

# Programowanie aplikacji w języku JavaScript

## *Phaser*



Darmowa biblioteka do gier:

- desktop i mobile
- grafika i animacje
- fizyka, system cząsteczek
- dźwięki

Strona:

<http://phaser.io/>

Przykłady:

<http://phaser.io/examples>

Dokumentacja:

<http://phaser.io/docs>

## // Struktura

```
var game = new Phaser.Game(300, 400, Phaser.AUTO, '',  
    {  
        preload: preload,  
        create: create,  
        update: update  
    }  
));
```

*game* – serce gry

```
function preload() {  
    game.stage.backgroundColor = '#48a';  
}
```

tu ładujemy zasoby

```
function create() {
```

tu tworzymy obiekty gry

```
function update() {
```

metoda uruchamiana co klatkę  
(aktualizowanie obiektów, zmiana położenia,  
przeliczanie AI, kontrolowanie wejścia, etc.)

```
// Dodawanie grafiki („duszka”)
```

```
function preload() {
```

lokacja zasobów

```
    game.load.baseURL = 'http://examples.phaser.io/assets/';  
    game.load.crossOrigin = 'anonymous';
```

```
    game.load.image('ball', 'games/breakout/ball.png');
```

```
}
```

ładowanie (asynchroniczne)  
obrazka do *cache*'u

```
var ball;
```

```
function create() {
```



```
    ball = game.add.sprite(250, 350, 'ball');  
    ball.anchor.set(0.5, 0.5);
```

ustawiamy punkt (0,0)

dodawanie *sprite*'a  
do sceny (*stage*)

```
}
```

```
// Ruch (fizyka)
```

```
function create() {
```

```
    game.physics.arcade.enable(ball);
```

```
    ball.body.velocity.x = 100;
```

```
    ball.body.velocity.y = -150;
```

```
    ball.body.collideWorldBounds = true;
```

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

```
}
```

uruchomienie fizyki dla  
sprite'a doda mu ciało (*body*)

to ciało (a nie sprite)  
ma prędkość

kolizja z krawędziami  
świata (ekranu)

odbijanie

```
// Paletka
```

```
function preload() {
```

ładujemy grafikę

```
    game.load.image('paddle', 'games/breakout/paddle.png');
```

```
}
```



```
var paddle;
```

```
function create() {
```

tworzymy sprite'a

```
    paddle = game.add.sprite(150, 380, 'paddle');  
    paddle.anchor.set(0.5);
```

```
}
```

```
// Sterowanie
```

```
var cursors;
```

paletka też powinna stać się obiektem fizycznym

```
function create() {
```

```
    game.physics.arcade.enable(paddle);
```

klawisze sterowania

```
    cursors = game.input.keyboard.createCursorKeys();
```

```
}
```

```
function update() {
```

kontrolujemy stan klawiszy

```
    paddle.body.velocity.x = 0;
```

```
    if (cursors.left.isDown) {
```

```
        paddle.body.velocity.x = -250;
```

```
    }
```

```
    else if (cursors.right.isDown) {
```

```
        paddle.body.velocity.x = 250;
```

```
    }
```

```
}
```

i aktualizujemy prędkość

```
// Odbijanie piłeczki
```

```
function create() {
```

```
    paddle.body.collideWorldBounds = true;  
    paddle.body.immovable = true;
```

```
}
```

```
function update() {
```

```
    game.physics.arcade.collide(ball, paddle);
```

```
}
```

inne ciała nie mogą  
go przesunąć

kolizja (zderzenie)  
piłki z paletką

**collide()** – testuje kolizję i nie dopuszcza do nakładania się obiektów  
**overlap()** – tylko testuje kolizję  
trzeci parametr – funkcja wywoływana w razie stwierdzenia kolizji

```
// Zmienny kąt odbicia
```

```
function update() {
```

```
    game.physics.arcade.collide(ball, paddle, ballHitsPaddle);
```

```
}
```

```
function ballHitsPaddle(ball, paddle) {
```

```
    ball.body.velocity.x = 5 * (ball.x - paddle.x);
```

```
}
```

metoda wywoływana  
w razie kolizji

zmiana prędkość poziomej  
w zależności od punktu  
uderzenia w paletkę



```
// Cegły
```

```
function preload() {
```

```
    game.load.image('brick', 'games/breakout/brick1.png');
```

```
}
```

```
var bricks;
```

```
function create() {
```

```
    bricks = game.add.physicsGroup();
```

```
    for (var y = 0; y < 4; ++y) {  
        for (var x = 0; x < 7; ++x) {
```

```
            var brick = bricks.create(24+x*36, 80+y*40, 'brick');  
            brick.body.immovable = true;
```

```
        }
```

```
    }
```

```
}
```

tworzy grupę sprite'ów z fizyką

tworzy w grupie  
pojedynczego sprite'a

```
// Zbijanie cegieł
```

kolizja cegieł z piłką



```
function update() {
```

```
    game.physics.arcade.collide(ball, bricks, ballHitsBrick);
```

