Labo 4 –Javascript orienté objet (2)

Exercice 1 : souffler les bougies

Le but de cet exercice est de modéliser une liste d'amis dont on retient le nom, le prénom, le sexe et la date de naissance, de la manipuler et de l'afficher sous diverses formes (1) en créant des objets Javascript et (2) en se servant des objets prédéfinis dans Javascript.

Vous trouverez plus d'informations sur les objets prédéfinis en Javascript et sur les méthodes qui sont disponibles à l'adresse suivante (une version française est également disponible en ligne).

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

Étape 1

Le projet sera organisé en deux fichiers :

- un document appelé `amis.html` qui correspondra à la page principale et
- un fichier `amis.js` qui contiendra une bonne partie du code Javascript (à savoir tout ce qui concerne les objets « Amis » ainsi que les diverses fonctions utilitaires nécessaires).

Contrairement aux laboratoires précédents, cet énoncé vous laisse relativement libres. Libres entre autres d'organiser votre code comme vous le voulez : à vous de déterminer quelles fonctions (et méthodes) annexes définir et comment structurer vos scripts. Pensez Clean Code, pensez « code facile à lire », pensez « code bien structuré », pensez « code facile à modifier ».

Dans un premier temps, concentrez-vous sur l'écriture d'une fonction constructrice Ami. Chaque objet « ami » comportera

- un nom,
- un prénom,
- un sexe stocké sous le format que vous trouvez le plus adéquat mais reçu sous la forme d'une chaîne de caractère (« f » et « F » signifiant féminin, « m » et « M » signifiant masculin), et
- une date de naissance stockée sous la forme d'un tableau associatif mais reçue par la fonction constructrice au format AAAAMMJJ.

Ainsi, si la variable adelaide correspond une certaine Adélaïde Berie, née le 30/10/1973, on aura

```
adelaide.nom → "Berie"
adelaide.prenom → "Adélaïde"
adelaide.sexe → (ce qui correspond à « féminin »)
adelaide.dateNaiss.jour → 30
adelaide.dateNaiss.mois → 10
adelaide.dateNaiss.annee → 1973
```

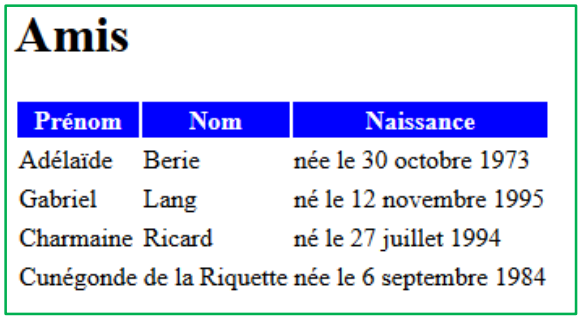
La fonction constructrice devra être compatible avec le script suivant, censé remplir un tableau appelé `amis`.

```
amis = [];
amis[0] = new Ami ("Adélaïde", "Berie", "F", 19731030);
amis[1] = new Ami ("Gabriel", "Lang", "m", 19951112);
amis[2] = new Ami ("Charmaine", "Ricard", "M", 19940727);
amis[3] = new Ami ("Cunégonde", "de la Riquette", "F", 19840906);
```

Ajoutez d'ailleurs les lignes de script reprises ci-dessus dans un script Javascript placé directement dans la partie « en-tête » du document HTML. Ce script initialisera la liste des amis à afficher (et sera complété au fil des étapes suivantes).

Étape 2

Dans un premier temps, ajoutez un titre (balise h1) « Amis » suivi d'un tableau dans le document HTML. Le tableau à produire devra correspondre à l'exemple donné ci-dessous.



Amis

Prénom	Nom	Naissance
Adélaïde	Berie	née le 30 octobre 1973
Gabriel	Lang	né le 12 novembre 1995
Charmaine	Ricard	né le 27 juillet 1994
Cunégonde	de la Riquette	née le 6 septembre 1984

Pour produire le tableau, vous pouvez utiliser un script inséré dans la partie « body » du document HTML et l'instruction `document.write`.

