DÉVELOPPEMENT INFORMATIQUE

LOGIQUE & PROGRAMMATION

LANGAGE JAVASCRIPT

Y. DELVIGNE CH. LAMBEAU

UNE AUTRE VISION DES OBJETS :

LES "TABLEAUX" ASSOCIATIFS

1. PRÉLIMINAIRES (A) ... RAPPELS ...

PRINCIPALES CARACTÉRISTIQUES D'UN OBJET

les propriétés sont des noms

les valeurs sont des primitives ou des objets

```
var personne = {
  nom       : 'Dugenou',
  prenom    : 'Henri',
  naissance : new Date(1985, 5, 16)
}
```

- on peut ajouter de nouvelles propriétés à tout moment ...

```
personne.sexe = 'M';  // nouvelle propriété
```

- mais chaque propriété est unique (pas de doublon ➔ modification)

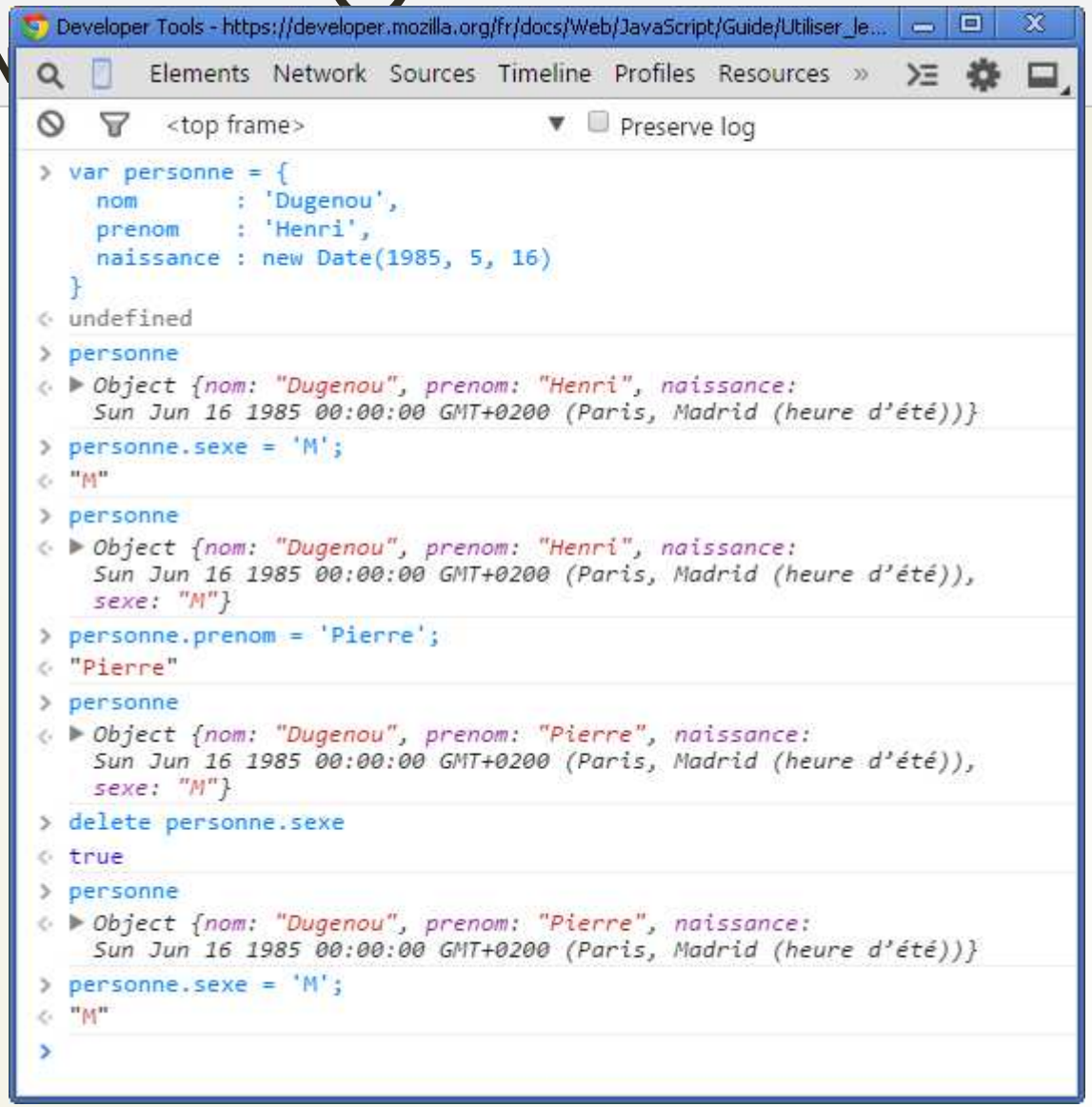
```
personne.prenom = 'Pierre'; // nouvelle valeur
```

