Teaching Presentation for CIT 261-week 2

JavaScript Objects

- Object Creation Functions
- Inheritance
- Properties
- Methods
- Instantiation

Cristina Irwin

JavaScript Object

An **object** is anything in the world that has these four things: **Identity** *(A unique name or identification number)*, **Data** *(These are characteristics of an object and are usually adjectives. These are also called attributes, properties, characteristics, fields, instance variables , or structure.)*, **Behavior** *(These are the things an object does and are usually verbs. These are also called functions, operations, methods, or procedures.)* and **Relationships** *(These are links or connections between objects).*

-The Handbook for Beginning programmers

Example of a JavaScript Object

var car = "Fiat"; This code assigns a simple value (Fiat) to a variable named car. **Car** is the object.

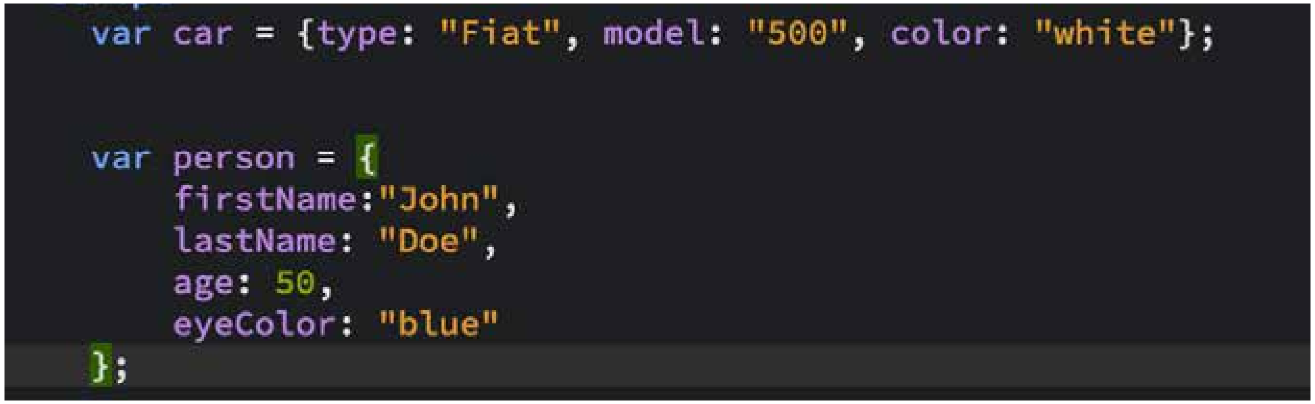
JavaScript objects are containers for named values called properties and methods. We will talk about those later.

https://www.w3schools.com/Js/js_objects.asp

JavaScript Object

Objects are variables, but they can contain values

Two ways Objects can be written with **name value pairs**



```
var car = {type: "Fiat", model: "500", color: "white"};


var person = {
    firstName:"John",
    lastName: "Doe",
    age: 50,
    eyeColor: "blue"
};
```

These are known as **name value pairs.** JavaScript objects are containers for named values called *properties* and *methods.*

Object Creation Functions

From w3schools

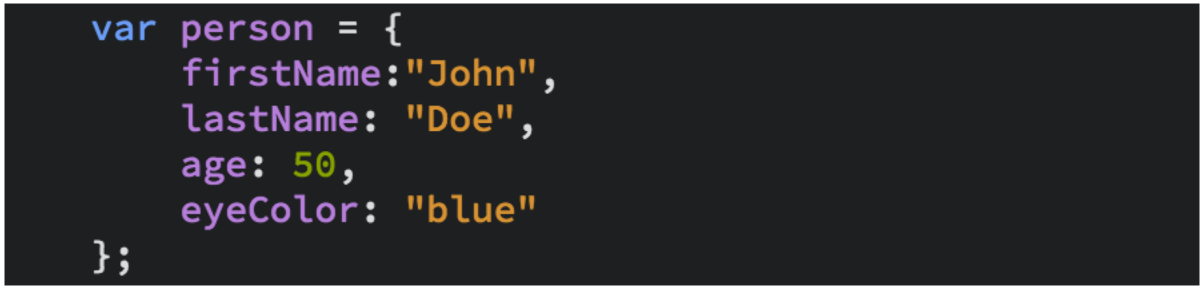
There are different ways to create a JavaScript Object.

