



Présenté par :

- Hona Yete
- Ahouefa Kpossou
- Franco Davy Irakoze



Université  
Lille1  
Sciences et Technologies



# Plan

I. Introduction à PharoJs

I.1 Le principe

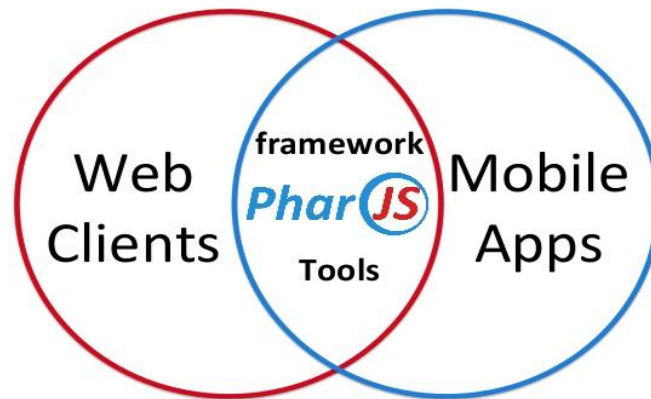
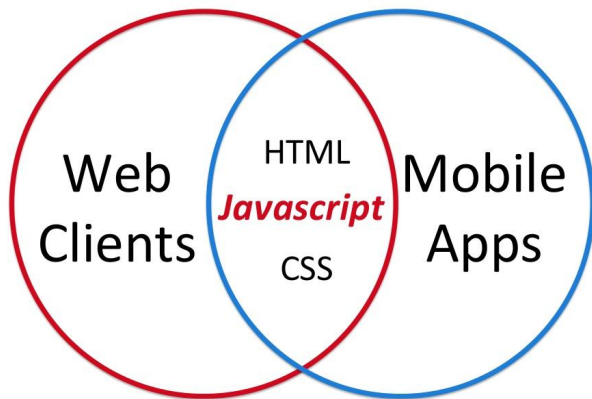
II. Présentation des packages

III. Package PharoJsApp

IV. Comment utiliser PharoJs

V. Design patterns

# PharoJs



# Le principe

Lifecycle with **PharJS**

**100%**

**Pharo**

**0%**

- Code + Test de model
- Code + Test d'intégration
- Test de portabilité
- Exporter pour la production

**0%**

**Javascript**

**100%**



# Packages de PharoJs

PharoJs regroupe :

- 8 Packages avec tests intégrés
- 2 packages sans tests internes
- 2 packages test correspondant aux tests des 2 packages ci- haut cités
- 308 classes

## PharoJsApp :

Contient la classe principale PjApplication .

## PharoJsTestFramework :

Framework qui sert à tester la librairie de PharoJs.

## PharoJsBridge & PharoJsBridgeTest :

Contiennent respectivement les classes permettant de créer un port sur un serveur et les tests des classes du package PharoJsBridge.

