Introduction to Web Programming

Chap. 4 / Javascript, part I

Anne Jeannin-Girardon, PhD | anne.jeannin@unistra.fr
Associate Professor, University of Strasbourg

Javascript, why ?

- Access the page content
- Modify the page content
- Program instructions followed by the browser
- React to user-triggered events

Examples : slideshow, form validation, filter information on a page for the user, …

What this chapter is not

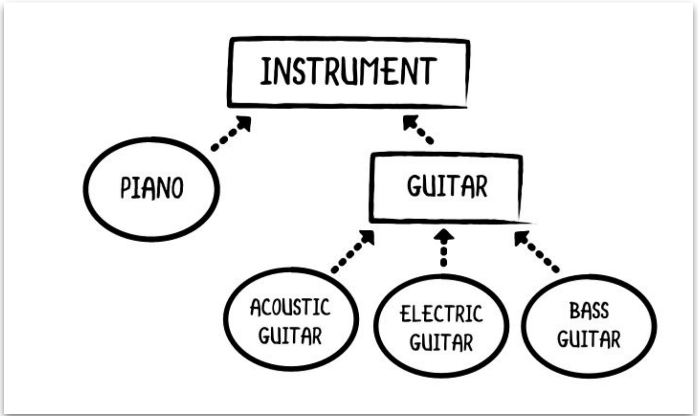
- It is NOT a course about algorithmic (I consider you all know about algorithmics basics, having followed the course Algorithms & Programming 1)
- So if you think you'll learn how to program, this is not the right place
- WE are going to learn about the JS language syntax and how we can use this language to manipulate web pages

Where will we write JS code ?

- In the HTML documents, using the tag `<script></script>` (anywhere in the document)
- In a javascript file, loaded using the tag `<script src=“sourceJS.js”></script>`
- We’ll see that loading JS scripts at the end of the HTML document is a good practice

JS is object oriented

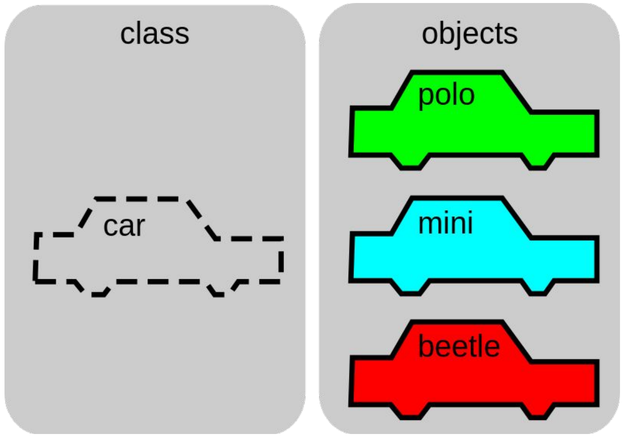
- Program = code + data
- Programs are either built around
  - The code (i.e. what is happening ?) => function-oriented model
  - The data (i.e. what is being affected ?) => object-oriented model
- OOP involves
  - Defining classes (blueprints for objects)
  - Creating objects (specific instances of the classes)
  - Writing applications manipulating those objects

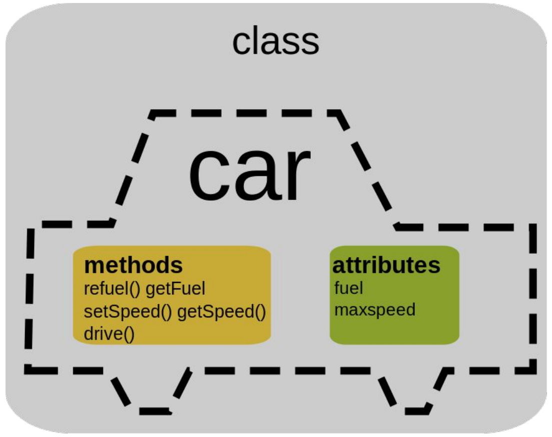


Source: https://www.raywenderlich.com

Some object concepts

- An object is a way to represent things (a car, a person, a building)
- Instances are built with the blueprint
- Each instance of an object has its own properties (name, eye color), its own methods (talk, walk)





