10 Tempo

Neste capítulo você vai aprender a acessar e controlar variáveis temporais, essenciais para a modelagem de quaisquer sistemas dinâmicos, da física à biologia à epidemiologia à economia. Além de aprender como formatar convenientemente e fazer cálculos com o tempo fornecido pelo sistema, vai descobrir como utilizar temporizadores para fazer com que ações sejam executadas em instantes pré-definidos ou a intervalos regulares.

`Date()`

Para obter a data e hora do sistema, emprega-se o objeto `Date`:

Listagem:

```
<script>
var agora = new Date()
document.write(agora);
</script>
```

Resultado:

Fri Nov 25 2016 20:05:00 GMT-0200 (Horário brasileiro de verão)

Note que a variável `agora` contém a data e hora do momento em que a instrução foi executada e **não** é constantemente atualizada.

O objeto `Date` tem vários métodos e aqui serão apresentados apenas os mais comuns. Alguns métodos retornam os diversos componentes de uma data separadamente:

Listagem:

```
<script>
var agora = new Date();
var dia   = agora.getDate();
var mes   = agora.getMonth();
var ano   = agora.getFullYear();
var hora  = agora.getHours();
var min   = agora.getMinutes();
var sec   = agora.getSeconds();
var milisec = agora.getMilliseconds();
str  = "Esta frase foi impressa em às ";
```

```
str += hora + " h " + min  + " m " + sec + " s ";
str += milisec + " ms de ";
str += dia + "/" + (mes+1) + "/" + ano;
document.write(str);
</script>
```

Resultado:

```
Esta frase foi impressa às 20 h 5 m 0 s 612 ms de 25/11/2016
```

> Note o uso do método `getFullYear()` ao invés de `getYear()`, que também existe. O problema deste último método é que em alguns sistemas operacionais ele devolve o ano com dois dígitos, contando a partir de 1900, o que faz com que "2009" fique "109". Note também que foi somada uma unidade ao mês no momento da impressão pois o método `getMonth()` retorna valores de 0 (janeiro) a 11 (dezembro).

O objeto `Date` também pode ser instanciado com uma data específica. Por exemplo, uma variável que contenha a representação das 12 horas e 10 minutos do dia 28 de outubro de 2015 pode ser criada utilizando a instrução `var umDiaQualquer = new Date(2015,9,28,12,10)`. Faça uma busca na internet (usando algo como "Date() JavaScript") para ver todos os tipos de parâmetros que podem ser passados para o construtor do objeto.

Sua vez.. (10-1)

Utilize os métodos do objeto `Date` para obter a data do sistema e determinar o tempo decorrido, em minutos, desde o início do mês até o momento em que seu programa é executado. Por exemplo, se seu programa fosse executado às 12h10m do dia 28 de outubro de 2015, deveria mostrar que passaram-se *dias* × 24 × 60 + *horas* × 60 + *minutos* = 41050 minutos desde as 00h00m do primeiro dia do mês.

A informação de tempo no sistema é de fato guardada como o número de milisegundos decorridos desde as 0 horas do dia 1o. de janeiro de 1970 e o objeto é convertido nos vários formatos pelos diferentes métodos. O script a seguir mostra como obter este número com o método `getTime()` do objeto `Date` no momento em que a página é carregada.

Listagem:

```
<script>
var agora = new Date();
document.write(agora.getTime() + " milissegundos.");
</script>
```

Resultado:

```
1480111500613 milissegundos.
```

Utilizando este método é possível subtrair os valores de objetos `Date` para obter o tempo decorrido entre dois eventos. O exemplo a seguir ilustra como obter o tempo de execução de um trecho de código que soma 1 milhão de números aleatórios. O script utiliza duas instâncias do objeto `Date`, uma obtida no início dos cálculos e outra no final.

Listagem:

```
<script>
```

```
var soma = 0;
var N = 1000000;
var inicio = new Date();
for (var i=0;i<N;i++)
   soma = soma + Math.random();
document.write("Soma de " + N + " nros. aleatórios: ");
document.write(soma + "<br>");
var final  = new Date();
document.write("Tempo de execução: ");
document.write((final.getTime()-inicio.getTime()) + " ms");
</script>
```

Resultado:

```
Soma de 1000000 nros. aleatórios: 500428.1851818529
Tempo de execução: 45 ms
```

Note que o resultado da soma deve ser sempre próximo a 500 mil, como era de se esperar ao evocar 1 milhão de vezes um gerador de números aleatórios que produz números entre 0 e 1.