1. Define and create a single object, using an **object literal.**
2. Define and create a single object, with the **keyword new.**
3. Define an **object constructor,** and then create objects of the constructed type.
4. Use and **Object.create().**

Object Literals

Object Creation Functions

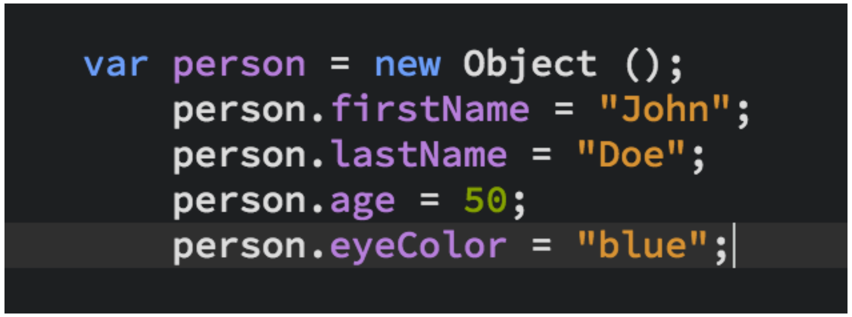
The most common way to create a JavaScript object is using an **object literal**. Literals are *global objects* which means they can be used throughout the program.



```
    var person = {
        firstName:"John",
        lastName: "Doe",
        age: 50,
        eyeColor: "blue"
    };
```

JavaScript Keyword *new*

Object Creation Functions



```
var person = new Object ();
    person.firstName = "John";
    person.lastName = "Doe";
    person.age = 50;
    person.eyeColor = "blue";
```

There is no need to use new Object().
For simplicity, readability and execution speed, use the object literal method.

JavaScript *Object Constructors*

Object Creation Functions

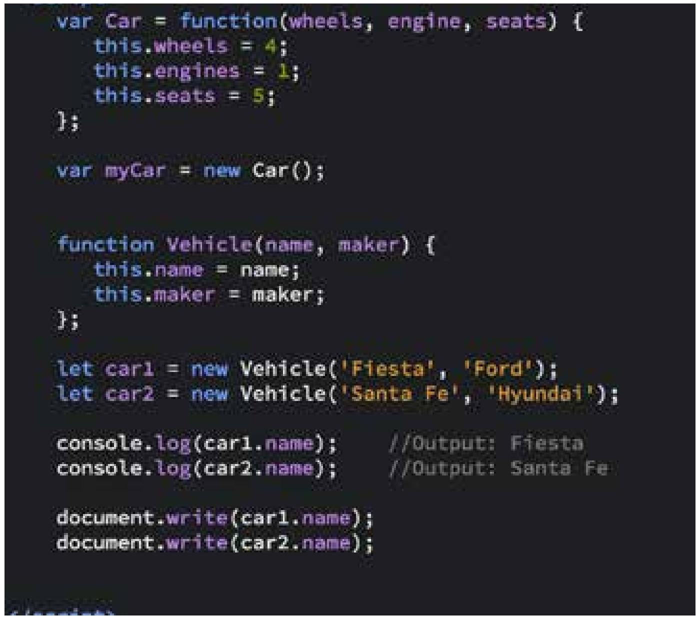
A constructor is a function that with the help of a new keyword *(this)* allows multiple objects to be created.

this and *new* keywords

this is used when in its a specific new object. All the references to this inside a constructor are referring to the new object

.

Use the *new* keyword to call a constructor. This is how JavaScript knows to create a new object.



```
var Car = function(wheels, engine, seats) {
   this.wheels = 4;
   this.engines = 1;
   this.seats = 5;
};

var myCar = new Car();


function Vehicle(name, maker) {
   this.name = name;
   this.maker = maker;
};

let car1 = new Vehicle('Fiesta', 'Ford');
let car2 = new Vehicle('Santa Fe', 'Hyundai');

console.log(car1.name);    //Output: Fiesta
console.log(car2.name);    //Output: Santa Fe

document.write(car1.name);
document.write(car2.name);
```

