Javascript

Prof. Fabio Ciravegna
Dipartimento di Informatica
Università di Torino
fabio.ciravegna@unito.it

UNIVERSITÀ DI TORINO

Outline

- What is JS/how to use it
- variables and scopes
- Datatypes
- Objects
- Arrays
- Booleans
- Comparisons
- Loops
- Functions
- The DOM
  - Document
  - Element
  - Events and Event Listeners
  - Navigation
  - Element creation
- JS Browser Browser Object Model (BOM)
  - window
  - location
  - history
  - timing

Outline 2

- Maps
- try/catch
- Strict mode
- How to debug
- More details on objects
- More details on function
- Classes
- …

JS

- JavaScript is the world's most popular programming language
- JavaScript is the programming language of the Web
- JavaScript is easy to learn
- It has a syntax that is very close to Java's
  - so it should be quite intuitive for you lot
- It has no main and classes are used sometimes
  - but not always as in java
- In the use with the browser
  - javascript is included in the HTML file
  - the functions are invoked directly from within the HTML

https://www.w3schools.com/js/default.asp

Where to?

- The JS code can be included
  - linking a separate JS file
    - <script src="myScript.js"></script>
- Directly in the HTML document using a <script></script> tag in either the `<head>` or the `<body>`
  - do not use it
    - always use a separate file
    - I will however use it in many examples to simplify reading
      - but you must never use it

Separate file

- In file index.js

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

- In the html file index.html

```
<!DOCTYPE html>
<html>
<head>
  <link src="index.js"></script>
</head>
<body>
  <h2>Demo JavaScript in Head</h2>
  <p id="demo">A Paragraph</p>
  <button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

Script in HTML

```
<!DOCTYPE html>
<html>
<head>
  <script>
    function myFunction() {
      document.getElementById("demo").innerHTML = "Paragrap[illegible]ed"
    }
  </script>
</head>
<body>
  <h2>Demo JavaScript in Head</h2>

  <p id="demo">A Paragraph</p>
  <button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

Variables

- JavaScript variables can be declared in 4 ways:
  - Automatically (do not use)
  - Using var (do not use)
  - Using let
  - Using const
- Javascript variables are not required to be declared but remember always to declare them using
  - const x = 3; // unchangeable value
  - let -> block level declaration

```
function myFunction(){
  let x = 5;
  let y = 6;
  let z = x + y;
  …
}
```

- var is no longer to be used - it was used in old browsers: it is rather dangerous because a variable defined with var is visible in every part of the programme, even if defined inside a block
  - never use it.

Let and const

- Variables defined with let and const
  - cannot be redeclared
  - must be declared before use
  - have block scope

JavaScript has 8 Datatypes

- 1. String
- 2. Number
- 3. Bigint
- 4. Boolean
- 5. Undefined
- 6. Null
- 7. Symbol
- 8. Object
  - 1. An object
  - 2. An array
  - 3. A date

JavaScript Types are Dynamic

- the same variable can be used to hold different data types

```
let x;          // Now x is undefined
x = 5;          // Now x is a Number
x = "John";     // Now x is a String
```

- STRINGS CAN BE DEFNED WITH "" OR ''

Objects

- Have properties and methods

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

You can access the objects fields in two ways:

```
    objectName.propertyName
    objectName["propertyName"]
```

Can also have methods

but they are object methods = no classes here

```
const person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

Arrays

```
const cars = ["Saab", "Volvo", "BMW"];

or

const cars = new Array("Saab", "Volvo", "BMW");
Try it Yourself »
```

or

```
const cars = [];
cars[0]= "Saab";
cars[1]= "Volvo";
cars[2]= "BMW";
```

note; arrays do not have a fixed size
- they are extensible

The easiest way to add a new element to an array is using the push() method which adds at the end

Arrays have a length **property**

- that is - while in Java length is a function (array.length()), in javascript it is a property

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let length = fruits.length;
```

Length can be used to add to the end of the array

```
const fruits = ["Banana", "Orange", "Apple"];
fruits[fruits.length] = "Lemon";  // Adds "Lemon" to fruits
```