- on peut supprimer une propriété ...

```
delete personne.sexe;
```

1. PRÉLIMINAIRES (A) ... RAPPELS ...

PRIN



1. PRÉLIMINAIRES (B)

QUELQUES UTILITAIRES BIENVENUS ...

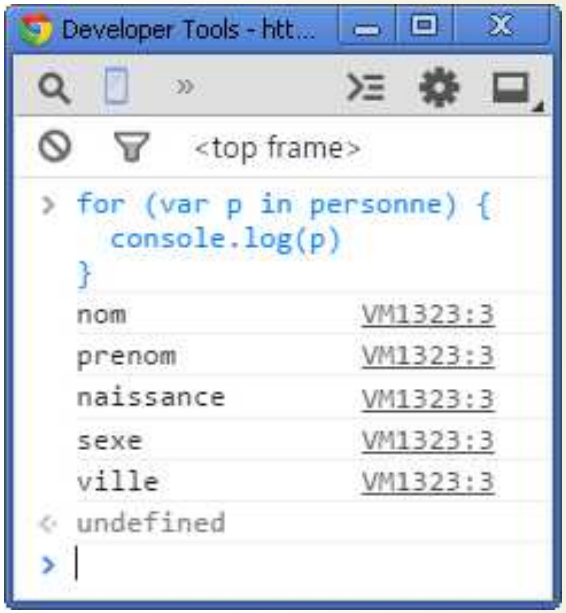
- l'opérateur **in** permet de savoir si une propriété existe

```
'sexe' in personne      // renvoie true
'adresse' in personne   // renvoie false
```

```
if (!('ville' in personne)) {
  personne.ville = 'Namur';
}
```

- l'itérateur **for ... in ...** énumère les propriétés

```
for (var p in personne) {
  console.log(p)
}
```



1. PRÉLIMINAIRES (C)

UNE AUTRE APPROCHE DES ACCESSEURS ...

- pour accéder à la valeur d'une propriété : l'accesseur **.** (dot)

```
personne.sexe = 'M';          // en écriture
console.log(personne.sexe);   // en lecture
```

- mais on peut aussi utiliser l'accesseur **[]**

```
personne['sexe'] = 'M';          // en écriture
console.log(personne['sexe']);   // en lecture
```

- cela permet l'accès indirect (via une variable)

```
var s = 'sexe';
console.log(personne[s]); // ceci fonctionne
console.log(personne.s);  // parce que cela non !
                          // s n'est pas une propriété
```

1. PRÉLIMINAIRES (D)

UNE AUTRE APPROCHE DES ACCESSEURS ...

- l'itérateur **for ... in ...** énumère les propriétés dans une variable ...

```
for (var p in personne) {
  console.log(p)
}
```

- ... et l'accesseur **[]** accède à la valeur de celle-ci

```
for (var p in personne) {
  console.log(p, personne[p])
}
```

```
nom Dugenou
prenom Pierre
naissance Sun Jun 16 1985 00:00:00 GMT+0200 (Paris, Madrid (heure d'été))
sexe M
ville Namur
```

1. PRÉLIMINAIRES (E)

ET UN DERNIER OUTIL ...

