CSC-318 Web Technology

(BSc CSIT, TU)

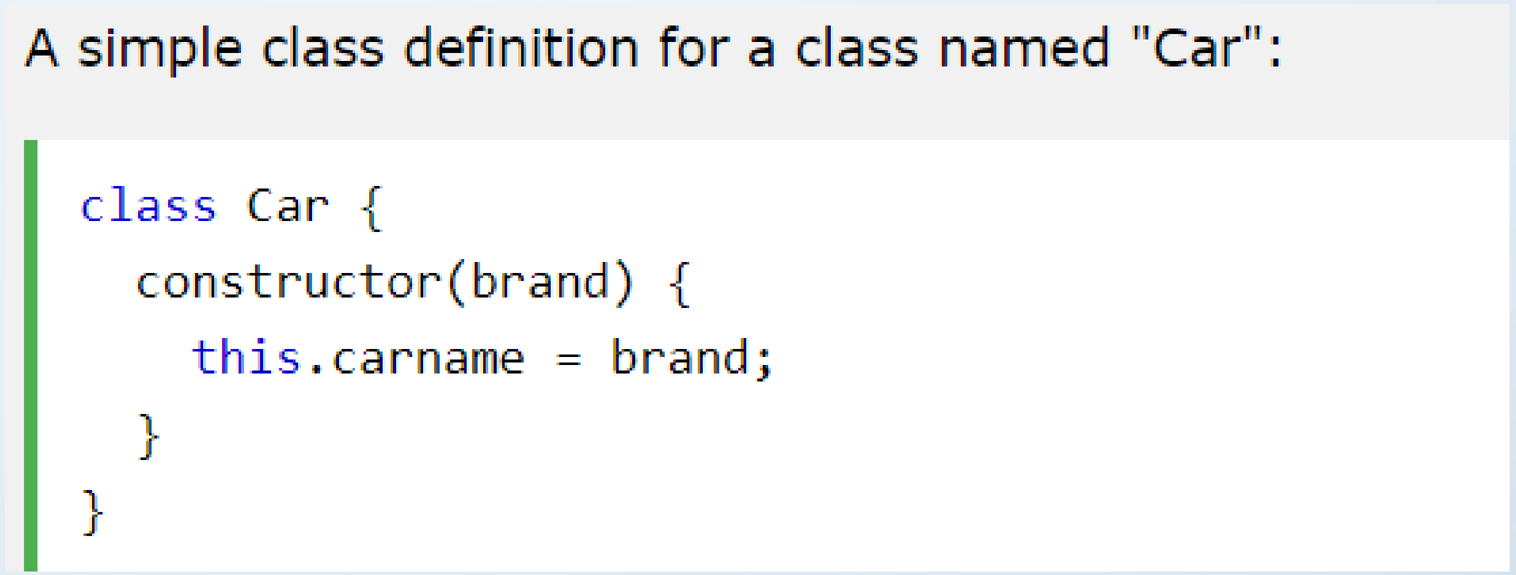
Ganesh Khatri
kh6ganesh@gmail.com

JavaScript Classes

- ES6, also known as ECMAScript2015, introduced classes
- A class is a type of function, but instead of using the keyword function to initiate it, we use the keyword class, and the properties are assigned inside a constructor() method

Class Definition

- Use the keyword class to create a class, and always add the constructor() method
- The constructor method is called each time the class object is initialized

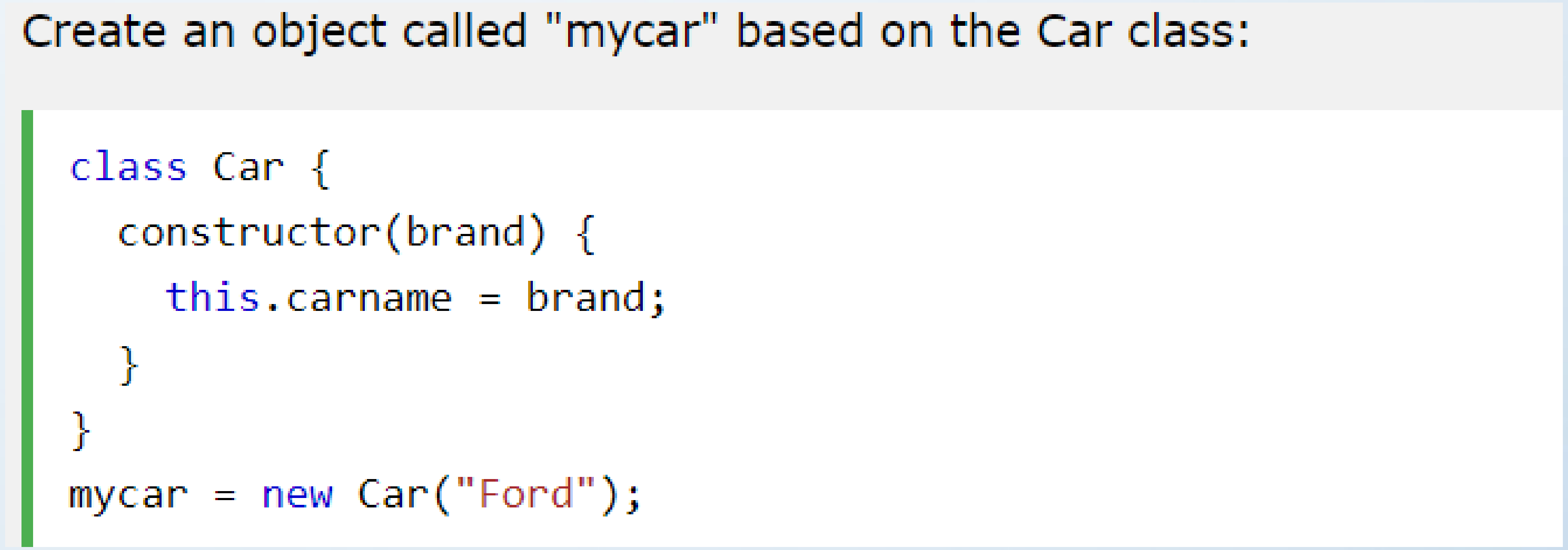


A simple class definition for a class named "Car":