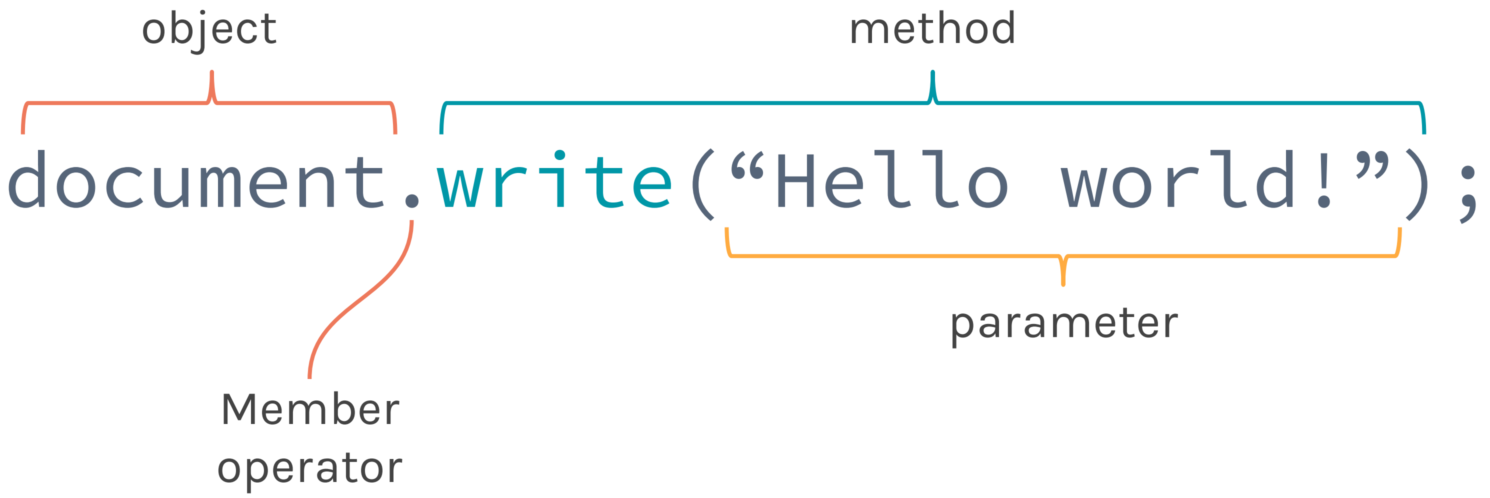
Source: https://en.wikibooks.org

JS objects: window and document

- JS window = window opened in a browser (instance)
- It has some properties (e.g. `document`, `console`, …)
- And some methods (`alert()`, `prompt()`, …)

- JS document = HTML document loaded in a browser
- Root node of the HTML document
- Has some properties (e.g. `doctype`) and methods (e.g. `getElementByID()`)

