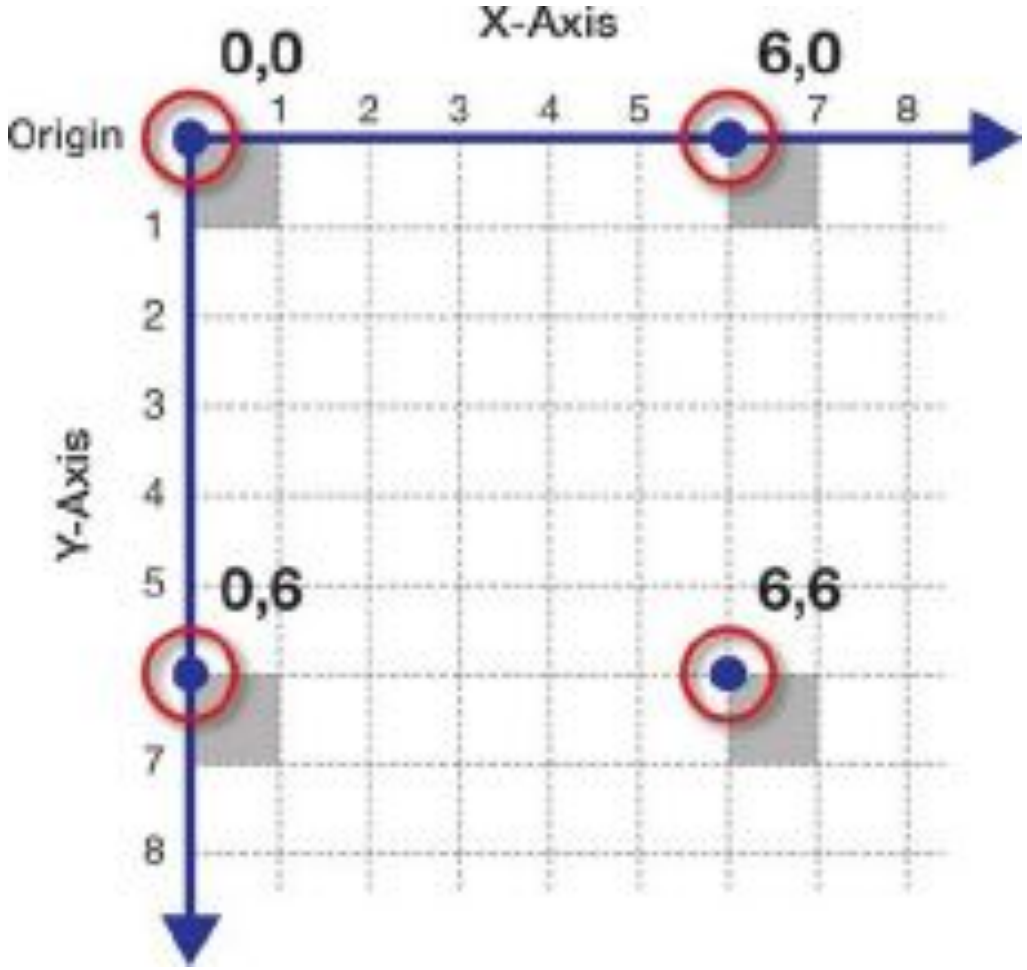
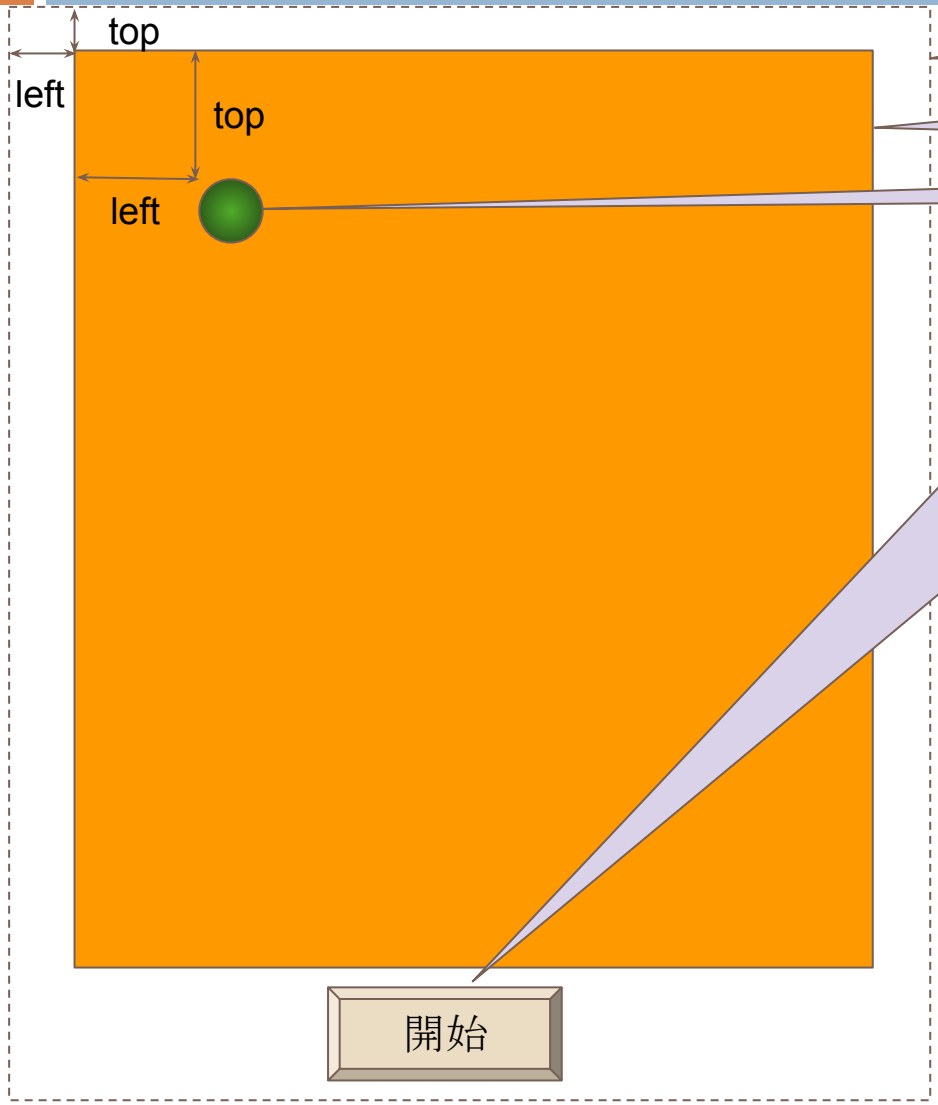
Animation

