Javascript

Objectifs

1. **Créer** des objets littéraux
2. **Lire et écrire** des propriétés sur les objets

C'est quoi un objet ?

Les objets en Javascript sont comme les objets nous entourent. Ce sont des sortes de **conteneurs** qui contiennent des propriétés qui caractérisent l'objet.

C'est quoi un objet ?

Les objets en Javascript sont comme les objets nous entourent. Ce sont des sortes de **conteneurs** qui contiennent des propriétés qui caractérisent l'objet.

Par exemple, prenons une pomme : une pomme a une couleur verte (*"#00FF00"*), un diamètre de 10cm, etc...

C'est quoi un objet ?

Les objets en Javascript sont comme les objets nous entourent. Ce sont des sortes de **conteneurs** qui contiennent des propriétés qui caractérisent l'objet.

Par exemple, prenons une pomme : une pomme a une couleur verte (*"#00FF00"*), un diamètre de 10cm, etc...

Essayons de décrire une pomme en JavaScript !

C'est quoi un objet ?

Les objets sont créés à l'aide d'accolades (🇬🇧 *curly braces*) : **{}**.

C'est quoi un objet ?

Les objets sont créés à l'aide d'accolades (🇬🇧 *curly braces*) : **{}**.

À l'intérieur de l'objet, on peut créer des **clés** *(ex : color)* et associer à ces clés des **valeurs** en séparant clé et valeur par (**:**)

C'est quoi un objet ?

Les objets sont créés à l'aide d'accolades (🇬🇧 *curly braces*) : **{}**.

À l'intérieur de l'objet, on peut créer des **clés** *(ex : color)* et associer à ces clés des **valeurs** en séparant clé et valeur par (**:**)

Chaque paire clé/valeur doit être **séparée par une virgule.**

C'est quoi un objet ?

Les objets sont créés à l'aide d'accolades (🇬🇧 *curly braces*) : **{}**.

À l'intérieur de l'objet, on peut créer des **clés** *(ex : color)* et associer à ces clés des **valeurs** en séparant clé et valeur par (**:**)

Chaque paire clé/valeur doit être **séparée par une virgule.**

La valeur peut être de n'importe quel type.

C'est quoi un objet ?

```
const apple = {
    color: "#00FF00",
    diameter: 10,
    isEaten: false,
    vitamins: ["A","B1", "B2", "B6", "C"],
    variety: {code: 576, name: "Granny Smith"},
    gather: function(){
        return "Here's one apple!";
    }
}
```

Par exemple, comme une pomme peut être source de plusieurs vitamines, on a utilisé **un tableau** pour les représenter.

JS

C'est quoi un objet ?

```
const apple = {
    color: "#00FF00",
    diameter: 10,
    isEaten: false,
    vitamins: ["A","B1", "B2", "B6", "C"],
    variety: {code: 576, name: "Granny Smith"},
    gather: function(){
        return "Here's one apple!";
    }
}
```

Par exemple, comme une pomme peut être source de plusieurs vitamines, on a utilisé **un tableau** pour les représenter.

Comme on peut le voir, un objet peut être imbriqué dans autre objet ! Ici, notre pomme appartient à une variété représentée par un objet.

JS

C'est quoi un objet ?

```
const apple = {
    color: "#00FF00",
    diameter: 10,
    isEaten: false,
    vitamins: ["A","B1", "B2", "B6", "C"],
    variety: {code: 576, name: "Granny Smith"},
    gather: function(){
        return "Here's one apple!";
    }
}
```

Par exemple, comme une pomme peut être source de plusieurs vitamines, on a utilisé **un tableau** pour les représenter.

Comme on peut le voir, un objet peut être imbriqué dans autre objet ! Ici, notre pomme appartient à une variété représentée par un objet.

Nous avons aussi donné à la pomme une **fonction (appelée méthode)** pour cueillir le fruit !

Accéder à la propriété d’un objet

Nous pouvons accéder à une propriété de l'objet en utilisant **.** ou **[ ]**.

Par exemple, si nous voulons accéder à la propriété **color** de **l'objet apple**, il suffit d'écrire **apple.color** ou **apple['color']**.

