Javascript

Xavier Le Pallec

Javascript 6, ECMAScript 6, ECMAScript 2015

Y était temps !

Master 2 Informatique - parcours e-Services

Let, var & scope

Scope for ever

Let, var & scope

Var

```
function testVar() {
    for (var i = 0; i < 10; i++) {
        setTimeout(
            function () {
                console.log(i);
            },
            2000
        )
    }
}
testVar();
```

```
10
10
10
10
10
10
10
10
10
10
```

Let, var & scope

Let

```
function testLet() {
    for (let i = 0; i < 10; i++) {
        setTimeout(
            function () {
                console.log(i);
            },
            2000
        )
    }
}
testLet();
```

```
0
1
2
3
4
5
6
7
8
9
```

Let, var & scope

Let

Var crée une variable qui est valable pour toute la fonction

Let crée une variable qui est valable pour tout le bloc où elle se trouve (ou pour une seule itération dans le cas d'une boucle)

Let

Imbrications

```
function testLetLet() {
    var tableau = [
        [0,1,2,3,4],
        [10,11,12,13,14],
        [20,21,22,23,24],
        [30,31,32,33,34]
    ]
    for (let i=0;i<4;i++) {
        var chaine="";
        sousTableau=tableau[i];
        for (let i=0;i<5;i++) {
            chaine+=sousTableau[i]+",";
        }
        console.log(chaine);
    }
}
testLetLet();
```

Imbrications non possibles avec des var

```
0,1,2,3,4,
10,11,12,13,14,
20,21,22,23,24,
30,31,32,33,34,
```

Scope

Des blocs sans mots-clés/structures

```
function test () {
    let i=1;
    console.log(i);
    {
        let i=2;
        console.log(i);
    }
    console.log(i);
}
test();
```

```
1
2
1
```

Scope

Idem pour les fonctions

```
{
    function f1() {
        console.log("1");
    }
    f1();
    {
        function f1() {
            console.log("2");
        }
        f1();
    }
    f1();
}
```

```
1
2
1
```

Arrow functions

Fonctions fléchées

Fonctions fléchées

Principe

```
[10,11,12,13,14].forEach(
    (element, index) => console.log(index+"="+element)
)
```

```
0=10
1=11
2=12
3=13
4=14
```

Fonctions fléchées

Valeur retournée

```
tableau=[25,10,2,90,55];
tableau.sort(
    (a, b) => (a>b?1:(a<b?-1:0))
)
console.log(tableau);
```

```
tableau=[25,10,2,90,55];
tableau.sort(
    (a, b) => {
        if (a>b)
            return 1;
        if (a<b)
            return -1;
        return 0;
    }
)
console.log(tableau);
```

```
[ 2, 10, 25, 55, 90 ]
```

Fonctions fléchées

Le this reste accessible !

```
objet = {
    x : 3,
    method : function () {
        function fois2 () {
            console.log(this);
        }
        fois2();
    }
}
objet.method();
```

```
Window
```

```
objet = {
    x : 3,
    method : function () {
       [1,2].forEach(
       () =>  console.log(this)
      );

    }
}
objet.method();
```

```
{x: 3, method: function}
{x: 3, method: function}
```

Paramètres et sous-ensembles

Paramètres

Valeur par défaut

```
function bonjour (texte, qui = "à tous", iteration = 1) {
    for (let i=0; i<iteration; i++)
        console.log(texte + " " + qui);
}
```

```
> bonjour ("salut")
salut à tous

> bonjour ("Hi","all",2)
Hi all
Hi all

> bonjour ("Hi",undefined,2)
Hi à tous
Hi à tous
```

Paramètres

Trois petits points

```
function ajouterBalise (baliseMere, typeBalise, ...attributs) {
    var balise=document.createElement(typeBalise);
    baliseMere.appendChild(balise);
    for (var i=0;i<attributs.length;i+=2)
        balise.setAttribute(attributs[i],attributs[i+1]);
}
```

```
> ajouterBalise(document.body,"textarea","rows",10,"cols",50)
```

Sous-ensembles

Toujours les trois petits points

```
att1 = ["rows",10,"cols",50];
att2 =  ["placeholder", "écrire ici", ...att1];
```

```
> ajouterBalise (document.body,"textarea",  ...att2);
```

```
att3 =  ["placeholder", "écrire ici", ...att1, "maxlength", 300];
```

Modèles de libellés

Template liberals

Modèles de libellés

Guillemet en biais, $ et \n

```
client = {
    nom : "Maqueron",
    ville : "Paris",
    preference : "Doigts d'honneur"
}
```

```
> console.log (`Bonjour Mr ${client.nom},
comme vous aimez les ${client.preference},
je peux vous en proposer pour un bon prix à $
{client.ville}`);

Bonjour Mr Maqueron,
comme vous aimez les Doigts d'honneur,
je peux vous en proposer pour un bon prix à Paris
```

Propriétés d'objet

Attention, ça tue !