web 座標



遊戲區域與主角



<img id="ball" class="ball">

<button id="start" class="startButton">

```
.gameZone {
    position: absolute;
    background-color: brown;
    left: 100px;
    top: 100px;
    width: 300px;
    height: 500px;
}
.ball {
    position: absolute;
    left: 50px;
    top: 30px;
}
.startButton {
    left: 200px;
    top: 610px;
    position: absolute;
}
```

練習—按鍵移動

```
$(document).keydown(function (e) {
    const position = $("#ball").position();
    let top = position.top;
    let left = position.left;
    switch (e.keyCode) {
        case 37:  //left
            left--;
            break;
        case 38:  //up
            top--;
            break;
        case 39:  //right
            left++;
            break;
        case 40:  //down
            top++
            break;
        default:
            break;
    }
    $("#ball").css("left", left);
    $("#ball").css("top", top);
});
```

事件驅動

取得位置

判斷按鍵

設定新位置

練習—自然向右移動

```
$(“#start”).click(function (e) {
    setInterval(startGame, 30);
});

function startGame(){
    const position = $("#ball").position();
    let top = position.top;
    let left = position.left;
    left++;
    $("#ball").css("left", left);
    $("#ball").css("top", top);
}
```

事件驅動

設定每30ms執行startGame

取得位置

設定新位置

如果要讓按鍵啟動後，自動變成停止鍵，停止鍵按後變成啟動鍵

```
let timer = underfined;
$(“#start”).click(function (e) {
    if(timer == undefined) {
        timer = setInterval(startGame, 30);
        $("#start").text("停止");