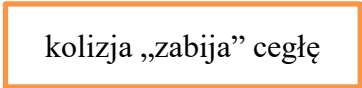

```
}
```

```
function ballHitsBrick(ball, brick) {
```

```
    brick.kill();
```

```
}
```

kolizja „zabija” cegłę



// Wyświetlanie tekstu

var gameOverText;

function create() {

gameoverText = game.add.text(game.world.centerX, 250,  
 'GAME OVER',  
 { font: "40px Arial", fill: "#ffffff", align: "center" });

gameoverText.anchor.setTo(0.5, 0.5);  
gameoverText.visible = false;

}

dodaje tekst

styl tekstu

```
// Test końca gry
```

```
function create() {
```

```
    game.physics.arcade.checkCollision.down = false;
```

```
    ball.checkWorldBounds = true;
```

```
    ball.events.onOutOfBounds.add(ballLost, this);
```

```
}
```

```
function ballLost() {
```

```
    gameOverText.visible = true;
```

```
}
```

bez kolizji na dole ekranu

zdarzenie podnoszone, gdy piłka  
znajdzie się poza ekranem

wyświetlenie napisu  
końca gry

## // Przykład 2.

```
var game = new Phaser.Game(500, 300, Phaser.AUTO, '',  
    { preload: preload, create: create, update: update });
```

```
function preload() {
```

```
    game.load.baseURL = 'http://examples.phaser.io/assets/';  
    game.load.crossOrigin = 'anonymous';
```

```
    game.load.image('background',  
        'games/starstruck/background.png');
```

```
}
```

```
function create() {
```

```
    game.add.tileSprite(0, 28, 500, 300, 'background');
```

```
}
```

sprite wypełniony  
powtarzającym się wzorem

tło



```
function preload() {  
    game.load.spritesheet('player',  
        'games/starstruck/dude.png', 32, 48);  
}
```

seria obrazków  
postaci gracza

rozmiar  
pojedynczego

```
var player;
```

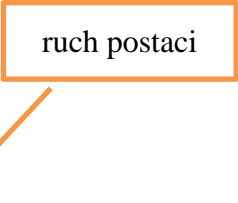
```
function create() {
```



```
    player = game.add.sprite(50, 100, 'player');  
    game.physics.arcade.enable(player);  
    player.body.collideWorldBounds = true;  
    player.body.gravity.y = 500;  
    player.body.bounce.y = 0.2;
```

gravitacja (można ustawić też globalną –  
game.physics.arcade.gravity)

```
var cursors;  
  
function create() {  
    cursors = game.input.keyboard.createCursorKeys();  
}  
  
function update() {  
    player.body.velocity.x = 0;  
    if (cursors.left.isDown)  
    {  
        player.body.velocity.x = -150;  
    }  
    else if (cursors.right.isDown)  
    {  
        player.body.velocity.x = 150;  
    }  
}
```



ruch postaci

```
var jumpButton;
```

```
function create() {
```

```
    jumpButton = game.input.keyboard.addKey(
        Phaser.Keyboard.SPACEBAR);
```

```
}
```

```
function update() {
```

```
    if (jumpButton.isDown &&
        (player.body.touching.down || player.body.onFloor()))
    {
        player.body.velocity.y = -250;
    }
```

```
}
```

klawisz skoku

o ile dotykamy  
podłoża

skok to nadanie  
pionowej prędkości



```
function create() {
```

```
    player.animations.add('left', [0, 1, 2, 3], 10, true);  
    player.animations.add('right', [5, 6, 7, 8], 10, true);
```

```
}
```

definicje  
animacji

numery klatek

```
function update() {
```

```
    if (cursors.left.isDown) {  
        player.animations.play('left');
```

```
    }
```

```
    else if (cursors.right.isDown) {  
        player.animations.play('right');
```

```
    }
```

```
    else {  
        player.animations.stop();  
        player.frame = 4;
```

```
    }
```

```
}
```

