to the setTimeout function.

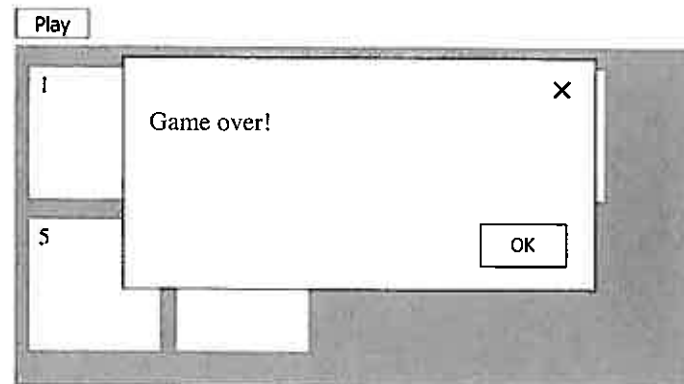
The if statement checks how many times our loop has run. If it is less than (<) 12 times, it keeps running by calling setTimeout again. If it is more than 12 times, the else statement runs and an alert pops up.



Save your new code and refresh your page. Press the play button and wait until the alert pops up.

Your game will now end after a certain period of time.

Security Ga



Don't forget you have to wait for 33 seconds before the alert will pop up!

8. Use CSS to show and hide the characters

During our game the guest and thief characters will flash up on-screen. The House of Volkov security team will have to spot and click on the thief before he disappears. We need to use CSS to make the characters appear and disappear as the game loop runs. To do this we have to learn a new CSS property that makes a HTML element appear or disappear from our browser. We can then use JavaScript to apply the property to our characters.

CSS property name	What does it do?	Example values
display	Changes how a HTML element appears on-screen	block; none;

The display CSS property has lots of different values. The two values we need for our game are "block" and "none". If you set the display property of a HTML element to none, the element will not appear on your screen. If you set the display property to block, the HTML element will display as a square.

In your text-editing program, add two new CSS classes called hidden and visible. Use the display property so your `<style>` block looks like this:



Save your code.



```
<style>
  #board {
    border: 1px solid black;
    background-color: gray;
    height: 350px;
    width: 650px;
  }
  .character {
    background-color: lightblue;
    width: 120px;
    height: 120px;
    padding: 10px;
    margin: 10px;
    float: left;
  }
  .hidden {
    display: none;
  }
  .visible {
    display: block;
  }
</style>
```


Display CSS
property

Let's have a quick
go at using this
new CSS property!

CODE SKILLS

► USING THE DISPLAY CSS PROPERTY

Try using the CSS display property to make HTML elements appear and disappear on your screen. You'll need this to make your characters display correctly in your game.

1. Open up your text-editing program. Create a new HTML file called **display.html**. Type out the code below, then save your HTML file and open it in your browser. Your page will look like this: 

```
<!DOCTYPE html>
<html>
<head>
  <title>Display</title>
</head>
<body>
  <div>Security</div>
  <div>Thief</div>
  <div>Guest</div>
</body>
</html>
```

display

Security
Thief
Guest

2. Reopen your file in your text-editing program. Modify your second `<div>` by adding the display CSS property, so it looks like this:

```
<body>
  <div>Security</div>
  <div style="display: none;">Thief</div>
  <div>Guest</div>
</body>
```

3. Save your file and refresh your page. The display CSS property and none value will make your second `<div>` disappear from your screen.



display

Security
Guest

Where did that
element go?



9. Animate the characters

We now need to apply our two new CSS classes (hidden and visible) to our game loop so the character `<div>` tags become animated. To do this we need our game loop to add and remove the new CSS classes to and from our character `<div>` tags.

- When our game runs, we want our characters to flash on and off the screen. We're currently looping around 12 times, so the character `<div>` tags need to be visible for 6 loops and invisible for 6 loops to create the flashing effect. The first thing we need is a new variable called `peopleVisible` to store this information. We need to set this variable to the value `false`, like this:

```
var peopleVisible = false
```

We need our variable to be set to `false` at the start so that the character `<div>` tags start off hidden on our screen. Then when the variable is set to `true`, the characters will appear on our screen.

When the game loop runs, we want the value stored in our new variable to alternate between `true` and `false`. To do this we have to use another JavaScript operator, called the not operator (`!`). You can use the not operator (`!`) to change the value of your variable.

Let's have a look at how we use the not operator (`!`) in our `<script>` block.