UNIVERSITÀ DI TORINO

Arrays methods

- Array toString() // prints values as comma separated values, e.g. "banana, fig, orange"
- Array pop() // removes the **last element**
- Array push() // inserts the element **at the end** of the array
- Array shift() //removes the first element and "shifts" all other elements to a lower index
- Array unshift() //adds a new element to an array (at the beginning), and "unshifts" older elements
- Array join() // joins all array elements into a string
  - fruits.join(" * "); —> "banana * fig *"
- Array delete()
- Array concat()
- Array flat()
- Array splice()
- Array slice()
-

UNIVERSITÀ DI TORINO

Array Iterators

- https://www.w3schools.com/js/js_array_iteration.asp

Booleans: Everything Without a "Value" is False

The Boolean value of **0** (zero) is **false**:

```
let x = 0;
Boolean(x);
```

The Boolean value of **-0** (minus zero) is **false**:

```
let x = -0;
Boolean(x);
```

The Boolean value of **""** (empty string) is **false**:

```
let x = "";
Boolean(x);
```

The Boolean value of **undefined** is **false**:

```
let x;
Boolean(x);
```

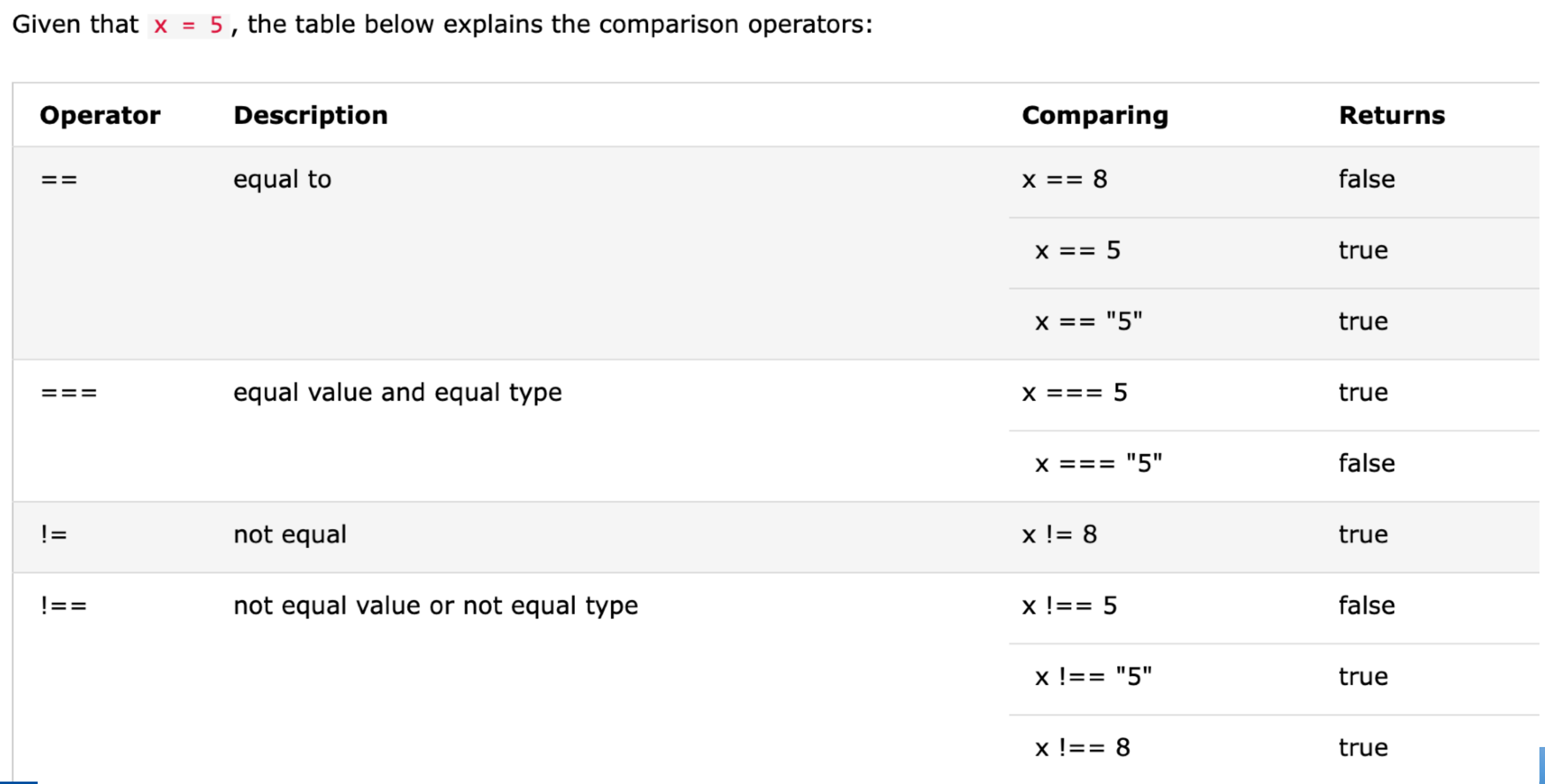
The Boolean value of **null** is **false**:

```
let x = null;
Boolean(x);
```

