JS : Structures de contrôle et gestion d'exceptions

L2 MPCIE - UE Développement Web

David Lesaint

david.lesaint@univ-angers.fr

Janvier 2019

Structures de contrôle

JS possède les structures de contrôle classiques :

- `if/else`
- `switch/case`
- `for` pour les tableaux
- `for/in` pour les objets
- `while`, `do/while`
- `break/continue`, `label`
- `throw/try/catch/finally`

Les blocs

Délimités par une paire d'accolades

Les variables déclarées avec `var` sont dans la portée de la fonction ou du script englobant (depuis ES6).

controle-bloc.js

```
1 var x = 1;
2 {
3   var x = 2;
4   let y= 3;
5   z = 4;
6 }
7 console.log(x); // 2
8 // console.log(y); ReferenceError
9 console.log(z); // 4
```

Les instructions conditionnelles

if...else

controle-if-else.js

```
1 if (condition_1) {
2   instruction_1;
3 } else if (condition_2) {
4   instruction_2;
5 } else if (condition_n) {
6   instruction_n;
7 } else {
8   dernière_instruction;
9 }
```

Les instructions conditionnelles

switch

controle-switch.js

```
1  switch (fruit) {
2  case "Orange":
3    console.log("Les oranges.");
4    break;
5  case "Pomme":
6  case "Banane":
7    console.log("Les pommes et bananes.");
8    break;
9  default:
10   console.log("Pas de " + fruit + ".");
11 }
```

Valeurs équivalentes à `false` en contexte booléen

- `false`
- `undefined`
- `null`
- `0`
- `NaN`
- `""` (la chaîne vide).

controle-falsy-values.js

```
1 var b = new Boolean(false);
2 if (b)  // condition vérifiée car b est un objet
3   console.log(b);
4 if (b == false)  // condition vérifiée
5   console.log(b);
```

Gestion des erreurs

Déclenchement

- Par le moteur JS pendant l'exécution.
- Programmatiquement avec `throw`.

Interception et traitement

Avec `try{...} catch(e){...} finally{...}` :

- Bloc `try` : code à surveiller.
- Bloc `catch` : interception et traitement de l'erreur (objet).
- Bloc `finally` (optionnel) : code à exécuter dans tous les cas (eg. libération de ressources).

Gestion des erreurs

Erreurs générales

Créées avec constructeur `Error([message[, file[, line]]])`

- `file` : par défaut, nom du fichier invoquant le constructeur.
- `line` : par défaut, numéro de ligne invoquant le construc.

Exceptions spécifiques

`RangeError`	Variable/paramètre numérique en dehors d'une plage de validité.
`ReferenceError`	Déréférencement d'une référence invalide.
`SyntaxError`	Erreur de syntaxe dans l'évaluation de code par la fonction `eval()`.
`TypeError`	Variable/paramètre de type invalide.
`URIError`	Paramètres invalides passées à `encodeURI()` ou à `decodeURI()`.

`Error.prototype`

- Propriétés : `message`, `name`.

Gestion des erreurs

Erreurs liées à l'accès au DOM

Créées avec constructeur `DOMException([message[, name]])`

- `name` représente une constante associée au type d'erreur DOM.

`DOMException.prototype`

- Propriétés : `code`, `message`, `name`.

Gestion des erreurs

controle-exception-range.js

```
1  var vérifier = function(num) {
2    if (num < MIN || num > MAX) {
3      throw new RangeError("Paramètre entre " + MIN + " et "
+ MAX);
4    }
5  };
6
7  try {
8    vérifier(500);
9  } catch (e) {
10   if (e instanceof RangeError) {
11     console.log(e.name + ": " + e.message);
12   }
13 }
```

Les boucles : instruction `for`

controle-for.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
  <form name="formulaire">
    <select id="music" name="music" multiple="multiple">
      <option selected="selected">R&B</option>
      <option>Jazz</option>
      <option>Blues</option>
      <option>New Age</option>
    </select>
    <button id="btn" type="button">Combien sont sélectionnés ?</button>
  </form>
  <script>
    function quantité(objet) {
      var q = 0;
      for (var i = 0; i < objet.options.length; i++) {
        if (objet.options[i].selected) {
          q++;
        }
      }
      return q;
    }

    var btn = document.getElementById("btn");
    btn.addEventListener("click", function() {
      alert('Nombre d\'options choisies : '
          + quantité(document.formulaire.music))
    });
  </script>
```

Les boucles : instructions `while` et `do...while`

controle-while-dowhile.js

```
1  var n = 0, x = 0;
2  while (n < 3) {
3    n++;
4    x += n;
5  }
6  console.log(x);
7
8  var i = 0;
9  do {
10   i += 1;
11 } while (i < 5);
12 console.log(i);
```

Les instructions `break` et `label`

Deux usages de `break` : avec ou sans label

- Sans : interrompt l'instruction `while`, `do...while`, `for` ou `switch` la plus imbriquée.
- Avec : interrompt l'instruction correspondante.

controle-break-label.js

```
1  var x = 0;
2  var z = 0;
3  labelAnnuleBoucle: while (true) {
4    console.log("Boucle externe : " + x);
5    x += 1;
6    z = 1;
7    while (true) {
8      console.log("Boucle interne : " + z);
9      z += 1;
10     if (z === 10 && x === 10) {
11       break labelAnnuleBoucle;
12     } else if (z === 10) {
13       break;
14     }
15   }
16 }
```

Les instructions `continue` et `label`

Deux usages de `continue` : avec ou sans label

- Sans : fait passer à l'exécution de l'itération suivante dans l'instruction `while`, `do...while`, `for` ou `switch` la plus imbriquée.
- Avec : fait passer à l'itération suivante de l'instruction correspondante.

controle-continue-label.js

```
1  var i = 0;
2  var n = 0;
3  while (i < 5) {
4     i++;
5     if (i == 3) {
6        continue;
7     }
8     n += i;
9  }
10 console.log(n);
```

L'instruction `for...in`

Pour itérer sur l'ensemble des propriétés *énumérables* d'un objet

Ne pas utiliser sur les tableaux !

controle-for-in.js

```
1  function afficher(obj) {
2      var result = "";
3      for (var i in obj) {
4        result += i + " = " + obj[i] + "\n";
5      }
6      result += "\n";
7      console.log(result);
8    }
9
10   teslaS = { fabricant:"Tesla", model:"S"};
11   afficher(teslaS);
```

L'instruction `for...of`

Pour itérer sur les objets *itérables*

- Dont `Array`, `Map`, `Set`, l'objet `arguments`.
- `for...of` utilise un mécanisme d'itération propre à l'objet pour énumérer les valeurs de ses différentes propriétés.

controle-for-of.js

```
1  let arr = [3, 5, 7];
2  arr.toto = "coucou";
3
4  for (let i in arr) {
5    console.log(i); // affiche 0, 1, 2, "toto" dans la console
6  }
7
8  for (let i of arr) {
9    console.log(i); // affiche 3, 5, 7 dans la console
10 }
```