Mapping Object

1 - foo的 连通成员变量 递归 做序列化 + Base64 + 压缩

普通类型序列话 - {key:value}
MappingObject类型- {key: “(MappingObject)|||Foo|||{value:123}”}

记录类的名字，后续做反序列的时候，通过类名字创建类

2 - 序列化之后的写入HTML，作为看不到的元素

```
<pre id=‘data-transfer’ style=‘display:none’>
    xxxxxxxxxxxxxxxxxxx
</pre>
```

后端

DB


```
Class Foo < MappingObject
   mapping_accessor :value

   def calc_value()
       return value**2
   end
end

foo=Foo.new()
foo.value = load_data_from_db()
RenderWrap[“foo”] = foo
RenderWrap.data
```

HTML

```
<pre id=‘data-transfer’
style=‘display:none’>
    xxxxxxxxxxxxxxxxxxx
</pre>
```

前端

```
$data[“foo”].calc_value
```

Mapping Object

3 - 加载解析的代码到前端Ruby-JS

$data = jsrb_undata($document.at_css('#data-transfer').text)

4 - 前端加载页面，运行编译后JS代码

用dom里面查找id=“data-transfer”的元素，获取text
然后text进入jsrb_undata函数，来初始化全局变量$data