JavaScript *Object.create()*

Object Creation Functions

www.htmlgoodies.com

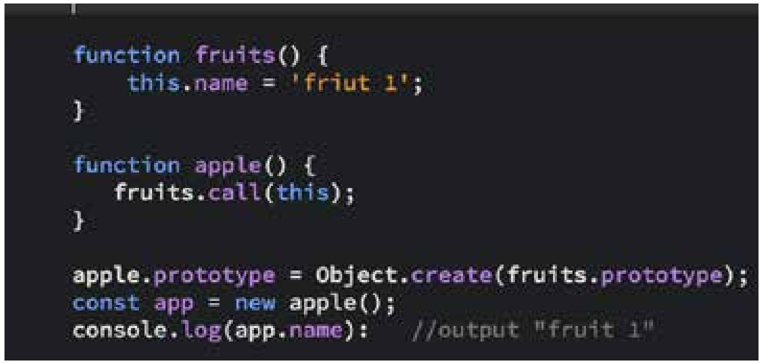
Object.create() is an excellent choice for creating an object without going through its constructor.

www.adripofjavascript.com

Using Object.create makes setting up inheritance chains simple because any object can be used as a prototype. Object.create is a feature of ECMAScript 5 and is not available in older browsers like IE8.

www.geeksforgeeks.org

Object.create() method is used to create a new object with the specified prototype object and properties. Object.create() method returns a new object with the specified prototype object and properties. Object.create() is used for implementing inheritance.



```
function fruits() {
    this.name = 'friut 1';
}

function apple() {
   fruits.call(this);
}

apple.prototype = Object.create(fruits.prototype);
const app = new apple();
console.log(app.name);   //output "fruit 1"
```

Inheritance

When an **object** *inherits* all the properties and methods of its parent object in order to become a specialized version of the parent object. In JavaScript the prototype property is used to establish subclasses that will *inherit* the main characteristics of the main parent classes. This results in a specialized subclass that can retain its own properties and methods as well as *inherit* all the properties and methods of the parent class.

www. adamkhoury.com

There are no classes in JavaScript, so inheritance is prototyped based. This means to implement inheritance in JavaScript, an object inherits from another object.

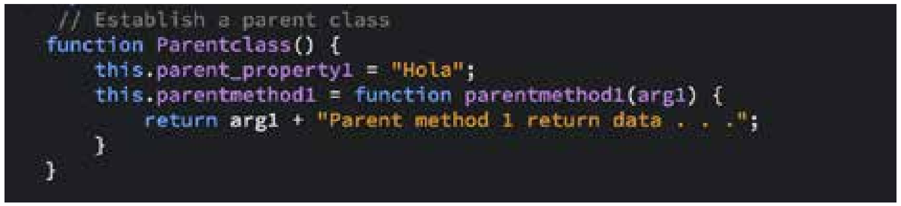
W3schools

All JavaScript objects inherit properties and methods from a prototype. Date objects inherit from Date prototype. Array objects inherit from Array.prototype. Person objects inherit from Person.prototype.

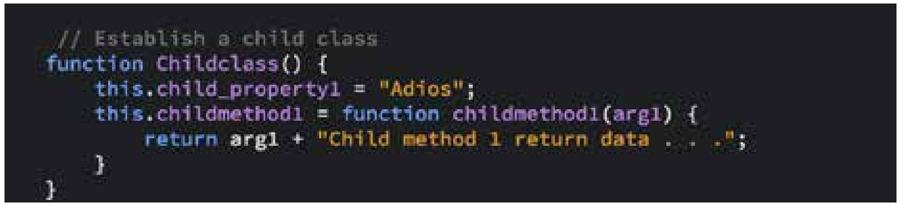
MDN Web Docs

All objects in JavaScript inherit from at least one other object. The object being inherited from is known as the prototype, and the inherited properties can be found in the prototype object of the constructor.

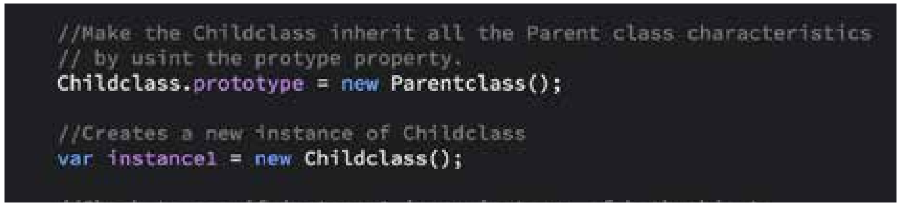
Inheritance



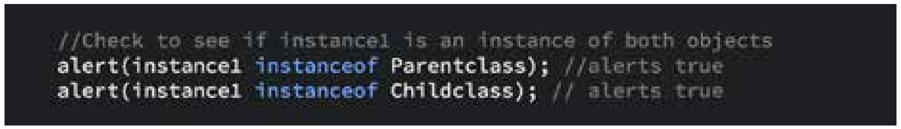
```
 // Establish a parent class
function Parentclass() {
    this.parent_property1 = "Hola";
    this.parentmethod1 = function parentmethod1(arg1) {
        return arg1 + "Parent method 1 return data . . .";
    }
}
```