Type these new pieces of code into your text-editing program and save your file.

```
<script>
  function startGame() {
    gameLoop();
  }
  var loops = 0;
  var peopleVisible = false;
  function gameLoop() {
    peopleVisible = !peopleVisible;
    loops++;
    if(loops < 12) {
      setTimeout(gameLoop, 3000);
    }
    else {
      alert("Game over!");
    }
  }
}</script>
```

We have used the not operator (!) to alternate the value of our new variable. If the value of our variable is false, when the variable runs in our gameLoop function the not operator (!) will change its value to true. And if the value of our variable is true, when the variable runs the not operator (!) will change its value to false.

Every time our loop runs, the loops variable will go up by one and the peopleVisible variable will alternate between true and false. As we loop round, our variables will look like this:



Loop number	Variable value
1	True
2	False
3	True
4	False
5	True
6	False
7	True
8	False
9	True
10	False
11	True
12	False
Game over	

Now we need to add a new function called flashCharacters to our <script> block that will make our character <div> tags appear on-screen when the variable value is true. We do this using our two new CSS classes. When the value is true we want to use the visible CSS class, and when the value is false we want to use the hidden CSS class. Turn over to find out exactly how to code this flashCharacters function.



This new function sounds complicated!

I like using loops!



10. Apply CSS using JavaScript

We need a new function that will apply a CSS class to the character <div> tags depending on the value of our variable. We should call this function `flashCharacters`. The function must find our game board, using `getElementById`, and store it in a variable called `board`. Then the function needs to pick a CSS class to apply to the character <div> tags. This will depend on the value of our `peopleVisible` variable.

To decide the CSS classes we create a second variable called `classToSet` that will store the two names of the CSS classes we need to set. We choose the CSS classes using an if and else statement. We set the value of our `classToSet`

variable to an **empty string**. This empty string is filled as the if and else statements run. If our `peopleVisible` variable is true, our `classToSet` variable will be set to the visible CSS class. If our `peopleVisible` variable is false, the else statement will change our `classToSet` variable to the hidden CSS class.

We then need to code a loop which counts the 6 character <div> tags. As the loop counts the <div> tags it assigns the value of our `classToSet` variable to each <div>. We do this using the `className` method. The `className` method lets you set a CSS class on a HTML element in JavaScript.

Our complete function now looks like this:

```
function flashCharacters() {  
  var board = document.getElementById("board");  
  var classToSet = "";  
  if(peopleVisible) {  
    classToSet = "character visible";  
  }  
  else {  
    classToSet = "character hidden";  
  }  
  for(var index = 0; index < 6; index++) {  
    board.children[index].className = classToSet;  
  }  
}
```

Did you notice?

In the variable called `index` inside our loop we have stored more than one piece of information. This kind of variable is called a collection. We have to use square brackets `[]` to get a value out of the collection to use in our code.

CODE WORDS

An **EMPTY STRING** is a piece of data, in the form of a sequence of characters, that has the value of 0. In JavaScript an empty string is represented as `" "`.

11. Simplify the conditional statements

This function looks quite complicated, so we should now try to simplify it. You can use another operator in JavaScript, called a conditional operator (?), to simplify the if and else statements. You use the conditional operator (?) like this:

Conditional operator

```
var variableName = condition ? value1 : value2;
```

This tells our browser that if the condition of our variable is true it should use value1, and if it is false use value2. So we can rewrite the code in our function like this:

Conditional operator

CSS class name

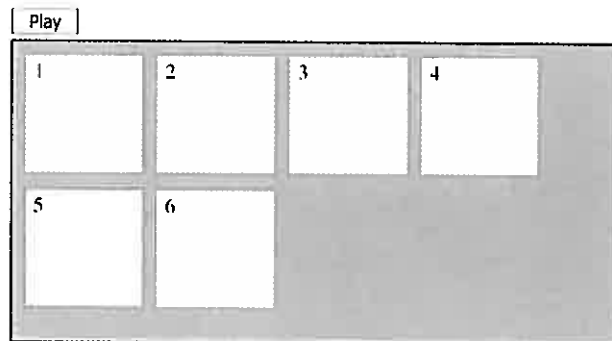
CSS class name

```
var classToSet = peopleVisible ? "character visible" : "character hidden";
```

Add your new flashCharacters function with the simplified if and else statements to your <script> block. Make sure your gameLoop function calls your new function too. Save your code and open it in your browser. Every 3 seconds your blue boxes will appear and disappear on your screen.