Accéder à la propriété d'un objet

Nous pouvons accéder à une propriété de l'objet en utilisant **.** ou **[ ]**.

Par exemple, si nous voulons accéder à la propriété **color** de **l'objet apple**, il suffit d'écrire **apple.color** ou **apple['color']**.

```
apple.color;
// "#00FF00"
apple['color'];
// "#00FF00"
```

Accéder à la propriété d'un objet

Nous pouvons accéder à une propriété de l'objet en utilisant **.** ou **[ ]**.

Par exemple, si nous voulons accéder à la propriété **color** de **l'objet apple**, il suffit d'écrire **apple.color** ou **apple['color']**.

```
apple.color;
// "#00FF00"
apple['color'];
// "#00FF00"
```

```
const apple = {
    color: "#00FF00",
    diameter: 10,
    isEaten: false,
    vitamins: ["A","B1", "B2", "B6", "C"],
    variety: {code: 576, name: "Granny Smith"},
    gather: function(){
        return "Here's one apple!";
    }
}
console.log(apple.color);
console.log(apple['diameter']);
console.log(apple.vitamins[2]);
console.log(apple.variety.name);
console.log(apple.gather());
```

JS

Accéder à la propriété d'un objet

Nous pouvons accéder à une propriété de l'objet en utilisant **.** ou **[ ]**.

Par exemple, si nous voulons accéder à la propriété **color** de **l'objet apple**, il suffit d'écrire **apple.color** ou **apple['color']**.

```
apple.color;
// "#00FF00"
apple['color'];
// "#00FF00"
```

```
const apple = {
    color: "#00FF00",
    diameter: 10,
    isEaten: false,
    vitamins: ["A","B1", "B2", "B6", "C"],
    variety: {code: 576, name: "Granny Smith"},
    gather: function(){
        return "Here's one apple!";
    }
}
console.log(apple.color);
console.log(apple['diameter']);
console.log(apple.vitamins[2]);
console.log(apple.variety.name);
console.log(apple.gather());
```

La plupart du temps, on utilisera un point pour accéder à une propriété (apple.color), mais les crochets peuvent aussi être très utiles par exemple dans le cas où l'on souhaite utiliser une variable pour accéder à une valeur.

JS

Accéder à la propriété d'un objet

Nous pouvons accéder à une propriété de l'objet en utilisant **.** ou **[ ]**.

Par exemple, si nous voulons accéder à la propriété **color** de **l'objet apple**, il suffit d'écrire **apple.color** ou **apple['color']**.

```
apple.color;
// "#00FF00"
apple['color'];
// "#00FF00"
```

```
const apple = {
    color: "#00FF00",
    diameter: 10,
    isEaten: false,
    vitamins: ["A","B1", "B2", "B6", "C"],
    variety: {code: 576, name: "Granny Smith"},
    gather: function(){
        return "Here's one apple!";
    }
}
console.log(apple.color);
console.log(apple['diameter']);
console.log(apple.vitamins[2]);
console.log(apple.variety.name);
console.log(apple.gather());
```

La plupart du temps, on utilisera un point pour accéder à une propriété (apple.color), mais les crochets peuvent aussi être très utiles par exemple dans le cas où l'on souhaite utiliser une variable pour accéder à une valeur.

```
const selectedProperty = prompt('Tape la propriété que tu veux afficher') ;
console.log(apple[selectedProperty]) ;
```

Ajouter ou modifier la propriété d'un objet

Pour ajouter une propriété à un objet, il suffit de la définir comme ceci :

Ajouter ou modifier la propriété d'un objet

Pour ajouter une propriété à un objet, il suffit de la définir comme ceci :

```
apple.growsOn = "Tree" ;
```

Ajouter ou modifier la propriété d'un objet

Pour ajouter une propriété à un objet, il suffit de la définir comme ceci :

```
apple.growsOn = "Tree" ;
```

De même, pour donner une autre valeur à la propriété d'un objet, il suffit d'utiliser le symbole égal **=.**

Ajouter ou modifier la propriété d'un objet

Pour ajouter une propriété à un objet, il suffit de la définir comme ceci :

```
apple.growsOn = "Tree" ;
```

De même, pour donner une autre valeur à la propriété d'un objet, il suffit d'utiliser le symbole égal =.

```
apple.color = "Red" ;
```

Supprimer une propriété

On peut utiliser **delete** pour supprimer une propriété.