- la méthode **Object.keys()** renvoie un Array des (noms des) propriétés

```
var tp = Object.keys(personne);
```

- un tableau donc, sur lequel on peut utiliser les méthodes de Array

```
tp.sort(); // liste alpha des propriétés

// liste alpha des propriétés avec leur valeur
for(var i in tp) {
  console.log(tp[i], personne[tp[i]]);
}
```

NE PAS SE FORCER À UTILISER
CE QU'ON CONNAÎT DÉJÀ !!

2. TABLEAU ASSOCIATIF ... QU'EST-CE QUE C'EST ?

- UNE STRUCTURE NON ORDONNÉE (UNE LISTE)
- CONSTITUÉE DE COUPLES CLÉS - VALEURS
- LES CLÉS SONT UNIQUES ET DE TYPE "STRING"
- LA VALEUR EST ACCESSIBLE DIRECTEMENT PAR LA CLÉ

```
consultations
- lundi    : 08h30 - 12h30
- mardi    : 13h00 - 18h00
- jeudi    : 08h30 - 12h30
- vendredi : 13h00 - 20h00
- samedi   : 09h00 - 12h00
```

```
cours
- t101 : Mathématiques
- t104 : Initiation
- t102 : Programmation
- t111 : Electricité
- t109 : Anglais
```

- on veut pouvoir répondre/faire **directement** :
  - *quelles sont les heures de consultation du jeudi ?*
  - *y a-t-il consultation le mercredi ?*
  - *quel est l'intitulé du cours t111 ?*
  - *je veux créer un nouveau cours t103 : Web 2.0*

2. TABLEAU ASSOCIATIF ... AVEC DES ARRAY (D'OBJETS) ?

```
var consultations = [
  {jr: 'lundi',    hr: '08h30 - 12h30'},
  {jr: 'mardi',    hr: '13h00 - 18h00'},
  {jr: 'jeudi',    hr: '08h30 - 12h30'},
  {jr: 'vendredi', hr: '13h00 - 20h00'},
  {jr: 'samedi',   hr: '09h00 - 12h00'}
];
```

```
var cours = [
  {code: 't101', lib: 'Mathématiques'},
  {code: 't104', lib: 'Initiation'},
  {code: 't102', lib: 'Programmation'},
  {code: 't111', lib: 'Electricité'},
  {code: 't109', lib: 'Anglais'}
];
```

- un tableau (Array)
  - est une structure **ordonnée** (des index 0, 1, ...)
  - dans laquelle l'accès est **positionnel** (via index)
  - donc : pour trouver **il faut chercher** (l'opposé de ce que l'on espère)

2. TABLEAU ASSOCIATIF ... AVEC DES ARRAY (D'OBJETS) ?

```
var consultations = [
  {jr: 'lundi',    hr: '08h30 - 12h30'},
  {jr: 'mardi',    hr: '13h00 - 18h00'},
  {jr: 'jeudi',    hr: '08h30 - 12h30'},
  {jr: 'vendredi', hr: '13h00 - 20h00'},
  {jr: 'samedi',   hr: '09h00 - 12h00'}
];
```

heures de consultation du jeudi ?

y a-t-il consultation le mercredi ?

```
function cherche(jour) {
  for(var j in consultations) {
    if(consultations[j].jr == jour) {
      return consultations[j].hr;
    }
  }
  return 'pas de consultation';
}

console.log(cherche('jeudi'));
console.log(cherche('mercredi'));
```

recherche : il faut passer par une itération du tableau en utilisant les index numériques

pas d'accès direct !

2. TABLEAU ASSOCIATIF ... AVEC DES ARRAY (D'OBJETS) ?

```
var consultations = [
  {jr: 'lundi',    hr: '08h30 - 12h30'},
  {jr: 'mardi',    hr: '13h00 - 18h00'},
  {jr: 'jeudi',    hr: '08h30 - 12h30'},
  {jr: 'vendredi', hr: '13h00 - 20h00'},
  {jr: 'samedi',   hr: '09h00 - 12h00'}
];
```

heures de consultation du jeudi ?
y a-t-il consultation le mercredi ?

```
function cherche(jour) {
  var t = consultations.filter(function(x) {
                                  return (x.jr == jour)
                               });
  if t.length != 0 return 'pas de consultation';
  return t[0].hr;
}
console.log(cherche('jeudi'));
console.log(cherche('mercredi'));
```

même avec l'API sur Array ...
il faut 'faire' ...
pas d'accès direct !