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
}
```

Class Definition

- Now you can create objects using the Car class:
- Create an object called "mycar" based on the Car class:



Create an object called "mycar" based on the Car class:

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
}
mycar = new Car("Ford");
```

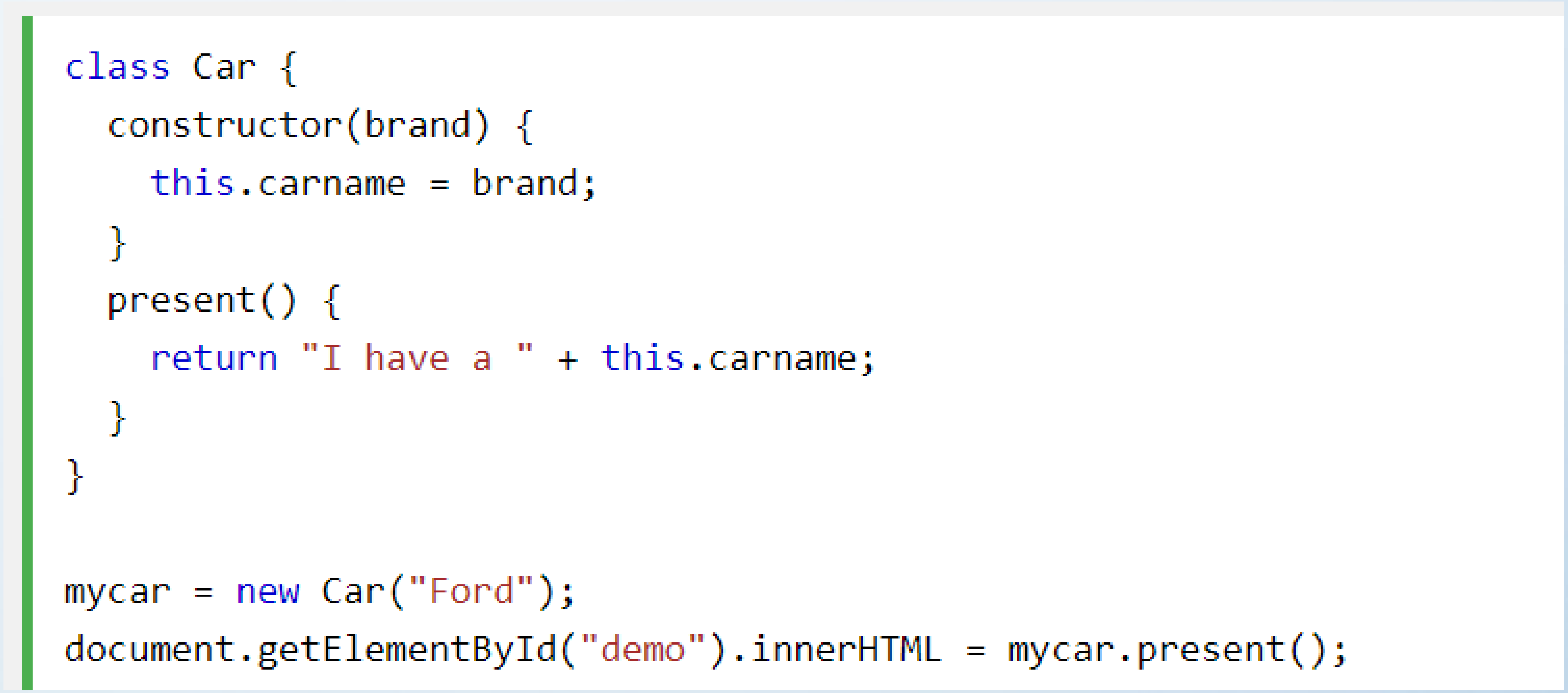
- **Note:** Constructor method is called automatically when the object is initialized

Methods

- The constructor method is special
- it is called automatically when a class is initiated
- it has to have the exact name "constructor", in fact, if you do not have a constructor method, JavaScript will add an invisible and empty constructor method
- You are also free to make your own methods, the syntax should be familiar