Calling a method of an object

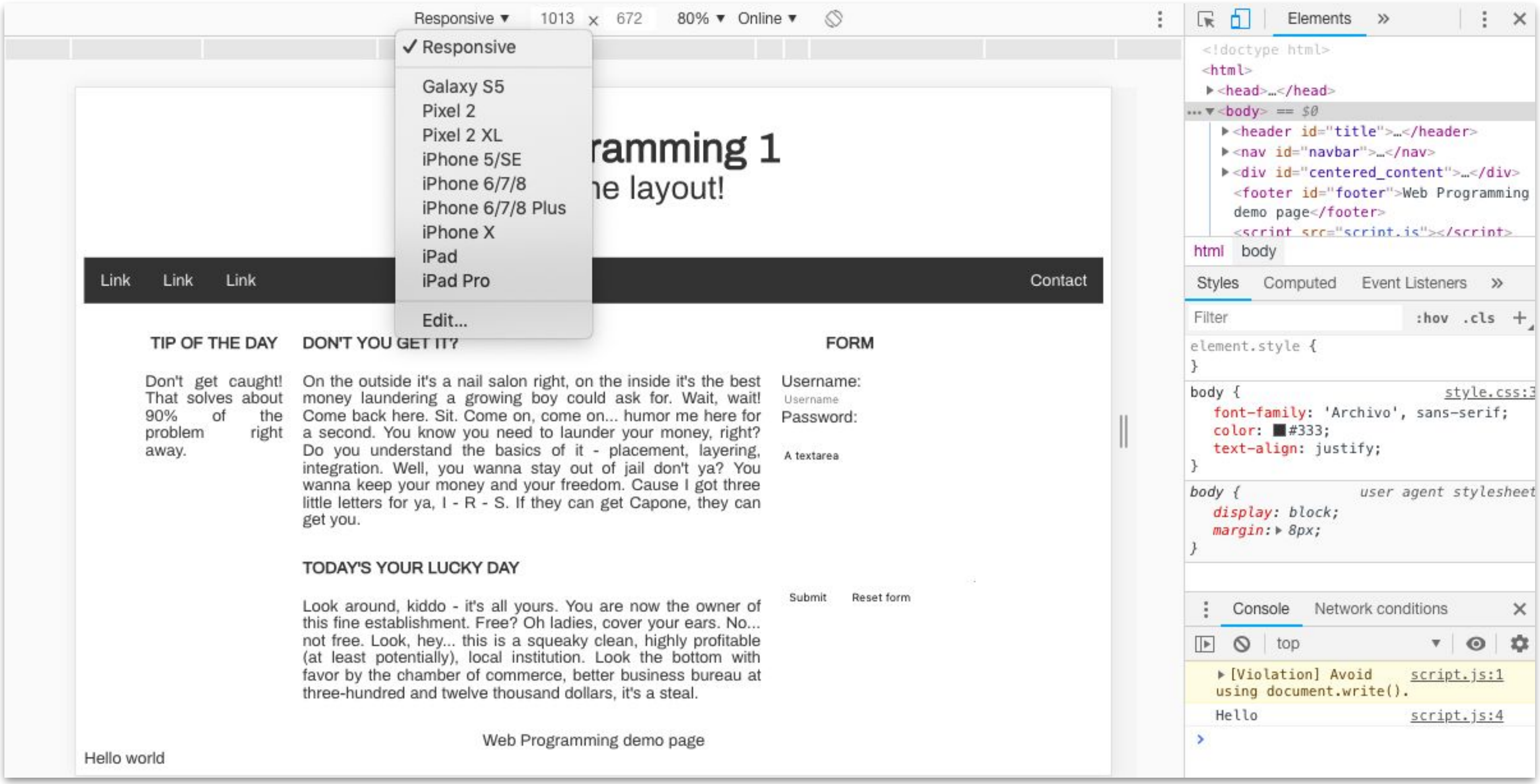


```
document.write(“Hello world!”);
```

First things first : outputting stuff

- No equivalent of printf (C) or print (Python) etc.
- We can use alert windows (meh…)
  - `alert(“Hello world!”);`
- We can write in the (resulting) document
  - `document.write(“Hello world!”);`
- We can use the browser console
  - `console.log(“Hello world!”);`

Browser developer tools



ramming 1
he layout!

Link Link Link Contact

TIP OF THE DAY

Don't get caught! That solves about 90% of the problem right away.

DON'T YOU GET IT?

On the outside it's a nail salon right, on the inside it's the best money laundering a growing boy could ask for. Wait, wait! Come back here. Sit. Come on, come on... humor me here for a second. You know you need to launder your money, right? Do you understand the basics of it - placement, layering, integration. Well, you wanna stay out of jail don't ya? You wanna keep your money and your freedom. Cause I got three little letters for ya, I - R - S. If they can get Capone, they can get you.

