nailed أصحاب الهدف

كل مشاكل ال waterfall في PureJS

Promise.all

هتخلي كل ال request تشتغل time same of The

"Paraller"

لكن نأخذ بالنا إن Promise.all بتشتغل لو ال requests مش معقدة على بعضها يعني زي categories, products, users,

إنما لو مثلا orders معقدة على userId يبقى مش هينفع

11 OOP JS

هنا عندنا حاجة في JS يتم تسميتها اسم object

- أي شيء يتم بناءه باستخدام JSON ممكن لو لاحظت إنه اختصار JavaScript object notation فهو فعلياً عبارة عن object

```
let x = {
    "name": "Saleza"
    "age": 342
}
console.log (typeof x) => object
```

ملحوظة: اخطأ في الفيديو لما قال إن ده JSON

ده الحقيقة object literal هو بنكتبه باستخدام {} literally

يعني باستخدام {} وليس باستخدام class أو constructor

=> يعني x عبارة عن object literal وليس JSON

لأن JSON عبارة عن string يُكتب ال key بداخل "name"

حيث لو قمت بكتابة " " حول key في object وقمت بالحفظ يحذفها تلقائياً

لذا هو ليس مهم في object بينما أساسي في JSON

• الفرق الثاني إنه لا يمكن كتابة method بداخل JSON فهو string

بينما object يمكن ذلك

الخلاصة: إن الـ object يشبه الـ JSON

JSON ≠ object

JSON = string

الـ JSON يشبه object فقط عندما نقوم بتحويله لـ Parse it

object

```
let x = {
    name : "Ahmed",
    age : 32
}
```

log(x.name) => Ahmed

JSON

```
let y = '{"name" : "Ahmed"}'
```

log(y.name) => undefined

```
let z = JSON.Parse(y)
```

log(y.name) => Ahmed

إذا JSON.Parse يأخذ

string = JSON = text الـ

JS object

الخلاصة

JSON.Parse( )

يتحول الـ JSON لـ JS object

- الطريقة الثانية لعمل object

باستخدام class

الـ class عبارة عن BluePrint قالب فيه خصائص و methods

```
class Animal { }
```

عند يبدأ بحرف Capital

```
let lion = new Animal( )
```

تكوين class من object

```
class Animal {
    move () {
        log ("moving")
    }
    Jumping () {
        log ("Jumping")
    }
}
```

method لما بنكتب — بنكتب اسم الـ method — Run يعني

```
let lion = new Animal ()
lion.move () => moving
```

كده اتكلمنا عن methods

بالنسبه للـ Properties حيث لكل حيوان "لون، اسم، ... "

```
class Animal {
    constructor () {
        log ("constructor")
    }
}
```

هي method بتشتغل تلقائياً عند عمل object
class يعني أول ما تعمل object
هتشتغل حتى لو ما استدعتش دي method

```
let lion = new Animal () => Constructor
```

هتطبع by default

هنضيف الـ Properties الخاصة بكل animal

```
constructor(name, color) {
    this.name = name
    this.color = color
```

this ترمز للـ object

اللي بيعتها لما اعمل الـ object

lion زي ما الـ بكريته
tigger ولو كريت تاني شيء

خلي بالك إسمها name
بداخل الـ animal كتابة
الوصول استخدم this

Class Animal {
Constructor(name, color) {

this.name= name
this.color= color

آكاديمية تمكين

}
move( ) {
console.log('Animal ${this.name} moved') }
Jump( ) {
log ('Animal ${this.name Jump}')}

}

let lion= new Animal("lion", "yello")
lion.move( ) => lion moved
let tigger =new Animal ("tigger", "black")
tigger.Jump() => tiger Jump

لمابقول this.name كأنك بتقول lion.name= lion
تشير إلى الـ object

للتوضيح أكتر

XMLHTTPRequest

let request = new XMLHTTPRequest( )
كلمة new معناها إنك object

بعت كأنك هنا الكلاس
Class XMLHTTP request {
xml( ) فاهم لأنك استدعيت ← Constructor( ) {
}
on load =null
- - - -
}

محمد و أميرة

شرح أكثر عن Constructor fun

ال - Constructor function هي أي function عادية الفرق الوحيد
انك لما بتعمل call or invoke بتحط كلمة new
- وده أول حرف فيها capital وده بالفعل ال built in fun اللي عندنا مثل Array, object
و String كلهم built in constructor function

Const Person = function

طب عندنا ٣ نوع function يشتغل أي نوع ؟
لا يشتغل على declaration, expression لكن arrow (no)
arrow function have problem with "this"

تذكير سريع ب deceleration & expression

Decleration	Expression
- عشان يكون فيها اسم	- بدون اسم عادي
function loadData ( ) { log ( ) }	let loadData = function
- لا تتخزن variable	- تتخزن في variable

so back to Constructor function

Const Person = function(Name, age) { }

new Person ("Nora", 30)

اللي بيحصل هنا هو اللي بيحصل بمجرد كتابته call

1- new

كلمة new بتعمل empty object {} new object

2- call function → empty {} بتاعه this = {}
فالله له مكونا

3- {} created object linked to prototype

4- function by default return object auto

Prototype

• أي function في JS ليها Property تسمى Prototype بما ذلك ال Constructor

• أي object بيتكريت عن طريق constructor fun يبقى شايف العلاقات الموجودة جوه ال Prototype بتاع ال Parent

Person.Prototype

لو ماقولنا أي function ليها by default Prototype

لو عايز اضيف function للـ class لكن ما اضيفهاش في class نفسه لأن كده كل مرة هكريت object هتتكريت معاه ملوث مرة ف ده مو أفضل حاجة

مثال =>

```
Person {
  this.getYear() {
      log("this year")
  }
}
```

ده bad practice وبرجعا متعملش كده

ف الأفضل نحطها في Prototype

~~Person.Prototype~~

```
Person.Prototype.getYear = function() {
      log(      )
}
```

ولو طبعت ال instance هتلاقي جواه Prototype وفيه ال functions اللي ضيفتها

معنى كده إن أي object علشان طريقة Person هو شايف كل ال methods الموجوده في PrototYPe

ملحوظة: إن ال functions الموجوده جوه ال PrototYPe مش مملوكة لـ object محدد هي بتتورث لكل ال objects

Codezetta

=> PrototYPe inheritance <=

يقدر يـ access ال Constructor fun بتاع ال PrototYPe

=> هنلاقي "__Proto__" جوه كل object اتكتبت

اتكتبت لما كتبنا object الخطوة التالتة

Link empty object with PrototYPe

```
Const Ahmed = new Person("Ahmed")
    ~~Ahmed.Pro~~
    Ahmed.__Proto__ === Person.PrototYPe
```

لو لاحظنا إن كل اللي كتبناه بدون "class" لأن ال class هي suger للتنظيم أكتر وتشبهًا باللغات الأخرى إنما الأصل Constructor