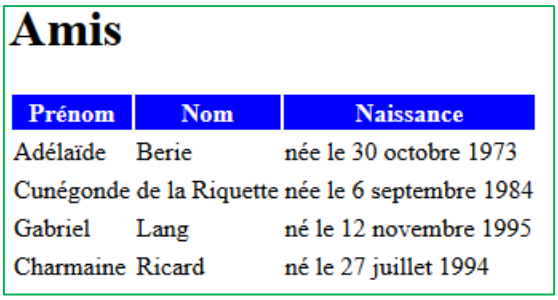
Quelques conseils/notes/observations :

- N'agglutinez pas tout votre code dans le script du document HTML. Pensez aux méthodes qui pourraient être définies pour chaque objet Ami et permettre de vous simplifier le travail.
- Parmi les fonctions constructrices prédéfinies en Java, il y a `Math`, qui contient diverses méthodes permettant de manipuler les nombres ; vous pourriez en avoir besoin.
- N'oubliez pas de définir les méthodes partagées par tous les objets « Amis » dans leur prototype commun ! Ne les répétez pas dans chaque objet !

Étape 3

Pour l'instant, les amis sont affichés dans l'ordre où ils sont encodés. Modifiez le script définissant le tableau `amis` en ajoutant une seule instruction à la fin du script, de manière à trier ce tableau selon les noms et prénoms des personnes. Pour ce faire, utilisez la méthode `Array.prototype.sort` (consultez sa documentation en ligne).

Vous devriez obtenir le résultat suivant.

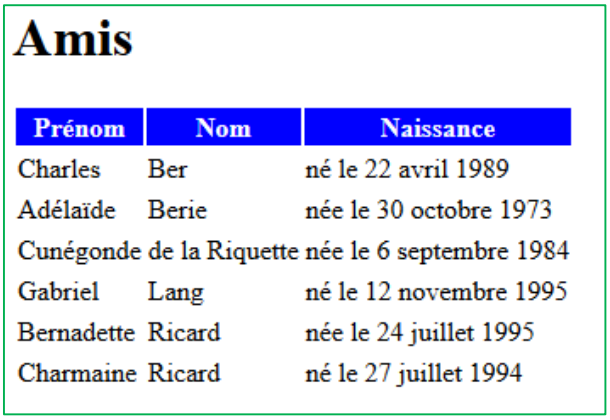


Amis

Prénom	Nom	Naissance
Adélaïde	Berie	née le 30 octobre 1973
Cunégonde	de la Riquette	née le 6 septembre 1984
Gabriel	Lang	né le 12 novembre 1995
Charmaine	Ricard	né le 27 juillet 1994

Ajoutez les deux amis suivants et vérifiez que votre script produit bien le résultat attendu.

```
amis[4] = new Ami ("Bernadette", "Ricard", "f", 19950724);
amis[5] = new Ami ("Charles", "Ber", "m", 19890422);
```



Amis

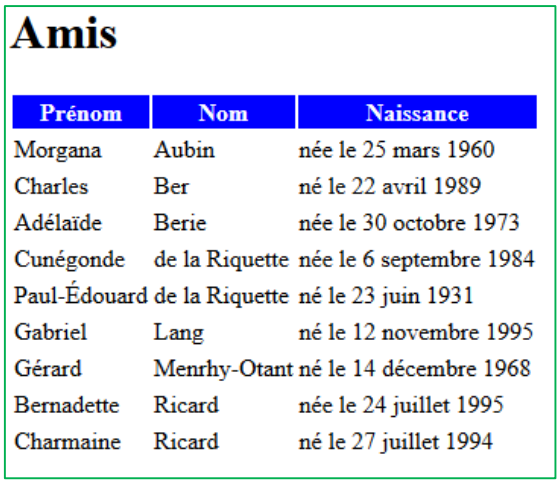
Prénom	Nom	Naissance
Charles	Ber	né le 22 avril 1989
Adélaïde	Berie	née le 30 octobre 1973
Cunégonde	de la Riquette	née le 6 septembre 1984
Gabriel	Lang	né le 12 novembre 1995
Bernadette	Ricard	née le 24 juillet 1995
Charmaine	Ricard	né le 27 juillet 1994

Étape 4

Afin de pouvoir récupérer certains amis qui avaient été encodés dans une ancienne liste, modifiez la fonction constructrice `Ami` afin qu'elle puisse être également appelée avec une unique chaîne de caractères reprenant le prénom et le nom (par exemple « Morgana Aubin »). Dans cette chaîne de caractères, on supposera que le prénom s'arrête juste avant le premier espace et que le nom commence dès après cet espace.