The Boolean value of **false** is (you guessed it) **false**:

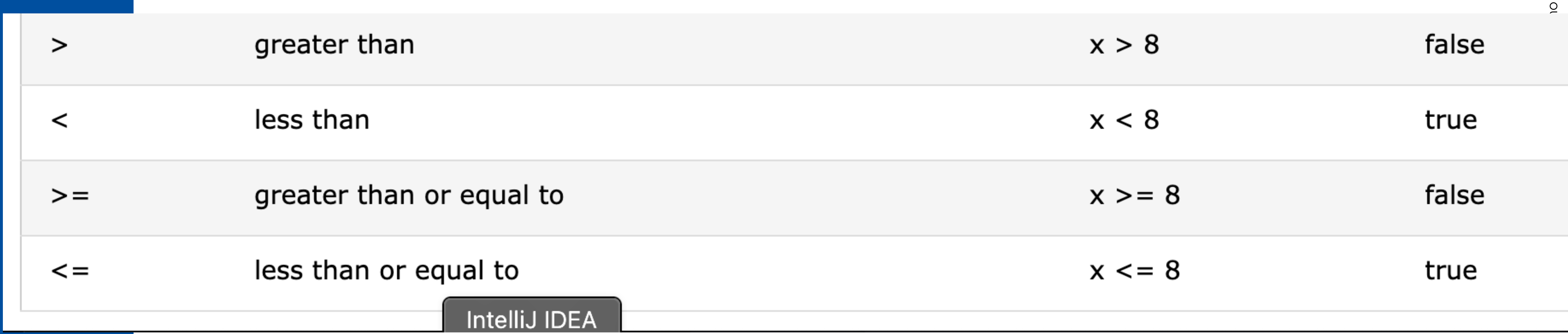
```
let x = false;
Boolean(x);
```

JavaScript Comparison



Given that `x = 5`, the table below explains the comparison operators:

Operator	Description	Comparing	Returns
==	equal to	x == 8	false
		x == 5	true
		x == "5"	true
===	equal value and equal type	x === 5	true
		x === "5"	false
!=	not equal	x != 8	true
!==	not equal value or not equal type	x !== 5	false
		x !== "5"	true
		x !== 8	true



>	greater than	x > 8	false
<	less than	x < 8	true
>=	greater than or equal to	x >= 8	false
<=	less than or equal to	x <= 8	true

UNIVERSITÀ DI TORINO

The Nullish Coalescing Operator (??)

The `??` operator returns the first argument if it is not **nullish** (`null` or `undefined`).

Otherwise it returns the second argument.

```
let name = null;
let text = "missing";
let result = name ?? text;
```

UNIVERSITÀ
DI TORINO

The Optional Chaining Operator (?.)

- The ?. operator returns undefined if an object is undefined or null (instead of throwing an error).

```
// Create an object:
const car = {type:"Fiat", model:"500", color:"white"};
// Ask for car name:
document.getElementById("demo").innerHTML = car?.name;
```

Loops

https://www.w3schools.com/js/js_loop_for.asp

UNIVERSITÀ DI TORINO

Looping Array Elements

- As in Java

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let fLen = fruits.length;

let text = "<ul>";
for (let i = 0; i < fLen; i++) {
  text += "<li>" + fruits[i] + "</li>";
}
text += "</ul>";
```

- But also differently as we will see