    }
    else{
            clearInterval(timer);
            timer = undefined;
            $("#start").text("開始");
    }
});
```

練習—按鍵改變方向

```
let timer = underfined;
$(“#start”).click(function (e) {
    if(timer == undefined) {
        timer = setInterval(startGame, 30);
        $("#start").text("停止");
    }
    else{
            clearInterval(timer);
            timer = undefined;
            $("#start").text("開始");
    }
});
```

```
let moveDirection = 39;
$(document).keydown(function (e) {
    moveDirection = e.keyCode;
});
```

```
function startGame(){
    const position = $("#ball").position();
    let top = position.top;
    let left = position.left;
    switch (moveDirection) {
        case 37:  //left
            left--;
            break;
        case 38:  //up
            top--;
            break;
        case 39:  //right
            left++;
            break;
        case 40:  //down
            top++
            break;
        default:
            break;
    }
    $("#ball").css("left", left);
    $("#ball").css("top", top);
}
```

requestAnimationFrame

- 採用系統時間間隔, 以保持最佳繪制效率, 不會因間隔時間過短, 造成過度繪制, 也不會因間隔時間過長, 造成動畫不流暢
- 讓所有網頁上的動畫效果具備统一的刷新機制
- 節省系統資源, 提高系統性能, 改善視覺效果
- 改善setInterval與setTimeout的缺點(均需設定時間)
- 使用方式

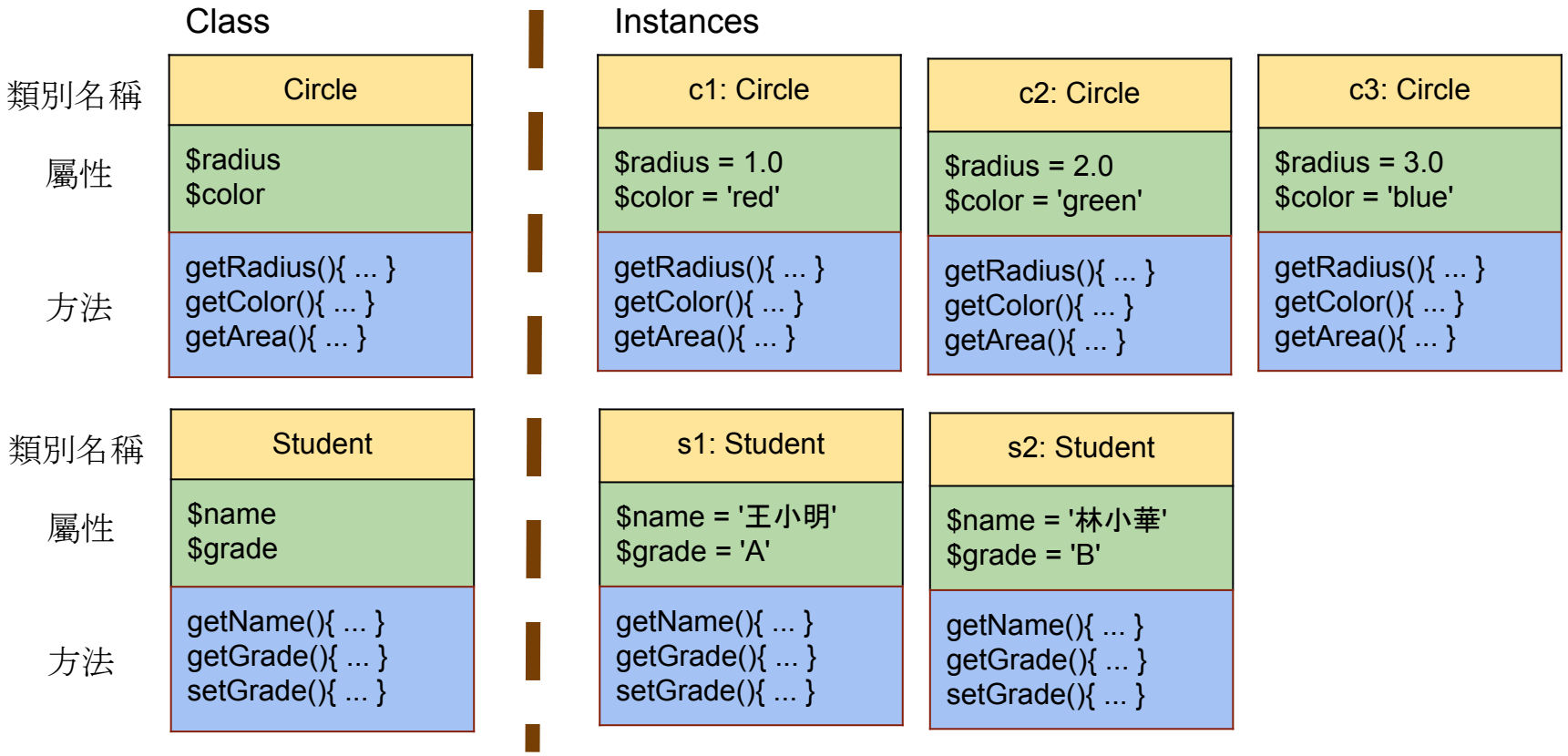
```
const requestID = requestAnimationFrame( ()=>animation() );

function animation(){
    ……
    ……
    ……
    requestAnimationFrame( ()=>animation() );
}
```

