Fonction Anonyme
Fonction Nommée
Fonction Fléchée

Fonction Nommée

Une fonction nommée est une fonction qui a un nom spécifique, comme ceci:

Syntaxe:

```
function nomDeMaFonction(param1, param2) {
  // instructions de ma fonction
}
```

Exemple:

```
function sommeEntier(x, y) {
    return x+y
  }
```

```
sommeEntier(2, 3);
```

Fonction Anonyme

Une fonction anonyme est une fonction qui n'a pas de nom spécifique. Elle est définie en utilisant une expression de fonction, comme ceci:

Syntaxe:

```
const maFonction = function(param1, param2, param3) {
    // instructions de ma fonction
  };
```

Exemple:

```
  const produit = function(x, y) {
      return x*y ;
    };
```

```
console.log(produit(2, 5));
```

Fonction Fléchée:

une fonction fléchée est une façon plus récente d'écrire une fonction anonyme. Elle est définie en utilisant une syntaxe de flèche, comme ceci:

Syntaxe:

```
const maFonctionFlechee = (param1, param2) => {
    // instructions de ma fonction
  };
```

Exemple:

```
 const produitEntier = (x, y) => {
     return x*y ;
   };
```

```
produitEntier(2, 5); // output: 10
```

Mot-clé "this"

En JavaScript, le mot-clé "this" est utilisé pour faire référence à l'objet courant dans lequel la fonction est appelée. Il est très utile pour accéder aux propriétés et aux méthodes d'un objet.

```
// Fonction nommée:

function somme() {
    return this.element1 + this.element2;
  }

  const obj = {
    element1: 2,
    element2: 3,
    calculerSomme: somme
  };

console.log(obj.calculerSomme()); // output: 5
```

Dans cette fonction, "this" est utilisé pour faire référence aux propriétés "element1" et "element2" de l'objet "obj". La fonction "somme" est définie à l'extérieur de l'objet et est ensuite assignée comme méthode de l'objet "obj". Lorsque la méthode "calculerSomme" est appelée sur l'objet "obj", la fonction "somme" est exécutée avec "this" qui fait référence aux propriétés de l'objet.

```
// Fonction Anonyme:

const objVar = {
    element1: 2,
    element2: 3,
    calculerSomme: function() {
      return this.element1 + this.element2;
    }
 };

 console.log(objVar.calculerSomme()); // output: 5
```

Dans cette fonction, "this" est également utilisé pour faire référence aux propriétés "element1" et "element2" de l'objet "objVar". La méthode "calculerSomme" est définie directement dans l'objet "objVar" en tant que fonction anonyme.

Note:

Si la fonction est appelée en tant que méthode d'un objet, "this" fait référence à l'objet. Si la fonction est appelée directement, "this" fait référence à la portée globale (par exemple, l'objet "window" dans un navigateur).

```
// Fonction fléchée:

const varObj = {
    element1: 2,
    element2: 3,
    calculerSomme: () => {
      return this.element1 + this.element2;
    }
  };

  console.log(varObj.calculerSomme()); // output: NaN
```

Dans cette fonction, "this" fait référence à la portée parente de la fonction fléchée, qui est la portée globale. Par conséquent, "this.element1" et "this.element2" ne sont pas définis, ce qui entraîne une erreur de "NaN" (Not a Number) lors de l'exécution de la fonction.

Différence de Comportement:

La différence de comportement entre les fonctions fléchées et les fonctions normales (nommées ou anonymes) en JavaScript est la façon dont elles gèrent le contexte "this".

Dans les fonctions normales, "this" est défini en fonction de la façon dont la fonction est appelée. Si la fonction est appelée en tant que méthode d'un objet, "this" fait référence à l'objet. Si la fonction est appelée directement, "this" fait référence à la portée globale.

Dans les fonctions fléchées, "this" n'a pas son propre contexte et il hérite du contexte "this" de la portée parente (la fonction ou le bloc dans lequel il est défini). Cela signifie que lorsque vous utilisez "this" à l'intérieur d'une fonction fléchée, la valeur de "this" est déterminée par la portée qui contient la fonction fléchée, et non par la façon dont la fonction est appelée.