odtworzenie  
animacji

zatrzymanie  
animacji

```
function preload() {
```

```
    game.load.image('platform', 'sprites/block.png')
```

```
}
```

```
var platforms;
```

```
function create() {
```

```
    platforms = game.add.physicsGroup();
```

```
    platforms.create(200, 240, 'platform');
```

```
    platforms.create(300, 190, 'platform');
```

```
    platforms.create(400, 140, 'platform');
```

```
    platforms.setAll('body.immovable', true);
```

```
    platforms.setAll('scale', { x: 0.5, y: 0.5 });
```

```
}
```

```
function update() {
```

```
    game.physics.arcade.collide(player, platforms);
```

```
}
```



dodawanie  
platform

operacja na wszystkich  
elementach grupy

kolizja z  
platformami

```
function create() {
```

```
    var bg = game.add.tileSprite(0, 28, 500, 300, 'background');  
    bg.fixedToCamera = true;
```

tło nie przemieszcza się

```
    game.world.setBounds(0, 0, 900, 300);  
    game.camera.follow(player);
```

świat gry większy niż ekran

```
    platforms.create(450, 90, 'platform');  
    platforms.create(500, 140, 'platform');  
    platforms.create(600, 190, 'platform');  
    platforms.create(700, 240, 'platform');
```

kamera podąża za  
postacią gracza

```
}
```

więcej platform

// Dźwięki

ładujemy pliki  
audio

```
function preload() {
```

```
    game.load.audio('step1', 'audio/SoundEffects/steps1.mp3');  
    game.load.audio('step2', 'audio/SoundEffects/steps2.mp3');  
    game.load.audio('jump', 'audio/SoundEffects/spaceman.wav');
```

```
}
```

```
var sndSteps = [];
```

```
var sndJump;
```

dodajemy dźwięki do  
menadżera odtwarzania

```
function create() {
```

```
    sndSteps[0] = game.sound.add('step1');  
    sndSteps[1] = game.sound.add('step2');  
    sndJump = game.sound.add('jump');
```

```
}
```

```
var stepIdx = 0;
```

```
function update() {
```

```
    if (jumpButton.isDown &&  
        (player.body.onFloor() || player.body.touching.down)) {  
  
        sndJump.play();  
    }
```

odtwarzamy dźwięk

```
    if ((cursors.left.isDown || cursors.right.isDown) &&  
        (player.body.onFloor() || player.body.touching.down)) {  
  
        if (!sndSteps[stepIdx].isPlaying) {  
            stepIdx = (stepIdx + 1) % sndSteps.length;  
            sndSteps[stepIdx].play();  
        }  
    }  
}
```

bardziej złożona sytuacja –  
naprzemienne odtwarzanie  
dwóch dźwięków

```
// Pule obiektów
```

```
var bullets;
```

```
function create() {
```

```
    bullets = game.add.physicsGroup();  
    bullets.createMultiple(30, 'bullet');
```

```
    bullets.setAll('anchor.y', 0.5);  
    bullets.setAll('outOfBoundsKill', true);  
    bullets.setAll('checkWorldBounds', true);
```

```
}
```

dodajemy grupę

i od razu wypełniamy  
ją obiektami

tak stworzone obiekty nie  
„istnieją” (exists = false)

można ustawić parametry  
stworzonych obiektów

dzięki temu obiekty poza  
„światem” będą zabijane  
(ale nie usuwane)

```
var nextBulletTime = 0;
```

gdy potrzebujemy obiektu...

```
function update() {
```

```
    if (fire.isDown && game.time.now >= nextBulletTime) {
```

```
        var bullet = bullets.getFirstExists(false);
```

bierzemy pierwszy wolny  
(nieistniejący)

```
        if (bullet) {  
            bullet.reset(player.body.x, player.body.y);  
            bullet.body.velocity.x = 400;  
            nextBulletTime = game.time.now + 100;
```

```
        }
```

reset przywraca go do  
życia w określonej pozycji

puszczamy w ruch i  
kontrolujemy, aby strzały nie  
pojawiały się za szybko

```
    }  
}
```

```
// Stany
```

```
var game = new Phaser.Game(300, 400, Phaser.AUTO, '');
```

```
var MainMenu = {  
  preload: function () { ... },  
  create: function () { ... },  
  update: function () { ... }  
};
```

stan może być  
obiekttem...

```
function Game() {  
  this.preload = function () { ... };  
  this.create = function () { ... };  
  this.update = function () {  
    if (...) {  
      game.state.start('Game');  
    }  
  };  
}
```