Javascript 匿名函數的寫法

類別與物件

- 類別 (Class)
  - 用來定義物件
- 物件 (Object)
  - 讓類別實體化



類別的組成

- 屬性(變數)
  - 描述靜態的資料值
- 方法(函數)
  - 可以執行的動作或指令
- 建構子(特定的函數)
  - 只在物件生成時執行的函數，通常用於設定物件的初始值

方法
BMI 說話 移動

屬性
身高 體重 眼睛顏色

類別與物件—範例

```
obj = new MyClass();

class MyClass{
    constructor(){ //建構子
     ……
    }
    aaa(){   //方法(函數)名稱
     ……
    }
    bbb(){   //方法(函數)名稱
     ……
    }
}
```

Class 範例

```
export default class Ball {
    constructor(){
        //設定初始狀態
    }
    start(){
        //開始動畫
    }
    stop(){
        //停止動畫
    }
    turn(direction){
        //轉向
    }
    score(){
        //算分數
    }
    getElement(){
        //取得html元件
    }
    animation(){
        //動畫
    }
    collision(){
        //偵測碰撞
    }
}
```

```
this.target = target;
this.obj = jQuery(`<img class="ball" id="ball" src="./dot.gif">`);
this.gameZone = $("#gameZone");
this.gameZone.append(this.obj);
this.direction = 37 + Math.floor(Math.random()*4);
this.start();
```

