Call / Apply / Bind

لو عندي Function زي دي كدا

```
function sayHi(){
console.log("HI");
{
sayHi();
```

هنا معنديش اي مشكله هتطبع Hi عادي

```
 function sayHi(){
console.log("HI");
{
sayHi.call();
```

هتطبع Hi بردو طب اي الفرق

الاولي هو عاملها call بالفعل بس دي short hand مدام مش حطيت اي parameter

.

```
const charOne={
    name:"Zeyad",
    health:90,
    fullHealth(){
        return (this.health=100);
    }
};
```

لو جيت اشغل ال health قبل م اشغل ال Function دي هيطبعلي 90

```
console.log(charOne.health)
charOne.fullHealth();
console.log(charOne.health)
```

```
هنا هيطبع
90
100
```

```
const chartwo={
    name:"Ahmed",
    health:70,
};
```

```
طب لو انا عندي object تاني ومحتاج اني استدعي ميثود بس من ال character الي فوق ومش
عايز اكتبه تاني عشان اطبق مفهوم ال . DRY
هنا بقا يجي دور ال .Call
```

```
charOne.fullHealth.call(chartwo)
```

```
هنا زي م يكون انا عملت Override علي ال function دي من ال object الي فوق واستخدمته
تحت.
```

```
console.log(chartwo.health)
charOne.fullHealth.call(chartwo)
console.log(chartwo.health)
```

هنا هتطبيع
70
100
ولو في parameter في ال Function الي فوق هبعتها جنب اسم ال object الي انا عاوز يتطبق عليه ال Function

```
charOne.fullHealth.call(chartwo,num1,num2)
```

زي كدا.

Apply :
نفس الي بتعمله ال Call بالظبط ولكن بدل م ببعت ال Argument بتاعتي ك Variables لا هبعتها ك Array .

```
charOne.fullHealth.apply(chartwo,[50,70])
```

بس هو ده كل الاختلاف .

Bind : بتاخد ال Argument علي هيئه Array طب اي الفرق بينها وبين ال Call الفرق انها بت return new function يعني الناتج منها لازم احفظه علي هيئه متغير والمتغير ده بيكون Function ف بعمله call عشان يشتغل.

```
const func= charOne.fullHealth.bind(chartwo,50,70)
func();
```

زي كدا بالظبط.

let const are hisoted

خلينا نبدء ب var الاول لو عندي متغير ومطبوع بالشكل ده

```
var Hi="Zeyad"
console.log(Hi);
```

هيطبع Zeyad عادي جدا.
طب لو عملت كدا :

```
console.log(Hi);
var Hi="Zeyad"
```

هيطبع undefined طب ليه لان ال decleration بس بيعمله hisoting بس ال inetialization بيفضل في مكانه عادي زي كدا

```
var Hi;
console.log(Hi);
var Hi="Zeyad"
```

طب ال const لا طبعا الموضوع بيختلف هنا مينفعش اعملها declare اصلا الا لما اعملها inetialization في الميموري بتاعتي ولو حاولت استخدمها او اطبعها هيديني ايرور ف هما مش بيتعملها hisoting .

```
const Hi;
```

دي كدا لوحدها مدياني ايرور ومش هكمل الا لما اديها قيمه.

طب وال let : زيها زي ال var بالظيط ممكن اعرفها من غير م اديلها قيمه فبالتالي بيتعملها hisoting وقت ال runtime ولكن هتدي undefined ومش هتدي قيمه الا لما اعملها inetialize.

Shallow and Deep Copy

الموضوع هنا شبيه بال Object , Class كدا ان انا عندي حاجه وعاوز اخد نسخه منها وابدء اني اعدل عليها

طب انا هاخد النسخه دي ازاي اصلا.في ميثود عندي اسمها copywithin

```
var arr1=[1,2,3,4]
var arr2=[];
arr2=arr1.copyWithin()
console.log(arr2)
```

Out put: [1, 2, 3, 4]

جميل اوي لحد هنا طب لو احتاجت اني اعدل ع ال arr2 دي اي الي هيحصل

```
var arr1=[1,2,3,4]
var arr2=[];
arr2=arr1.copyWithin()
arr2.pop();
console.log(arr2)
```

Out Put : [1, 2, 3]

طب لما جيت اطبع arr1 لقيت ان التعديل الي عملته ع arr2 سمع في arr1 ازاي

```
console.log(arr2)
console.log(arr1)
```

Out put : [1,2,3]]

[1,2,3]

طب م دي مشكله انا عاوز اخد نسخه واعدل عليها مش اعدل ع ال النسخه الرئيسيه بتاعتي

طب م كدا انا مخدتش نسخه بقا دنا خدت الclass زي م هو لا هي كل الحكايه ان ال تعريف ال arr بس بيتخزن في ال stack طب والقيم الي عندي لا دي بتتخزن معايا فال heap وبفضل باصص عليها ب حاجه اسمها reference طب وبعدين كل الفكره اني لما اخدت كوبي هنا انا خدت حاجه اسمها deep copy وهي اني بقيت بال arr2 دي ببص ع نفس ال reference بتاع ال arr1 ف اي تعديل هيحصل في واحد منهم هيسمع فالتاني عادي

والمشكله الي زي دي في لغه C Sharp اسمها Diamond Problem

Shallow :

وهي اني عاوز اخد نسخه حقيقيه واعدل عليه مش الاصل

```
var arr1=[1,2,3,4]
var arr2=[...arr1];
arr2.push(5)
console.log(arr2)
console.log(arr1)
```

```
Out Put : [1, 2, 3, 4, 5]
[1,2,3,4]
```

هنا حليت المشكله بتاعتي وهي اني عملت Reference جديد فال heap

في طريقه تانيه اسمها json

```
var arr1=[1,2,3,4]
var arr2=JSON.parse(JSON.stringify(arr1))
arr2.push(5)
console.log(arr2)
console.log(arr1)
```

```
Out Put : [1, 2, 3, 4, 5]
[1,2,3,4]
```

وهنا نفس الفكره الاتنين array ب اتنين reference مختلفين .