TODAY'S YOUR LUCKY DAY

Look around, kiddo - it's all yours. You are now the owner of this fine establishment. Free? Oh ladies, cover your ears. No... not free. Look, hey... this is a squeaky clean, highly profitable (at least potentially), local institution. Look the bottom with favor by the chamber of commerce, better business bureau at three-hundred and twelve thousand dollars, it's a steal.

FORM

Username:
Username
Password:

A textarea

Submit Reset form

Web Programming demo page

Hello world

```
<!doctype html>
<html>
 ▸<head>…</head>
…▾<body> == $0
   ▸<header id="title">…</header>
   ▸<nav id="navbar">…</nav>
   ▸<div id="centered_content">…</div>
    <footer id="footer">Web Programming
    demo page</footer>
    <script src="script.js"></script>
```

html body

```
element.style {
}
body {                                   style.css:3
    font-family: 'Archivo', sans-serif;
    color: ■#333;
    text-align: justify;
}
body {                          user agent stylesheet
    display: block;
    margin:▸ 8px;
}
```

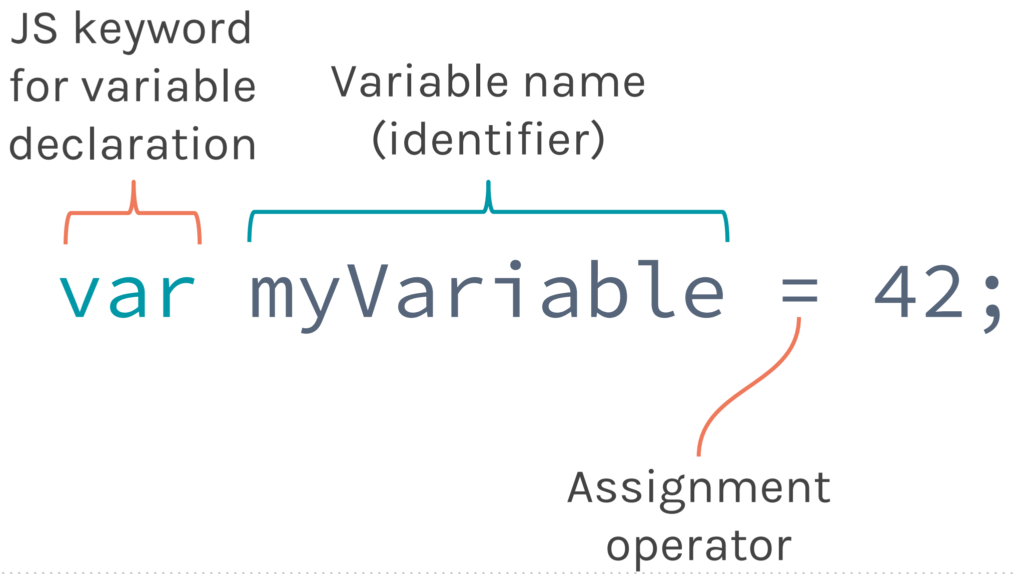
```
▸[Violation] Avoid              script.js:1
 using document.write().
 Hello                          script.js:4
>
```

Javascript statements

- As in any programming language, JS programs are made of statements
- Statements end with a semicolon `;`
- Statements can be grouped within blocks delimited by curly braces `{}`
- JS is case sensitive
- Comments are just like C comments :
  - `// single line comment`
  - `/* multiple lines comment */`

Javascript variables

- Used to store data ; stored values can change over time
- Variable must be declared before we can use them


JS keyword for variable declaration
Variable name (identifier)
Assignment operator

```
var myVariable = 42;
```

Declaring variables : var or let ?