Methods

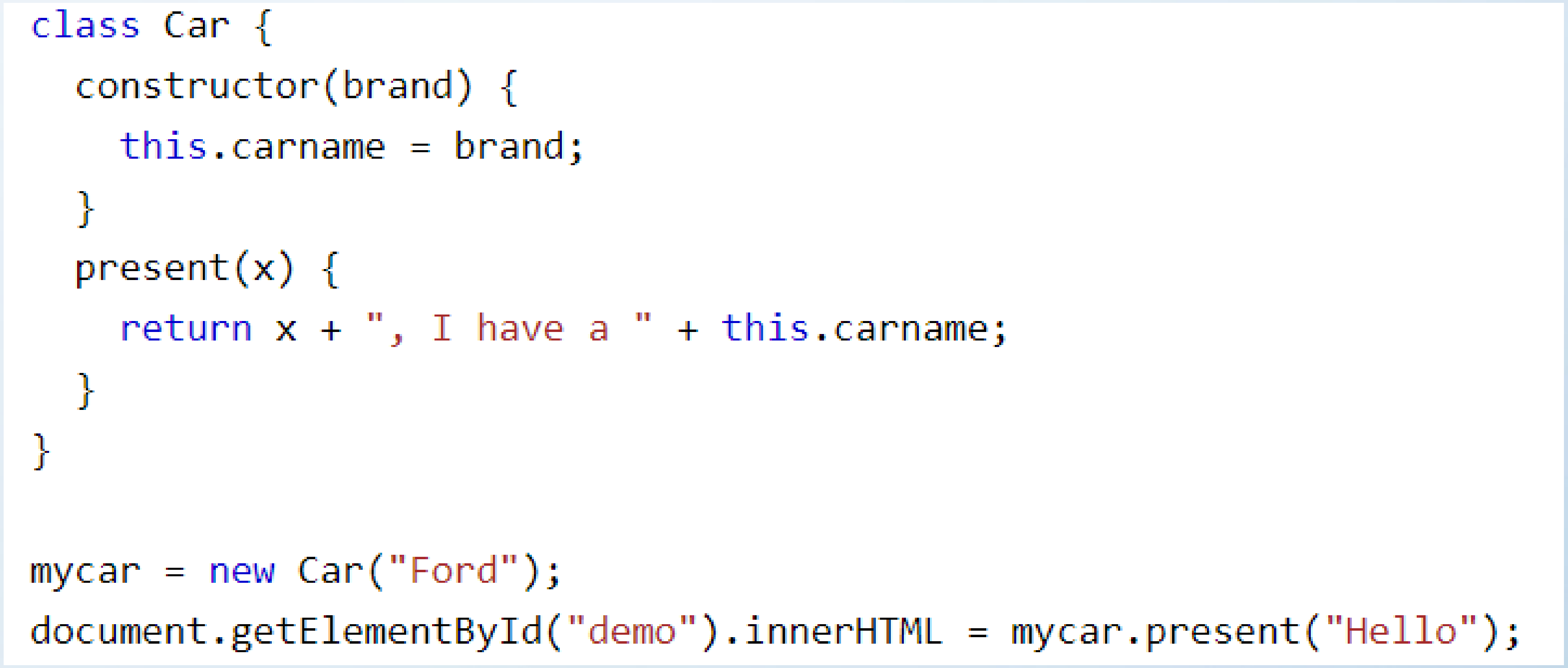
- Create a method named "present"



```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present() {
    return "I have a " + this.carname;
  }
}

mycar = new Car("Ford");
document.getElementById("demo").innerHTML = mycar.present();
```

Methods

- you call the method by referring to the object's method name followed by parentheses (any parameters would go inside the parentheses)
- Send a parameter to the "present()" method:



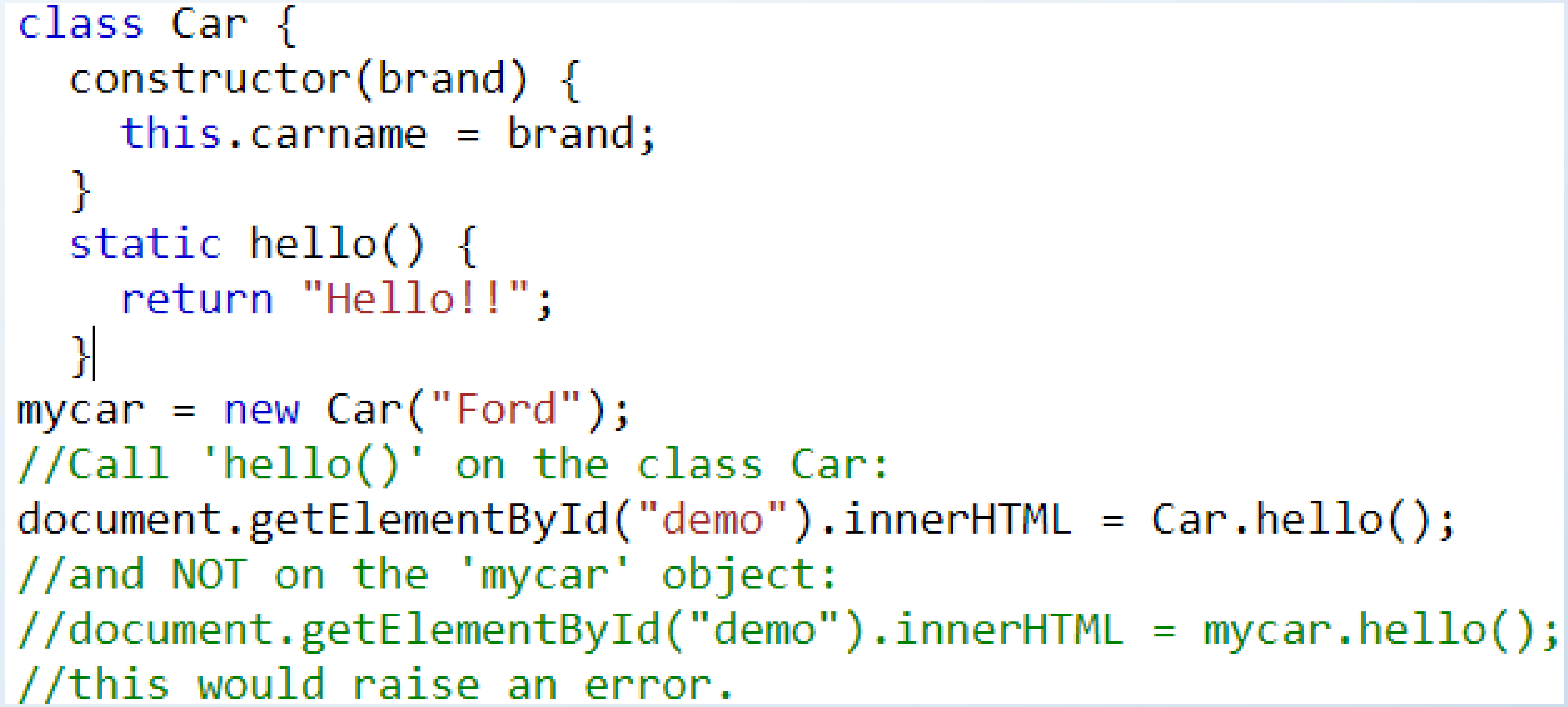
```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present(x) {
    return x + ", I have a " + this.carname;
  }
}

mycar = new Car("Ford");
document.getElementById("demo").innerHTML = mycar.present("Hello");
```

