Camelrace

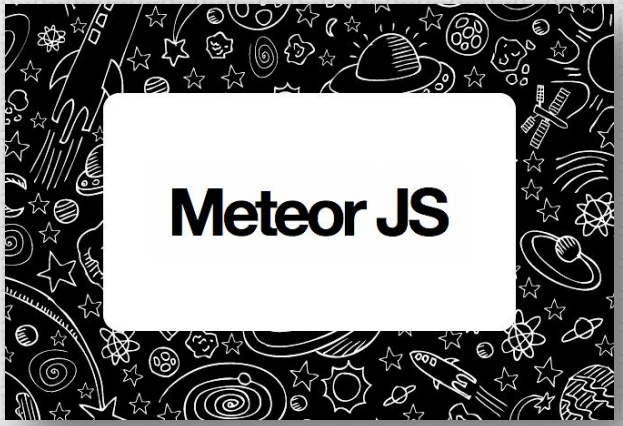
Matthew Demyttenaere
Robin Geldolf
Ewout Merckx

Index

- Problem
- Used technologies
- Realizations
- Demo
- Not realized
- Future / reflection
- Conclusion

Problem

Used technologies



=

Node.js™

+

Spark.js (Hot Code pushes)

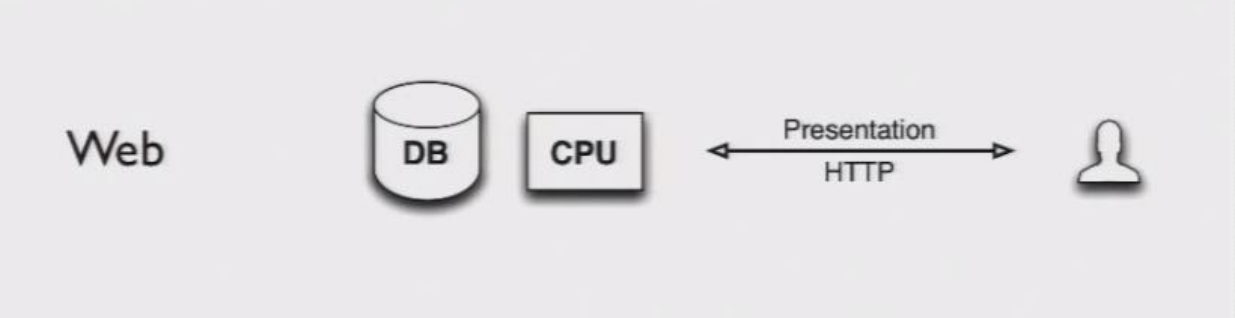
+

Meteor's Smart Packages & Meteorite

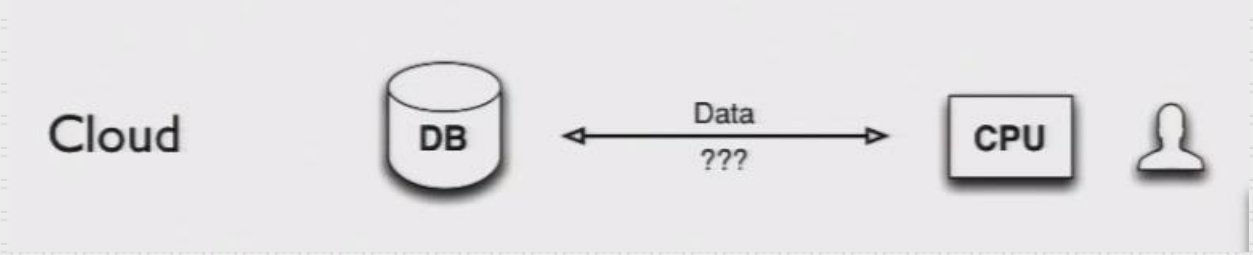
+

mongoDB

Used technologies



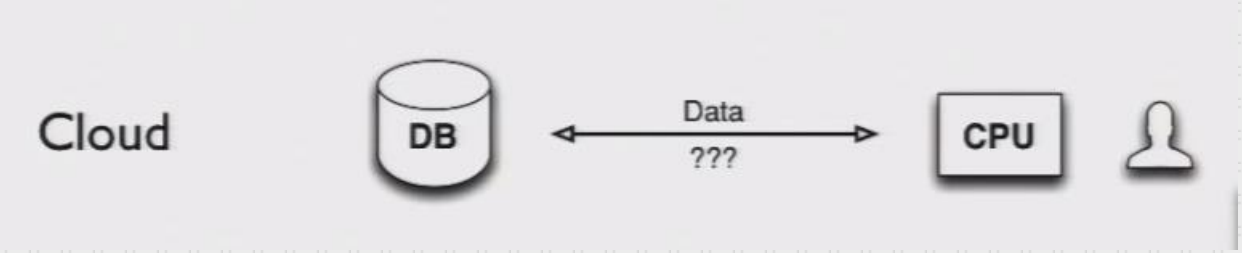
PHP / Rails



JSON / AJAX

Used technologies

Page is requesting data without a page refresh



DDP

```
// On the server
Meteor.publish('Games', function () {
              return Games.find();
});
```

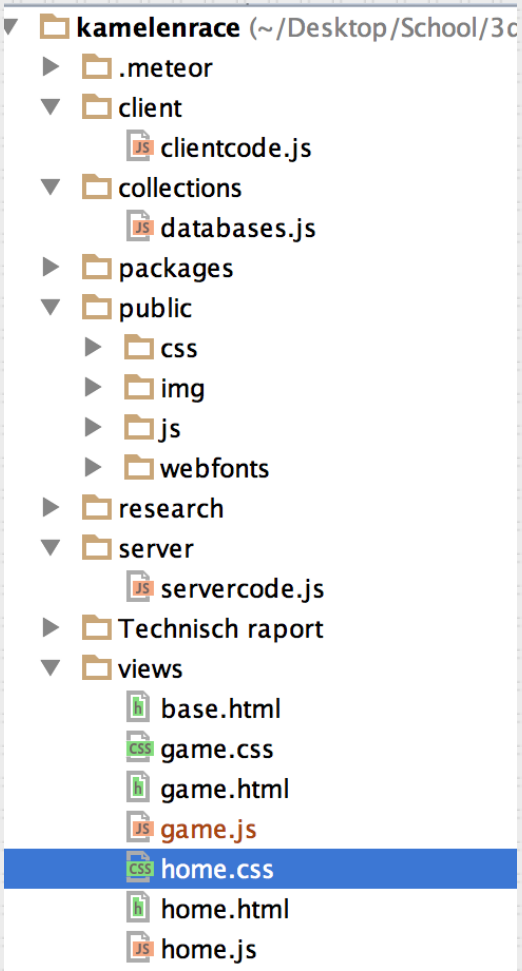
JSON & Distributed Data Protocol (DDP)

```
// On the client
Games = new Meteor.Collection('Games');