Supprimer une propriété

On peut utiliser **delete** pour supprimer une propriété.

```
delete apple.name;
```

JS

Supprimer une propriété

On peut utiliser **delete** pour supprimer une propriété.

```
delete apple.name;
```

```
const apple = {
    color: "#00FF00",
    diameter: 10,
    isEaten: false
}

delete apple.color;
console.log(apple);
```

JS

Supprimer une propriété

On peut utiliser **delete** pour supprimer une propriété.

```
delete apple.name;
```

```
const apple = {
    color: "#00FF00",
    diameter: 10,
    isEaten: false
}

delete apple.color;
console.log(apple);
```

```
▼ Object i
    diameter: 10
    isEaten: false
```

Combiner des tableaux et des objets

Et si nous pouvions **combiner des tableaux avec des objets ?**

Combiner des tableaux et des objets

Et si nous pouvions **combiner des tableaux avec des objets ?**

Par exemple, nous avons beaucoup de fruits différents, pas seulement des pommes.

Combiner des tableaux et des objets

Et si nous pouvions **combiner des tableaux avec des objets ?**

Par exemple, nous avons beaucoup de fruits différents, pas seulement des pommes.

Et si nous voulions décrire **tous les fruits ?**

Combiner des tableaux et des objets

Et si nous pouvions **combiner des tableaux avec des objets ?**

Par exemple, nous avons beaucoup de fruits différents, pas seulement des pommes.

Et si nous voulions décrire **tous les fruits ?**

Et bien, nous pourrions mettre nos objets dans un tableau !

JS

Combiner des tableaux et des objets

Et si nous pouvions **combiner des tableaux avec des objets ?**

Par exemple, nous avons beaucoup de fruits différents, pas seulement des pommes.

Et si nous voulions décrire **tous les fruits ?**

Et bien, nous pourrions mettre nos objets dans un tableau !

```
const fruits = [
    {name: "apple", color: "green"},
    {name: "Pineapple", color:"yellow"},
    {name: "Orange", color:"orange"},
    {name: "Cherry", color:"red"},
]

console.log(fruits[0].color);
```

Combiner des tableaux et des objets

Et si nous pouvions **combiner des tableaux avec des objets ?**

Par exemple, nous avons beaucoup de fruits différents, pas seulement des pommes.

Et si nous voulions décrire **tous les fruits ?**

Et bien, nous pourrions mettre nos objets dans un tableau !

