正课:

1. 事件处理:

页面加载后事件

鼠标事件

模拟触发

2. 动画:

3. 简化类数组对象操作

4. 封装自定义插件:

1. 添加自定义函数:

2. 封装自定义插件:

1. 事件处理:

1. 页面加载后事件: 2个

window.onload=function(){...}

必须等待所有网页内容(html,css,js,图片)加载完成，才触发

DOMContentLoaded

只要html和js加载完成就可提前触发

不必等待css和图片加载完成

DOMContentLoaded比onload更早触发

总结:

1. 绝大多数不依赖css的初始化操作和事件绑定都应该在DOMContentLoaded阶段提前执行

2. 只有依赖于css的初始化操作，必须放在loaded中等待所有内容加载完成才能执行

如何在DOMContentLoaded阶段执行操作:

$(document).ready(()=>{

... ...

})

鄙视: window.onload vs $(document).ready()

1. window.onload必须等待所有内容(html,css,js,图片)加载完成才能触发——晚

$(document).ready()只要DOM内容(html和js)加载完成就提前触发——早

2. onload只能添加一个处理函数

$(document).ready() 可反复添加多个处理函数

更简化: $().ready(()=>{...})

更简化: $(()=>{...})

总结: 直接放入$()中的回调函数自动在DOM内容加载后提前触发

更简单: 放在body结尾的script中的代码，自动就是DOM加载后提前触发

2. 鼠标事件:

mouseover mouseout

进出子元素，同样会触发父元素的进出事件

问题: 进入子元素，同样也在父元素内，没必要反复触发进出事件

解决: mouseenter mouseleave

进出子元素，不再触发父元素的进出事件

如果同时绑定mouseenter和mouseleave:

$(...).hover(

e=>{...}, //mouseenter

e=>{...} //mouseleave

)

更简化: 如果将进入和移出，统一为一个函数

$(...).hover(e=>{...})

3. 模拟触发:

用程序代替用户的操作，触发其它元素的上的事件

2种:

1. 可直接调用其它元素上的事件处理函数:

$(...).事件名()

2. 用trigger()

$(...).trigger("事件名")

2. 动画: 2大类  
 1. 简单动画: 3种效果:

1. 显示/隐藏:

$(...).show/hide/toggle()

如果没有参数, 默认使用display:none/block瞬间显示隐藏

如果要加动画: 参数: (动画持续时间)

可以为: slow/normal/fast/ 毫秒数

原理:

用定时器逐渐修改width/height/opacity

问题: 1. 程序中定时器的效率比css中transition的效率要低

2. 效果是写死的，无法维护

总结: 今后优先选择css transition和boot中的效果实现动画。复杂动画才用程序中的定时器

2. 上滑下滑:

$(...).slideUp/slideDown/slideToggle()

3. 淡入淡出:

$(...).fadeIn/fadeOut/fadeToggle()

动画函数的回调函数: 专门在动画结束后自动执行

$(...).动画函数(时间,fn)

问题: 如果连续执行的函数特别多，会导致回调的层级很深——callback hell

解决: 动画队列:

jQuery会为同一个元素上反复调用的多个动画函数创建执行队列。前一个函数执行完，后一个函数才能开始。

2. 万能动画: 可对任意css属性执行动画效果

$(...).animate({

css属性:目标值,

... : ... ,

},时间,callback)

仅支持纯数字值的属性:

width/height top/left margin/padding

opacity borderWidth fontSize ...

不支持: 颜色, 字体, CSS3中的transform

排队与并发:

排队: 对同一个元素反复调用的动画函数，自动排队执行。

并发: 放在同一个animate内的要修改的多个css属性，同时改变。

停止动画:

transition: 无法控制停止

解决: animate()+stop()

$(...).stop() 停止正在执行的动画

问题: 无法清除队列中剩余的动画，剩余动画依然会执行

解决: $(...).top(true) true: 清空队列

延迟执行: $(...).delay(时间).XXX()

3. 简化类数组对象:

为什么: 1. jQuery对象本身就是类数组对象

2. 类数组对象无法使用数组强大的API

所以,jQuery中提供了操作类数组对象的简化版API

包括:

$(...).size() $(...).length

$(...).get(i) $(...)[i]

$(...).each((i,elem=)>{...}) 类似于数组的forEach

可对jQuery对象中封装的每个DOM元素执行相同的操作

其中: i当前位置, elem 当前DOM元素

在jQuery3.0中可用for...of代替

$(...).index(elem) 类似于数组的indexOf

查找elem在整个jQuery结果集合中的下标位置。如果找不到，返回-1

4. 添加自定义函数:

何时: 如果jQuery提供的函数不够用，就可添加自定义函数:

如何:

1. 必须放在jQuery.fn原型对象中

jQuery.fn.自定义函数=function(){

this//指将来调用函数的.前的jQuery对象

}

2. 判断是否引入了jQuery类型

if(typeof jQuery!=="function")

throw new Error(...)