Meteor.subcribe('Games');
// On the client data is now in cache. Fast local DB
//  DDP now intelligently polls your database to pick up changes and push them down to
the client.
```

Used technologies

Smart package: meteor-router

```
// In the common code (clientcode.js)
  Meteor.Router.add({
        '/': 'home',


        '/game/:_id': { to: 'game', and: function (id) {
            Session.set('GameId', id);
        }},
        '*': 'not_found'
    });
```



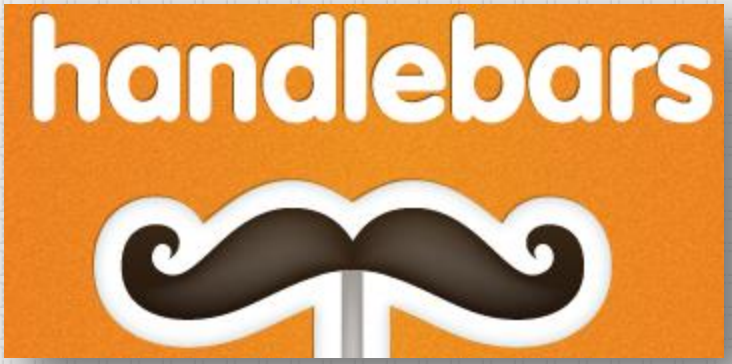
Used technologies

Meteor – Security

Server

```
// On the server
Games.allow({
   insert:function (userId, doc) {
      return (userId && doc.playerId === userId);
   },
   update: function (userId,doc) {
     return doc.playerId === userId;
   }
   remove: function (userId, doc) {
    return false ;
   }
});
```