## PharoJsCoreLibraries & PharoJsCoreLibrariesTest :

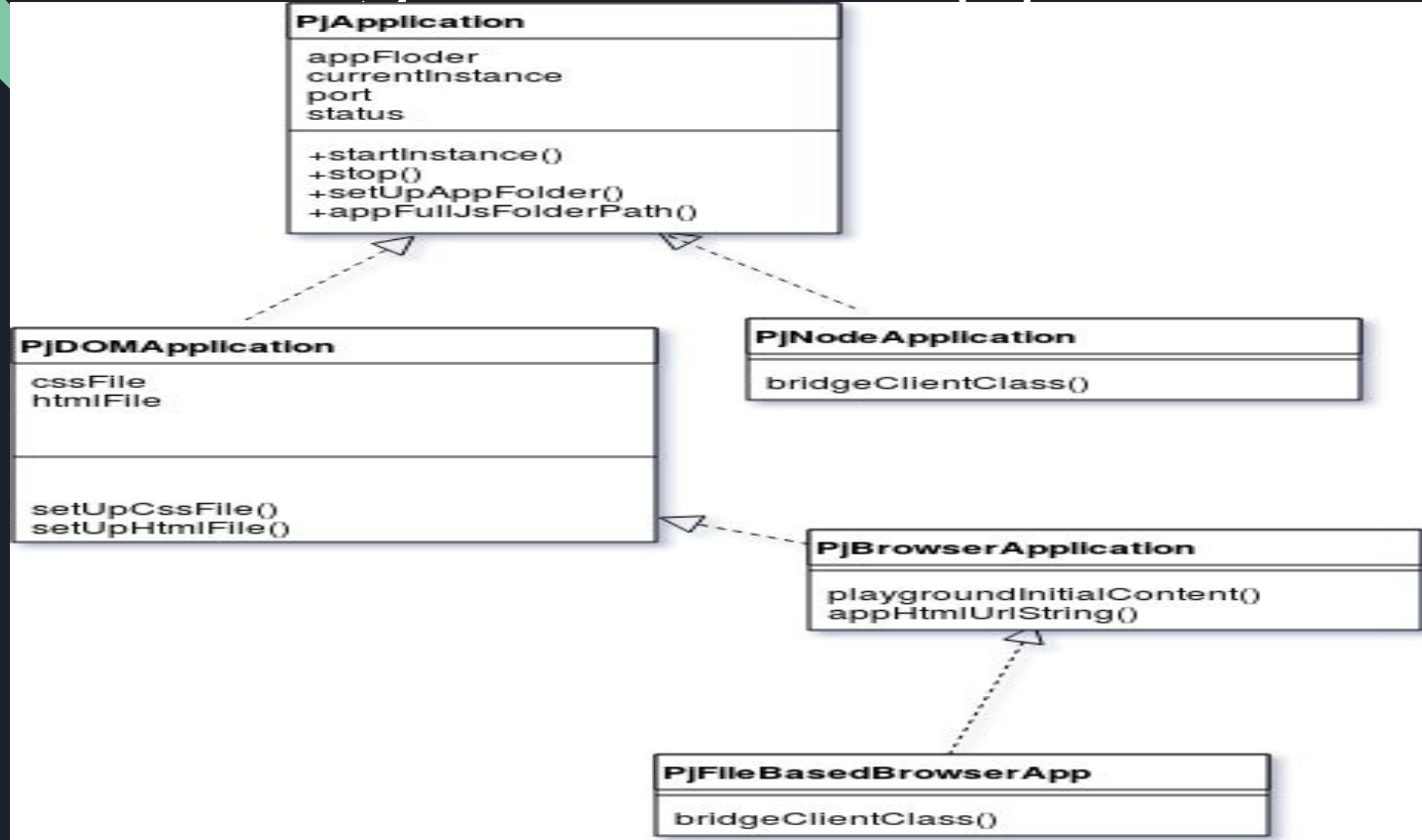
Contiennent respectivement les classes permettant de créer les librairies de PharoJs et les tests du package PharoJsCoreLibraries .

**PharoJsTranspiler** : Contient les classes qui vont se charger de migrer le code pharo en JavaScript.

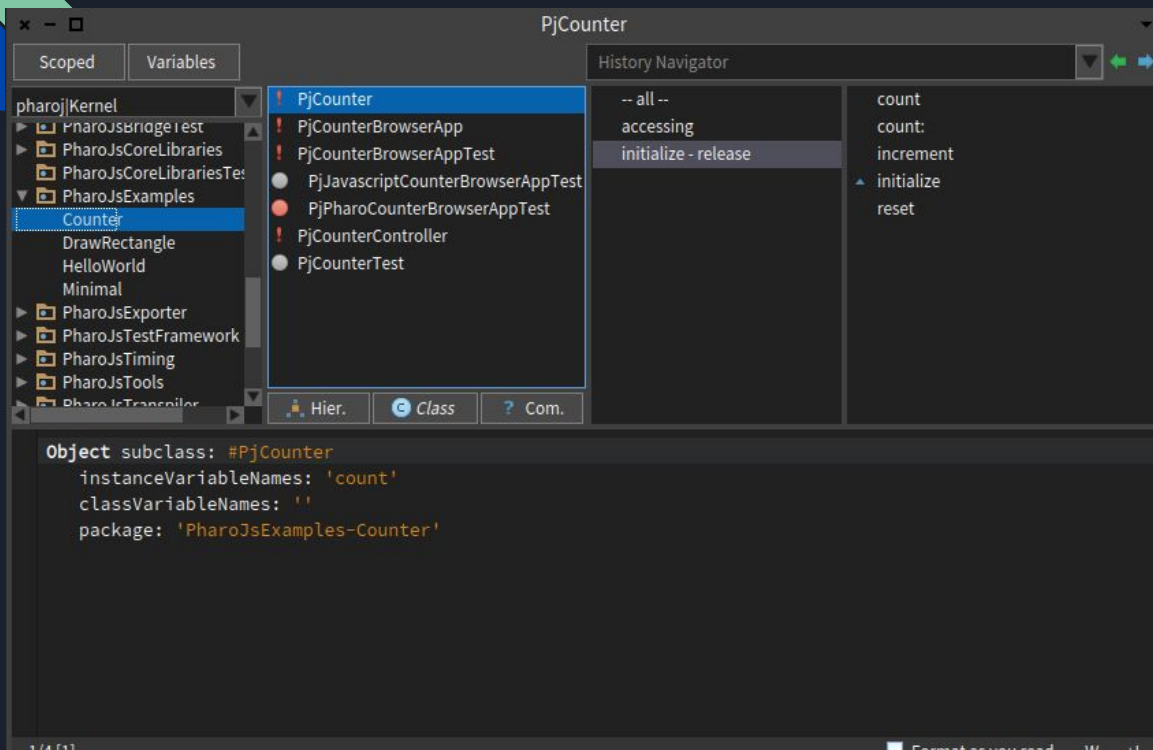
## PharoJsTools :