Sua vez.. (10-2)

Modifique o exemplo anterior para tentar estimar o tempo de execução de uma chamada à função (a) seno, (b) logaritmo e (c) raiz quadrada (que podem variar bastante de computador para computador).

Uma potencial aplicação do objeto `Date` e seus métodos é o cálculo da atividade de uma fonte radioativa em um determinado instante sabendo sua atividade em outro instante anterior. A lei do decaimento radioativo faz com que a atividade $A(t)$ de uma amostra radioativa em um determinado instante de tempo t seja dada por:

$$A(t) = A_0\, e^{-\lambda t}$$

onde A_0 é a atividade medida em um instante conhecido e λ é a constante de decaimento, relacionada à meia-vida $T_{1/2} = \ln(2)/\lambda$ do radioisótopo. O script a seguir assume que uma amostra de cobalto-60 ($T_{1/2}$ = 5,3 anos, $\lambda = 0.13$ ano^{-1}) tinha, em 15 de julho de 2005, uma atividade de 10 GBq, ou 10×10^9 decaimentos por segundo, e calcula qual a sua atividade em uma data especificada pelo usuário:

Listagem:

```
<script>
function calculaAtividade() {

   var ano = document.getElementById("ano").value;
   var mes = document.getElementById("mes").value;
   var dia = document.getElementById("dia").value;

   var dataUsr = new Date(ano,mes,dia);
   var dataRef = new Date(2005,7,15);

   var dif = (dataUsr.getTime() - dataRef.getTime());
   dif = dif/1000/60/60/24/365;
   var ativ = 1e10 * Math.exp(-0.13*dif);

   document.getElementById("atividade").value = ativ.toPrecision(3);

}
</script>
<p>
Cobalto-60 (<i>T</i><sub>1/2</sub> = 5,3 anos)<br>
Atividade em 15 de julho de 2005: 10 &times; 10<sup>9</sup> dec/s<br>
Atividade em:
```

```
<input type="text" size="1" value="" id="dia">/
<input type="text" size="1" value="" id="mes">/
<input type="text" size="1" value="" id="ano">
<input type="button" value=" OK " onClick="calculaAtividade()">
<input type="text" size="5" value="" id="atividade" disabled> dec/s
</p>
```

Resultado:

Cobalto-60 ($T_{1/2}$ = 5,3 anos)
Atividade em 15 de julho de 2005: 10×10^9 dec/s
Atividade em: [ ] / [ ] / [ ] OK [ ] dec/s

No exemplo, são criados dois objetos `Date` que recebem como parâmetros a data digitada pelo usuário e a data da medida de referência da atividade da fonte. Em seguida, é utilizado o método `getTime()` para obter ambas as datas em milissegundos desde 1o. de janeiro de 1970 e calcular a sua diferença, que na linha seguinte é convertida em anos. Com a diferença em anos, é possível utilizar a equação para a atividade com a constante de decaimento dada também em anos.

Note o atributo `disabled`, utilizado no elemento de texto que mostra o resultado dos cálculos. Este atributo faz com que o usuário não possa modificar o valor do campo.

Sua vez.. (10-3)

Modifique o exemplo anterior de modo que a data e a atividade de calibração também possam ser fornecidas pelo usuário.

`setTimeout`

O método `setTimeout(`*`expressão`*`,`*`ms`*`)` executa a `expressão` passado um certo tempo `ms`, especificado em milissegundos. O método devolve uma referência que pode ser utilizada para cancelar a tarefa com o método `clearTimeout(`*`ref`*`)`. O exemplo a seguir sorteia dois números e aguarda 10 segundos para que o usuário escreva o resultado de sua soma e aperte o botão "OK". Se o resultado correto for digitado antes disso, permanecerá na caixa de texto; caso contrário, será substituído por asteriscos.

Listagem:

```
<script>

function escreveSoma(soma) {
   document.getElementById("soma").value = "***";
}

function checaSoma() {
   var num = parseInt(document.getElementById("soma").value);
   if (num==soma) clearTimeout(timeout);
}
var timeout = setTimeout("escreveSoma(soma)",10000);

var num1 = parseInt(Math.random()*100);
var num2 = parseInt(Math.random()*100);
var soma = num1 + num2;
document.write(num1  + " + " + num2 + " = ");

</script>

<input type="text" size="5" value="" id="soma">
```

```
<input type="button" value=" OK " onClick="checaSoma()">
```

Resultado:

22 + 71 = [ ] OK

Note que a referência retornada pela chamada a `setTimeout` é guardada na variável `timeout`, que é passada como parâmetro para o método `clearTimeout()`, que desativará a chamada à função caso o usuário acerte a resposta.