Used technologies



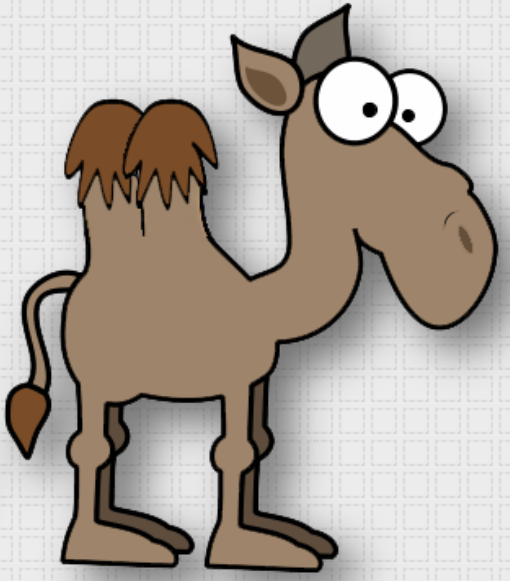

```
<template name="home">
    <svg id="TitleText"></svg>
    <section id="setUsernameBox">
    </section>

    {{> welcomeCamel}}

</template>

<template name="welcomeCamel">
    <svg id="camelRunningBy">
    </svg>
</template>
```

Used technologies

Snap.svg

Used technologies

Camel.svg



Camel.svg

Used technologies

Used technologies

chrome

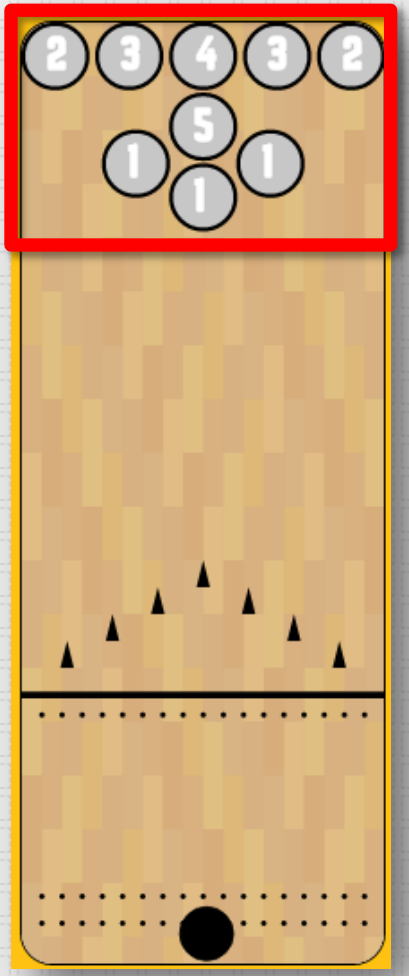
Realisations

Realisations

```
// Gives us a "class" of a circle where we can use all the properties
function Circle(middlepointX, middlepointY, radius, nr) {
    this.middlepointX = middlepointX;
    this.middlepointY = middlepointY;
    this.radius = radius;
    this.nr = nr;
}
```

