Buzz words:

|  |  |
| --- | --- |
| Isomorphic code | Works everywhere |
| dynamic vs static language | Variables can change types vs must remain in the same type as it was declared at first. |
| type safety | Helps you not changing types of a variables after it was made. |
| Compiler vs. transpiler | Compiling is taking high level to low level. Transpiler is simply making one high level into another high level. |
| JIT language | Just In Time. |

23/04/18 – 9:15

JS is a JIT language – Just In Time.

23/04/18 14:30

Hoisting : in a **scope**, any variable that would be declared inside the scope will actually, behind the scenes, be declared at the beginning of the scope. If the variables gets a value while being declared, the **declaration** will be performed first, and the **hasama** is made at the respected written line.

When doing a logical test of && between 2 "true" things (strings, numbers, so on), it will return the last thing evaluated.

1 && "bye" && "hello" && 0 && "ok"

Will return ok.

1 && "bye" && "hello" && **null** & 0 && "ok"

Will return null, because it's like "false", and thus the evaluation stops there.

24/04/2018 9:00

Because there are several scenarios, like null, 'undefined' and such, the best way to check for something if it's true or false is with !!var.

Bitwise (p. 56 in slide 1):

A binary action between 2 variables that results in true or false.

Binary numbers example / trick to calculate:

|  |  |
| --- | --- |
| 0110 |  |
| 2^3 2^2 2^1 2^0 |  |
| 8421 | 8\*0 + 4\* 1 + 2\* 1 + 1\* 0 = 4 + 2 = 6 |

|  |  |
| --- | --- |
| 1111 |  |
| 2^3 2^2 2^1 2^0 |  |
| 8421 | 8\*1 + 4\* 1 + 2\* 1 + 1\* 1 = 8 + 4 + 2 + 1 = 15 |

|  |  |
| --- | --- |
| 1 | 1 = 2^0 = 1 |
| 2 | 10 = 2^1 = 2 |
| 3 | 11= 2^1 + 2^0 = 3 |
| 4 | 100 = 2^2 = 4 |
| 5 | 101 = 2^2 + 2^0 = 5 |
| 6 | 110 = 2^2 + 2^1 = 4 + 2 = 6 |
| 7 | 111 = 2^2 + 2^1 +2^0 = 4 + 2 +1 = 7 |
| 8 | 1000 = 2^3 = 8 |
| 9 | 1001 = 2^3 + 2^0 = 9 |
| 10 | 1010 = 2^3 + 2^1 = 8 + 2 = 10 |