Propriétés d'objet

Nom et valeur en un coup

```
var nom = "Maqueron";
let ville = "Paris";
client = { nom, ville };
```

```
> client
{nom: "Maqueron", ville: "Paris"}
```

Propriétés d'objet

Définition et valeur dynamique

```
function Objet (pt1, valeur1, pt2, valeur2) {
    return { pt1 : valeur1, pt2 : valeur2};
}

function Objet6 (pt1, valeur1, pt2, valeur2) {
    return { [pt1] : valeur1, [pt2] : valeur2};
    // avant ça marchait pas
}
```

```
> Objet ("nom", "Maqueron", "ville", "Paris")
{pt1: "Maqueron", pt2: "Paris"}

> Objet6 ("nom", "Maqueron", "ville", "Paris")
{nom: "Maqueron", ville: "Paris"}
```

Propriétés d'objet

Méthode

```
client = {
    nom : "Maqueron",
    ville : "Paris",
    sayHello () {
        console.log("Bonjour la France !");
    }
}
```

```
> client.sayHello()
Bonjour la France !
```

Affectations tout azimut

WTF!

Affectations tout azimut

Avec les tableaux

```
var [a,,b] = [1,2,3];
var monTableau = [4,5,6];
[c,d] = monTableau;

console.log(a);
console.log(b);
console.log(c);
console.log(d);
```

```
1
3
4
5
```

Affectations tout azimut

Avec les objets

```
var client = { nom : "Aznavour", prenom : "Charles" , age : 94 };
var {a,b,c} = client;
var {prenom,age,nom} = client;

console.log(a);
console.log(b);
console.log(c);
console.log(nom);
console.log(prenom);
console.log(age);
```

```
undefined
undefined
undefined
Aznavour
Charles
94
```

Affectations tout azimut

Avec les objets++

```
var client = {
    nom: "Aznavour",
    prenom: "Charles",
    age: 94,
    adresse: {
        ville: "Mouriès",
        dpt: 13
    }
};
var {prenom: a, age: b, nom: c, adresse: {ville: d}} = client;

console.log(a);
console.log(b);
console.log(c);
console.log(d);
```

```
Charles
94
Aznavour
Mouriès
```

Affectations tout azimut

Objet en paramètres

```
mots = [
    { mot : "Collomb", occurences : 24},
    { mot : "Aznavour", occurences : 15}
]
function afficherClassement ({mot, occurences}) {
    console.log (occurences+" fois le mot "+mot);
}

mots.forEach(
    element => afficherClassement(element)
);
```

```
24 fois le mot Collomb
15 fois le mot Aznavour
```

Affectations tout azimut

Objet en paramètres 2

```
mots = [
    { mot : "Collomb", occurences : 24},
    { mot : "Aznavour", occurences : 15}
]
function afficherClassement ({mot : m, occurences : o}) {
    console.log (o+" fois le mot "+m);
}

mots.forEach(
    element => afficherClassement(element)
);
```

```
24 fois le mot Collomb
15 fois le mot Aznavour
```

Affectations tout azimut

Tableau en paramètres

```
mots = [
    [ "Collomb", 24],
    [ "Aznavour", 15 ]
]
function afficherClassement ([mot, occurences]) {
    console.log (occurences+" fois le mot "+mot);
}

mots.forEach(
    element => afficherClassement(element)
);
```

```
24 fois le mot Collomb
15 fois le mot Aznavour
```

Affectations tout azimut

Objet en paramètres et valeur par défaut

```
mots = [
    { mot : "Collomb", occurences : 24},
    { mot : "Aznavour", occurences : 15},
    { occurences : 15},
    { mot : "Macron"},
]
function afficherClassement ({mot = "mot absent", occurences = 0}) {
    console.log (occurences+" fois le mot "+mot);
}

mots.forEach(
    element => afficherClassement(element)
);
```

```
24 fois le mot Collomb
15 fois le mot Aznavour
15 fois le mot mot absent
0 fois le mot Macron
```

Affectations tout azimut

Ça marche aussi avec les tableaux

```
mots = [
    [ "Collomb", 24],
    [ "Aznavour", 15 ],
    [, 15],
    ["Macron"]
];

function afficherClassement ([mot = "mot absent", occurences = 0]) {
    console.log (occurences+" fois le mot "+mot);
}

mots.forEach(
    element => afficherClassement(element)
);
```

```
24 fois le mot Collomb
15 fois le mot Aznavour
15 fois le mot mot absent
0 fois le mot Macron
```