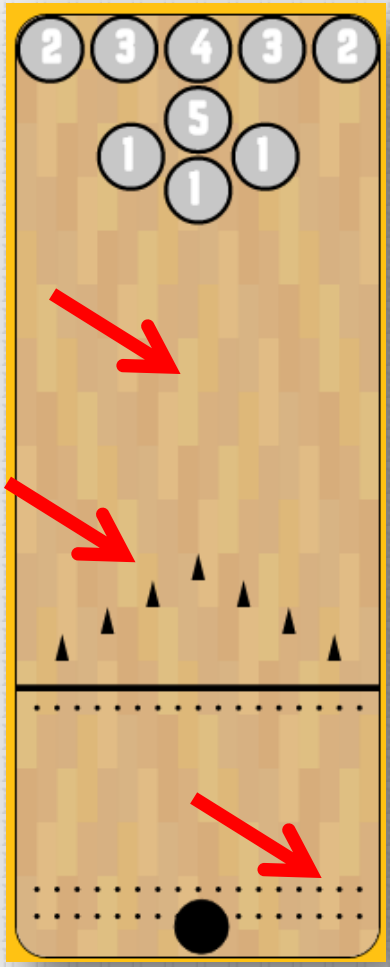
```
// the snap object with width and height
snapobj = Snap('#ballTrowLocation');
ballFieldWidth = $('#ballTrowLocation').width();
ballFieldHeight = $('#ballTrowLocation').height();

// we make all the holes
holesArray = new Array();
holesArray[0] = new Circle(ballFieldWidth / 2, 130, ballRadius + 4, '1');
```

```
//we show the holes at the top
var i;
for (i = 0; i < holesArray.length; i++) {
    var hole = snapobj.circle(holesArray[i].middlepointX, holesArray[i].middlepointY,
              holesArray[i].radius);
}
```



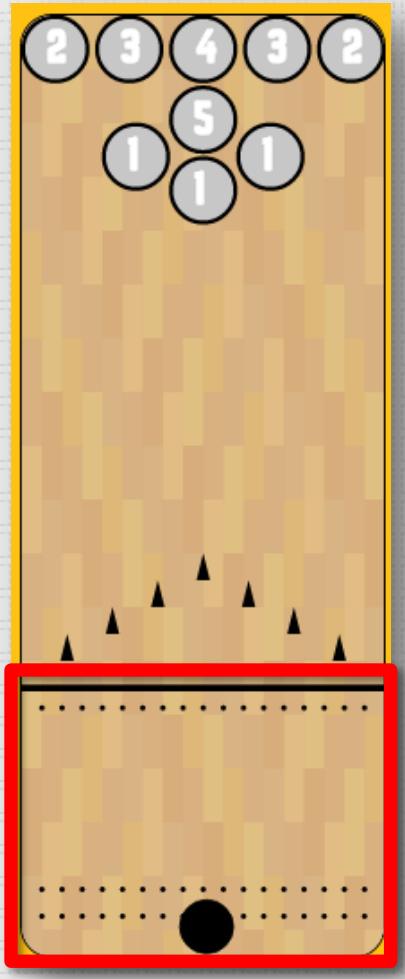
Realisations

```
function showBackground() {
    // Makes the wooden background
    var counter = 0;
    var i;
    for (i = 0; i < ballFieldWidth; i += 15) {

        // If i is even, create an indent of -120
        // and make then pieces of wood
        var j;
        for ((i % 30 == 0 ? j = 0 : j = -120) ;
             j < ballFieldHeight; j += 80) {
            var woodblock = snapobj.rect(i, j, 16, 80);
            var colour;
            switch (counter % 7) {
                case 0:
                    colour = '#D7AD7B'
                    break;
                // all the different colors
            }
            woodblock.attr({
                fill: colour
            });
            counter++;
        }
        // code for the dots
    }
    // code for the triangles
}
```



