JS [leuke naam]

Uitleg

## Constructor

De constructor is een methode die automatisch aangeroepen wordt als een object wordt gecreëerd. De constructor kan ook een parameter ontvangen die dan meegegeven moet worden als het object aangemaakt wordt.

### Voorbeeld

class Car {

constructor() {

alert('Car is ready');

}

}

myCar = new Car(); // alert will be shown when this code is executed

class Car {

constructor(name, year) {

this.name = name;

this.year = year;

alert('Car ' + this.name + ' from year ' + this.year + ' is ready');

}

}

myCar1 = new Car("Ford", 2014); // alert with car details is shown

myCar2 = new Car("Audi", 2019); // alert with car details is shown

### Bronnen voor meer informatie

https://www.w3schools.com/Js/js\_classes.asp

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes

## **Super**

De super methode voert de constructor methode uit en kan er gebruik van maken.

### Voorbeeld

Class Fish {

constructor(habitat, length) {

this.habitat = habitat

this.length = length

}

}

class Trout extends Fish {

constructor(habitat, length, variety) {

super(habitat, length)

this.variety = variety

}

}

var grouper = new Fish("saltwater", "26in");

console.log(grouper);

let rainbowTrout = new Trout("freshwater", "14in", "rainbow");

console.log(rainbowTrout);

### Resultaat

#### grouper

{"habitat":"saltwater","length":"26in"}

#### rainbow trout

{"habitat":"freshwater","length":"14in","variety":"rainbow"}

#### Bronnen voor meer informatie

<https://css-tricks.com/what-is-super-in-javascript/>  
<https://www.w3schools.com/jsref/jsref_class_super.asp>

## 

## **Extend**

De extend methode haalt de parent class properties + methods van de parent class op en kan er gebruik van maken.

### Voorbeeld

class Monster {

this.health = 100;

}

class Monkey extends Monster{

this.bananaCount = 5;

}

#### Bronnen voor meer informatie

<https://stackoverflow.com/questions/15192722/javascript-extending-class>   
<https://www.w3schools.com/jsref/jsref_class_extends.asp>

## **New**

Met New maak je een nieuw object aan.

### Voorbeeld

class Person {

constructor(fname, lname) {

this.firstName = fname;

this.lastName = lname;

}

}

const person = new Person('testFirstName', 'testLastName');

#### Bronnen voor meer informatie

<https://www.freecodecamp.org/news/a-complete-guide-to-creating-objects-in-javascript-b0e2450655e8/>

<https://www.w3schools.com/js/js_object_definition.asp>

## **This**

Wanneer This wordt aangeroepen dan gaat het over de nieuwe functie. Dus als er een nieuwe instantie wordt gemaakt met bv var player2 = new Player dan zal alles met this slaan op dit object en niet op player class op zichzelf.

### Voorbeeld

function Person(fn, ln) {

this.first\_name = fn;

this.last\_name = ln;

this.displayName = function() {

console.log(`Name: ${this.first\_name} ${this.last\_name}`);

}

}

let person = new Person("John", "Reed");

person.displayName(); // Prints Name: John Reed

let person2 = new Person("Paul", "Adams");

person2.displayName(); // Prints Name: Paul Adams

#### Bronnen voor meer informatie

<https://medium.com/better-programming/understanding-the-this-keyword-in-javascript-cb76d4c7c5e8>

<https://codeburst.io/all-about-this-and-new-keywords-in-javascript-38039f71780c#:~:text=Patro%20on%20Unsplash-,What%20is%20%E2%80%9Cthis%E2%80%9D%20keyword%20in%20JavaScript,how%20the%20function%20is%20called>.

<https://www.w3schools.com/js/js_this.asp>

## Let

Let kan gebruikt worden als variabele binnen een block scope. Dit wilt zeggen dat een let bv aangeroepen kan worden in een if loop binnen de {} Als de let daarbuiten gedefinieerd wordt zal er een error verschijnen.

### Voorbeeld

**var x = 10;**

**// Here x is 10**

**{**

**let x = 2;**

**// Here x is 2**

**}**

**// Here x is 10**

#### Bronnen voor meer informatie

<https://www.w3schools.com/js/js_let.asp>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let#:~:text=Description,function%20regardless%20of%20block%20scope>.

## Const

Geeft waarde aan een variabele die niet meer veranderd. Bijvoorbeeld speler snelheid. In dit geval is rennen niet van toepassing dus is beweging altijd dezelfde snelheid.

### Voorbeeld

const speed = 2.5;

speed = 3 → error

const person = { age: 20 };

person.age = 30; // OK

console.log(person.age); // 30

Even though the person variable is a constant, you can change the value of its property.