Pour vérifier votre code, ajoutez les amis suivants.

```
amis[6] = new Ami ("Morgana Aubin", "F", 19600325);
amis[7] = new Ami ("Paul-Édouard de la Riquette", "m", 19310623);
amis[8] = new Ami ("Gérard Menrhy-Otant", "M", 19681214);
```



Amis

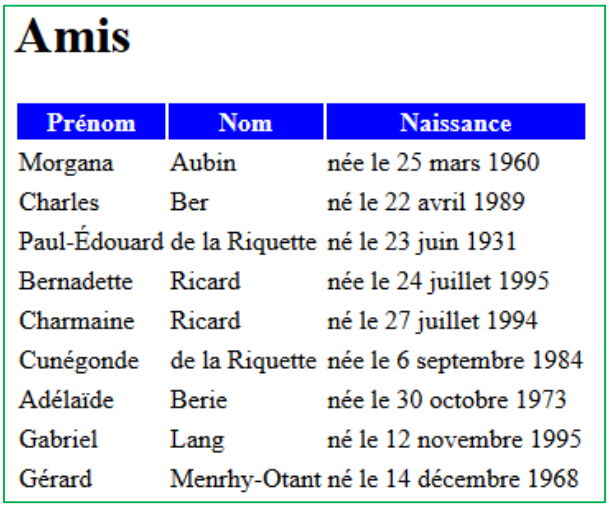
Prénom	Nom	Naissance
Morgana	Aubin	née le 25 mars 1960
Charles	Ber	né le 22 avril 1989
Adélaïde	Berie	née le 30 octobre 1973
Cunégonde	de la Riquette	née le 6 septembre 1984
Paul-Édouard	de la Riquette	né le 23 juin 1931
Gabriel	Lang	né le 12 novembre 1995
Gérard	Menrhy-Otant	né le 14 décembre 1968
Bernadette	Ricard	née le 24 juillet 1995
Charmaine	Ricard	né le 27 juillet 1994

Étape 5

Modifiez le tableau pour que les amis soient affichés non plus par ordre alphabétique sur leurs nom et prénom mais plutôt par ordre chronologique sur leur date d'anniversaire (on négligera l'année).

Pour ce faire, commencez par définir une fonction `compareDates(d1,d2)` qui compare deux dates (c'est-à-dire deux tableaux associatifs contenant des propriétés jour, mois et annee) et renvoie -1 si `d1` vient avant `d2` dans l'année, 0 s'il s'agit de dates identiques et 1 si `d1` vient après `d2` dans l'année.

Utilisez-la ensuite pour créer une fonction `compareAmiChrono(ami1,ami2)` qui compare deux amis selon leur date d'anniversaire.



Amis

Prénom	Nom	Naissance
Morgana	Aubin	née le 25 mars 1960
Charles	Ber	né le 22 avril 1989
Paul-Édouard	de la Riquette	né le 23 juin 1931
Bernadette	Ricard	née le 24 juillet 1995
Charmaine	Ricard	né le 27 juillet 1994
Cunégonde	de la Riquette	née le 6 septembre 1984
Adélaïde	Berie	née le 30 octobre 1973
Gabriel	Lang	né le 12 novembre 1995
Gérard	Menrhy-Otant	né le 14 décembre 1968

Étape 6

Ajoutez trois colonnes aux tableaux. Ces colonnes seront intitulées « 2016 », « 2017 » et « 2018 » et indiqueront, pour chacune de ces trois années,

- l'âge que l'ami en question atteindra au moment de son anniversaire ;
- le jour qui correspond à son anniversaire (lundi, mardi, mercredi, jeudi, vendredi, samedi ou dimanche) (servez-vous des méthodes du constructeur prédéfini `Date`).