- To declare a variable, one can use either `var` or `let`
- Outside functions, variables declared with either keyword are global
- In functions and block, the scope might be different…
- `var` allows variable redeclaration

Variable scope

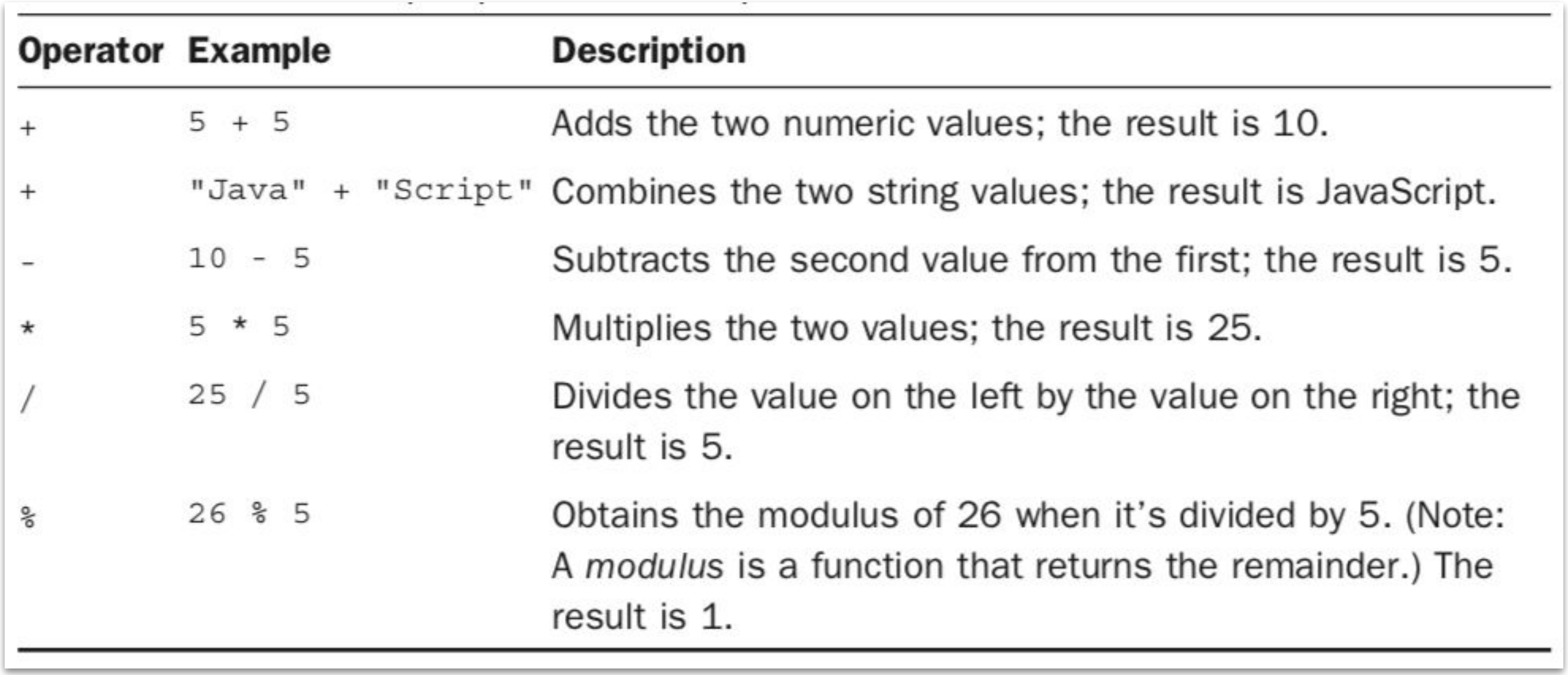
```
// Global scope
var v1;
let v2;

function doStuff() {
  var mint = 13; // function scope
  for(let i = 0; i < 10; i++){
    // mint is visible here
    // i is visible here
  }
  // mint is still visible here
  // i is no longer visible
}
```