- Help
- Inspector
- Playground
- Tutorial

# Package PharoJsApp







## PjCounter

- a une variable d'instance **count** qui désigne l'élément compté
- possède deux accesseurs
- possède une méthode **initialize**
- possède une méthode **increment** qui se charge d'ajouter 1 à la valeur précédente de 1.
- **reset** qui se charge de ramener la valeur à compter à 0

### increment

```
self count: self count + 1
```

### reset

```
self count: 0
```

The screenshot shows the Pharo IDE interface with the **PjCounterController** class selected in the **History Navigator**. The **Class** tab is active, showing the class hierarchy and instance variables. The instance variables are `counter` and `countDisplay`. The package is `PharoJsExamples-Counter`.

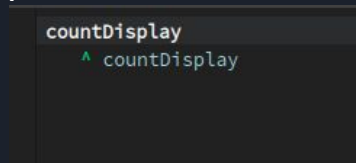
```
Object subclass: #PjCounterController
  instanceVariableNames: 'counter countDisplay'
  classVariableNames: ''
  package: 'PharoJsExamples-Counter'
```

The bottom status bar shows "1/4 [1]" and "Format as you read W +L".

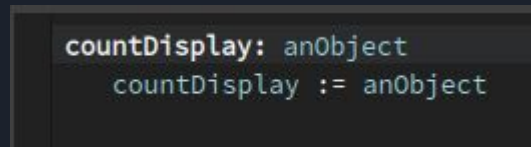
## PjCounterController :

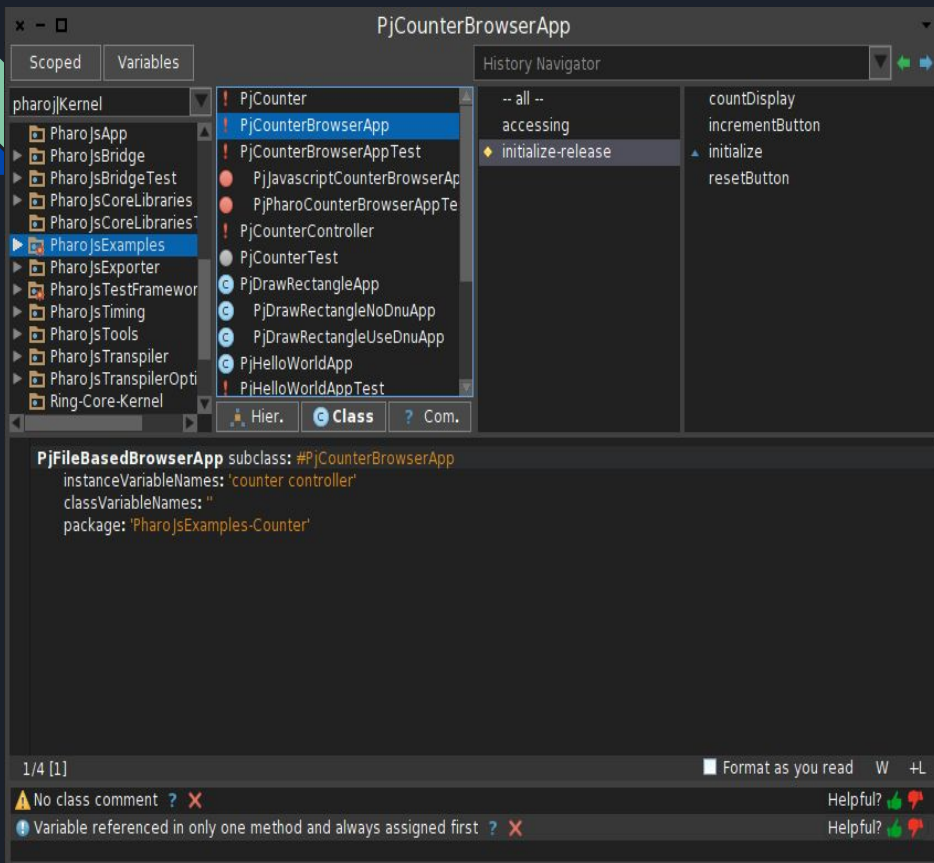
- a deux variables d'instance un objet de PjCounter et countDisplay
- dispose des accesseurs pour les variables d'instances.
- dispose d'une méthode unupdateDisplay qui met à jour le contenu de countDisplay
- possède les méthodes increment et reset pour la manipulation de la donnée calculée

