Mind your languages

Pierre Chifflier

5 juin 2014

Langages ?

- Question reçue :
- “Si je fais mon projet avec le langage $LANG, est-ce que c’est bon pour vous ?”
- Du coup, on préfère quel langage ?

L'addition a ses mystères en JavaScript. Enfin, le symbole +

Source (snippets/js/weirdeval.js)

```
[]    <- array                    {}     <- Object

[] + []    = 0
[] + {}    = "[object Object]"
{} + []    = 0
{} + {}    = NaN
({} + {}) = "[object Object][object Object]"
```

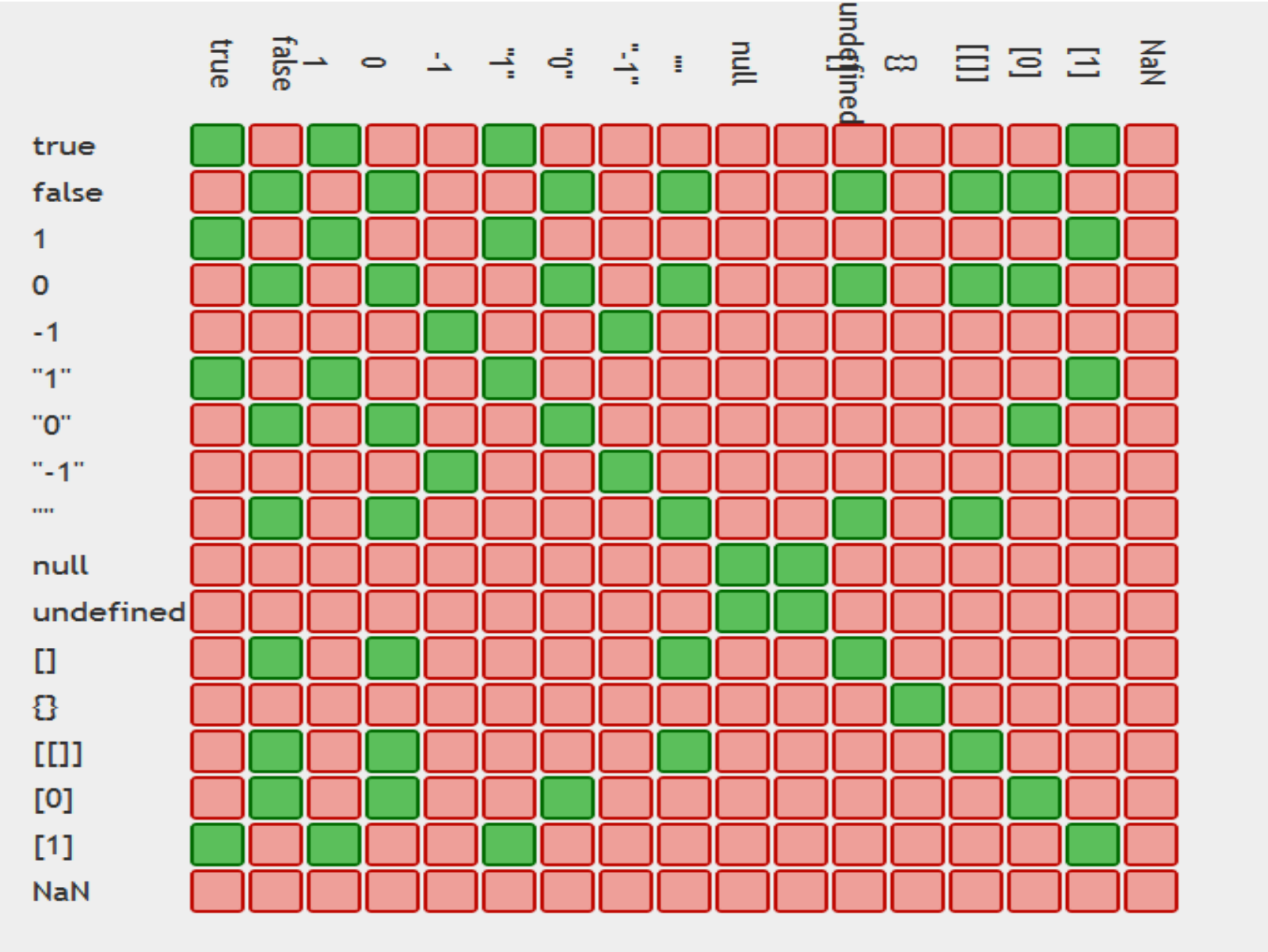
Heureusement, l'égalité aussi...