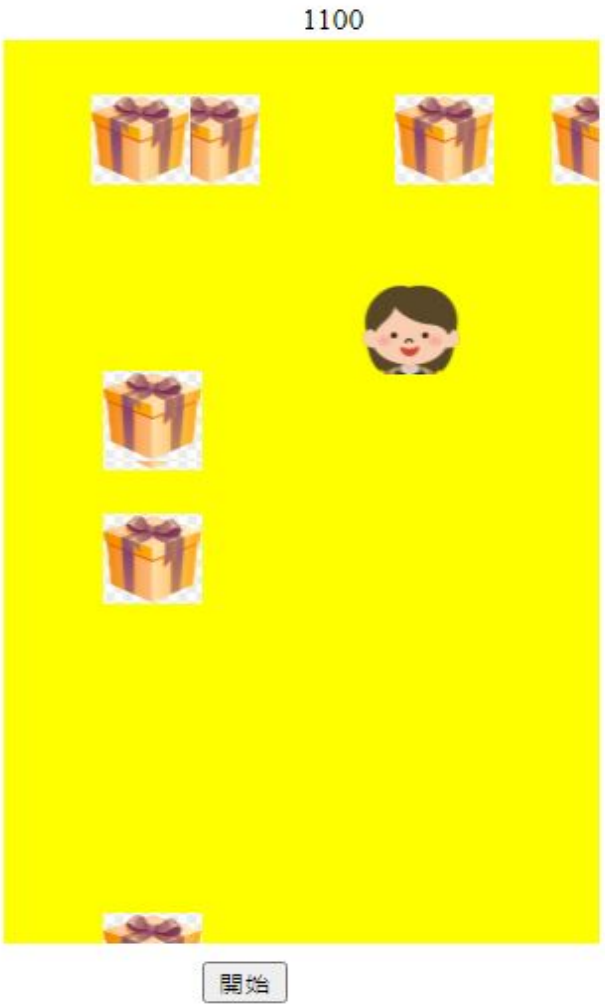
```
requestAnimationFrame(()=>this.animation());
```

```
const position = this.obj.position();
let top = position.top;
let left = position.left;
switch (this.direction) {
    case 37:  //left
        left--;
        break;
    case 38:  //up
        top--;
        break;
    case 39:  //right
        left++;
        break;
    case 40:  //down
        top++
        break;
    default:
        break;
}
this.obj.css("left", left);
this.obj.css("top", top);
requestAnimationFrame(()=>this.animation());
```

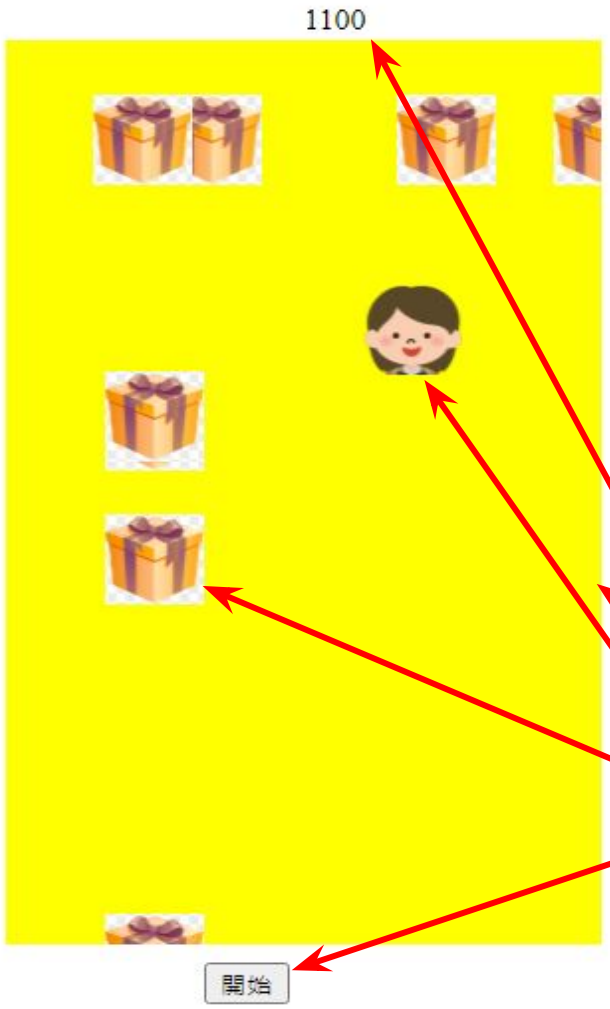
使用屬性或方法均要this.

