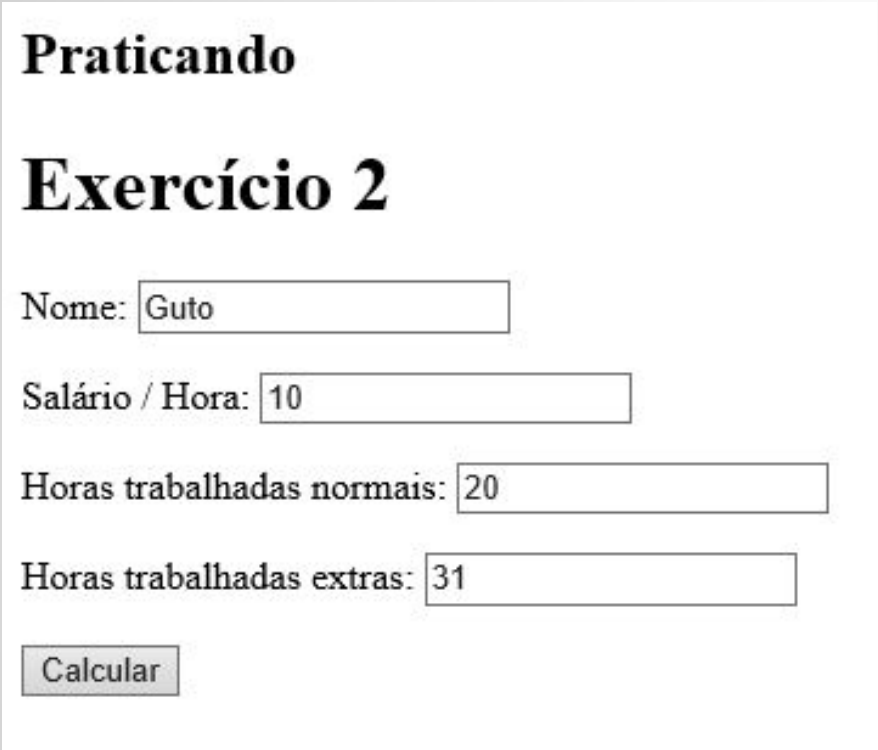

Calcular

SHora * SHNormal + (SHora * HTExtras) * 2

JavaScript e jQuery

Fundamentos e Biblioteca

funções temporais

setTimeout

Em JavaScript, podemos **criar** um ***timer*** para **executar** um trecho de código **após um certo tempo**, ou ainda executar algo de tempos em tempos.

```
// executa a minhaFuncao daqui um segundo
setTimeout(minhaFuncao, 1000);
```

A função **setTimeout** permite que agendemos alguma função para execução no futuro e recebe o nome da função a ser executada e o número de milissegundos a esperar:

```
setTimeout(NomeDaFuncao, Tempo);
```

```
21
22 function mensagem(){
23     alert('teste');
24 }
25
26 setTimeout(mensagem, 5000);
```

```
22 function mensagem(){
23     alert('teste');
24     setTimeout(mensagem, 5000);
25 }
26
27 setTimeout(mensagem, 5000);
```

setInterval

Se for um código **recorrente**, podemos usar o setInterval que **recebe os mesmos argumentos** mas executa a função **indefinidamente** de tempos em tempos:

```
// executa a minhaFuncao de 5 em 5 seg
setInterval(minhaFuncao, 5000);
```

É uma função útil para, por exemplo, implementar um **banner** rotativo, como faremos no exercício a seguir.

```
21
22 function mensagem(){
23     alert('teste');
24 }
25
26 setInterval(mensagem, 5000);
```

clearInterval

As funções temporais **devolvem** um **objeto** que representa o **agendamento** que foi feito. É possível usá-lo para cancelar a execução no futuro.

```
// agenda uma execução qualquer
var timer = setInterval(minhaFuncao, 1000);

// cancela execução
clearInterval(timer);
```

É especialmente interessante para o caso do *interval* que pode ser cancelado de sua execução infinita:

```
21
22 function mensagem(){
23   alert('teste');
24 }
25
26 var timer = setInterval(mensagem, 5000);
27
28 clearInterval(timer);
```

exercício: cronômetro


the End