Data types

- Main types are numbers, strings and booleans (true / false)
- Number examples : `42  13.37`
- String examples : `“Hello world”   ‘Hello world’`
- There are also objects, arrays, undefined and null
- No need to specify the type of data a variable will hold when you declare them !

```
var v; // undefined
v = 42;
v = “Hello”;
var foo = ‘I\’m ok’; // escape character
var bar = “Hello\nworld”; // newline
```

Operators and expressions



Operator	Example	Description
+	`5 + 5`	Adds the two numeric values; the result is 10.
+	`"Java" + "Script"`	Combines the two string values; the result is JavaScript.
-	`10 - 5`	Subtracts the second value from the first; the result is 5.
*	`5 * 5`	Multiplies the two values; the result is 25.
/	`25 / 5`	Divides the value on the left by the value on the right; the result is 5.
%	`26 % 5`	Obtains the modulus of 26 when it's divided by 5. (Note: A *modulus* is a function that returns the remainder.) The result is 1.

Source: Lemay et al., HTML, CSS & JavaScript web publishing in one hour a day

A little fun with variables

What will be the data type and the value holded
in the following variable :

```
var foo = 40+2+"Hello"+1+3+3+7;
```

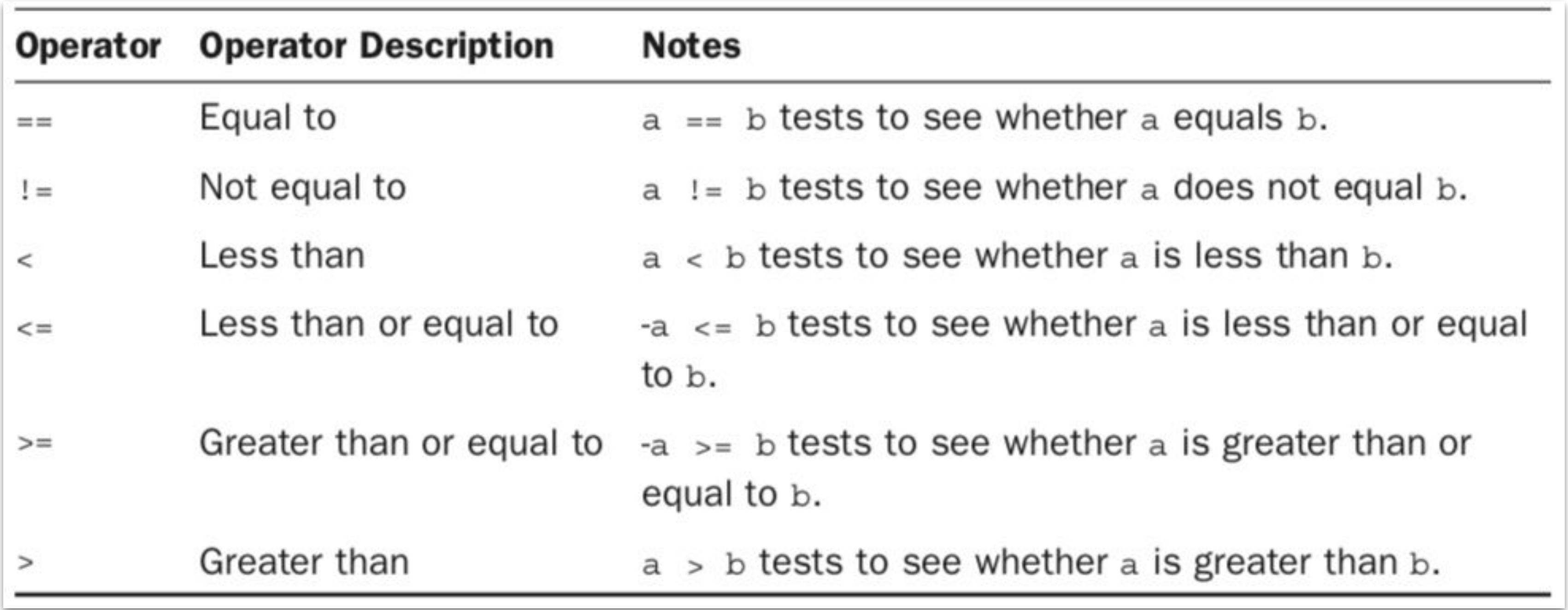
Arrays

- Arrays store sets of values
- Items in an array have an index
- The first index is `0` !
- Arrays have methods (Array is a JS class)