Static Methods

- Static methods are defined on the class itself, and not on the prototype
- That means you cannot call a static method on the object (mycar), but on the class (Car)

Static Methods

- Example : Create a static method and call it on the class:



```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  static hello() {
    return "Hello!!";
  }
mycar = new Car("Ford");
//Call 'hello()' on the class Car:
document.getElementById("demo").innerHTML = Car.hello();
//and NOT on the 'mycar' object:
//document.getElementById("demo").innerHTML = mycar.hello();
//this would raise an error.
```

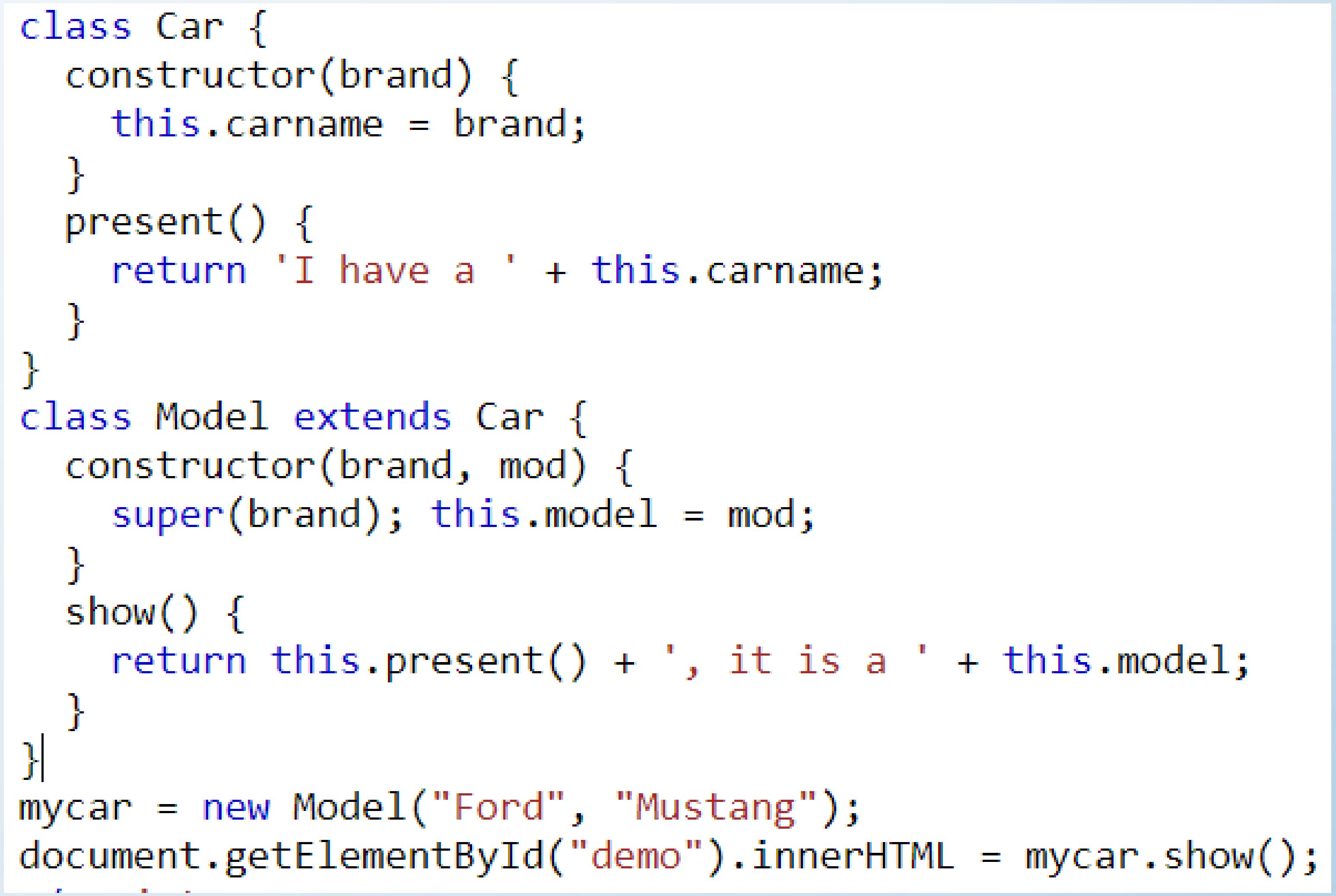
Static Methods

- If you want to use the mycar object inside the static method, you can send it as a parameter:

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  static hello(x) {
    return "Hello " + x.carname;
  }
}

mycar = new Car("Ford");

document.getElementById("demo").innerHTML = Car.hello(mycar);
```

Inheritance

- To create a class inheritance, use the extends keyword
- A class created with a class inheritance inherits all the methods from another class