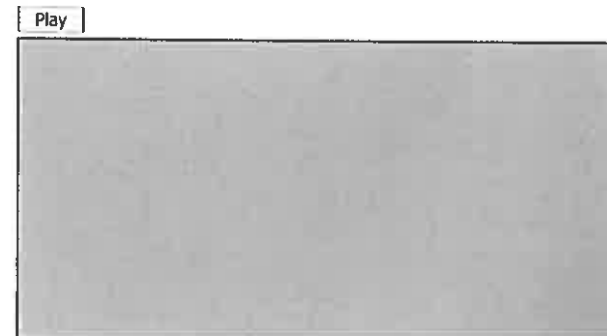


Security G



Turn over to see what our whole code block now looks like.

Security G





Check your code in your text-editing program. The code you need for your game will look like the block below. You have to refresh your page between each play of the game. Make any changes you need and save your file.

Don't forget you can view the code on the *Get Coding!* website too!



```
<!DOCTYPE html>
<html>
<head>
  <title>Security Game</title>
  <style>
    #board {
      border: 1px solid black;
      background-color: gray;
      height: 350px;
      width: 650px;
    }
    .character {
      background-color: lightblue;
      width: 120px;
      height: 120px;
      padding: 10px;
      margin: 10px;
      float: left;
    }
    .hidden {
      display: none;
    }
    .visible {
      display: block;
    }
  </style>
</head>
```

```

<body>
  <input type="button" value="Play" onclick="startGame()"/>
  <div id="board">
    <div class="character">1</div>
    <div class="character">2</div>
    <div class="character">3</div>
    <div class="character">4</div>
    <div class="character">5</div>
    <div class="character">6</div>
  </div>
  <script>
    function startGame() {
      gameLoop();
    }
    var loops = 0;
    var peopleVisible = false;
    function gameLoop() {
      peopleVisible = !peopleVisible;
      flashCharacters();
      loops++;
      if(loops < 12) {
        setTimeout(gameLoop, 3000);
      }
      else {
        alert("Game over!");
      }
    }
    function flashCharacters() {
      var board = document.getElementById("board");
      var classToSet = peopleVisible ? "character visible" : "character hidden";
      for(var index = 0; index < 6; index++) {
        board.children[index].className = classToSet;
      }
    }
  </script>
</body>
</html>

```

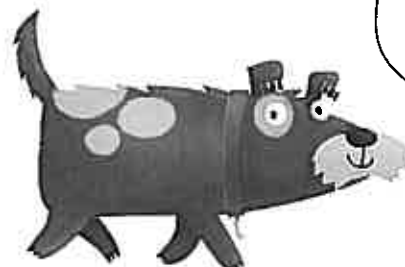
But which characters
are the guests and
which is the thief?



Function call

Simplified if
statement

Turn over to
create the
characters!



12. create the thief

At the moment in our game, every 3 seconds our characters flash on and off the screen. What we need to do now is change the position of our characters on-screen every time our game loop runs. Every 3 seconds the characters need to move to a different place on the game board. And we need to make one of the characters the thief.

We first need to add a function that creates a new set of characters, in different positions, every time our game loop runs. Let's call this function `createCharacters`. Start by adding the function call in your `gameLoop` function, like this:

```
function gameLoop() {  
    peopleVisible = !peopleVisible;  
    createCharacters();  
    flashCharacters();  
    loops++;  
    if(loops < 12) {  
        setTimeout(gameLoop, 3000);  
    }  
    else {  
        alert("Game over!");  
    }  
}
```

Function call

Then code the new `createCharacters` function before your `flashCharacters` function:

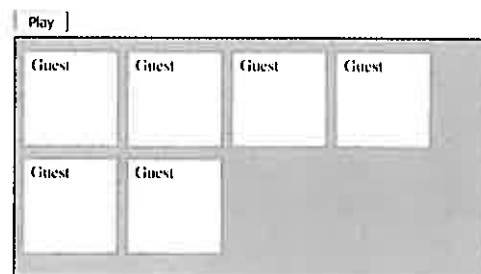
```
function createCharacters() {  
    var board = document.getElementById("board");  
    for(var index = 0; index < 6; index++) {  
        board.children[index].innerHTML = "Guest";  
    }  
}
```



This is very similar to the function we coded earlier to set the CSS class of each character `<div>`. But this time we've used `innerHTML`, like we used in Mission 3, to set the value of each of the characters to "Guest".



Save your HTML file and refresh your page. Now when you press *Play*, you'll see that each of the character `<div>` tags is labelled "Guest".