`setInterval`

O método `setInterval(`*`expressão`*`,`*`ms`*`)` executa repetidamente a *`expressão`* a cada intervalo de tempo *`ms`*, dado em milissegundos (diferentemente do método `setTimeout()` que executa a expressão uma única vez). O exemplo a seguir utiliza o `setInterval()` para implementar um relógio no documento, atualizado uma vez por segundo.

Listagem:

```
<script>
function relogio() {
   var agora = new Date();
   var hora = agora.getHours();
   var min  = agora.getMinutes();
   var sec  = agora.getSeconds();
   var str  = hora + ":" + min + ":" + sec;
   document.getElementById("relogio").innerHTML = str;
}
setInterval("relogio()",1000);
</script>
<div id="relogio"></div>
```

Resultado:

20:5:4

Note o uso da propriedade `innerHTML` para atribuir a informação da hora atual à divisão inicialmente definida como vazia no documento.

Sua vez.. (10-4)

Modifique o exemplo anterior de modo que mostre o número de segundos que decorreram desde que o documento foi carregado no navegador, atualizando a informação a cada segundo.

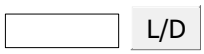
Assim como o método `clearTimeout()` cancela o método `setTimeout()`, o método `clearInterval()` cancela o `setInterval()`. No exemplo a seguir, ambos são utilizados para implementar um cronômetro simples.

Listagem:

```
<script>
```

```
var inicio;
var agora;
var crono = null;

function LigaDesliga() {
   if (crono) {
      clearInterval(crono);
      crono = null;
   }
   else {
      inicio = new Date();
      crono = setInterval("escreveTempo()",10);
   }
}

function escreveTempo() {
   agora = new Date();
   var tempo = (agora.getTime() - inicio.getTime())/1000;
   document.getElementById("tempo").value = tempo.toFixed(2);
}

</script>

<p style='text-align:center'>
<input type='text' value='' id='tempo' size='5'>
<input type='button' value=' L/D ' onClick='LigaDesliga()'>
</p>
```

Resultado:



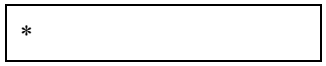
Note o uso da palavra reservada `null` para inicializar a variável global `crono` e para atribuir-lhe valor na instrução `if` no corpo da função `LigaDesliga()`. Na instrução `if`, `null` é equivalente a `false`.

O controle do tempo com o método `setInterval()` é essencial para a produção de animações. O exemplo a seguir mostra um asterisco em um movimento harmônico simples na coordenada horizontal enquanto permanece com a coordenada vertical constante.

Listagem:

```
<script>

function moveCaracter() {
   theta = theta + Math.PI/20;
   var x = parseInt(95 - 90 * Math.cos(theta));
   var y = 10;
   var str = "<div style='position:relative;";
   str += "left:" + x + ";top:" + y + "'>";
   str += "*";
   str += "</div>";
   document.getElementById("caixa").innerHTML = str;
   cnt++;
   if (cnt>100) clearInterval(intervalo);
}
var theta = 0;
var cnt = 0;
var intervalo = setInterval("moveCaracter()",100);

</script>

<p>
<div id="caixa"
     style="margin-left: 2em;border:1px solid black;
          width:200; height:35">
</div>
</p>
```

Resultado:



No exemplo, é definida uma divisão (`<div>...</div>`) identificada como `caixa` no documento. Essa divisão tem 200 pixels de largura, 35 pixels de altura e uma borda sólida preta com 1 pixel de espessura, compondo a "caixa" em que o asterisco oscila. O método `setInterval` faz com que a função `moveCaracter()` seja chamada a cada 100 milissegundos. A função `moveCaracter()` cria uma nova divisão que será colocada dentro da divisão existente no documento pelo uso do `innerHTML`. Nessa divisão, os atributos de estilo `position:relative`, `left:`*x* e `top:`*y* são utilizados para fazer com que o asterisco seja impresso cada vez em uma posição diferente. O movimento cessa após 100 interações.

Sua vez.. (10-5)

Modifique o exemplo anterior de modo a produzir uma caixa quadrada de largura e altura iguais a 200 pixels, substitua o asterisco pela letra "o" minúscula e faça com que ela ande em círculos, com o ângulo variando 60 graus por segundo (6 graus a cada 100 milissegundos).

