JS 6

Cookies (biscoitos) são dados armazenados em pequenos arquivos de texto no computador cliente. Note que quando o servidor manda uma página web para um browser a conexão é desfeita e o servidor esquece tudo referente àquele usuário. Cookies foram inventados para resolver esse problema, ou seja, para "lembrar informações sobre aquele usuário". Por exemplo, quando um usuário visita um site seu nome pode ser armazenado como um cookie. Na próxima vez que esse usuário visitar aquele site o cookie "relembra" o nome do usuário. Os cookies são armazenados em pares nome-valor como `nome=maria`. Quando o browser requisita uma página web de um servidor, os cookies que pertencem a essa página são anexados à requisição. Dessa maneira o servidor consegue lembrar as informações sobre o usuário.

o JS pode criar, ler e excluir cookies através da propriedade `document.cookie`. Para criar a síntaxe é

```
document.cookie = "username=Maria Silva";
```

Pode ser incluída uma data de expiração. Por default o cookie é excluído quando o browser é encerrado. Ficaria

```
document.cookie="username=Maria Silva;
expires=Tue, 01 Jan 2015 12:00:00 UTC";
```

Há ainda o parâmetro `path` que indica qual o caminho para a gravação.

Para ler um cookie com JS, a síntaxe é

```
var x = document.cookie;
```

Haverá um retorno em um único string de todos os cookies, como em cookie1=valor; cookie2=valor; cookie3=valor;

Para trocar um cookie procede-se como se ele estivesse sendo criado. Com isso o cookie velho é sobreescrito.

Para excluir um cookie, basta expirá-lo com uma data já passada, por exemplo

```
document.cookie = "username=;
expires=Thu, 01 Jan 1970 00:00:00 UTC";
```

Note que não é preciso definir valor quando o cookie é exluído.

Agora um exemplo para criar um cookie

```
function setCookie(nome, valo, dias) {
 var d = new Date();
 d.setTime(d.getTime() + (dias*24*60*60*1000));
 var expi = "expires="+d.toUTCString();
 document.cookie = nome + "="
   + valo + "; " + expi;}
```

Já para obter um cookie, a função é:

```
function getCookie(cname) {
 var name = cname + "=";
 var ca = document.cookie.split(';');
 for(var i=0; i<ca.length; i++) {
  var c = ca[i];
  while (c.charAt(0)==' ') c = c.substring(1);
   if (c.indexOf(name) == 0) return
    c.substring(name.length,c.length);
  }
 return "";}
```

Explicando o exemplo acima: Primeiro cria-se a variável name contendo a literal que vai se buscar no cookie, seguida de um "=". Depois, em ca corta-se o cookie (document.cookie) em partes separadas por ";". Para cada parte, ela é colocada em c. Daí, avança-se nos caracteres em branco até chegar ao primeiro não branco. Ele tem sua literal comparada com "name". Se igual, devolve-se o complemento do nome, que é o que se buscava.

Já a função para checar um cookie é

```
function checkCookie() {
 var user=getCookie("username");
 if (user != "") {
  alert("Welcome again " + user);
  } else {
  user = prompt("Please enter your name:","");
  if (user != "" && user != null) {
   setCookie("username", user, 30); } } }
```

Exercício 1

Crie um cookie com seu nome e endereço no computador cliente. Depois investigue onde ele está no computador.

Agora o assunto são os frameworks (bibliotecas). Trata-se de um esforço para produzir funções de uso comum visando diminuir o esforço para escrever aplicativos.

A primeira é JQUERY. Para acessá-la escreva

```
<script src="http://code.jquery.com/
 jquery-2.0.3.min.js">
```

A principal função jquery é a função `$()`. Se você passar objetos DOM a esta função, ela retorna objetos jQuery com funcionalidades adicionais. Veja uma função feita em JS puro

```
function myFunction() {
 var obj = document.getElementById("h01");
 obj.innerHTML = "Hello jQuery"; }
onload = fun1;
```

Agora, compare a mesma função feita em JQuery:

```
function fun1() {
    $("#h01").html("Oi jQuery"); }
$(document).ready(fun1);
```