Now we need to add some code to our `createCharacters` function that will randomly pick one of the 6 character `<div>` tags to be the thief, every time the game loop runs. We need to write code that will pick a random number. There isn't an easy way of picking a random number using JavaScript, so the next piece of code you need is quite complicated. Carefully type this new code at the end of your `createCharacters` function, so that it looks like this:

```
function createCharacters() {
  var board = document.getElementById("board");
  for(var index = 0; index < 6; index++) {
    board.children[index].innerHTML = "Guest";
  }
  // Math API
  var randomNumber = Math.floor(Math.random() * 6) + 1;
  board.children[randomNumber-1].innerHTML = "Thief";
}
```

Here we are using a new API called the Math API. It works in exactly the same way as the DOM and localStorage APIs that we used in Mission 3. Using the Math API lets you access handy maths functions that have been built in to your web browser. To find a random number you have to do the following calculation:

```
Math.floor(Math.random() * BIGGEST_NUMBER) + SMALLEST_NUMBER;
```

We have 6 characters in our game, so our biggest number is 6 and our smallest number is 1. We need to code:

```
Math.floor(Math.random() * 6) + 1;
```

We must get
the diamond
back!

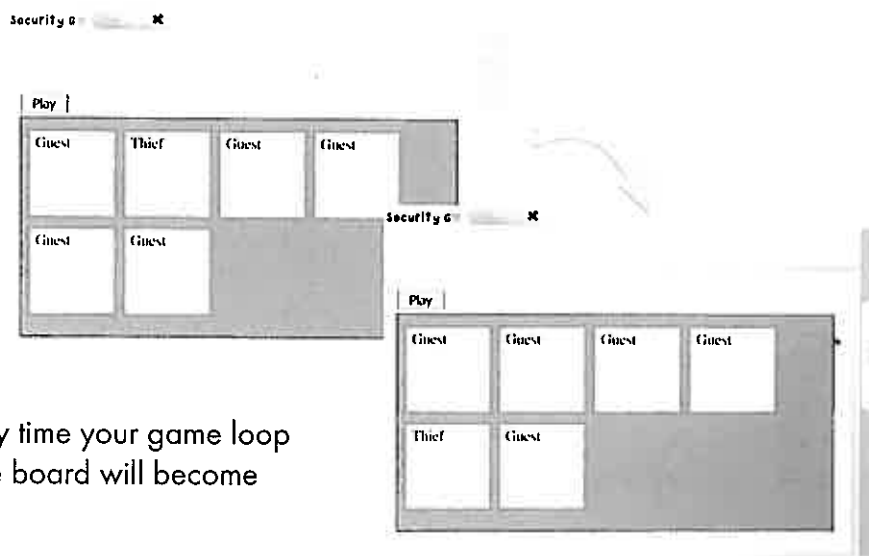
We then store the result of our calculation in a variable. Then we can use this variable in our next line of code. We use innerHTML to set the value of whichever <div> matches our random number, like this:

```
board.children[randomNumber-1].innerHTML = "Thief"
```

Because our numbers in JavaScript count up from 0, we're going to subtract 1 from our random number, so we'll only ever get the numbers 0, 1, 2, 3, 4 and 5.



Save your code. Your new createCharacter function will have added a thief to your game. Now when you press *Play*, every time your game loop runs a random character on your game board will become the thief.



13. create a score

Now we have a game board that shows us a different set of characters every 3 seconds. One of those characters is the thief that the House of Volkov's security team need to catch. What we need to do now is create a way for the user to click

on the thief. When the user clicks, they catch the thief and score a point.

We first have to add a variable called `gameScore`, to keep track of the score. It goes underneath the `startGame` function at the top of our `<script>` block:

```
var loops = 0;
var peopleVisible = false;
var gameScore = 0;
```

Every time the player clicks on the thief, we need to add 1 point to their score. And to make sure the player is paying attention, every time they click on a guest we should take away 2 points. To create our scoring system we need to add an onclick to every character, every time our game loop runs. Add this new code to your `createCharacter` function:

```
function createCharacters() {
  var board = document.getElementById("board");
  for(var index = 0; index < 6; index++) {
    board.children[index].innerHTML = "Guest";
    board.children[index].onclick = function() {
      gameScore += -2;
    }
  }
  var randomNumber = Math.floor(Math.random() * 6) + 1;
  board.children[randomNumber-1].innerHTML = "Thief";
  board.children[randomNumber-1].onclick = function() {
    gameScore++;
  }
}
```

Variable Operator Onclick

Variable Operator Onclick

We're using the `onclick` just like we did in Mission 3. When a guest or thief is created, an `onclick` is added. We're using two new arithmetic operators to change the value of our `gameScore` variable. If the user clicks on a guest, the `+=` operator means the value that follows is added to the value in the variable. In this case, since the value that follows is `-2`, it means that 2 is subtracted from the `gameScore` value. If the user clicks on a thief, the increment operator (`++`) means 1 is added to the value in the variable.