2. TABLEAU ASSOCIATIF ... AVEC DES ARRAY (D'OBJETS) ?

```
var cours = [
  {code: 't101', lib: 'Mathématiques'},
  {code: 't104', lib: 'Initiation'},
  {code: 't102', lib: 'Programmation'},
  {code: 't111', lib: 'Electricité'},
  {code: 't109', lib: 'Anglais'}
];
```

intitulé du cours t111 ?

créer t103 : Web 2.0

```
function cherche(cde) {
  for(var c in cours) {
    if(cours[c].code == cde) {
      return cours[c].lib;
    }
  }
  return 'inconnu';
}

console.log(cherche('t111'));

if(cherche('t103') == 'inconnu'){
  cours.push({code:'t103', lib:'Web 2.0'});
}
```

recherche : il faut passer par une itération du tableau en utilisant les index numériques
pas d'accès direct !

ajout : assez facile mais il faut une recherche préalable d'inexistence

2. TABLEAU ASSOCIATIF ... AVEC DES ARRAY (D'OBJETS) ?

```
var cours = [
  {code: 't101', lib: 'Mathématiques'},
  {code: 't104', lib: 'Initiation'},
  {code: 't102', lib: 'Programmation'},
  {code: 't111', lib: 'Electricité'},
  {code: 't109', lib: 'Anglais'}
];
```

intitulé du cours t111 ?

créer t103 : Web 2.0

```
function existe(cde) {
  var t = cours.filter(function(x) {
                          return (x.code == cde);
                       });
  return t.length != 0;
}

console.log(existe('t111'));

if(! existe('t103')) {
  cours.push({code:'t103', lib:'Web 2.0'});
}
```

alternative via API
pas d'accès direct !

2. TABLEAU ASSOCIATIF ... FORCER AVEC DES ARRAY ?

```
var consultations = [];

consultations['lundi']    = '08h30 - 12h30';
consultations['mardi']    = '13h00 - 18h00';
consultations['jeudi']    = '08h30 - 12h30';
consultations['vendredi'] = '13h00 - 20h00';
consultations['samedi']   = '09h00 - 12h00';
```

clé

valeur

```
var cours = [];

cours['t101'] = 'Mathématiques';
cours['t104'] = 'Initiation';
cours['t102'] = 'Programmation';
cours['t111'] = 'Electricité';
cours['t109'] = 'Anglais';
```

plus simple à première vue

on sait faire ça avec un Array ?

oui hélas !

```
console.log(consultations['jeudi']);     // '08h30 - 12h30'
console.log(consultations['mercredi']); // undefined

console.log(cours['t111']);              // Electricité
if(cours['t103'] == undefined){          // si n'existe pas
  cours['t103'] = 'Web 2.0';             // crée
}
```

accès direct, non ?

oui mais ...

2. TABLEAU ASSOCIATIF ... FORCER AVEC DES ARRAY ?

```
var cours = [];

cours['t101'] = 'Mathématiques';
cours['t104'] = 'Initiation';
cours['t102'] = 'Programmation';
cours['t111'] = 'Electricité';
cours['t109'] = 'Anglais';
```

clé

valeur

- en faisant cela, le tableau a perdu son statut d'Array !

```
cours;                                  // [] !!
console.log(cours.length);              // 0 !!
cours.sort();                           // [] !!
cours.map(function(x)
          { return x.toUpper();})       // aucun effet
```

- tout Array dont on n'exploite pas explicitement les index numériques perd ses propriétés et ses méthodes !

PRÊT(E)S À PENSER 'AUTREMENT' ??

