

Aller plus loin

JS





LE PROTOTYPAGE

2

3

- Tout type est “wrappé” par un objet
- Tout objet contient un propriété **prototype** (ou **__proto__**)
- **prototype** contient l'ensemble des méthodes et propriétés accessibles de l'objet hérité
- Héritage

```
const Object2 =  
Object.create(Object1);
```


Déclaration d'un objet

- Accolade (A)

```
const myObject = {  
  p1: "foo"  
};
```

- Constructeur (C)

```
function MyObject() {  
  this.p1 = "foo";  
}
```

```
const myObject = new  
MyObject()
```

5

Publique

Propriétés

- (A)
{
 p1: "foo"
};

- (C)
function MyObject() {
 this.p1 = "foo";
}

Méthodes

- (A)
{
 m1: function() {}
};

- (C)
function MyObject() {
 this.m1 = function() {};
}

Prototype

MyObject.prototype.p1 = "foo";
MyObject.prototype.m1 = function() {};

6

Privé

Propriétés

- (A)

Impossible

- (C)

```
function MyObject() {  
  let pr1 = "foo";  
}
```

Méthodes

- (A)

Impossible

- (C)

```
function MyObject() {  
  let mr1 = function() {};  
}
```

selecting=false;e.selected=true;e.startselected=true;e.
tend(a.ui.selectable,{version:"1.8.16"})))(jQuery);
a.widget("ui.sortable",a.ui.mouse,{widgetEventPrefix:"sort
rent",axis:false,connectWith:false,containment:false,curs
:false,helper:"original",items:">
alse,placeholder:false,revert:false,scroll:true,scrollSens
ptions;this.containerCache={};this.element.addClass("ui-se
);this.floating=this.items.length?d.axis=="x"||/left|right
is.items[0].item.css("display")):false;this.offset=this.el
abled").removeData("sortable").unbind(".sortable");this.no
n this},_setOption:function(d,c){if(d=="disabled")this._
this.options[d]=c;this.widget()[c?"addClass":"removeClass"]
_setOption.apply(this,arguments)},_mouseCapture:func
s.options.disabled||this.options.type=="static"return false
this,"sortable-item")==h){e=a(this);return false};if(a.dat
lse;a(this).options.handle,e).find("*").andSelf().each(func
currentItem=e;this._removeCurrentFromItems();return true);
currentContainer=this;this.refreshPositions();this.helper.offset
this.offset=this.currentItem.offset();this.helper.css("position","abs
relative:this._getRelativeOffset();this.helper.css("position","abs

7

Privilégié

Propriétés

- (A)

Impossible

- (C)

Impossible

Méthodes

- (A)

Impossible

- (C)

```
function MyObject() {  
  const pr1 = "foo";  
  this.m11 =function() {  
    console.log(pr1);  
  };  
}
```



Exercice 3 - Prototypons

8

Reprendre l'exercice 1 - Manipulation des chaînes

- Rendre toutes les fonctions accessibles pour chaque String, excepté `prop_access`

Exemples:

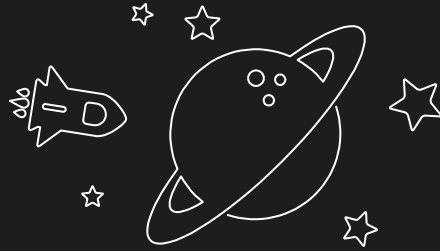
```
ucfirst("ma chaine") => "ma chaine".ucfirst()
```

```
vig("ma chaine", "ma clé") => "ma chaine".vig("ma clé")
```

- Rendre `prop_access` accessible pour chaque Object

Exemple:

```
prop_access(object, "animal.type.name") => object.prop_access("animal.type.name")
```



Exercise 4

Here we go!

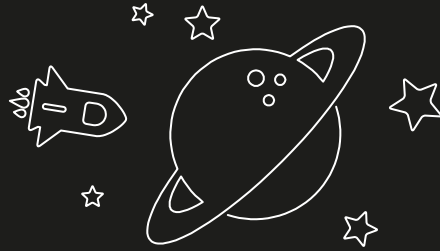
10

“

<https://github.com/kmarques/esgi-javascript/blob/master/exercise-4.md>

11

Commit : [DONE] exercice 4



Les Exceptions

12

13

- 7 types d'exceptions

EvalError : erreur dans un `eval()`

RangeError : utilisation d'un nombre en dehors des valeurs possibles

ReferenceError : utilisation d'une variable non déclarée ou hors scope

SyntaxError : erreur de syntaxe dans le code

TypeError : utilisation d'une fonction n'appartenant pas au type

URIError : utilisation de caractères illégaux dans une fonction URI

Error : Erreur générale, base de toutes les autres

14

Gestion des exceptions

try {

Permet d'exécuter des instructions à risque
throw "exc" | *new Error("exc");*

} catch (error) {

Catch les exceptions levées

If (error instanceof TypeError)
ou

If (error.name === "ReferenceError")
console.error(error.message);