We also need to change the alert message when the game ends, so that it tells the user their score. Change the else statement in your gameLoop function so it uses the gameScore variable value, like this:

```
function gameLoop() {  
  peopleVisible = !peopleVisible;  
  createCharacters();  
  flashCharacters();  
  loops++;  
  if(loops < 12) {  
    setTimeout(gameLoop, 3000);  
  }  
  else {  
    alert("You scored " + gameScore);  
  }  
}
```

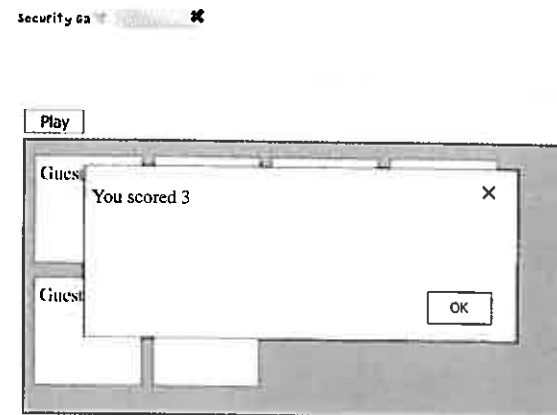
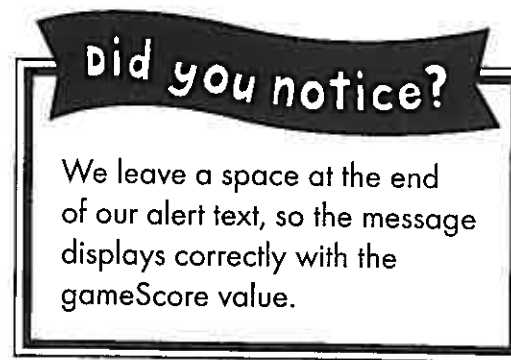
Variable



Save your HTML file and refresh your page. Try clicking on the thief every time it flashes on-screen. See how many points you score at the end of the game. Remember you need to refresh your page between each play of the game.



Hurray! The diamond will surely be safe now!



14. Simplify the code

Our game now works and the House of Volkov's security team can soon begin their training. But you might have noticed that our createCharacters and flashCharacters functions are quite similar. One creates our characters and the other adds a CSS

class to them. We can now combine these functions to make our code simpler. Simplifying your code as you work through a problem is a really common thing to do when you're coding. It makes it much easier to understand your code.

We should change our createCharacters function so it adds the CSS class as well as creating the characters. Change the function so it looks like this:

```
function createCharacters() {  
  var board = document.getElementById("board");  
  var classToSet = peopleVisible ? "character visible" : "character hidden";  
  for(var index = 0; index < 6; index++) {  
    board.children[index].className = classToSet;    CSS classes  
    board.children[index].innerHTML = "Guest";      added  
    board.children[index].onclick = function() {  
      gameScore += -2;  
    }  
  }  
  var randomNumber = Math.floor(Math.random() * 6) + 1;  
  board.children[randomNumber-1].innerHTML = "Thief";  
  board.children[randomNumber-1].onclick = function() {  
    gameScore++;  
  }  
}
```

This function now does everything. Every time the game loop runs, it picks up the correct CSS class, creates the characters and then sets the onclick for each character. We no longer need the flashCharacters function, so we can delete it. Also delete the function call from your gameLoop function so it looks like this:

```
function gameLoop() {  
  peopleVisible = !peopleVisible;  
  createCharacters();  
  loops++;  
  if(loops < 12) {  
    setTimeout(gameLoop, 3000);  
  }  
  else {  
    alert("You scored " + gameScore);  
  }  
}
```



Save your code.
Your game will work in
exactly the same way.

Function call for
flashCharacters removed

Smart coding,
my friend!