3. TABLEAU ASSOCIATIF ... COMMENT FAIRE ALORS ???

- penser les données autrement :
  puisque les clés
  - ➢ sont alphanumériques
  - ➢ sont uniques

```
var cours = [];

cours['t101'] = 'Mathématiques';
cours['t104'] = 'Initiation';
cours['t102'] = 'Programmation';
cours['t111'] = 'Electricité';
cours['t109'] = 'Anglais';
```

clés

valeurs

➔ alors elle peuvent convenir comme propriétés **d'objet**

```
var cours = {
  t101 : 'Mathématiques',
  t104 : 'Initiation',
  t102 : 'Programmation',
  t111 : 'Electricité',
  t109 : 'Anglais'
}
```

propriétés

valeurs

3. TABLEAU ASSOCIATIF ... FINALEMENT ? AVEC OBJECT !

```
var cours = {
  t101 : 'Mathématiques',
  t104 : 'Initiation',
  t102 : 'Programmation',
  t111 : 'Electricité',
  t109 : 'Anglais'
}
```

clé — valeur

```
var consultations = {
  lundi    : '08h30 - 12h30',
  mardi    : '13h00 - 18h00',
  jeudi    : '08h30 - 12h30',
  vendredi : '13h00 - 20h00',
  samedi   : '09h00 - 12h00'
}
```

```
console.log(consultations.jeudi);       // '08h30 - 12h30'
console.log('mercredi' in consultations); // false

console.log(cours.t111);                // Electricité
if(!('t103' in cours)){                 // si n'existe pas
  cours.t103 = 'Web 2.0';                 // créer
}
```

- un Object est tout naturellement un tableau associatif :
  le couple clé-valeur est offert par le couple propriété-valeur

3. TABLEAU ASSOCIATIF ... FINALEMENT ? AVEC OBJECT !

```
var cours = {
  t101 : 'Mathématiques',
  t104 : 'Initiation',
  t102 : 'Programmation',
  t111 : 'Electricité',
  t109 : 'Anglais'
}
```

clé

valeur

```
var consultations = {
  lundi    : '08h30 - 12h30',
  mardi    : '13h00 - 18h00',
  jeudi    : '08h30 - 12h30',
  vendredi : '13h00 - 20h00',
  samedi   : '09h00 - 12h00'
}
```

- et les autres fonctionnalités ?
  un Array est le bienvenu, mais comme outil de travail ...

```
crs = Object.keys(cours); // tableau des codes

console.log(crs.length); // 6 (après ajout)

crs.sort();                    // trier les codes

for(var c in crs) { // liste triée code et libellés
  console.log(crs[c], cours[crs[c]]);
}
```

3. TABLEAU ASSOCIATIF ... IMBRICATIONS ...

```
var cours = {
  t101 : { lib : 'Mathématiques', ects: 6 },
  t104 : { lib : 'Initiation',    ects: 1 },
  t102 : { lib : 'Programmation', ects: 11 },
  t111 : { lib : 'Electricité',   ects: 6 },
  t109 : { lib : 'Anglais',       ects: 4 }
}
```

- et les autres fonctionnalités ?
  un Array est le bienvenu, mais comme outil de travail ...

```
if (!('t103' in cours)){
  cours.t103 = {lib : 'Connectique', ects: 1};
}
crs = Object.keys(cours).sort(); // tableau des codes
for(var c in crs) { // liste triée code et libellés
  console.log(crs[c], cours[crs[c]].lib);
}
```

4. TABLEAU ASSOCIATIF ... ON GÉNÉRALISE ?

```
var cours1T = {
  t101 : 'Mathématiques',
  t104 : 'Initiation',
  t102 : 'Programmation',
  t111 : 'Electricité',
  t109 : 'Anglais'
}
```

```
var cours2T = {
  t202 : 'Java',
  t200 : 'Bases de données',
  t205 : 'jQuery Ajax',
  t201 : 'Réseaux',
  t204 : 'Electronique'
}
```

```
var cours = {
  t1 : cours1T,
  t2 : cours2T,
  t3 : { t301 : 'Traitement de signal',
         t303 : 'Routage & Commutation',
         t302 : 'Sécurité',
         t398 : 'Stages',
         t399 : "Travail de Fin d'Etudes"
       }
}
```

```
console.log(cours.t1.t102); // Programmation

console.log(Object.keys(cours.t3).length); // 5
```