Realisations

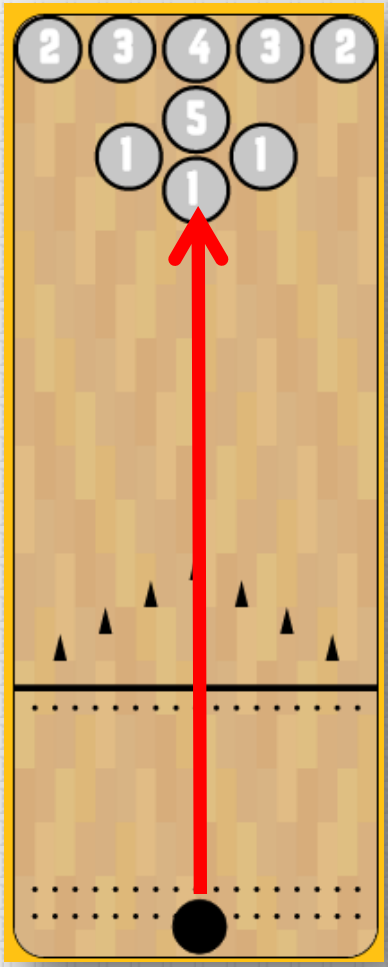
```
function hoverInCanvas() {
    // Eventlistener for when there is a click on the canvas
    canvas.mousedown(onMouseDown);
}

// Method for when the mouse is down
function onMouseDown(mouseEvent) {
    // If the mouse is on the ball, then you can move the ball
    if (getDistance(mouseEvent.offsetX, mouseEvent.offsetY, ball.node.cx.baseVal.value,
              ball.node.cy.baseVal.value) <= ballRadius
              && mouseEvent.offsetY > 450) {
        isMouseDown = true;
        oldX = ball.node.cx.baseVal.value;
        oldY = ball.node.cy.baseVal.value;
        oldTime = 0;
        time = Date.now();

        // add additional event listeners for dragging
        canvas.mouseup(onMouseUp);
        canvas.mousemove(onMouseMove);
    }
}

// When the mouse hovers out of the canvas, removes al the eventlisteners
// The ball will be thrown
function hoverOutCanvas() {
    onMouseUp();
    canvas.unmousedown(onMouseDown);
}
```



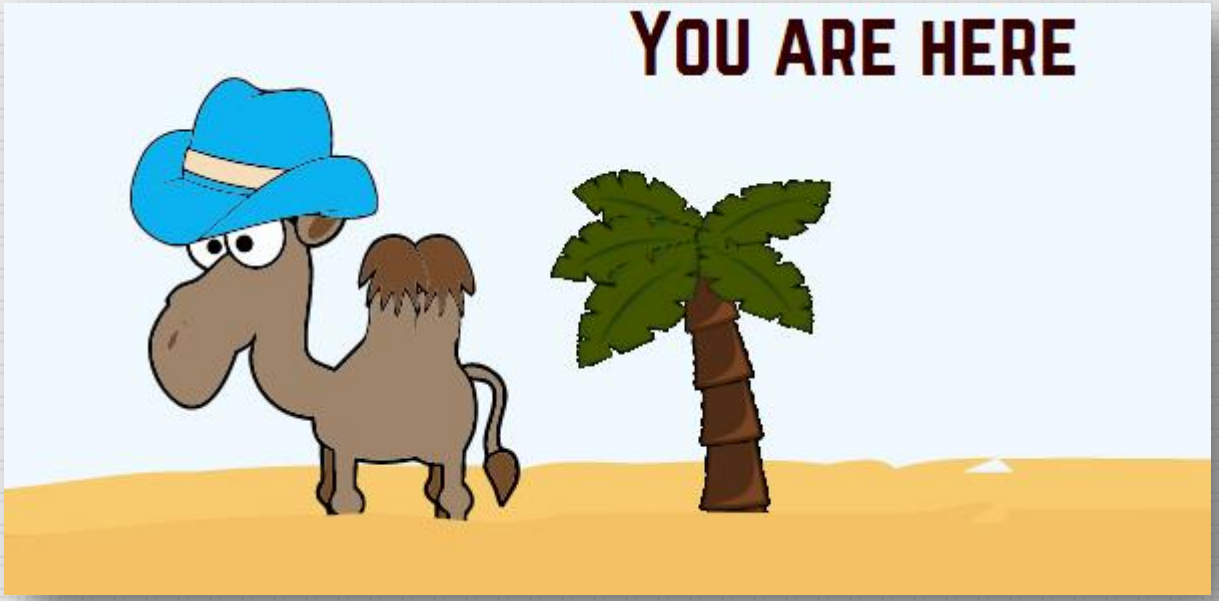
Realisations

```
// Check if the ball is above a hole
var i;
for (i = 0; i < holesArray.length; i++) {
    if (holesArray[i].radius > getDistance(ball.node.cx.baseVal.value,
ball.node.cy.baseVal.value, holesArray[i].middlepointX, holesArray[i].middlepointY)) {
        isAboveHole = true;
        console.log('The ball is above hole ' + holesArray[i].nr);
        ballGoesInHole(holesArray[i]);
        return;
    }
}
```

```
// The ball goes to the middle of the hole
function ballGoesInHole(hole) {
    console.log("Ball went into hole: " + hole.nr);
    //Get the current state of the game
    var tempGame = Games.findOne({}, { GameId: Session.get("GameId") });
    //Increase the current location
    tempGame.Players[Session.get("PlayerId")].CurrentLocation += parseInt(hole.nr) * 50;
    //Update the game in the DB
    Games.update(tempGame._id, tempGame);

    //Reset the ball location
    ball.animate({ cx: hole.middlepointX, cy: hole.middlepointY, r: 0 }, 1000);
    setTimeout(resetBall, 1000);
}
```