15. Designing the game with css

We've got the basic structure of our game working now, but it doesn't look very fun on-screen. Let's use our CSS skills from Mission 1 to design it. First let's see if we can make our thief look more distinctive when he flashes up on-screen. Let's add a new CSS class called thief to our `<style>` block, like this:

```
.character {  
    background-color: lightblue;  
    width: 120px;  
    height: 120px;  
    padding: 10px;  
    margin: 10px;  
    float: left;  
}  
.thief {  
    background-color: red;  
}
```

Then we need to apply this CSS class to our JavaScript. Every time the thief is added we need to add this new CSS class. To do that we need to make this change to our code:

```
var randomNumber = Math.floor(Math.random() * 6) + 1;  
board.children[randomNumber-1].innerHTML = "Thief";  
board.children[randomNumber-1].onclick = function() {  
    gameScore++;  
}  
board.children[randomNumber-1].className = classToSet + " thief";
```

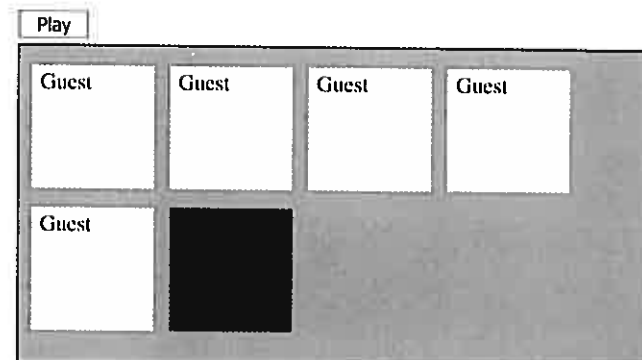


Every time our game loop runs, this will apply our thief CSS class to our thief `<div>` and make it red. Save your HTML file. Your thief will now be different to the guest.

Did you notice?

There is a space in the code " thief ". This space is essential because we are using two CSS class names: the CSS class we've picked using our classToSet variable (hidden or visible), and the CSS class thief. We need the space so it doesn't end up as "hiddenthief".

Security 53



16. Use images for the characters

If we want to make our character <div> tags look a bit more exciting, we're going to have to learn some new CSS properties. The new properties are very simple and work in exactly the same way as the ones you learnt in Mission 1.

First of all, go to the Get Coding! website and find these two images:

Save these images with all your other HTML files in your **Coding** folder on your desktop. You've done this before in Mission 1, so go back to pages 28-30 if you need a reminder.

Now we're going to change the CSS classes in our <style> block. We're going to use two new CSS properties: the background and background-size CSS properties.



CSS property	What does it do?	Example values
background	Sets the background of a HTML element to an image	url('image.jpg'); none;
background-size	Sets the size of the background image in a HTML element	cover; 650px;

We can use these two new CSS properties in our character and thief CSS classes, like this:

```
.character {
  background: url('guest.jpg');
  background-size: cover;
  width: 120px;
  height: 120px;
  padding: 10px;
  margin: 10px;
  float: left;
}
.thief {
  background: url('thief.jpg');
  background-size: cover;
}
```

Background property

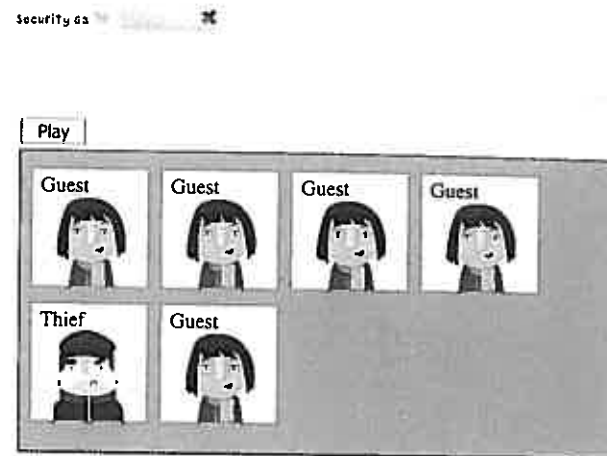
Background-size property



Using the background CSS property, we ask our browser to use saved images as the background for our <div> tags. Then using the background-size CSS property, we ask our browser to make sure that the image file is set to a big enough size to cover the whole background of each <div>.



Save your HTML file and see what the new CSS properties have done to your game.