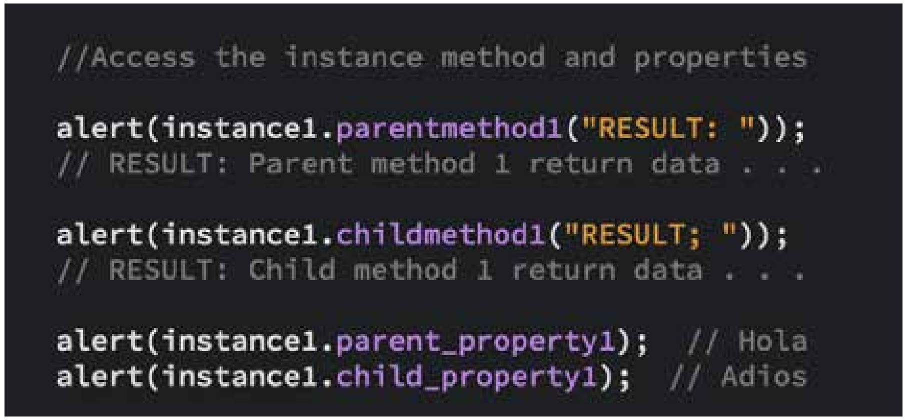
```
 // Establish a child class
function Childclass() {
    this.child_property1 = "Adios";
    this.childmethod1 = function childmethod1(arg1) {
        return arg1 + "Child method 1 return data . . .";
    }
}
```



```
 //Make the Childclass inherit all the Parent class characteristics
 // by usint the protype property.
 Childclass.prototype = new Parentclass();

 //Creates a new instance of Childclass
 var instance1 = new Childclass();
```



```
//Check to see if instance1 is an instance of both objects
alert(instance1 instanceof Parentclass); //alerts true
alert(instance1 instanceof Childclass); // alerts true
```



```
//Access the instance method and properties

alert(instance1.parentmethod1("RESULT: "));
// RESULT: Parent method 1 return data . . .

alert(instance1.childmethod1("RESULT; "));
// RESULT: Child method 1 return data . . .

alert(instance1.parent_property1);  // Hola
alert(instance1.child_property1);  // Adios
```

Objects have Properties

Properties

w3schools.com
Properties are the values associated with a JavaScript object. A JavaScript object is a collection of unordered properties. Properties can usually be changed, added, and deleted, but some are read only.

mmtus
Property is a way to get information regarding a specific variable.

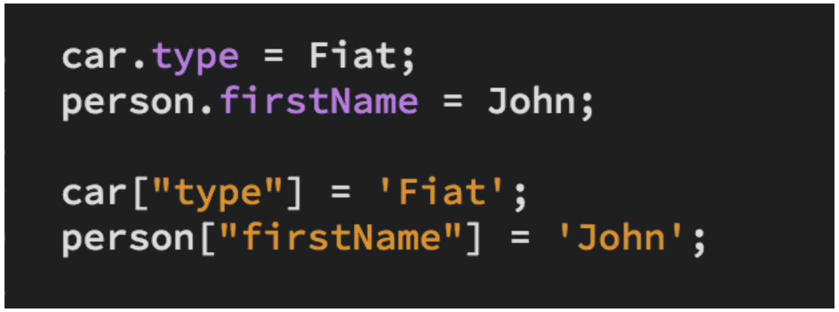
Zenva-https://www.youtube.com/watch?v=mxlcjTB_i98
Objects have different properties. Some properties can be numbers, strings and arrays. Properties can be functions.