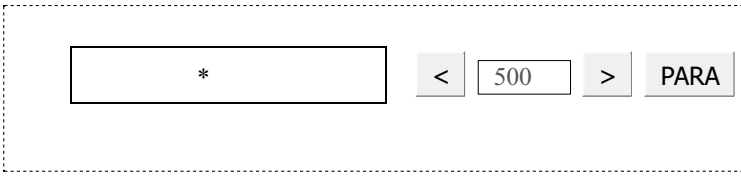
Algumas modificações nesse exemplo permitem incluir controles para acelerar, desacelerar e parar o movimento, como mostrado a seguir.

Listagem:

```
<script>

var passo = 500;

function alteraPasso(opt) {
   switch (opt) {
      case  0:
         clearInterval(intervaloPasso);
         break;
      case -1:
         clearInterval(intervaloPasso);
         if (passo>=200) passo = passo - 100;
         intervaloPasso = setInterval("moveCaracterPasso()",passo);
         break;
      case +1:
         clearInterval(intervaloPasso);
         if (passo<=400) passo = passo + 100;
         intervaloPasso = setInterval("moveCaracterPasso()",passo);
         break;
   }
   document.getElementById("passo").value = "  " + passo;
}

var theta = 0;

function moveCaracterPasso() {
   theta = theta + Math.PI/20;
   var x = parseInt(95 - 90 * Math.cos(theta));
   var y = 10;
   var str = "<div style='position:relative;";
   str += "left:" + x + ";top:" + y + "'>";
   str += "*";
   str += "</div>";
   document.getElementById("caixaPasso").innerHTML = str;
}

</script>
```

```
<p>
<div id="caixaPasso"
     style="margin-left:2em;border:1px solid black;
            width:200; height:35;">
</div>

<div style="position: relative; top: -35; left:250">
<input type="button" value=" < " onClick="alteraPasso(-1);">
<input type="text" value="" id="passo" size="3" disabled>
<input type="button" value=" > " onClick="alteraPasso(+1);">
<input type="button" value=" PARA " onClick="alteraPasso(0);">
</div>
</p>

<script>
document.getElementById("passo").value = "  " + passo;
intervaloPasso = setInterval("moveCaracterPasso()",passo);
</script>
```

Resultado:



As alterações merecem alguns comentários adicionais:

- Qualquer que seja o botão pressionado, a função `alteraPasso()` sempre desliga o `intervaloPasso` para depois, se for o caso das opções −1 e +1, religá-lo. Se a variável que contém sua referência receber um novo valor, a referência antiga será perdida e não será mais possível desligar o temporizador relacionado a ela.

- Para que ficassem ao lado da caixa onde o asterisco se move, os botões foram colocados dentro de uma divisão para a qual foram especificadas coordenadas relativas ao canto superior esquerdo do elemento anterior; por isso o valor negativo do atributo `top`.

- Há um trecho de script antes e outro depois das divisões com a caixa e os controles. O trecho que coloca o valor do passo na caixa de texto entre os controles utiliza o método `getElementById()`, que busca no documento o elemento solicitado. Caso esta instrução apareça antes da divisão, o método não encontra o elemento solicitado e um erro é gerado. Nada impede, no entanto, que todas as instruções sejam colocadas em um único bloco de script depois da definição da caixa.

- Foram colocados limitadores para que os valores do passo ficassem entre 100 e 500, impedindo que atinja valores nulos ou negativos, ou que o movimento fique lento demais.

Sua vez... (10-6)

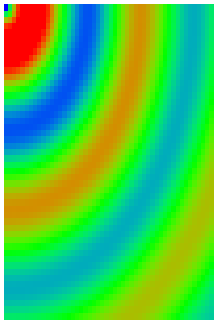
Modifique o exemplo anterior de modo a incluir mais um botão com a legenda `MOVE` e que, ao ser acionado, chame a função `alteraPasso()` com um outro parâmetro (por exemplo, `"m"`). Modifique a função `alteraPasso()` incluindo um novo `case` que responda a esse novo parâmetro, fazendo com que o asterisco passe a mover-se se estiver parado.

O exemplo a seguir modela uma onda bidimensional em movimento.