Realisations



Realisations

Creating field

```
//creating field
s = Snap("#backgroundRaceField");
```

Add text to field

```
s.text(320, 15, "You are here").attr({ fill: "#300","font-size": "16px" });
```

Add image to field

```
Snap.load("../img/BlueCamel.svg", onBlueCamelSVGLoaded);
function onBlueCamelSVGLoaded(f) {
    camelBlue = s.group().transform(startLocationBlue).append(f);
```

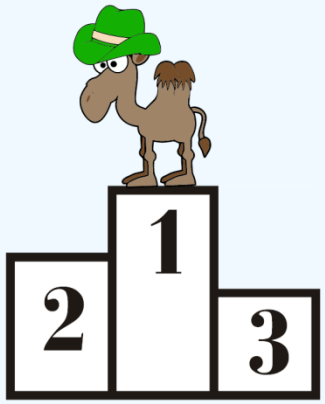
Animate image

```
this.animate({
    transform: "t" + [590 - currentGame.Players[index].CurrentLocation, ys[index]] + "s" + [0.25]
}, parseInt(2000), function checkwin() {
```

Realisations

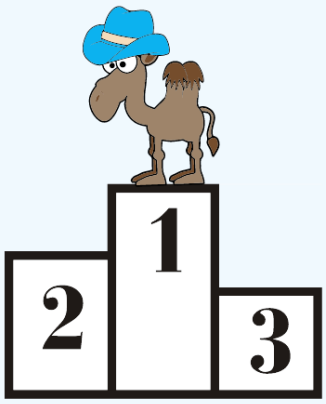
You win!

NEW GAME...



You win!

NEW GAME...



YOU LOSE ,BETTER LUCK NEXT TIME!

NEW GAME...

Realisations

Check finished

```
if (currentGame.Players[index].CurrentLocation >= 590) {
```

Check win/lose

```
if (currentGame.Players[index].PlayerId == (parseInt(Session.get("PlayerId")))) {
```

Add win/lose message

```
var anim = s.group().transform("t" + [590 / 2 - 50, 0] + "s" + [0.25]).append(f);
```

```
var lostMessage = s.text(70, 250, "You lose ,better luck next time!").attr({
```

New game

```
var btn = s.text(270, 400, "New game...").attr({
    fill: "#900",
    "font-size": "50px"

});
```