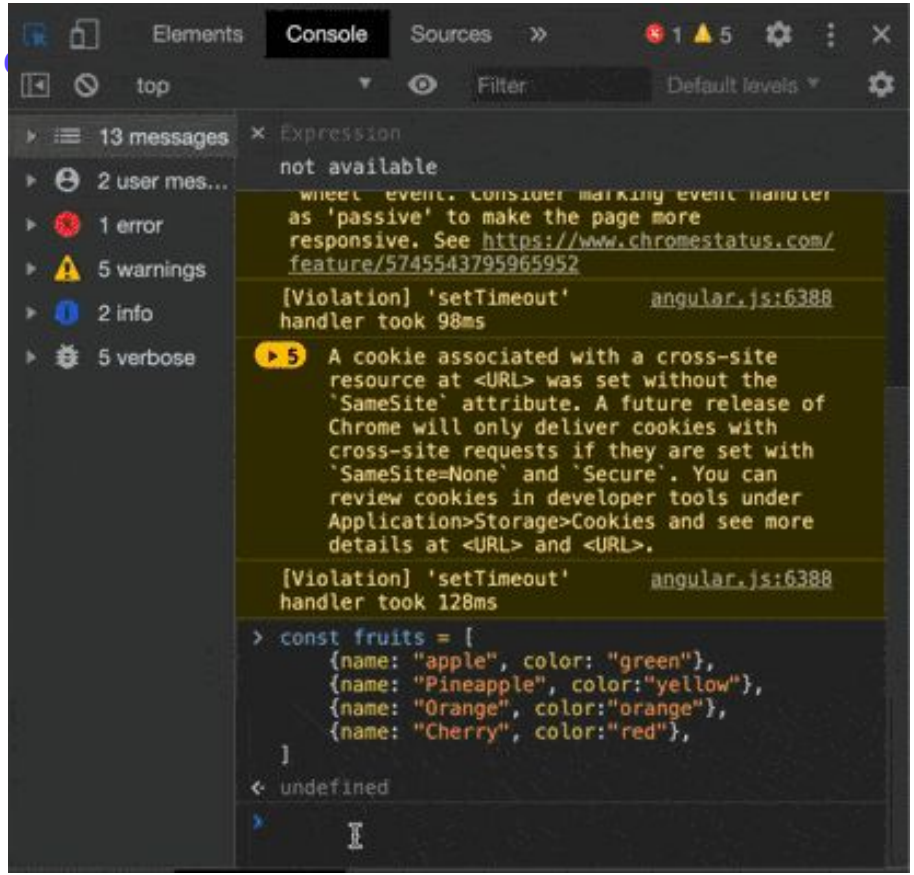
```
const fruits = [
    {name: "apple", color: "green"},
    {name: "Pineapple", color:"yellow"},
    {name: "Orange", color:"orange"},
    {name: "Cherry", color:"red"},
]

console.log(fruits[0].color);
```

```
green
```

Objets dans la console chrome

Si on affiche ce tableau d'objets dans la console du navigateur, on peut naviguer à l’intérieur en cliquant sur les petites flèches à gauche.

JS


Elements
Console
Sources
1 5
top
Filter
Default levels
13 messages
2 user mes...
1 error
5 warnings
2 info
5 verbose
Expression
not available
as 'passive' to make the page more
responsive. See https://www.chromestatus.com/
feature/5745543795965952
[Violation] 'setTimeout'
handler took 98ms
angular.js:6388
5
A cookie associated with a cross-site
resource at <URL> was set without the
`SameSite` attribute. A future release of
Chrome will only deliver cookies with
cross-site requests if they are set with
`SameSite=None` and `Secure`. You can
review cookies in developer tools under
Application>Storage>Cookies and see more
details at <URL> and <URL>.
[Violation] 'setTimeout'
handler took 128ms
angular.js:6388
const fruits = [
{name: "apple", color: "green"},
{name: "Pineapple", color:"yellow"},
{name: "Orange", color:"orange"},
{name: "Cherry", color:"red"},
]
undefined

What is “this” ?

Le mot-clé **this** se réfère à l’**objet JavaScript représentant le contexte dans lequel le code courant est exécuté.**

Plus précisément, **this** désigne ce qui précède le **.** lors de l'appel d'une méthode.

What is “this” ?

Le mot-clé **this** se réfère à l’**objet JavaScript représentant le contexte dans lequel le code courant est exécuté.**

Plus précisément, **this** désigne ce qui précède le **.** lors de l'appel d'une méthode.

```
const person1 = {
    name : "Bob",
    age : 30,
    sayHello : function(){
        console.log(`Hi, I'm ${this.name}`) ;
    }
}

person1.sayHello() ;
// this se réfère à l'objet auquel il appartient.
// This.name signifie "le nom de person1"
```

What is “this” ?

Le mot-clé **this** se réfère à l’**objet JavaScript représentant le contexte dans lequel le code courant est exécuté.**

Plus précisément, **this** désigne ce qui précède le **.** lors de l'appel d'une méthode.

```
const person1 = {
    name : "Bob",
    age : 30,
    sayHello : function(){
        console.log(`Hi, I'm ${this.name}`) ;
    }
}

person1.sayHello() ;
// this se réfère à l'objet auquel il appartient.
// This.name signifie "le nom de person1"
```

Pousser la notion d’objets :

>> Ressource 1

>> Ressource 2

Pratiquons !