serious #**js**

to **infinity** and beyond!

plaatje: http://www.disneypictures.tk/disney-pictures/buzz-lightyear-pictures.html

welke vragen heb je over de vids

some concepts

Athlete

dayOfBirth
name
surname
stamina
strength
length

http://1.bp.blogspot.com/-G43gdM3tW1M/T2c0_siXA_I/AAAAAAAADmU/yG351RcUuxI/s1600/sporty+female+athlete.jpg

We gaan in op **encapsulation** en denken na over hoe we data in object kunnen **setten** en **getten**

Defining properties

```
var person1 = {
    name: "Nicholas"
};

var person2 = new Object();
person2.name = "Nicholas";


person1.age = "Redacted";

person2.age = "Redacted";


person1.name = "Greg";

person2.name = "Michael";
```

Defining properties

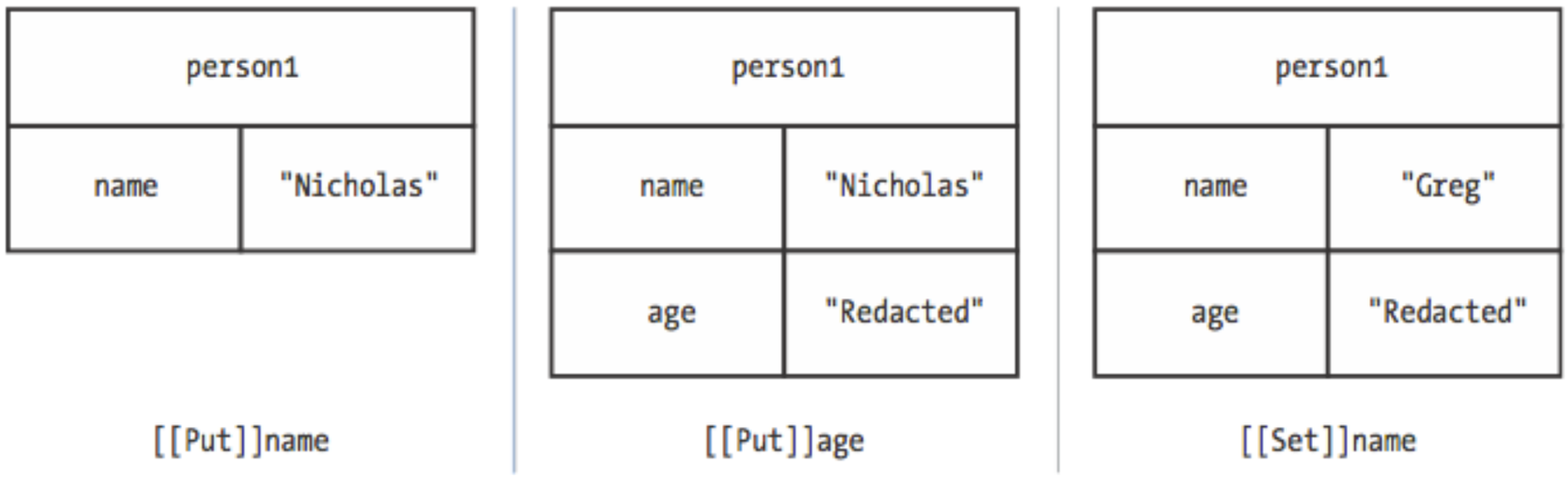

Figure 3-1: Adding and changing properties of an object

Detecting proporties

```
var person1 = {
    name: "Nicholas",
    sayName: function() {
        console.log(this.name);
    }
};

console.log("name" in person1);

console.log(person1.hasOwnProperty("name"));

console.log("toString" in person1);


console.log(person1.hasOwnProperty(“toString"));
```

assignment

Maak nu eens Athlete object met 1 property, ‘name’
met een assessor en mutator.
Gebruik: `object.defineProperty()`

Probeer nu eens meerder properties toe te voegen.

Detecting proporties

```
var person1 = {
    name: "Nicholas"
};

console.log("name" in person1);
console.log(person1.propertyIsEnumerable(“name"));

var properties = Object.keys(person1);

console.log("length" in properties);
console.log(properties.propertyIsEnumerable("length"));
```

Property attributes

```
var person1 = {
    _name: "Nicholas",
    get name() {
          console.log("Reading name”);
          return this._name;
    },
    set name(value) {
          console.log("Setting name to %s", value);
        this._name = value;
    }
};

console.log(person1.name);
person1.name = "Greg";
console.log(person1.name);
```

Accessor property attributes

```
var person1 = {
    _name: "Nicholas"
};

Object.defineProperty(person1, "name", {
    get: function() {
        console.log("Reading name");
        return this._name;
    },
    set: function(value) {
        console.log("Setting name to %s", value);
        this._name = value;
    },
    enumerable: true,
    configurable: true
});
```

Defining multiple properties

```
var person1 = {};

Object.defineProperties(person1, {

// data property to store data
    _name: {
        value: “Nicholas",
        enumerable: true,
        configurable: true,
        writable: true
    },

                                    // accessor property
    name: {
        get: function() {
            console.log("Reading name");
            return this._name;
        },
        set: function(value) {
            console.log("Setting name to
                %s", value);
                this._name = value;
        },
        enumerable: true,
        configurable: true
    }
})
```

Prevent Object Modification

```
var person1 = {
        name: "Nicholas"
};
console.log(Object.isExtensible(person1));

Object.preventExtensions(person1);
console.log(Object.isExtensible(person1));

person1.sayName = function() {
    console.log(this.name);
};

console.log("sayName" in person1);
```

assignment

Voeg aan Athlete de volgende properties toe:
dayOfBirth, surname, stamina, strength, length

Set bewust de enumerable, configurable en writable.

Gebruik: `object.defineProperties()` **en** `Set, Get.`

assignment

Voeg een simpele factory patterns toe om objecten te maken.

assignment

Verbeter je app. Probeer OOP principe van encapsulation (volgens ECMA5) zoveel mogelijk toe te passen.

Deel het via GITHUB.