Source (snippets/js/unification2.js)

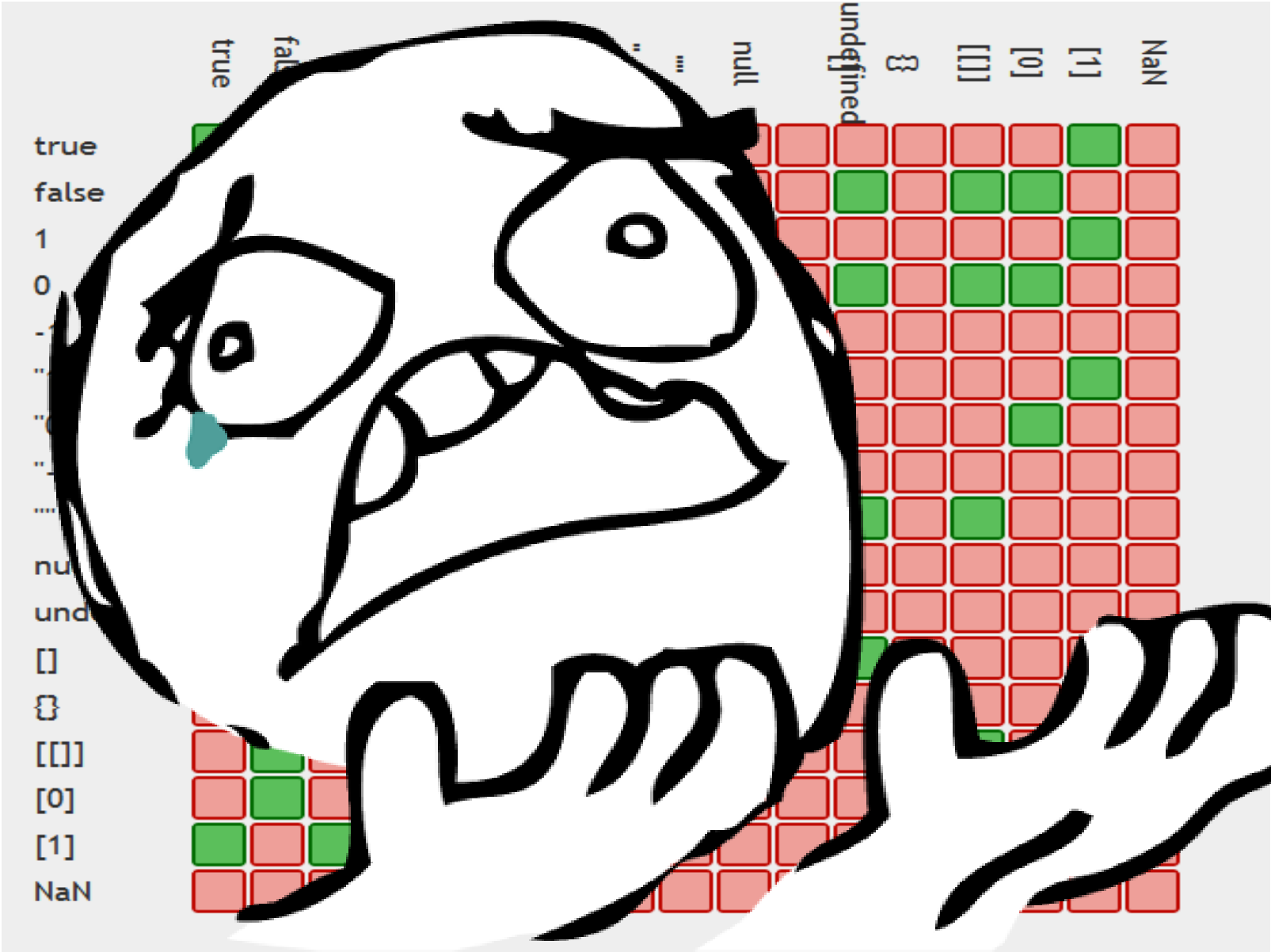
```
[1] == [1] ? false ! (mais [1]==true)

0 == '0' ?   true
0 == "0.0" ? true
'0' == "0.0" ? false !
```