Arrays

```
// Using a constructor:
var a1 = new Array(10); // 10 slots
a1[0] = 42;

var a2 = ['blue', 'orange', 'gray'];
var a3 = [42, a2, true, "hello"];

var l = a3.length;
a3.pop(); // remove last element
a3.push("world"); // append element
a3.shift(); // remove first element
and shift the rest to the left
a3.unshift(10); // insert in the
beginning of the array
/* Final array:
10 a2 true "world" */
```

Comparison operators



Operator	Operator Description	Notes
`==`	Equal to	`a == b` tests to see whether `a` equals `b`.
`!=`	Not equal to	`a != b` tests to see whether `a` does not equal `b`.
`<`	Less than	`a < b` tests to see whether `a` is less than `b`.
`<=`	Less than or equal to	-`a <= b` tests to see whether `a` is less than or equal to `b`.
`>=`	Greater than or equal to	-`a >= b` tests to see whether `a` is greater than or equal to `b`.
`>`	Greater than	`a > b` tests to see whether `a` is greater than `b`.

=== equal to tests both values & type

Source: Lemay et al., HTML, CSS & JavaScript web publishing in one hour a day

Control structure: conditional

```
if (condition) {
     instructions
} else {
     instructions
}
```

Conditional

```
var color = "blue";

if (color == "red"){
  console.log("Red color");
} else if (color == "green"){
  console.log("Green color");
} else {
  console.log("Another color");
}
```

Control structure: loops

Initialisation — Loop termination — Next iteration

```
for (exp1; exp2; exp3) {
    instructions
}

while (exp){
    instructions;
}
```

Loops

```
for (let i = 0; i < 10; i++){
  console.log(i);
}

var i = 0;
while (i < 10) {
  console.log(i);
  i = i + 1;
}
```

Control in loops

- `break` : interrupt the loop
- `continue` : skip current loop iteration

Loops

```
for (let i = 0; i < 10; i++){
  if (i == 4) break;
  console.log(i);
}
// 0 1 2 3

for (let i = 0; i < 10; i++){
  if (i == 4) continue;
  console.log(i);
}
// 0 1 2 3 5 6 7 8 9
```

Functions

$\overbrace{\texttt{function}}^{\text{JS keyword}}\ \overbrace{\texttt{sayHi()}}^{\text{Function name}}$

```
function sayHi() {
  console.log("Hi !");
}
```

Calling a function

- Functions are useful for "storing" instructions for a specific task
- Once defined, it can be called as many times as needed
- Functions can have parameters
- Functions can return a value

Functions

```
function sayHi(){
  console.log("Hi!");
}

function area(width, height){
  return width*height;
}

sayHi();
var w = 13.37;
var h = 2;
var s = area(w,h);
```

Prelude to HTML/JS interactions

HTML

```
<form>
  <input type="text"   id="fieldContent"/>
  <input type="button" id="ajoutItem" value="Ajouter"/>
</form>
<span class="output"/><span/>
```

Javascript

```
var bouton = document.getElementById("ajoutItem");
bouton.addEventListener("click", ecrireItem);

function ecrireItem(){
  var outputElt   = document.getElementById("output");
  var content     = document.getElementById("fieldContent").value;
  outputElt.innerHTML += content + "<br>";
}
```

Chapter recap

- Notion of object-oriented programs
- Variables, data types
- Arrays
- Control structure
- Functions