```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present() {
    return 'I have a ' + this.carname;
  }
}
class Model extends Car {
  constructor(brand, mod) {
    super(brand); this.model = mod;
  }
  show() {
    return this.present() + ', it is a ' + this.model;
  }
}
mycar = new Model("Ford", "Mustang");
document.getElementById("demo").innerHTML = mycar.show();
```

Inheritance

- The super() method refers to the parent class
- By calling the super() method in the constructor method, we call the parent's constructor method and gets access to the parent's properties and methods
- Inheritance is useful for code reusability: reuse properties and methods of an existing class when you create a new class

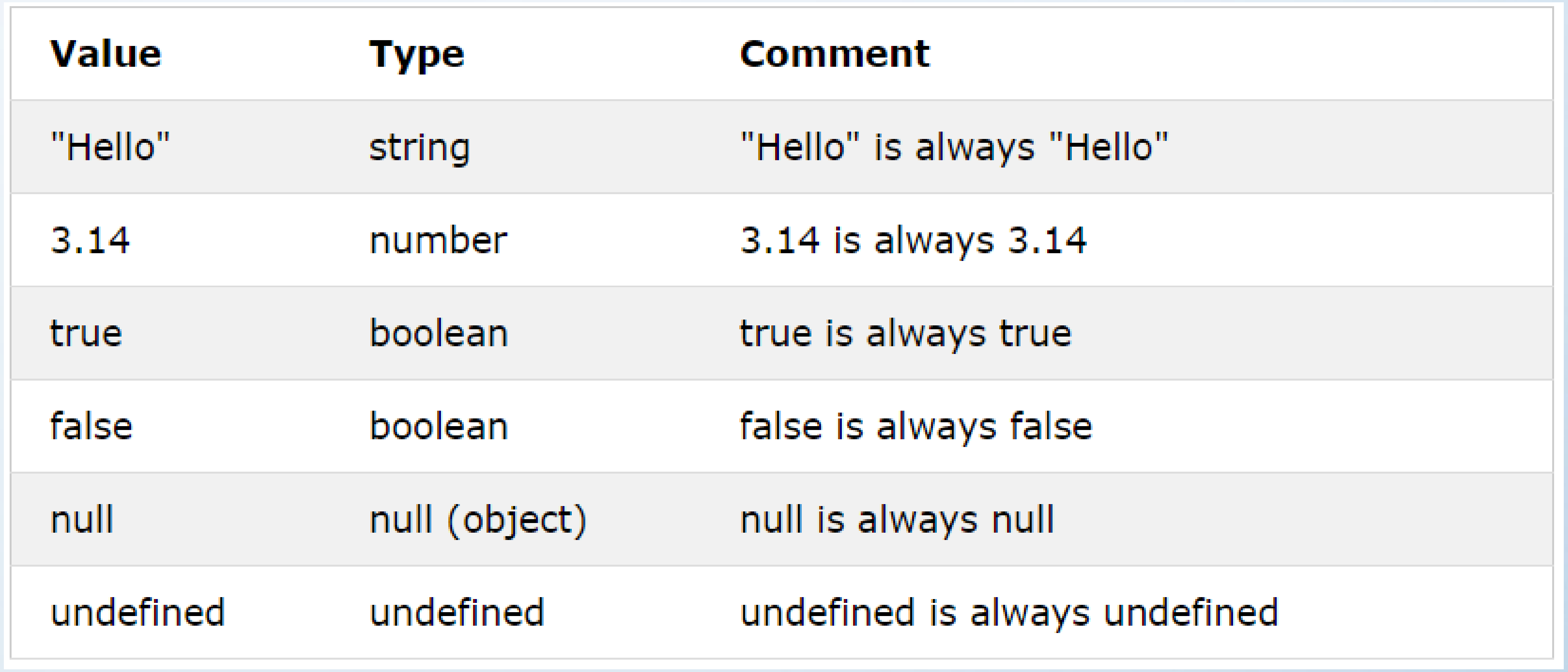
JavaScript Objects

- In JavaScript, objects are king. If you understand objects, you understand JavaScript.
- In JavaScript, almost "everything" is an object
  - Booleans can be objects (if defined with the new keyword)
  - Numbers can be objects (if defined with the new keyword)
  - Strings can be objects (if defined with the new keyword)
  - Dates are always objects
  - Maths are always objects
  - Regular expressions are always objects
  - Arrays are always objects
  - Functions are always objects
  - Objects are always objects

JavaScript Primitives

- A primitive value is a value that has no properties or methods.
- A primitive data type is data that has a primitive value.
- JavaScript defines 5 types of primitive data types:
  - string
  - number
  - boolean
  - null
  - undefined
- Primitive values are immutable (they are hardcoded and therefore cannot be changed).

JavaScript Primitives



Value	Type	Comment
"Hello"	string	"Hello" is always "Hello"
3.14	number	3.14 is always 3.14
true	boolean	true is always true
false	boolean	false is always false
null	null (object)	null is always null
undefined	undefined	undefined is always undefined

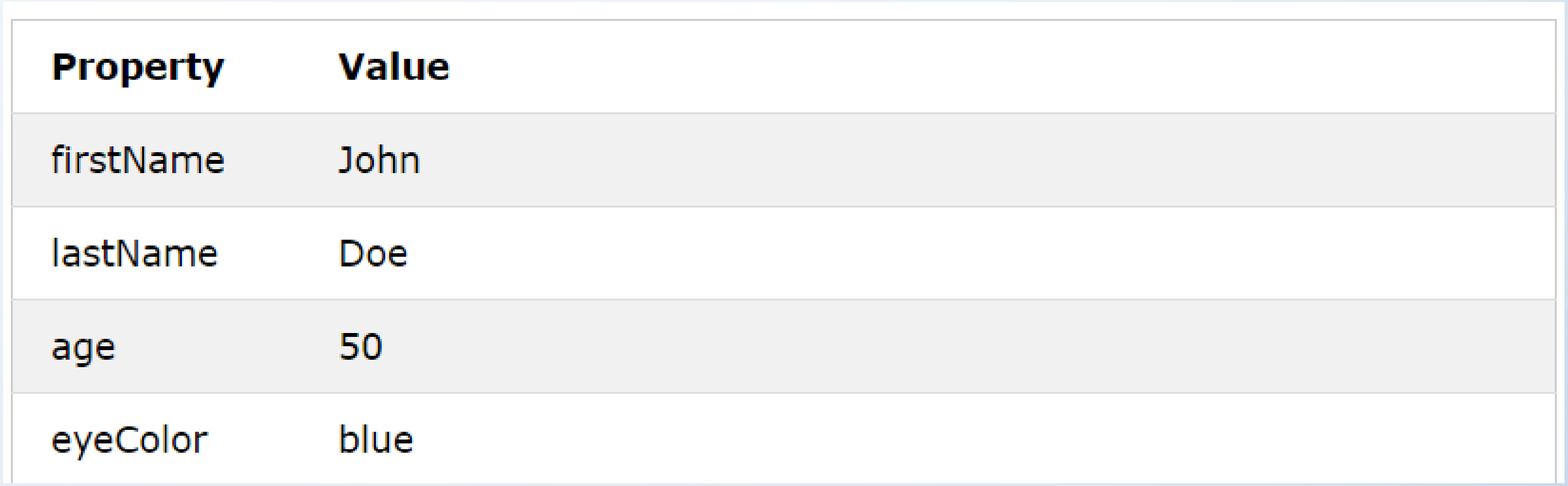
Objects are Variables