Les classes

Les vraies

Les classes

La base - après - avant

```
class Planning {
    constructor(baliseMere, listeJours, largeurJour, creneaux) {
        this.baliseMere = baliseMere;
        this.listeJours = listeJours;
        ...
    }

    construitColonnesJours() {
        this.baliseColonnesJours = {};
        for (var i = 0; i < this.listeJours.length; i
           ...
        }
    }

    trierCreneaux() {
        this.creneaux.sort(
            function (creneau1, creneau2) {
                ...
                return 0;
            }
        )
    }

    afficherCreneaux() {
        for (var i = 0; i < this.creneaux.length; i++
            ...
        }
    }
}
```

```
function Planning
  (baliseMere, listeJours, largeurJour, creneaux) {
    this.baliseMere=baliseMere;
    this.listeJours=listeJours;
    ...
}

Planning.prototype.construitColonnesJours = function () {
    this.baliseColonnesJours={};
    for (var i=0;i<this.listeJours.length;i++) {
        ...
    }
}

Planning.prototype.trierCreneaux = function () {
    this.creneaux.sort (
        function (creneau1, creneau2) {
            ...
            return 0;
        }
    )
}

Planning.prototype.afficherCreneaux = function () {
    for (var i=0;i<this.creneaux.length;i++) {
        ...
    }
}
```

Les classes

L'héritage - après - avant

```
class PlanningColore extends Planning {
    constructor
    (baliseMere, listeJours, largeurJour,
     creneaux, listeCouleurs) {
        super(baliseMere, listeJours, largeurJour, creneaux);
        this.listeCouleurs = listeCouleurs;
    }
}
```

```
function PlanningColore (
        baliseMere, listeJours, largeurJour,
            creneaux, listeCouleurs )
{
    Planning.call(this, baliseMere, listeJours, largeurJour, creneaux);
    this.listeCouleurs = listeCouleurs;
}
PlanningColor.prototype=new Planning();
```

Les classes

Super... pour les méthodes

```
class PlanningColore extends Planning {
    constructor
    (baliseMere, listeJours, largeurJour,
     creneaux, listeCouleurs) {
        ...
    }
    afficherCreneaux() {
        super.afficherCreneaux();
        ...
    }
}
```

```
PlanningColor.prototype.afficherCreneaux = function () {
    Planning.prototype.afficherCreneaux.call(this);
    ...
}
```

Les classes

Méthode statique

```
class Planning {
    constructor(baliseMere, listeJours, largeurJour, creneaux) {
        this.baliseMere = baliseMere;
        this.listeJours = listeJours;
        if (largeurJour==undefined)
            this.largeurJour =
                Planning.largeurDeBase(); // this.largeurDeBase ne fonctionne pas
        else
            this.largeurJour = largeurJour;
        this.construitColonnesJours();

        this.creneaux = creneaux;
        this.trierCreneaux();

        this.afficherCreneaux();
    }

    static largeurDeBase () {
        return 2;
    }
```

```
Planning.largeurDeBase = function () {
    return 2;
}
```

Les classes

Les get/set

```
class PanelCreneau {
    constructor(baliseMere, creneau) {
        ...
    }

    creerPanel() {
        ...
    }

    set entete(texte) {
        this.panelHeading.appendChild(document.createTextNode(texte));
    }

    set corps(texte) {
        this.panelBody.appendChild(document.createTextNode(texte));
        this.panelBody.appendChild(document.createElement("BR"));
    }

    set pied(texte) {
        this.panelFooter.appendChild(document.createTextNode(texte));
    }
}
```

Les classes

On peut aussi le faire à des objets

```
var client = {
    nom: "Aznavour",
    prenom: "Charles",
    get age () { return 94},
    set age (value) { console.log("rien à faire, c'est 94");},
    adresse: {
        ville: "Mouriès",
        dpt: 13
    }
};

console.log(client.age);
client.age=12;
```

```
94
rien à faire, c'est 94
```

Les classes

En interne du constructeur

```
class Point {
    constructor (x,y) {
        var _x=x;
        var _y=y;
        Object.defineProperty(this,"x", {
            get : function () { return _x;},
            set : function (value) {_x=value;}
            }
        )
        Object.defineProperty(this,"y", {
                get : function () { return _y;},
                set : function (value) {_y=value;}
            }
        )
    }
}
```

```
> p=new Point(50,10);
> console.log(p.x);
50
> p.x=4;
> console.log(p.x);
4
> console.log(p._x);
undefined
> console.log(_x);
undefined
```