Listagem:

```
<div id="onda"></div>

<script>
function zCor(z) {
    var z1 = -100;
    var z2 = +100;
    var zm = (z1+z2)/2;
    var r = 0;
    var g = 0;
    var b = 0;
    if (z<=zm) r = parseInt(255*Math.cos((z-z1)/(zm-z1)*Math.PI/2));
    g = parseInt(255*Math.sin((z-z1)/(z2-z1)*Math.PI));
    if (z>=zm) b = parseInt(255*Math.sin((z-zm)/(z2-zm)*Math.PI/2));
    return "rgb(" + r + "," + g + "," + b + ")";
}

function tabela() {
   tempo = tempo + 0.01;
   var str = "";
   str += "<p>";
   str += "<table style='font-size: 1pt; line-height: 1pt'";
   str += "        border='0' cellpadding='1' cellspacing='0'";
   str += "        align='center'>";
   for (x=0;x<=1;x=x+0.02) {
      str += "<tr>";
      for (y=0;y<=1;y=y+0.02) {
         var r = Math.sqrt(x*x+y*y);
         // linha abaixo foi modificada
         var z = 50*Math.cos(4*Math.PI*(r - 2.5*tempo))/Math.sqrt(r);
         if (z>100) z = 100; else if (z<-100) z = -100;
         str += "<td style='background-color:" + zCor(z) + "'>";
         str += "  " + "</td>";
      }
      str += "</tr>";
   }
   str += "</table>";
   str += "</p>";
   document.getElementById("onda").innerHTML = str;
   if (tempo>0.5) clearInterval(tabInterval);
}

var tempo = 0;
var tabInterval = setInterval("tabela()",50);

</script>
```

Resultado:



A primeira providência foi criar uma `<div>` vazia para receber a tabela com a onda renovada a cada passo do temporizador, o que é feito quase no final do script atribuindo-se `str` (que contém a tabela) à propriedade `innerHTML` da divisão, acessada através da chamada a `document.getElementaById()`. O trecho que constrói a tabela (e que contém a física da onda) deve ser colocado dentro de uma função que pode ser passada como parâmetro para o método `setInterval()`.

> Usar uma tabela é uma maneira exótica e ineficiente de construir a onda e lhe dar movimento. No capítulo seguinte, sobre desenhos e animações, veremos uma maneira muito mais razoável e eficiente de fazer isso.

Exercícios

1. Adapte o script do decaimento radioativo para calcular a diferença aproximada (sem levar em conta anos bissextos) em dias entre a sua data de nascimento, digitada nas caixas de texto, e o dia de hoje.

2. Modifique o script do asterisco oscilante para incluir um segundo asterisco fazendo um movimento harmônico simples na direção vertical com controle de velocidade independente.

3. Faça um script que mostre um planeta girando em torno do Sol. Simule o planeta com uma letra "o" e o planeta com um ".".

4. Encontre figuras do Sol, da Terra e da Lua e modifique o script anterior para simular o movimento da Terra em torno do Sol e da Lua em torno da Terra.

5. Faça um script que simule um caça-níqueis com três elementos e cada elemento com 3 opções (três letras ou três figuras, por exemplo), tal que as opções sejam trocadas a cada 100 milissegundos e que o primeiro elemento pare 3 segundos após o acionamento, o segundo após 4 segundos e o terceiro após 5 segundos.

6. O lançamento oblíquo é um movimento bidimensional caracterizado por uma posição de lançamento (x_o, y_o), um ângulo de lançamento θ com a horizontal e uma velocidade de lançamento de módulo v_o. O movimento na direção horizontal (direção x) é retilíneo uniforme (MRU) e o movimento na direção vertical (direção y) é uniformemente variado (MRUV) devido à força da gravidade. A equação de movimento para o lançamento oblíquo pode ser escrita como:

   $$\mathbf{r}(t) = x(t)\,\mathbf{i} + y(t)\,\mathbf{j}$$

   onde

$$x(t) = x_o + v_{xo}\, t$$
$$y(t) = y_o + v_{yo}\, t + \tfrac{1}{2}\, a\, t^2$$

onde x_o e y_o são as coordenadas do ponto de lançamento, $v_{xo} = v_o \cos\theta$, $v_{yo} = v_o \sin\theta$ as componentes da velocidade inicial e a a aceleração, no caso a da gravidade.

A velocidade em qualquer instante é dada por:

$$\mathbf{v}(t) = v_x(t)\, \mathbf{i} + v_y(t)\, \mathbf{j}$$

onde

$$v_x(t) = v_{xo}$$
$$v_y(t) = v_{yo} + a\, t$$

Faça um script que simule com uma letra ou uma figura um lançamento oblíquo em que o usuário escolhe o módulo da velocidade inicial e o ângulo de lançamento.