...albo  
konstruktorem

zmiana stanu

dodajemy stany  
do menadżera

```
game.state.add('MainMenu', Welcome, true);  
game.state.add('Game', Game);  
game.state.add('Over', ...);
```

stan domyślny



```
// Przyciski
```

```
function preload() {  
    game.load.spritesheet('button',  
        'buttons/button_sprite_sheet.png', 193, 71);  
}
```

grafika z  
przyciskami

```
var button;
```

DOWN

OUT

OVER

```
function create() {
```

zdarzenie  
kliknięcia

```
    button = game.add.button(x, y, 'button',  
        actionOnClick,  
        this,  
        2, 1, 0);
```

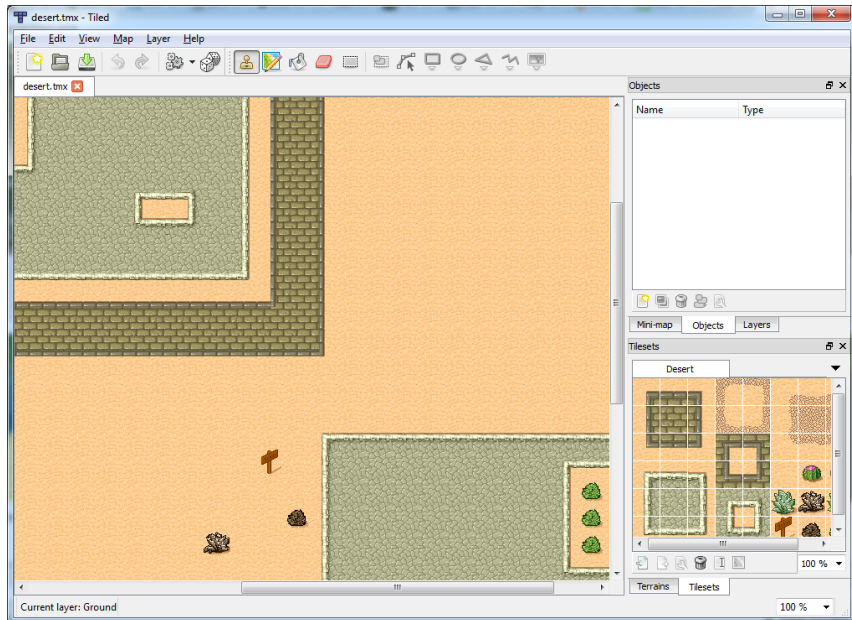
numery grafik do stanów  
*over, out i down*

```
    button.onInputOver.add(over, this);
```

zdarzenie  
najechnia myszą

```
}  
function over() { ... }  
function actionOnClick() { ... }
```

// Mapy tworzone w edytorze Tiled (<http://www.mapeditor.org/>)



```
function preload() {
```

```
    game.load.tilemap(  
        'mario',  
        'tilemaps/maps/super_mario.json',  
        null,  
        Phaser.Tilemap.TILED_JSON);
```

mapa (w pliku  
JSON)

format danych

```
    game.load.image('tiles',  
        'tilemaps/tiles/super_mario.png');
```

tileset (grafiki  
elementów mapy)



```
var map;  
var layer;
```

```
function create() {
```

```
    map = game.add.tilemap('mario');
```

```
    map.addTilesetImage('SuperMarioBros-World1-1', 'tiles');
```

```
    layer = map.createLayer('World1');
```

```
    layer.resizeWorld();
```

```
}
```

ładowanie mapy

nazwa tilesetu z  
edytora

klucz załadowanego  
tilesetu

dopasowanie rozmiaru  
świata do warstwy

warstwa funkcjonuje  
jak sprite (i jest  
dodawana do sceny)

```
// Bitmap Fonts
```

```
function preload() {
```

```
    game.load.bitmapFont('desyrel',  
        'fonts/bitmapFonts/desyrel.png',  
        'fonts/bitmapFonts/desyrel.xml');
```

```
}
```

fS}{][IGJ@jB)(  
5QDX/\YU+AH  
RdP9P9!C3\$5b  
WLM67&FEKQ?  
4k9IS8ZNTLh%  
><#iOV01x2ie  
avwoz\*cmu+r  
n;|≡m|~—

grafika z czcionką

plik opisu czcionki

Bitmap Fonts można generować np. przy pomocy:

<http://www.angelcode.com/products/bmfont/>

```
txt = game.add.bitmapText(200, 100, 'desyrel', 'Ala ma kota', 64);
```