Properties

objectname.property name

obj	Property	Property Value
car	type	Fiat
	model	500
	color	white
person	firstName	John
	lastName	Doe
	age	50
	eyeColor	blue

You can access the properties by either



```
car.type = Fiat;
person.firstName = John;

car["type"] = 'Fiat';
person["firstName"] = 'John';
```

Objects have Methods

Methods

Allows us to do something to the data inside the container

Zenva-https://www.youtube.com/watch?v=mxlcjTB_i98

Objects have different properties. Some properties can be numbers, strings and arrays. Properties can be functions. Functions are just another variable added to an object. Those functions are called Methods.

obj	Property	Property Value	
person	**fullName**	function() {return this.firstName + " " + this.lastName;}	**method**
	firstName	John	**property**
	lastName	Doe	**property**

Objects have Methods

Methods

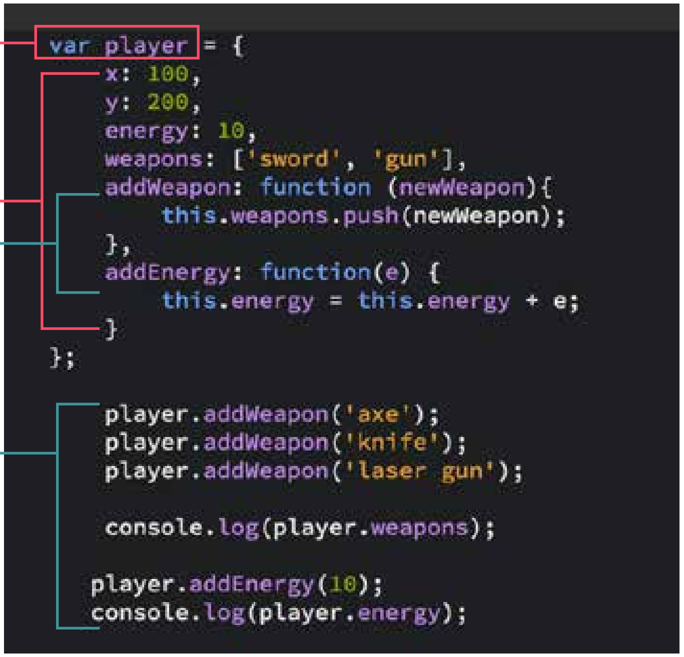
Allows us to Do something to the data inside the container

Zenva-https://www.youtube.com/watch?v=mxlcjTB_i98

console.log is an example of a method that is available to us while working on the browser. It is a method that shows things on the console. We can create our own method. Methods are where we bring together objects and functions and it basically giving superpowers to our objects so we can do anything that we want by having their own properties as functions. You access the object properties by using the key word *this.*

Objects have Methods

Methods

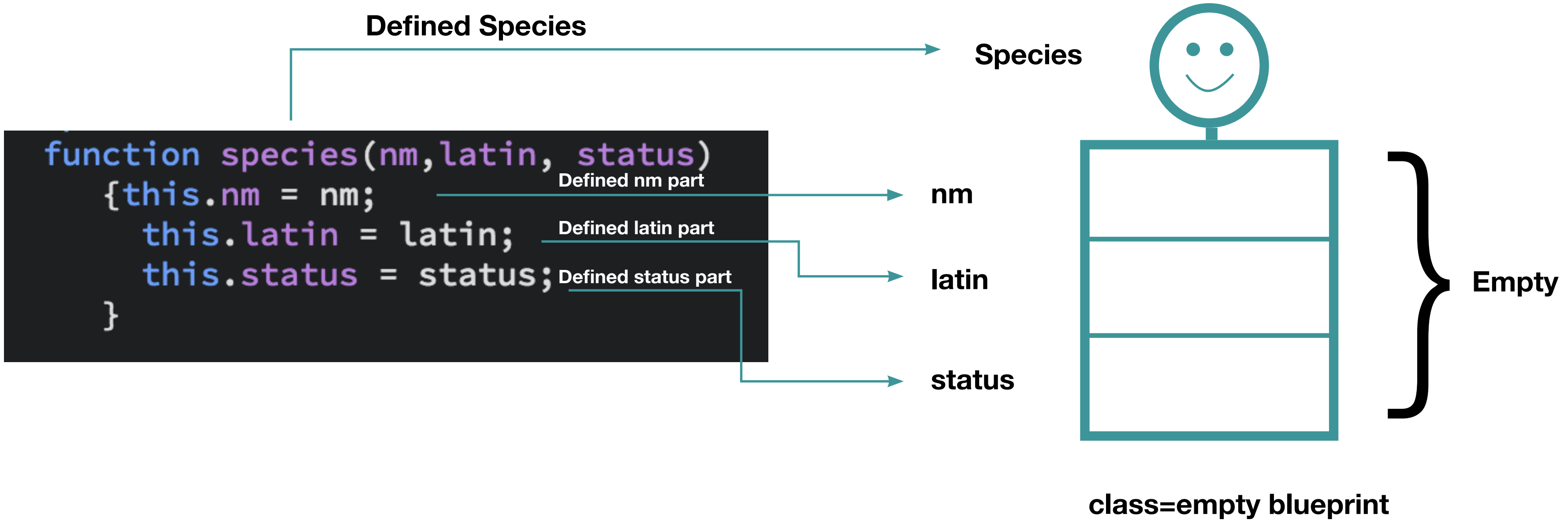


```
var player = {
    x: 100,
    y: 200,
    energy: 10,
    weapons: ['sword', 'gun'],
    addWeapon: function (newWeapon){
        this.weapons.push(newWeapon);
    },
    addEnergy: function(e) {
        this.energy = this.energy + e;
    }
};

    player.addWeapon('axe');
    player.addWeapon('knife');
    player.addWeapon('laser gun');

    console.log(player.weapons);

   player.addEnergy(10);
   console.log(player.energy);
```