- JavaScript variables can contain single values:
- Objects are variables too. But objects can contain many values.
- The values are written as name : value pairs (name and value separated by a colon).

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

Object Properties

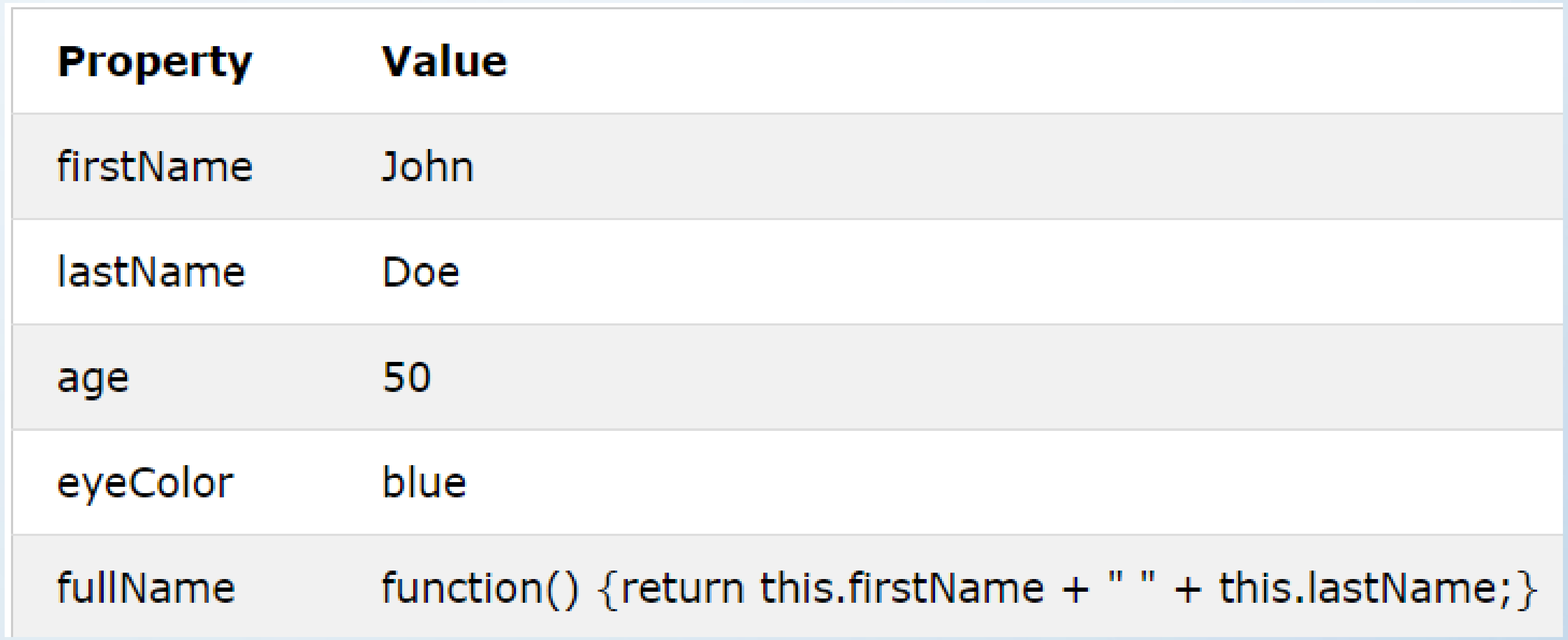
- The named values, in JavaScript objects, are called properties



Property	**Value**
firstName	John
lastName	Doe
age	50
eyeColor	blue

Object Methods

- Methods are actions that can be performed on objects
- Object properties can be both primitive values, other objects, and functions
- An object method is an object property containing a function definition



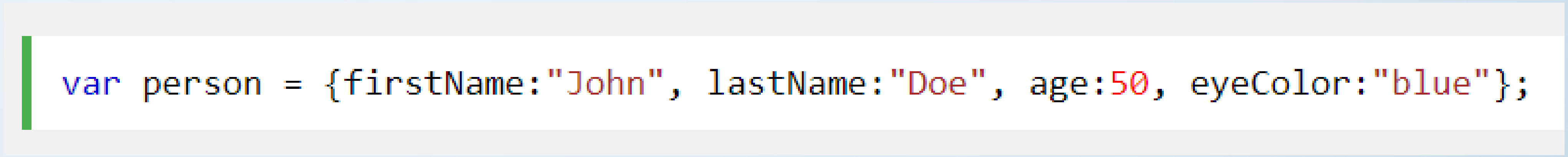
Property	Value
firstName	John
lastName	Doe
age	50
eyeColor	blue
fullName	function() {return this.firstName + " " + this.lastName;}

Creating a JavaScript Object

- With JavaScript, you can define and create your own objects
- There are different ways to create new objects:
  - Define and create a single object, using an object literal.
  - Define and create a single object, with the keyword new.
  - Define an object constructor, and then create objects of the constructed type

Using an Object Literal

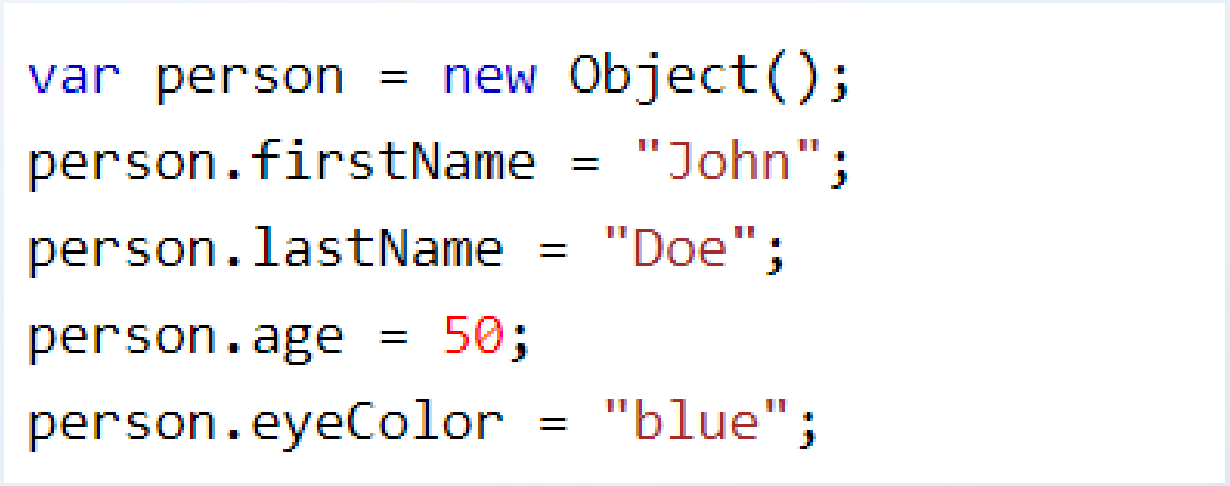
- This is the easiest way to create a JavaScript Object.
- Using an object literal, you both define and create an object in one statement.
- An object literal is a list of name:value pairs (like age:50) inside curly braces {}.
- The following example creates a new JavaScript object with four properties:



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

Using the JavaScript Keyword new

- The following example also creates a new JavaScript object with four properties:



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

JavaScript Objects are Mutable

- Objects are mutable: They are addressed by reference, not by value.
- If person is an object, the following statement will not create a copy of person:

```
var x = person;  // This will not create a copy of person.
```

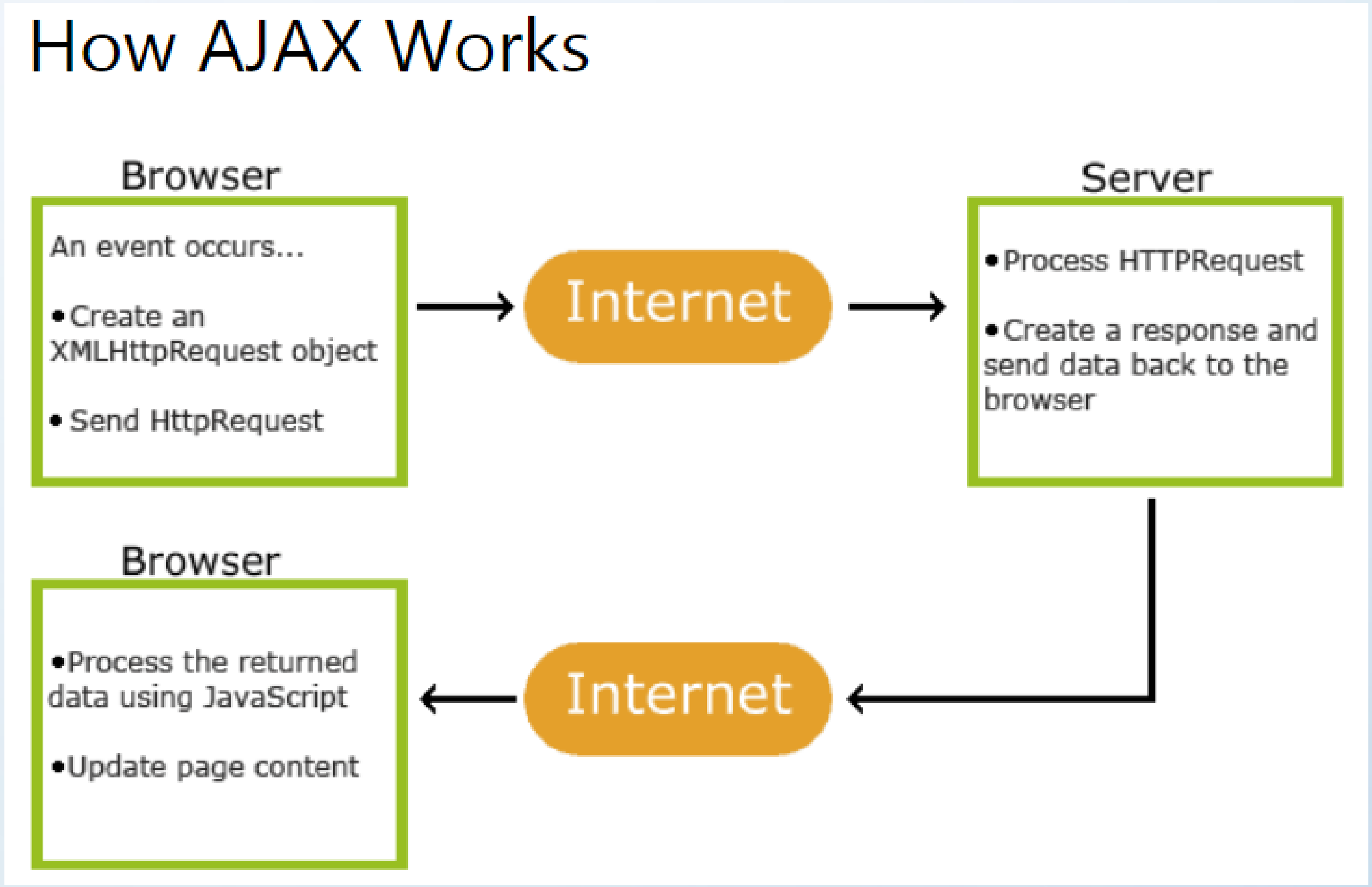
- The object x is not a copy of person. It is person. Both x and person are the same object
- Any changes to x will also change person, because x and person are the same object

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

var x = person;
x.age = 10;           // This will change both x.age and person.age
```

