# Day04

### jQuery基础

* jQuery仅仅是在应用层
* jQuery是JavaScript函数库，封装了一堆JavaScript函数
* Write less do more
* jQuery的使用精髓
* 选择元素
* 循环元素
* 链式调用
* jQuery的官方版本 1 2 3
* 正在时用的版本 1.11.3 兼容IE6
* jQuery对象是个 类数组

### jQuery源码

1. (**function** () {
2. **function** jQuery(selector) {
3. **return** **new** jQuery.prototype.init(selector)
4. }
5. // 原型 构造函数
6. jQuery.prototype.init=**function**(selector){ // id class dom对象 包含dom对象的数组
7. // 选择元素的时候，只需要去考拉id和class
8. // document.getElementById()   直接是一个dom元素
9. // document.getElementsByClassName()  包含dom元素的类数组对象
10. // indexof字符串方法，检索子串首次出现的位置 如果没有检测到 出现-1
12. **this**.length=0 // 初始化length长度
14. **if** (selector==**null**){
15. **return** **this**
16. }
17. **if** (**typeof**(selector)=="string" && selector.indexOf('.') != -1){
18. **var** dom=document.getElementsByClassName(selector.slice(1)) // 包含dom元素的类数组对象
19. } **else** **if** (**typeof**(selector)=="string" && selector.indexOf('#') != -1) {
20. **var** dom=document.getElementById(selector.slice(1)) // dom对象
21. }
22. // 判断dom变量到底为类数组还是dom对象
23. console.log(dom)
24. **if** (selector **instanceof** Element || dom.length == undefined){ // dom对象
25. **this**[0]=dom || selector
26. **this**.length++
27. } **else** { // 类数组
28. **for** (**var** i = 0; i < dom.length; i++) {
29. **this**[i]=dom[i]
30. **this**.length++
31. }
32. }
33. }
34. jQuery.prototype.css=**function** (config) { // config是一个对象
35. **for** (**var** i = 0; i < **this**.length; i++) {
36. **for** (**var** attr **in** config) {
37. **this**[i].style[attr]=config[attr]
38. }
39. }
40. // 返回jQuery对象 进行链式调用
41. **return** **this**
42. }
43. // 扩展get方法
44. jQuery.prototype.get=**function** (num) {
45. // if (num==null) {
46. //     // 类数组转换成数组
47. //     console.log(Array.prototype.slice.call(this,0))
48. //     return [].slice.call(this,0)
49. // } else {
50. //     if (num>=0){
51. //         return this[num]
52. //     } else {
53. //         return this[num+this.length]
54. //     }
55. // }
56. **return** num==**null**?([].slice.call(**this**,0)):(num>=0?**this**[num]:**this**[num+**this**.length])
57. }
58. // 扩展eq方法
59. jQuery.prototype.eq=**function** (num) {
60. **var** dom = num == **null** ? ([].slice.call(**this**,0)):(num>=0?**this**[num]:**this**[num+**this**.length])
61. // 给dom对象进行包装 包装成jQuery对象
62. **return** jQuery(dom)  // 不管这个dom是dom对象还是包含dom的jQuery对象
63. }
64. // 独立封装prevObject
65. jQuery.prototype.pushStack=**function**(dom){
66. **if** (dom.constructor!=jQuery){ // 判断是否是jQuery对象，如果不是jQuery对象，进行包装成jQuery对象
67. dom=jQuery(dom)
68. }
69. dom.prevObject=**this**
70. **return** dom
71. }
72. // 扩展add方法
73. jQuery.prototype.add=**function**(selector){
74. **var** curObj=jQuery(selector)  // add传递的jQuery对象
75. **var** baseObj=**this** //  原始的jQuery对象
76. **var** newObj=jQuery() // 新的jQuery对象
77. **for** (**var** i = 0; i < curObj.length; i++) {
78. newObj[newObj.length++]=curObj[i]
79. }
80. **for** (**var** i = 0; i < baseObj.length; i++) {
81. newObj[newObj.length++]=baseObj[i]
82. }
83. **this**.pushStack(newObj)
84. **return** newObj
85. }
86. // 扩展end方法
87. jQuery.prototype.end=**function**(){
88. **return** **this**.prevObject
89. }
90. // 原型共享 不然jQuery出来的实例地下没有办法使用css
91. jQuery.prototype.init.prototype=jQuery.prototype
92. // 函数里面的函数 外面可以访问
93. // 简单闭包
94. **return** window.$=window.jQuery=jQuery
95. })()