} finally {

Permet d'exécuter des instructions même si
des exceptions sont levées

}

15

- Possibilité de créer ses propres exceptions

```
function MyError(param1, param2, ...) {  
    // ... Votre code vient ici
```

```
    const instance = new Error("custom message");  
    instance.name = "MyError";  
    Object.setPrototypeOf(  
        instance, Object.getPrototypeOf(this)  
    );  
    if (Error.captureStackTrace) {  
        Error.captureStackTrace(instance, MyError);  
    }  
    return instance;  
}
```



Exercise 5

THE Exception

16

`./exercice-5/exception.js`

- Reprendre l'exercice 3, uniquement la méthode `prop_access`

- Créer une exception `UndefinedPropertyError(path, property, array)`

Message: "Property '{property}' not exist for path '{path}', expected one of : [available object properties]"

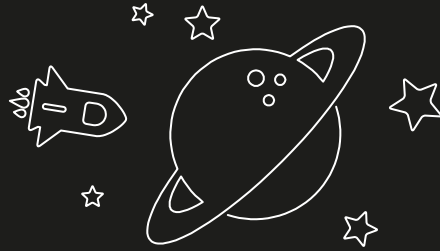
Ex: `UndefinedPropertyError('animal.gender', 'gender', {type: "dog", name: "spoky"})`

=> "Property 'gender' not exist for path 'animal.gender', expected one of : [type, name]"

- Créer une fonction **test** qui catch l'exception et affiche

Si exception, **"Exception caught"**

Sinon, **`JSON.stringify(valeur retourné)`**



Les Promises

18



19

- Une *Promise* est un objet
- Permet d'exécuter des opérations "asynchrones"
- Complétion/Rejet sont toujours exécutés en fin de boucle événementielle
- Permet de rendre le code plus lisible et plus réactif

20

```
const promise = new Promise(
  function(resolve, reject) {
    // Do Something
    if (cond) {
      resolve(result)
    } else {
      reject(result)
    }
  }
);
```

resolve => représente la bonne complétion d'une promise

reject => représente le rejet d'une promise

Les notions de complétion et rejet sont définies par le développeur

21

Le chaînage et la composition

Chaînage

`.then(resolve, reject)`

`.catch(reject) <=> .then(undefined, reject)`

Composition

`Promise.all([prom1, prom2, prom3])`

Attend la fin de toutes les complétions

S'arrête au premier rejet

`Promise.race([prom1, prom2, prom3])`

S'arrête à la fin de la première complétion

S'arrête au premier rejet

22

Nouvelle écriture

async

Permet d'exécuter du code séquentielle avec des Promises tout en restant dans un contexte de parallélisation

Valeur de retour => resolve

Exception => reject

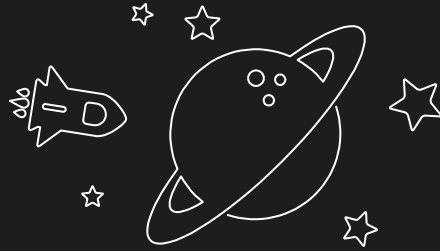
await

Attend la fin d'exécution d'une Promise

resolve => Valeur de retour

reject => Lève une exception

Attention!! *await* n'est disponible que dans une fonction *async*



Exercise 6

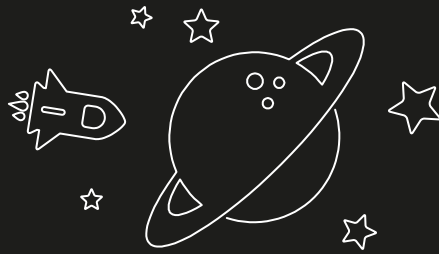
Je promets

23

`./exercice-6/promise.js`

Simulation d'appels serveurs

- Créer une Promise `getStudents` qui récupère une liste d'étudiants entre 1 et 2 secondes
EX: [{ name: "Dupont", cours: [1, 3, 5] }, { name: "Lea", cours: [2, 4] }, { name: "Charles", cours: [1] }]
- Créer une Promise `getCourses` qui récupère une liste de cours entre 2 et 4 secondes
EX: [{ id: 1, name: "JS" }, { id: 2, name: "PHP" }, { id: 3, name: "C#" }, { id: 4, name: "F#" }, { id: 5, name: "CSS" }]
- Créer une Promise qui mappe à l'ensemble des étudiants les cours associés entre 1 et 4 secondes
EX: [{ name: "Lea", cours: [{ id: 2, name: "PHP" }, { id: 4, name: "F#" }] }, ...]
- Créer une Promise qui contrôle le temps d'accès global
 - Celle-ci doit rejeter si le temps max dépasse 7 secondes
- Afficher la fonction et le temps estimé pour chaque Promise
EX: "getStudents:2"
- Afficher "Merge OK" si tout s'est bien passé sinon "Timeout"