## • un getter



## • un setter





**PjCounterBrowserApp**

History Navigator

countDisplay  
incrementButton  
resetButton

**PjCounterBrowserApp** subclass: #PjCounterBrowserApp  
instanceVariableNames: 'counter controller'  
classVariableNames: ''  
package: 'PharoJsExamples-Counter'

1/4 [1] Format as you read W +L

No class comment ? ✗ Helpful? 🟢 🟡 🟠 📌  
Variable referenced in only one method and always assigned first ? ✗ Helpful? 🟢 🟡 🟠 📌

**PjCounterBrowserApp** est la classe principale possède :

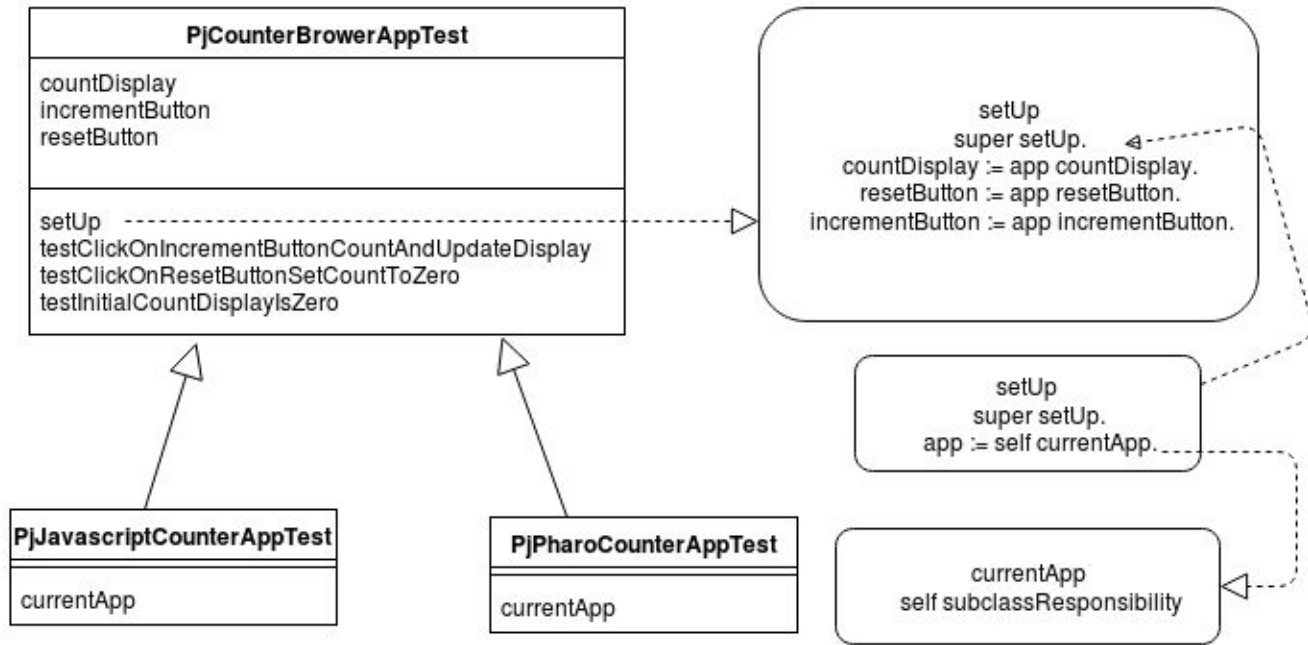
- deux variables d'instances
- des accesseurs
- une méthode initialize pour l'initialisation

```
incrementButton
^ self domElementAt: 'incrementButton'
```