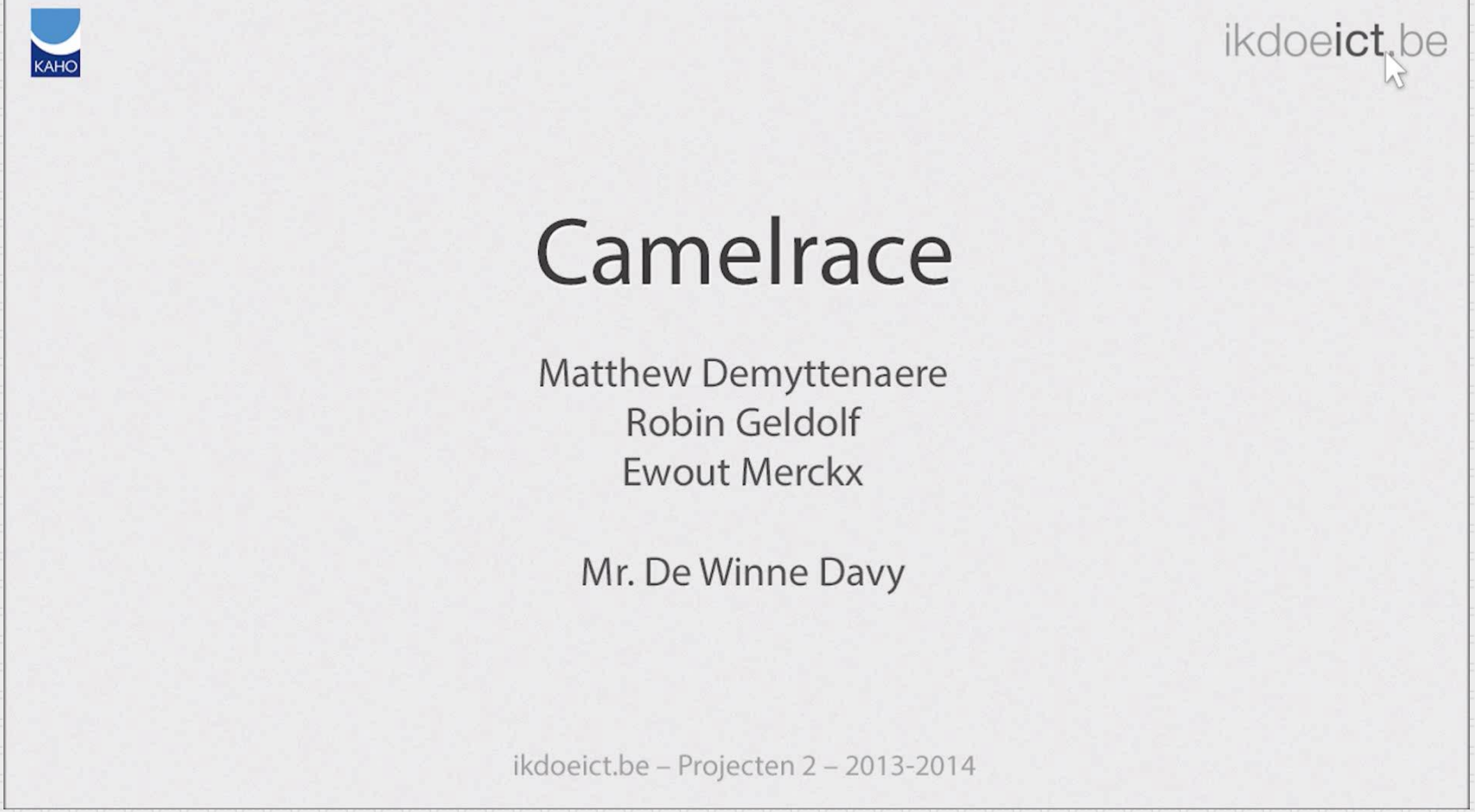
```
btn.click(function () {
    Meteor.Router.to('/');

});
```

Realisations

EWOUT: HELLO
ROBIN: HI
EWOUT: GL & HF !

your message

Demo


KAHO
ikdoeict.be
Camelrace
Matthew Demyttenaere
Robin Geldolf
Ewout Merckx
Mr. De Winne Davy
ikdoeict.be – Projecten 2 – 2013-2014

Not realized

Future and reflection

- Integration on smart phone
- Responsive design
- Play vs. Computer/AI

Conclusion

- A multiplayer game
- Working with meteor and node.js
- SVG images
- HTML5 + CSS3 + JS
- MongoDB