遊戲腳本

- 玩家部份
  - 玩家可由鍵盤控制方向，並自行移動
  - 當玩家碰到邊界時，則遊戲停上
- 禮品部份
  - 遊戲中會自動產生禮品，且禮品也會自行移動，當玩家碰到時，則分數加100
  - 禮品碰到邊界時，會自動反彈
  - 禮品出現位置是隨機出現的



遊戲畫面設計



score

gameZone

player

gift

start

index.html

```
<html lang="en">
<head>
    <script src="https://...../jquery.min.js"></script>
    <script type="module" src="index.js"></script>
    <link rel="stylesheet" href="game.css">
</head>
<body>
    <div id="root"></div>
</body>
</html>
```

game.css

```
.gameZone {
    position: absolute;
    background-color: yellow;
    left: 100px;
    top: 100px;
    width: 300px;
    height: 500px;
    overflow: hidden;
}
.gift {
    position: absolute;
}
.player {
    position: absolute;
    left: 100px;
    top: 30px;
}
.start {
    left: 200px;
    top: 610px;
    position: absolute;
}
.score {
    left: 250px;
    top: 80px;
    position: absolute;
}
```

index.js

```
import Gift from './gift.js';
import Player from './player.js';

$(document).ready(function () {
    const startPage = `
    <div class="score" id="score">0</div>
    <div class="gameZone" id="gameZone"></div>
    <button id="start" class="startButton">開始</button>
    `;
    $("#root").html(startPage);
    const gifts = new Array();
    const player = new Player();
    $(document).keydown(function (e) {
        player.turn(e.keyCode);
    });
    $("#start").click(function (e) {
        requestAnimationFrame(()=>animation());
    });
    function animation(){
        if(parseInt(Math.random()*60)==0)
            gifts.push(new Gift(player.getElement()));
        gifts.forEach(Element=>{
            Element.move();
        });
        if(player.move()==200)
            requestAnimationFrame(()=>animation());
    }
});
```

匯入Player, Gift (一般class命名均以大寫開頭)

作畫面

產生player, 準備gifts陣列

依據按鍵, 改變player的方向

開始作動畫

1.由於requestAnimationFrame大約每秒更新60次, Math.random()會產生0~1的數, 乘60意即約每秒產生一個gift

2.每個gift都作一次move()

3.player作一次move(), 但player可能Gameover, 所以必需判斷狀態

player.js (1/2)

```
export default class Player {
    constructor(){
        this.element = jQuery(`<img class="player" id="player" src="./player.png">`);
        this.gameZone = $("#gameZone");
        this.gameZone.append(this.element);
        this.direction = 37 + Math.floor(Math.random()*4);
    }
    turn(direction){
        this.direction = direction;
    }
    getElement(){
        return this.element;
    }
    move(){
        // 下一頁說明
    }
}
```

Player的初始狀態

改變方向

取得player裡的 html element

player.js - 邊界偵測 (2/2)

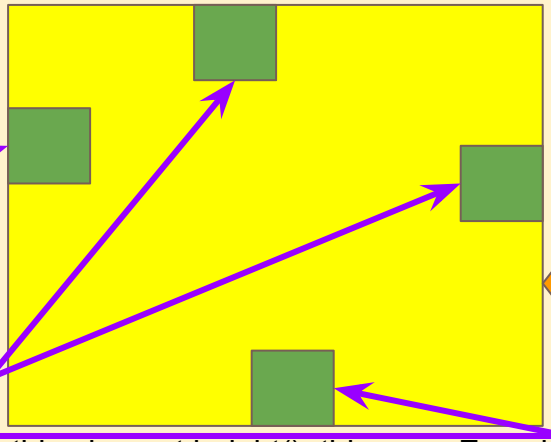
```
move(){
    const position = this.element.position();
    let top = position.top;
    let left = position.left;
    switch (this.direction) {
        case 37:  //left
            left--;
            break;
        case 38:  //up
            top--;
            break;
        case 39:  //right
            left++;
            break;
        case 40:  //down
            top++
            break;
        default:
            break;
    }
    this.element.css("left", left);
    this.element.css("top", top);
    if(left<0 || left+this.element.width()>this.gameZone.width() || top<0 || top+this.element.height()>this.gameZone.height()){
        return -1;
    }
    else{
        return 200;
    }
}
```