Now we have the images set as the background of the character <div> tags, let's remove the guest and thief labels. After all, we don't want it to be too easy for the House of Volkov's security team. Remove the text by changing these two lines in your <script> block:

```
board.children[index].innerHTML = "Guest";
```

```
board.children[randomNumber-1].innerHTML = "Thief";
```

Change them to empty strings, like this:

```
board.children[index].innerHTML = "";
```

```
board.children[randomNumber-1].innerHTML = "";
```

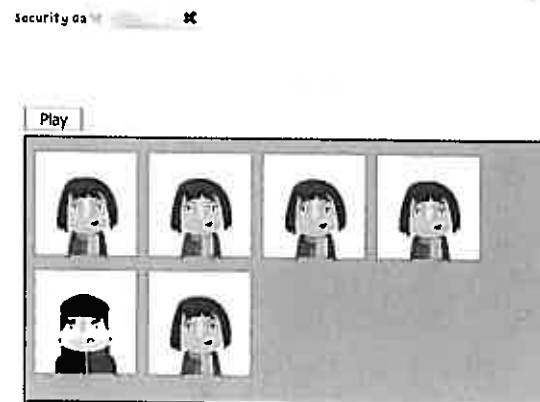


Save your HTML file. Now your game will just be made up of pictures.



I don't like the look of that thief.

The game is looking great!



17. change the game board

Now we know how to use the background and background-size CSS properties, we can easily change our game board from the grey colour to an image. Go to the *Get Coding!* website and find the background image. Save it in your **Coding** folder.

All we then have to do is add the new background and background-size properties to the board CSS class in our <style> block. Don't forget to delete the background-color CSS class, as it is no longer needed. Your code will look like this:

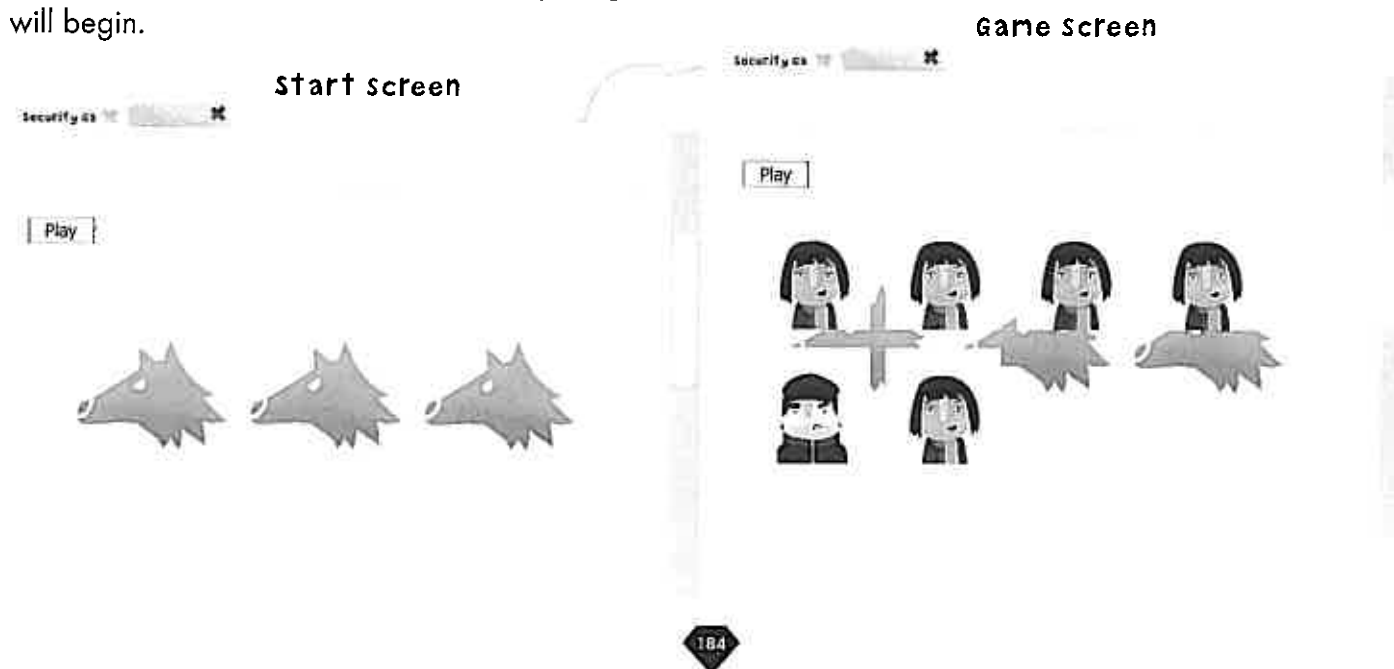
We should also make sure that at the start of the game, when it loads in our browser, we just see the game board. Then when we press play, we see the characters. To do this we need to add the CSS display property to the character CSS class in our <style> block, like this:



Save your HTML file. When your game loads, you will see the new background for your game board. When you press *Play* your characters will load on-screen and your game will begin.

```
#board {  
  background: url('background.jpg');  
  background-size: cover;  
  border: 1px solid black;  
  height: 350px;  
  width: 650px;  
}
```

```
.character {  
  background: url('guest.jpg');  
  background-size: cover;  
  width: 120px;  
  height: 120px;  
  padding: 10px;  
  margin: 10px;  
  float: left;  
  display: none;  
}
```



18. Make the game harder

You might have noticed that the 3 second delay makes it easy to click on the thief. We now need to make the game harder so it's more of a challenge. The characters need to flash on-screen for a much shorter time so there will be less time to click on the thief. To do this we need to change the `setTimeout` call, like this:

```
setTimeout(gameLoop, peopleVisible ? 1000 : 3000);
```

We're using a simplified if statement, like we did earlier in the mission. We've changed our `setTimeout` call so that if `peopleVisible` is true, our `gameLoop` function will be called in 1 second. If `peopleVisible` is false, our `gameLoop` function will be called in 3 seconds. So now our characters will only flash on-screen for 1 second, but when they're invisible they will be hidden for 3 seconds.

Save your code and start playing your finished game. Can you score 6 points?



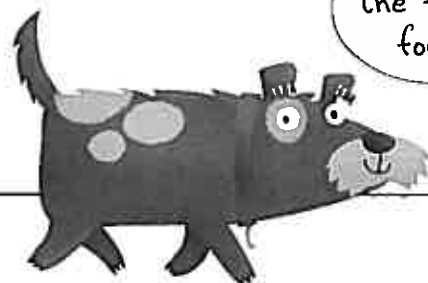
DO IT YOURSELF TASK YOUR FINISHED GAME

Coding a game is challenging and you've mastered it in this mission. Professor Bairstone will be delighted. The game will test the security team's reaction times.

Security Training Game Brief

Check that you have correctly coded all the things below in your **securitygame.html** file and that it's saved in your **Coding** folder. Don't forget you can change the speed of your game, making it faster and even harder for the user.

- ♦ A game board
- ♦ 5 guests
- ♦ 1 thief
- ♦ A play button
- ♦ A score alert



Turn over to see
the full code block
for the game.


```

<!DOCTYPE html>
<html>
<head>
  <title>Security Game</title>
  <style>
    #board {
      background: url('background.jpg');
      background-size: cover;
      border: 1px solid black;
      height: 350px;
      width: 650px;
    }
    .character {
      background: url('guest.jpg');
      background-size: cover;
      width: 120px;
      height: 120px;
      padding: 10px;
      margin: 10px;
      float: left;
      display: none;
    }
    .thief {
      background: url('thief.jpg');
      background-size: cover;
    }
    .hidden {
      display: none;
    }
    .visible {
      display: block;
    }
  </style>
</head>
<body>
  <input type="button" value="Play" onclick="startGame()"/>
  <div id="board">
    <div class="character">1</div>
    <div class="character">2</div>
    <div class="character">3</div>
    <div class="character">4</div>
  </div>
</body>
</html>

```



```

<div class="character">5</div>
<div class="character">6</div>
</div>
<script>
  function startGame() {
    gameLoop();
  }
  var loops = 0;
  var peopleVisible = false;
  var gameScore = 0;
  function gameLoop() {
    peopleVisible = !peopleVisible;
    createCharacters();
    loops++;
    if(loops < 12) {
      setTimeout(gameLoop, peopleVisible ? 1000 : 3000);
    }
    else {
      alert("You scored " + gameScore);
    }
  }
  function createCharacters() {
    var board = document.getElementById("board");
    var classToSet = peopleVisible ? "character visible" : "character hidden";
    for(var index = 0; index < 6; index++) {
      board.children[index].className = classToSet;
      board.children[index].innerHTML = "";
      board.children[index].onclick = function() {
        gameScore += -2;
      }
    }
    var randomNumber = Math.floor(Math.random() * 6) + 1;
    board.children[randomNumber-1].innerHTML = "";
    board.children[randomNumber-1].onclick = function() {
      gameScore++;
    }
    board.children[randomNumber-1].className = classToSet + " thief";
  }
</script>
</body>
</html>

```

FUTURE CODE SKILLS

This mission was an important step in learning about JavaScript game programming. You used a game loop, an essential part of nearly all computer games, to power your game and create an interactive experience for the user. This isn't an easy thing to do, so well done! You can now use your knowledge about game loops to build more complicated games that will respond to your user in different ways.