WEB APIS

25

26

DOM API

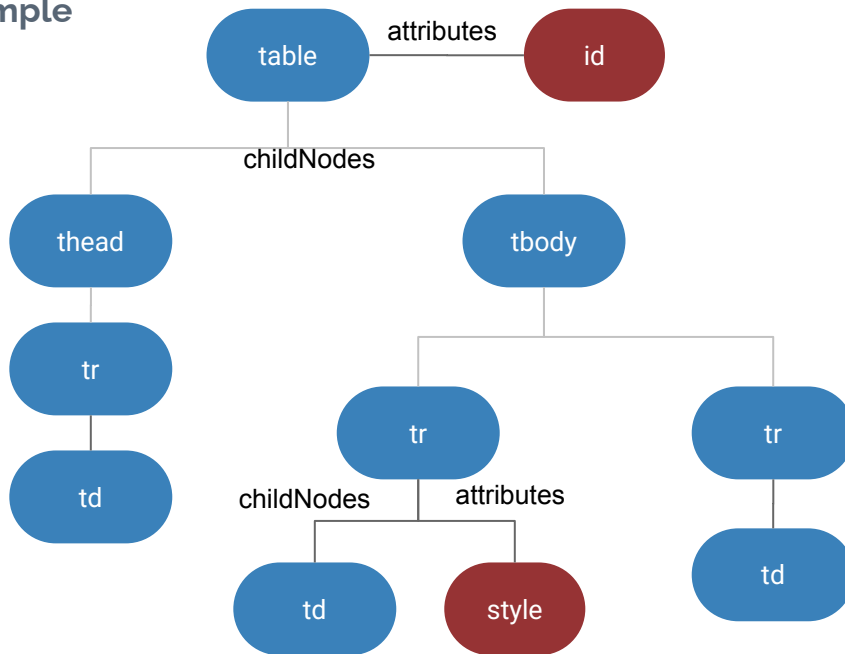
- Représentation sous forme d'AST d'un document HTML ou XML
- Deux types de noeuds
 - Element : représente un tag dans le code source
 - Attribute : représente un attribut d'un tag
- Chaque noeud du DOM est un Object contenant ses propres méthodes/propriétés selon son type
- L'API DOM est standardisée par la W3C
 - => garanti une même base fonctionnelle quelque soit le langage


```

<table id="table1">
  <thead>
    <tr>
      <td></td>
    </tr>
  </thead>
  <tbody>
    <tr style="border: 1px solid black">
      <td></td>
    </tr>
    <tr>
      <td></td>
    </tr>
  </tbody>
</table>

```

Exemple



Manipulation du DOM

Les accesseurs

Éléments globaux

window : fenêtre/onglet du DOM

document : ensemble du DOM

Sélecteur DOM

document.getElementById(monId)

=> élément avec ID monId

{element}.getElementsByClassName(maClass)

=> sous-éléments avec Class maClasse

{element}.getElementsByTagName(monTag)

=> sous-éléments avec Tag monTag

Sélecteur CSS

{element}.querySelector(cssSelector)

=> 1er sous-élément correspondant

{element}.querySelectorAll(cssSelector)

=> tous les sous-éléments correspondants

Les modifieurs

Création

document.createElement("tag name")

=> crée un noeud de type "tag name"

document.createTextNode("mon texte")

=> crée un noeud de type texte ayant pour valeur "mon texte"

Insertion

{elementParent}.appendChild(newElement)

=> ajoute un sous-élément à la fin

{elementParent}.insertBefore(newElem, refElem)

=> ajoute un sous-élément avant un sous-élément existant

Suppression

{elementParent}.removeChild(ElemToDelete)

=> supprime un sous-élément

Manipulation d'un DOMElement

{element}.parentNode

=> accède à l'élément parent

{element}.childNodes

=> accède aux sous-éléments

{element}.attributes

=> accède à tous les attributs

{element}.getAttribute("monAttr")

=> accède à l'attribut "monAttr"

{element}.className

=> accède à toutes les classes sous forme de String

{element}.classList

=> accède à toutes les classes sous forme de List

{element}.style

=> accède à l'object style de l'élément
=> facilite sa manipulation

Gestion des événements

<div onmouseover="myFunc();" />

=> exécute une fonction au hover

{element}.onclick = func

=> exécute une fonction au click

{element}.addEventListener("monEvent", func)

=> exécute func lors de l'événement "monEvent"

{element}.removeEventListener("monEvent", func)

=> supprimer func lors de l'événement "monEvent"

{element}.dispatchEvent(event)

=> diffuse l'événement event à l'élément

Création

const event = new Event("customName");

=> crée un event de type customName

**const event = new CustomEvent("customName",
{detail: customData});**

=> crée un event de type customName avec des données complémentaires

Event modifieurs

event.preventDefault()

=> annule l'effet d'un event

=> ex: annuler un click, une saisie clavier

event.stopPropagation()

=> annule la propagation d'un event vers l'élément target