A última linha acima passa o DOM HTML para o jQ:`$(document)`. A função JQ retorna um objeto jQ, onde `ready()` é um método. Já que funções são variáveis em jQ, fun1 é passada como variável ao método ready() do jQ. O jQ retorna um objeto diferente do DOM que foi passado a ele. Com métodos e propriedades diferentes das do objeto DOM. Para testar o jQuery faça

```
<html> <head> <script
src="http://code.jquery.com/jquery-2.0.3.min.js"
> </script> <script>
function fun1() {
    $("#h01").html("Oi jQuery")  }
$(document).ready(fun1);
</script> </head> <body>
<h1 id="h01"></h1> </body> </html>
```

Agora troque o comando central por

```
$("#h01").attr("style","color:red")
 .html("Oi jQuery")
```

A síntaxe jQuery é

```
$(seletor).ação()
```

Onde $ define um comando jQ; seletor indica qual o elemento do HTML vai ser manipulado e ação indica o que fazer nesse elemento. Exemplos

$(this).hide()	apaga o elemento atual
$("p").hide()	apaga todos os elementos `<p>`
$(".test").hide()	apaga todos elementos com classe="test"
$("#test").hide()	apaga todos elementos com id="test"

Todos os métodos do jQ precisam estar dentro do evento ready document

```
$(document).ready(function(){
   // métodos jQuery vão aqui
});
```

Isto é para prevenir que algum código jQ seja iniciado antes que o documento seja completamente carregado. Um jeito mais barato de fazer o mesmo

```
$(function(){
  // métodos jQuery vão aqui
});
```

Alguns exemplos de seletores jQuery

se fizer	seleciona:
$("*")	todos os elementos
$(this)	o elemento HTML corrente
$("p.in")	elementos `<p>` da classe=in
$("p:first")	o primeiro elemento `<p>`
$("ul li:first")	primeiro li da primeira ul
$("ul li:first-child")	o primeiro li de todas as ul
$("[href]")	todos elementos que tem atributo href
$("a[target='x']")	elementos `<a>` com target=x
$("a[target!='x']")	elementos `<a>` com target≠x
$(":button")	todos os elementos button e elementos `<input>` do tipo button
$("tr:even")	todos eventos tr pares
$("tr:odd")	todos eventos tr ímpares

Os eventos jQuery são respostas a ações tomadas pelo usuário. Eis alguns:
mouse: click, dblclick, mouseenter, mouseleave
keyboard: keypress; keydown, keyup
form: submit, change, focus
document/window: load, resize, scroll, unload...
Para definir o que acontece quando um evento ocorre, faça

```
$("p").click(function(){
  // escreva o que acontecerá
});
```

Neste caso, quando o usuário clicar em algum parágrafo `<p>` acontecerá o que estiver na função. Veja um exemplo

```
$("p").click(function(){
  $(this).hide();
});
```

Existem 3 métodos jQuery para manipular o DOM que são `text()`: cria ou retorna o texto de elementos selecionados; `html()`: cria ou retorna tudo (incluindo a marcação html) e `val()`: cria ou retorna o valor de campos de formulário. Veja no exemplo

```
<head>
<script src=endereço do jQ"></script>
<script> $(document).ready(function(){
  $("#btn1").click(function(){
    alert("Text: " + $("#test").text());  });
  $("#btn2").click(function(){
    alert("HTML: " + $("#test").html());  });
});
</script> </head> <body>
<p id="test">Algum texto <b>negrito</b></p>
<button id="btn1">Mostra Texto</button>
<button id="btn2">Mostra HTML</button> </body>
```

As suas funções podem ser deixadas em um arquivo a parte então, pode fazer:

```
<head> <script
src="http://code.jquery.com/jquery-2.0.3.min.js"
> </script>
<script src="minhas.js"></script> </head>
```

Exercício 2

Crie uma página HTML simples (título, 4 ou 5 parágrafos, alguns cabeçalhos, 2 ou 3 links e uma tabela. Depois brinque à vontade manipulando os elementos HTML através do jQ.

☞ Para você fazer

1. Escreva aqui e se não couber continue no verso desta folha um resumo das coisas estudadas nesta aula:

   _____

   _____

2. Imprima os arquivos exemplos desta folha (gerador de cookies e arquivos html e js da manipulação do exercício 2) grampeie-os nesta folha e devolva tudo ao professor.

.