Interviewee 3: [redacted]
Undergraduate student [redacted]
Works as Full stack Developer and Infrastructure
JS experience: 3 years. Elementary JS knowledge
Other languages: Python, C, C++, Java, Go
Maintains already existing JS modules/frameworkds
JS Pros: Asynchronous REST calls in the browser. Allows developers to push work into the client and reduce server load
JS Cons: Floating point in JS is 64-bits - GREAT remark by rafael (Adriano's reference: (2) Bartek Szopka: Everything you never wanted to know about JavaScript numbers -- JSConf EU 2013 - YouTube); 'this' keyword (Indeed it is **very** problematic in JS) is very counter-intuitive. It tries to simplify developer's life too much (type coercion).
Does JS lead to hard-to-understand code? Yes, it is a bit too complex to read.
Any JS construct that is particularly difficult to understand? Type coercion.

Atom	Preferred version
Arithmetic as Logic	Without atom
Assignment as Value*	Without atom
Automatic Semicolon Insertion	Without atom
Comma operator	Without atom
Ternary operator**	With atom
Implicit predicate***	Without atom
Logic as Control Flow	Without atom
Omitted Curly Braces	Without atom
Post Increment	Without atom
Pre Increment	Without atom

*Easier to understand, but 'dirty writing' in the sense that there are too many lines

** prefers to write side A, "mas o lado B é mais fácil de ler para aquele que não tem um histórico de programação ne" - This is just the third one, but there seems to be a trend that developers prefer to just write it in the way that's easier for them

***Note on implicit predicate for future interviews: Both expressions (Side A and Side B) inside the if statement evaluate to false, which is why the pieces of code are equivalent. The actual values for V1 and V2 do not really matter. What matters is how difficult it is to evaluate the condition inside the if statement.

This interviewee strongly stresses the fact that type coercion can be very problematic in JS. And in fact it may remain unclear even to experienced developers. This is why JS has two ‘equality’ operators, == and ===.

Complicated JS constructs: ‘this’. **Adriano`s opinion**: He seems to be referring to three very peculiar JS functions: bind(), apply(), and call(), which take this as a parameter and ‘reassigns’ it. It was very difficult for me to understand this at first, and it cost me my first job interview in JS.
Another big problem: == vs ===. Same problem as above (type coercion). This interviewee seems to suffer hard with type coercion, and it can in fact be messy.