IFT1015 Programmation 1

Type et expressions booléennes

Marc Feeley
(avec ajouts de Aaron Courville et Pascal Vincent)

Booléen

- Un **booléen** (**boolean**) c'est une donnée qui ne peut prendre que 2 valeurs : **vrai** et **faux**
- C'est utile pour représenter la **valeur de vérité** d'une proposition ou **condition**, et pour faire une **exécution conditionnelle**
- P.ex. : si «la variable **`solde`** est négative» alors afficher un message
- Il faut **un seul bit** pour encoder un booléen :
  - Par convention : vrai : 1 et faux : 0

Booléen

- Un programme peut contenir des **constantes littérales booléennes**
- Syntaxe : `true` et `false`

```
> true
  true
> false
  false
```

Opérateurs de comparaison

- Plusieurs opérateurs sont disponibles pour comparer deux valeurs
- La valeur d'une expression de comparaison c'est toujours un booléen (soit **`true`** ou **`false`**)
  - x **==** y **`true`** si et seulement si $x = y$
  - x **!=** y **`true`** si et seulement si $x \neq y$
  - x **<** y **`true`** si et seulement si $x < y$
  - x **>** y **`true`** si et seulement si $x > y$
  - x **<=** y **`true`** si et seulement si $x \leq y$
  - x **>=** y **`true`** si et seulement si $x \geq y$
  - x **===** y **`true`** $\Leftrightarrow$ $x = y$ et x, y ont le même type

Opérateurs de comparaison

- Si x et y sont des **nombres**, alors la comparaison se fait en fonction de l'**ordre numérique**

```
> -6 < 2
  true
> 6 < 2
  false
> 2 >= 2
  true
> 2 != 2
  false
> 2 == 2
  true
```

Opérateurs de comparaison

- Cas spécial : la valeur numérique NaN est différente de toute valeur (incluant elle même!)

```
> var x = 1 / 0
> x
  Infinity
> x == x
  true
> var y = 0 / 0
> y
  NaN
> y == y
  false
```

Opérateurs de comparaison

- Si x et y sont des **textes**, alors la comparaison se fait en fonction de l'**ordre alphabétique**

```
> "aubergine" < "poire"
  true
> "juliette" < "david"
  false
> "david" >= "david"
  true
> "david" != "david"
  false
> "david" == "david"
  true
```

Opérateurs de comparaison

- Deux textes x et y sont égaux seulement si ces textes contiennent exactement les même caractères et dans la même séquence

```
> "david" == "david"
  true
> "david" == "David"
  false
> "david" == "  david  "
  false
```

Opérateurs de comparaison

- Si x et y sont des **booléens**, alors la comparaison se fait en fonction de leur **encodage numérique**

```
> false < true
  true
> true < false
  false
> false >= false
  true
> false != false
  false
> false == false
  true
```

Opérateurs de comparaison

- Les opérateurs de comparaison font au besoin la conversion de textes numériques et booléens en nombres mais pas l'opérateur d'**égalité stricte** $x === y$

```
> "2" == 2
  true
> "2" === 2
  false
> true == 1
  true
> true === 1
  false
```

Opérateurs booléens

Opérateurs booléens

- JS a un ensemble d'**opérateurs prédéfinis sur les booléens** qui sont utiles pour exprimer des conditions composées
- **Négation** (NON): **!** *‹expression›*
  - Vraie ssi *‹expression›* est fausse
- **Conjonction** (ET): *‹expression›* **&&** *‹expression›*
  - Vraie ssi les deux *‹expression›* sont vraies
- **Disjonction** (OU): *‹expression›* **||** *‹expression›*
  - Fausse ssi les deux *‹expression›* sont fausses

Opérateurs logiques

booléens v.s. bit-à-bit

Opérateurs booléens:

Effectués sur des booléens

Opérateur ! «NON»

Expression	Valeur
!false	true
!true	false

Opérateur || «OU»

Expression	Valeur
false || false	**false**
false || true	true
true || false	true
true || true	true

Il suffit qu'un des deux (ou les deux) soit vrai pour que le résultat soit vrai.

Opérateur && «ET»

Expression	Valeur
false && false	false
false && true	false
true && false	false
true && true	**true**

Il faut que les deux soient vrai pour que le résultat soit vrai.

Opérateurs bit-à-bit:

Effectué sur **chaque bit** de la représentation binaire d'un nombre.

Opérateur ~ «NON»
Inverse *tous* les bits .

Opérateur | «OU»

Expression	Valeur
0 | 0	**0**
0 | 1	1
1 | 0	1
1 | 1	1

Opérateur & «ET»

Expression	Valeur
0 & 0	0
0 & 1	0
1 & 0	0
1 & 1	**1**

Exemples d'expressions booléennes

- (3>4) || (3!=4)

  true

- (3>4) || ("Bonjour"=="bonjour")

  false

- ((3-1==2) && ("B">"Allo")) || ("Allo".length==2)

  true

Conversion des booléens

- Les opérateurs **+**, **-**, *****, /, <,>,… convertissent les booléens en nombres ou textes automatiquement

```
> 33 * false
  0
> 33 + true
  34
> true + true
  2
> "allo " + true
  "allo true"
```

Attention!

- 5<=20<=10
  vaut **true.** Pourquoi?
- Ce qui se passe lors de l'évaluation:
  (5<=20)<=10
  true <=10
  1 <= 10
  true
- Comment corriger? L'intention: 5<=20 «**ET**» 20<=10
  (5<=20) && (20<=10) vaut **false**.

Quel est ton type?

- On peut examiner de quel type est une valeur (d'une expression ou d'une variable)
- fonction **typeof**(...)

```
> typeof(3.5)
"number"
> var a = 10
> typeof("a")
"string"
> typeof(a)
"number"
> typeof("a"==a)
"boolean"
```

Exercice: variables et expressions

- Indiquez l**a valeur** que contiendra la variable **et son**

```
> var a = 1+3;
> var b = “”;
> var c = b+“b”;
> var d = a+b+c;
> var e = +a+b;
> var f = a-+b;
> var g = +(a+b);
> var h = (e==f);
> var i = (e===f);
> var j = !(!i);
> var k = (“b”===c);
> var l = d+d+d;
> var m = l.charAt(0)+l.substring(0,3);
```