取得現在位置

根據目前方向作位置的調整

設定新位置

判斷是否撞到gameZone的邊界，
如果是return -1，否則return 200



gift.js (1/3)

```
export default class Gift {
    constructor(player){
        this.player = player;
        this.element = jQuery(`<img class="gift" src="./gift.png">`);
        this.gameZone = $("#gameZone");
        this.gameZone.append(this.element);
        this.direction = 37 + Math.floor(Math.random()*4);
        const left = Math.floor(Math.random()*(this.gameZone.width()-this.element.width()));
        const top = Math.floor(Math.random()*(this.gameZone.height()-this.element.height()));
        this.element.css("left", left);
        this.element.css("top", top);
    }
    removeElement(){
        this.element.remove();
    }
    score(){
        let score = parseInt( $("#score").text() );
        score += 100;
        $("#score").text(score);
    }
    move(){
        // 下一頁說明
    }
    collision(){
        // 下一頁說明
    }
}
```

Gift的初始狀態

當gift與player碰撞時，gift會消失

取得player裡的 html element

gift.js - 邊界偵測 (2/3)

```
move(){
    const position = this.element.position();
    let top = position.top;
    let left = position.left;
    switch (this.direction) {
        case 37:  //left
            left--;
            break;
        case 38:  //up
            top--;
            break;
        case 39:  //right
            left++;
            break;
        case 40:  //down
            top++
            break;
        default:
            break;
    }
```

取得現在位置

根據目前方向作位置的調整

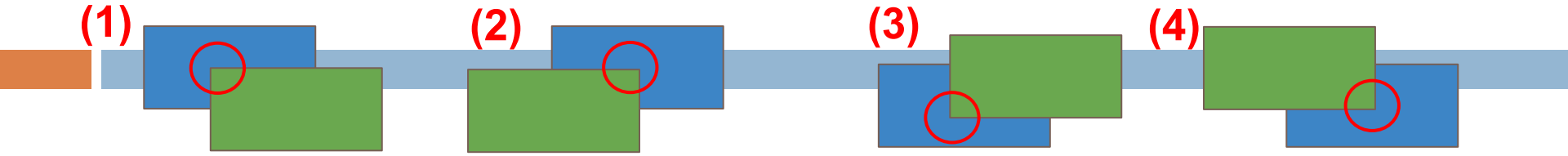
判斷是否撞到gameZone的邊界，作必要的方向改變

設定新位置

判斷是否撞到player，以便移除元件，並作加分

```
    if(left<0){
        left = 0;
        this.direction = 39;
    }
    if (left+this.element.width()>this.gameZone.width()) {
        left--;
        this.direction = 37;
    }
    if(top<0){
        top = 0;
        this.direction = 40;
    }
    if(top+this.element.height()>this.gameZone.height()){
        top--;
        this.direction = 38;
    }
    this.element.css("left", left);
    this.element.css("top", top);
    if(this.collision()){
        this.removeElement();
        this.score();
        return -1;
    }
    else{
        return 200;
    }
}
```

gift.js - 碰撞偵測 (3/3)




```
collision(){
    const playerLeft = this.player.position().left;
    const playerTop = this.player.position().top;
    const playerWidth = this.player.width();
    const playerHeight = this.player.height();
    const elementLeft = this.element.position().left;
    const elementTop = this.element.position().top;
    const elementWidth = this.element.width();
    const elementHeight = this.element.height();

(1) if( elementLeft>=playerLeft && elementLeft<=(playerLeft+playerWidth) && elementTop>=playerTop &&
elementTop<=(playerTop+playerHeight))
        return true;
(2) if( (elementLeft+elementWidth)>=playerLeft && (elementLeft+elementWidth)<=(playerLeft+playerWidth) &&
elementTop>=playerTop && elementTop<=(playerTop+playerHeight))
        return true;
(3) if( elementLeft>=playerLeft && elementLeft<=(playerLeft+playerWidth) && (elementTop+elementHeight)>=playerTop &&
(elementTop+elementHeight)<=(playerTop+playerHeight))
        return true;
(4) if( (elementLeft+elementWidth)>=playerLeft && (elementLeft+elementWidth)<=(playerLeft+playerWidth) &&
(elementTop+elementHeight)>=playerTop && (elementTop+elementHeight)<=(playerTop+playerHeight))
        return true;
    return false;
}
```