[JavaScript] Certains sont plus égaux que d'autres



[JavaScript] Certains sont plus égaux que d'autres



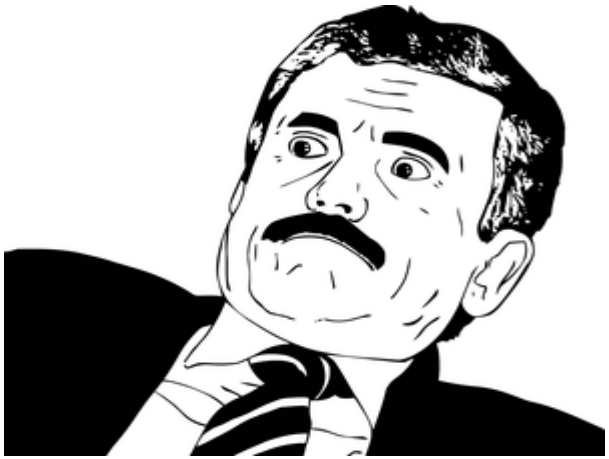
La crypto en JS (dans le navigateur) semble certainement une bonne idée :

- OpenPGP.js
- keybase.io
- Heartbleed and javascript crypto

Source (snippets/php/castincr.php)

```
$x="2d8"; print(++$x."\n");   2d9 (chaine)
          print(++$x."\n");   2e0 (chaine)
          print(++$x."\n");   3 (flottant)

$x="2d8"; print($x+1);        3 (entier)
```



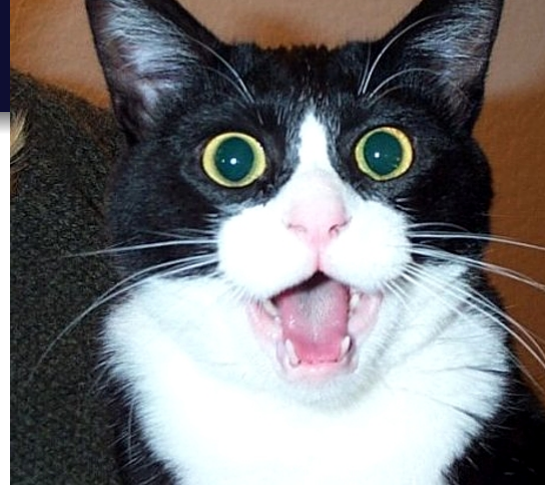
Quel rapport avec la sécurité ?

Source (snippets/php/hash.php)

```
$h1= md5 ('QNKCDZO');
$h2= md5 ('240610708');
$h3= md5 ('A169818202');
$h4= md5 ('aaaaaaaaaaaaumdozb');
$h5= sha1('badthingsrealmlavznik');
```

Lesquels sont égaux ?

A. Aucun, bien sûr	**C.** h1, h3 et h4
B. h3 et h5	**D.** La réponse D



La réponse D

Tous !

En Php :

```
'0e830400451993494058024219903391' ==
'0e462097431906509019562988736854' ==
'0e590126417109547563244339779435' ==
'000e994639666666707280479226342 4' ==
'00e6350478108627283429100248932178194894'
```

[Java] L'égalité a ses limites

Source (snippets/java/IntegerBoxing.java)

```
Integer a1 = 42; Integer a2 = 42;
if (a1 == a2)
    System.out.println("a1 == a2");

Integer b1 = 1000; Integer b2 = 1000;
if (b1 == b2)
    System.out.println("b1 == b2");
```

[Java] L'égalité a ses limites

Source (snippets/java/IntegerBoxing.java)

```
Integer a1 = 42; Integer a2 = 42
if (a1 == a2)
    System.out.println("a1 == a2");

Integer b1 = 1000; Integer b2 =
if (b1 == b2)
    System.out.println("b1 == b2");
```

a1==a2 mais $b1 != b2$

(à cause de Integer boxing + cache + .equals)

Source (snippets/ocaml/mutablebool.ml)

```
let t=string_of_bool true in
  t.[0]<-'f';
  t.[1]<-'a';
  t.[3]<-'x';;

Printf.printf "1=1 est %b\n" (1=1);;
```

Source (snippets/ocaml/mutablebool.ml)

```
let t=string_of_bool true in
  t.[0]<-'f';
  t.[1]<-'a';
  t.[3]<-'x';;

Printf.printf "1=1 est %b\n" (1
```

1=1 est **faux**

Ce serait moins drôle si `Char.escaped` n'était pas aussi concerné

Conclusion (presque) sérieuse

- Coder sécurisé, c'est comme jongler avec des tronçonneuses
- **Le langage n'est pas qu'un outil**
- Les bugs/features **ont** un impact sur les fonctions de sécurité
- Devs de langages : STOP avec le ==, ===, ...
- Wanted : plus d'exemples !
- Présentation complète : Mind your languages (95+ slides ...) sur ssi.gouv.fr (publications)