Think of an instance of

Instantiation

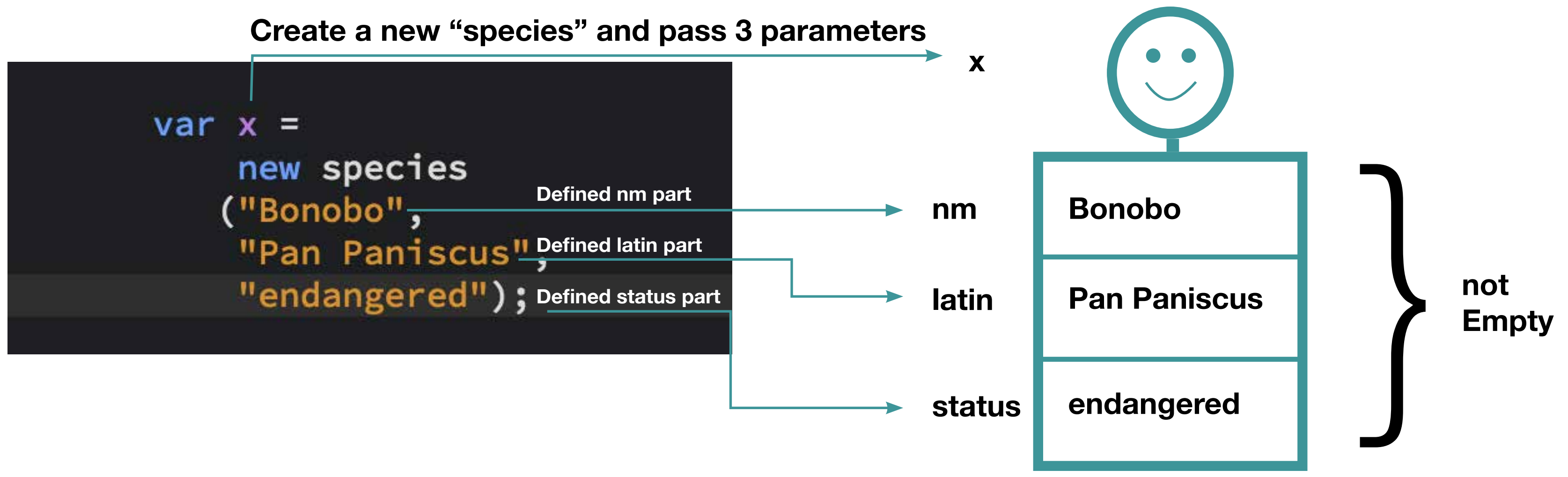
Create a new version



```
function species(nm,latin, status)
   {this.nm = nm;
     this.latin = latin;
     this.status = status;
   }
```

Think of an instance of

Instantiation


```
var x =
    new species
   ("Bonobo",
    "Pan Paniscus",
    "endangered");
```