Amis

Prénom	Nom	Naissance	2016	2017	2018
Morgana	Aubin	née le 25 mars 1960	56 ans (vendredi)	57 ans (samedi)	58 ans (dimanche)
Charles	Ber	né le 22 avril 1989	27 ans (vendredi)	28 ans (samedi)	29 ans (dimanche)
Paul-Édouard	de la Riquette	né le 23 juin 1931	85 ans (jeudi)	86 ans (vendredi)	87 ans (samedi)
Bernadette	Ricard	née le 24 juillet 1995	21 ans (dimanche)	22 ans (lundi)	23 ans (mardi)
Charmaine	Ricard	né le 27 juillet 1994	22 ans (mercredi)	23 ans (jeudi)	24 ans (vendredi)
Cunégonde	de la Riquette	née le 6 septembre 1984	32 ans (mardi)	33 ans (mercredi)	34 ans (jeudi)
Adélaïde	Berie	née le 30 octobre 1973	43 ans (dimanche)	44 ans (lundi)	45 ans (mardi)
Gabriel	Lang	né le 12 novembre 1995	21 ans (samedi)	22 ans (dimanche)	23 ans (lundi)
Gérard	Menrhy-Otant	né le 14 décembre 1968	48 ans (mercredi)	49 ans (jeudi)	50 ans (vendredi)

Exercice 2 : arrays et strings en pagaille

Cet exercice présente quelques-unes des méthodes prédéfinies sur les tableaux et sur les chaînes de caractères. Comme précédemment, référez-vous à la documentation en ligne sur mozilla.org pour obtenir plus de détails quant aux spécifications de ces méthodes.

Étape 1

Examinez la définition de fonction suivante et déterminez ce qu'elle fait.

```
function distance (tab,x) {
  return tab.lastIndexOf(x) - tab.indexOf(x);
}

distance([0,1,2,3,4,3,2,1,0], 4)
distance([0,1,2,3,4,3,2,1,0], 2)
```

Étape 2

Faites de même avec la définition suivante.

```
function motsLexico (s) {
  return s.split(" ").sort();
}

motsLexico("le chien noir aboie toute la nuit");
motsLexico("toujours penser clean code");
```

Étape 3

Et la fonction décrite ci-dessous ?

```
function memeSigne (tab) {
  return tab.every(function (x) { return x > 0; })
    || tab.every(function (x) { return x < 0; });
}

memeSigne([1,2,3,4,5,6]);
memeSigne([-2,-5,-7,-34,-1]);
memeSigne([1,-2,3,-4,5,-6]);
```

Étape 4

Et cette fonction ?

```
function carreDesPairs (tab) {
  return tab.filter(function (x) { return x % 2 == 0; })
    .map(function (x) { return x * x; });
}

carreDesPairs([1,2,3,4,5]);
carreDesPairs([5,6,7]);
```

Étape 5

Une petite dernière ?

```
function plusLong (s) {
  function plusLongDeDeux (s1,s2) {
    return s1.length > s2.length ? s1 : s2;
  }
  return s.split(" ").reduce(plusLongDeDeux);
}

plusLong("le chien noir aboie toutes les nuits");
plusLong("toujours penser clean code");
```

Étape 6

À vous de jouer… En utilisant les méthodes présentées ci-dessus ou d'autres méthodes prédéfinies, définissez des fonctions répondant aux spécifications suivantes.

1. Une fonction `tousNamur(tab)` qui reçoit un tableau de codes postaux et qui indique s'ils correspondent bien tous à la province de Namur (c'est-à-dire s'ils sont bien tous compris entre 5000 et 5999 au sens large).
   Utilisez « every ».

```
tousNamur([5000,5500,5600,5200]) → true
tousNamur([1000,5600,4000]) → false
```

2. Une fonction `motsAvecInitiale(s,lettre)` qui reçoit une chaîne de caractères comportant plusieurs mots séparés par des virgules et une lettre et qui renvoie un tableau reprenant tous les mots de la chaîne qui commencent par la lettre en question.
   Utilisez « filter » et « startsWith ».

```
motsAvecInitiale("le,chien,est,un,caniche","c") → ["chien","caniche"]
motsAvecInitiale("trois,plus,quatre,égal,sept","h") → []
```

3. Une fonction `pascalCase(mots)` qui reçoit un tableau de mots et renvoie une unique chaîne de caractères obtenue en concaténant ces mots selon les règles du PascalCase (chaque mot commence par une majuscule et le reste du mot est en minuscules).
   Utilisez « map », « substring », « toLowerCase » et « toUpperCase ».

```
pascalCase(["age","CAPITAINE","naVIre"]) → "AgeCapitaineNavire"
pascalCase(["nb","jeux","PROMOTION","ajd"]) → "NbJeuxPromotionAjd"
```