However, you cannot reassign a different value to the person constant like this:

person = {age: 40}; // TypeError

#### Bronnen voor meer informatie <https://www.javascripttutorial.net/es6/javascript-const/> <https://medium.com/javascript-scene/javascript-es6-var-let-or-const-ba58b8dcde75#:~:text=In%20JavaScript%2C%20%60const%60%20means,object%20can%20have%20properties%20mutated>.)

**Errors**

## Properties Tiled

Om ervoor te zorgen dat de bomen een andere collision hebben dan de stenen en de struiken (omdat de bomen een ander formaat zijn) heb ik de volgende code geprobeerd.

var setYOrigin = resource.properties.find(p=>p.name == 'setYOrigin').value;

resourceItem.x += resourceItem.width/2;

resourceItem.y += resourceItem.height/2;

resourceitem.y = resourceItem.y + reourceItem.height \* (setYOrigin - 0.5);

setYOrigin is de property naam in Tiled. Deze code werkt niet omdat find undefined is.

Het gebruik van <https://developer.aliyun.com/mirror/npm/package/phaser-tilemap-plus> deze plugin heeft ook geen effect.

Het voorbeeld wat hier staat: <https://phaser.io/examples/v2/tilemaps/tile-properties> heeft helaas ook geen effect.

Ook sites zoals stackoverflow en <https://www.html5gamedevs.com/topic/8415-tiled-tile-properties/> kwamen helaas niet met het antwoord.

Oplossing voor nu (13-10-2020 12:30)

Voor nu blijft de collision even staan. In principe werkt het alleen is het formaat nog niet naar behoren bij de bomen. Dit heeft verder geen impact op andere code.

### Oplossing

Binnen Tiled stond een instelling niet goed waardoor de .find de properties niet kon vinden omdat deze ook niet meegegeven zijn. Deze oplossing heeft ook meteen het probleem opgelost dat resource.type niet gevonden kon worden. Nu is het niet meer nodig om de objecten handmatig een naam te geven.

**setCollideWorldBounds()**

Tijdens het programmeren kwamen zowel Mijke als ik erachter dat setCollideWorldBounds niet werkend te krijgen is. Zo hebben we verschillende voorbeelden gepakt en deze geprobeerd.

### Oplossing

Niet perse een oplossing maar we zijn erachter gekomen dat setCollideWorldBounds alleen toegepast kan worden op de arcade game modes en niet op de Physics Matter game modes die ik hanteer voor de game. Dit omdat binnen de Physics Matter modes het gebruikelijk is dat je van de map kan lopen om vervolgens naar het volgende gebied te gaan. Het is niet handig om de game modes om te zetten omdat ik dan meer kwijt ben om alle code om te zetten en ik dan meer functionaliteiten kwijt ben. Als “tijdelijke” oplossing wordt er gebruik gemaakt van water om de map heen zodat de speler niet van de map kan lopen.

Code Dump

# Als JSON tileset niet werkt.

this.load.image('tiles','assets/images/tileSetNat.png');

this.load.tilemapCSV('map', 'assets/images/tileset3.csv');

}

create(){

// this.player = new Phaser.Physics.Matter.Sprite(this.matter.world, 0,0, 'female','townsfolk\_f\_idle\_1'); //creates player

const map = this.make.tilemap({key:'map', tileWidth:32, tileHeight:});

# Resources handmatig op de map zetten

var tree = new Phaser.Physics.Matter.Sprite(this.matter.world, 50, 50, 'resources', 'tree'); // Loads resources

var rock = new Phaser.Physics.Matter.Sprite(this.matter.world, 150, 150, 'resources', 'rock');

tree.setStatic(true); // Player can't move them when bumps into them

rock.setStatic(true);

this.add.existing(tree); // Adds to the map

this.add.existing(rock);

Arduino code

const int buttonPin01 = 13;

const int buttonPin02 = 12;

const int buttonPin03 = 10;

const int buttonPin04 = 9;

void setup() {

// put your setup code here, to run once:

Serial.begin(9600); // Starts the serial communication

pinMode(buttonPin01, INPUT);

pinMode(buttonPin02, INPUT);

pinMode(buttonPin03, INPUT);

pinMode(buttonPin04, INPUT);

digitalWrite(buttonPin01, HIGH);

digitalWrite(buttonPin02, HIGH);

digitalWrite(buttonPin03, HIGH);

digitalWrite(buttonPin04, HIGH);

}

void loop() {

// put your main code here, to run repeatedly:

if(digitalRead(buttonPin01) == LOW)

{

Serial.write(1);

Serial.flush();

delay(20);

}

if(digitalRead(buttonPin02) == LOW)

{

Serial.write(2);

Serial.flush();

delay(20);

}

if(digitalRead(buttonPin03) == LOW)

{

Serial.write(3);

Serial.flush();

delay(20);

}

if(digitalRead(buttonPin04) == LOW)

{

Serial.write(4);

Serial.flush();

delay(20);

}

}