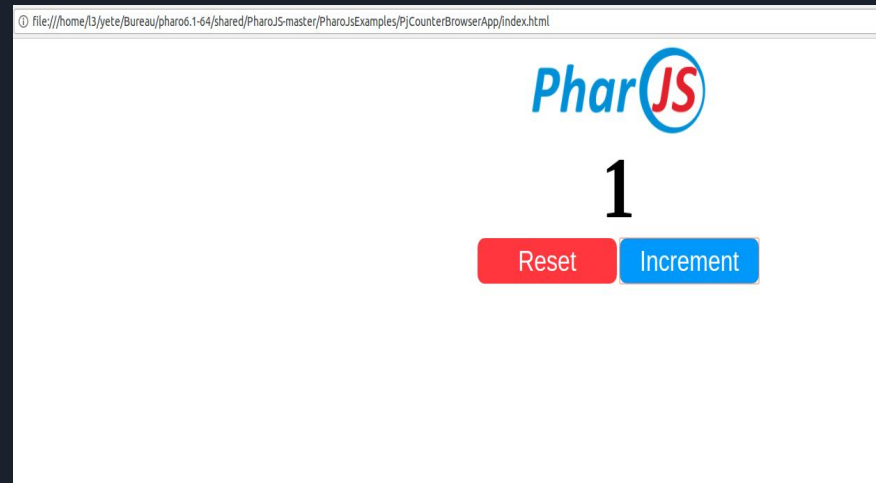
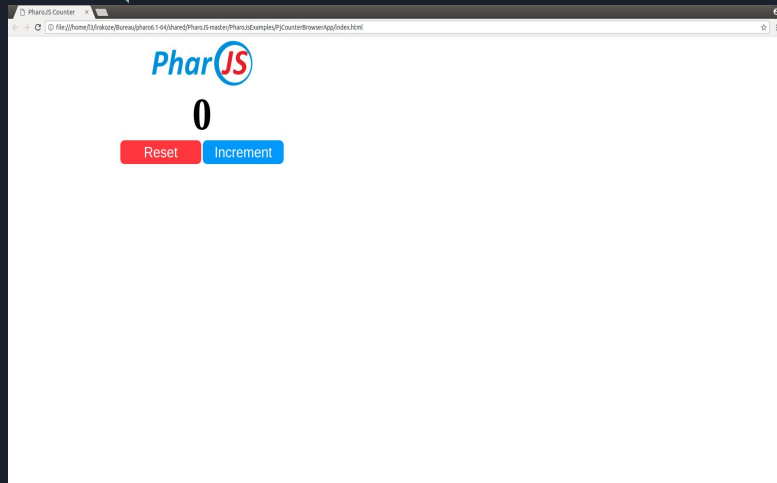
```
initialize
super initialize.
counter := PjCounter new.
controller := PjCounterController new.
controller counter: counter.
controller countDisplay: self countDisplay.
self resetButton addEventListener: #click block: [ controller reset ].
self incrementButton addEventListener: #click block: [ controller increment ]
```

2/8 [2] -- 2/8 [19] Form

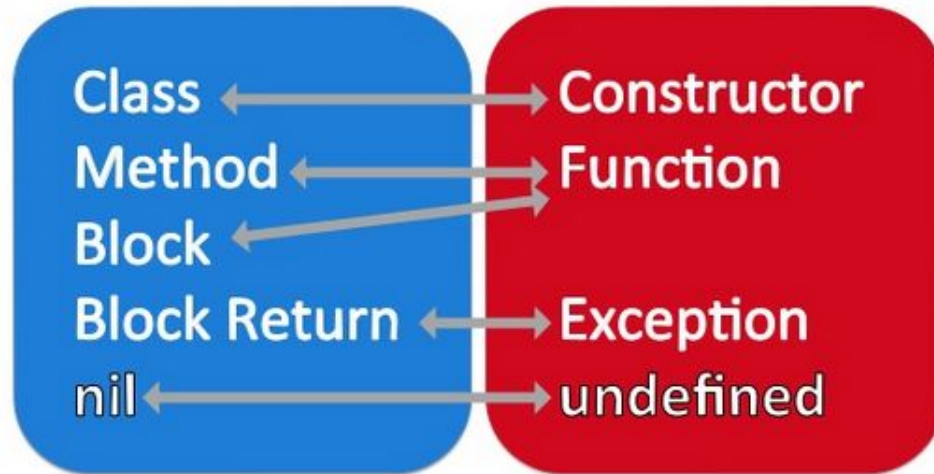
Inconsistent method classification ? ✗



# Application PjCounter



## Mapping **Pharo** to **Javascript**



Design pattern:

- Template Method
- Lazy initialization