AJAX Introduction

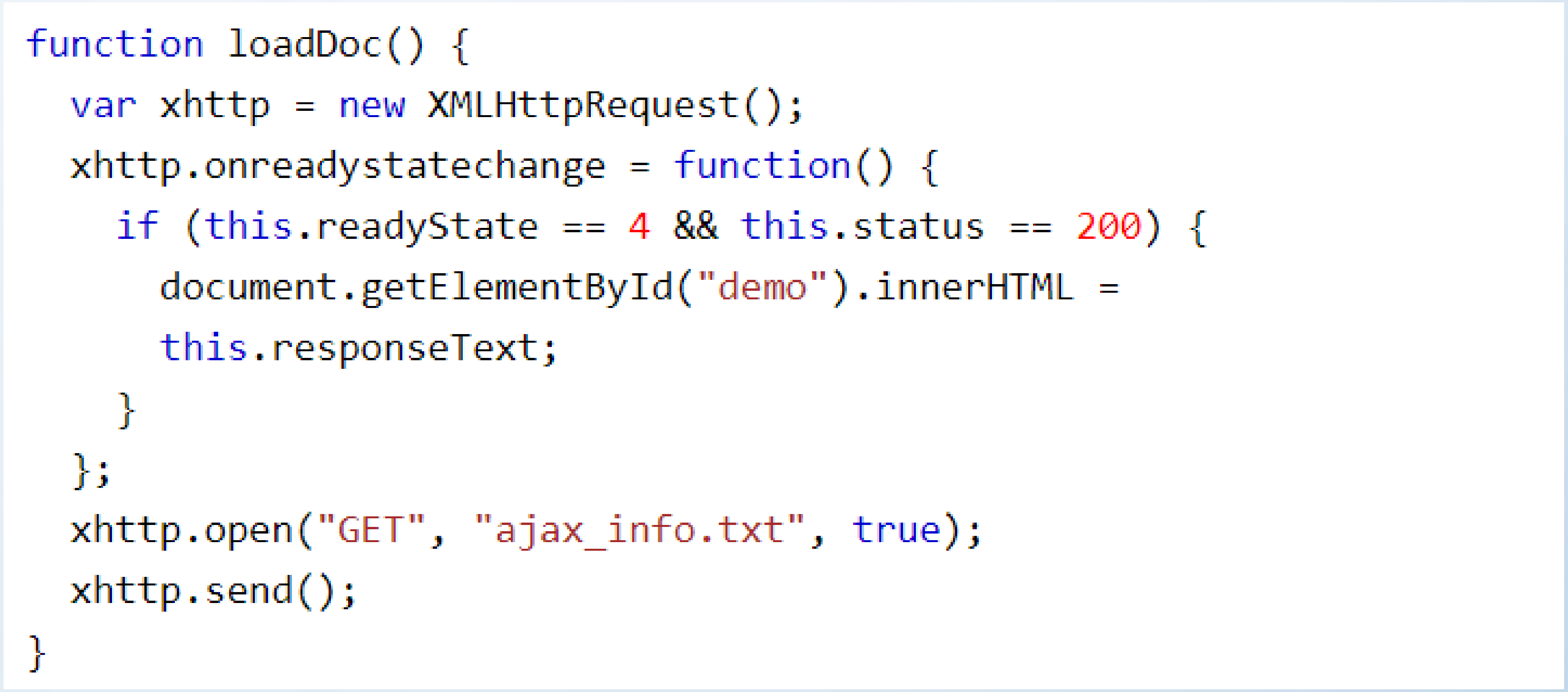
- AJAX = Asynchronous JavaScript And XML
- ajax is not a programming language
- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)
- AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text
- AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes
- This means that it is possible to update parts of a web page, without reloading the whole page

AJAX Introduction


How AJAX Works
Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Browser
• Process the returned data using JavaScript
• Update page content
Internet

AJAX - The XMLHttpRequest Object

- All modern browsers support the XMLHttpRequest object



```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```