

Interface com HTML

A forma mais convencional de lidar com entradas de dados do usuário é através do uso de HTML. Usualmente, as aplicações Web utilizam HTML e Javascript para lidar com as interfaces de usuário.

Uma forma simples de fazer uso dessa parceria é criar um arquivo HTML e codificar a(s) função(ões) Javascript dentro desse próprio arquivo. Dessa forma, você está escrevendo um **código HTML** que **executa um script** para processar os dados.



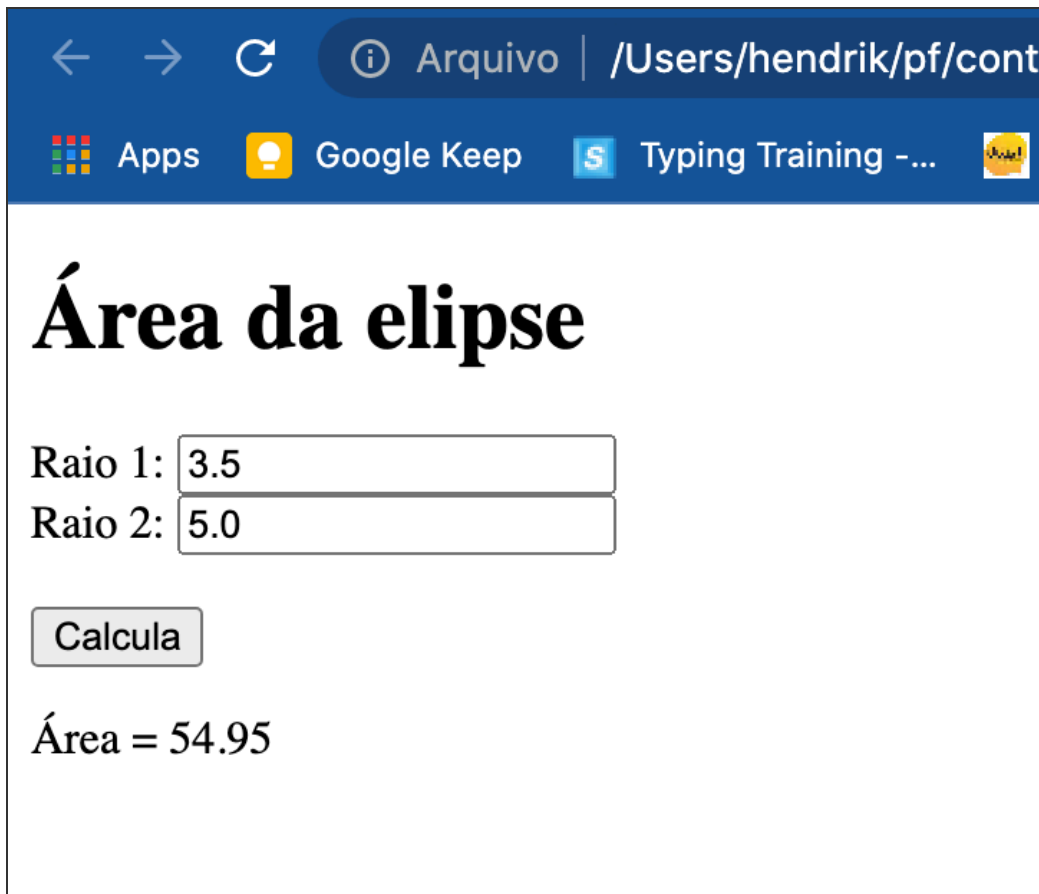
[EXEMPLO] Programa para calcular a área de uma elipse com leitura de dados do usuário via HTML.

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h1>Área da elipse</h1>
6 Raio 1: <input id=input1 type="number"><br>
7 Raio 2: <input id=input2 type="number"><br><br>
8 <button onclick="calcula()">Calcula</button>
9 <p>Área = <span id="output"></span></p>
10
11 <script>
12     const fareasEli = (x, y, pi=3.14) => pi*x*y
13
14     const calcula = () => {
15         const r1 = document.getElementById("input1").value
16         const r2 = document.getElementById("input2").value
17         const resultado = fareasEli(r1,r2)
18         document.getElementById("output").innerHTML = resultado
19     }
20 </script>
21 </body>
22 </html>
```



Nesse caso, é importante conhecer quais elementos HTML melhor se adequam à interface desejada. No código acima, por exemplo, foram utilizados dois elementos `input` numéricos, para receber os valores dos raios (linhas 6 e 7), um `button` (linha 8) que ao ser clicado informa que função Javascript deve ser executada, e um elemento `span` (linha 9) que cria uma área de texto para exibir o resultado.

Quando a função `calcula` é executada, os valores digitados pelo usuário nos dois elementos de `input` HTML são acessados (linhas 15 e 16) e usados na função `areaEli` (linha 17). O resultado então é inserido no espaço de texto que havia sido definido pelo elemento `span` referenciado com o `id` `output` (linha 18).



← → ↻ ⓘ Arquivo | /Users/hendrik/pf/cont

Apps Google Keep Typing Training -...

Área da elipse

Raio 1:

Raio 2:

Área = 54.95

Organizando o código e adicionando estilo

É possível separar o script JS em um arquivo separado e acessá-lo a partir do documento HTML. No código a seguir, o trecho entre as tags `script` do código anterior foi transferido para o arquivo denominado `entradahtml2.js` e essa indicação é feita na linha 4.

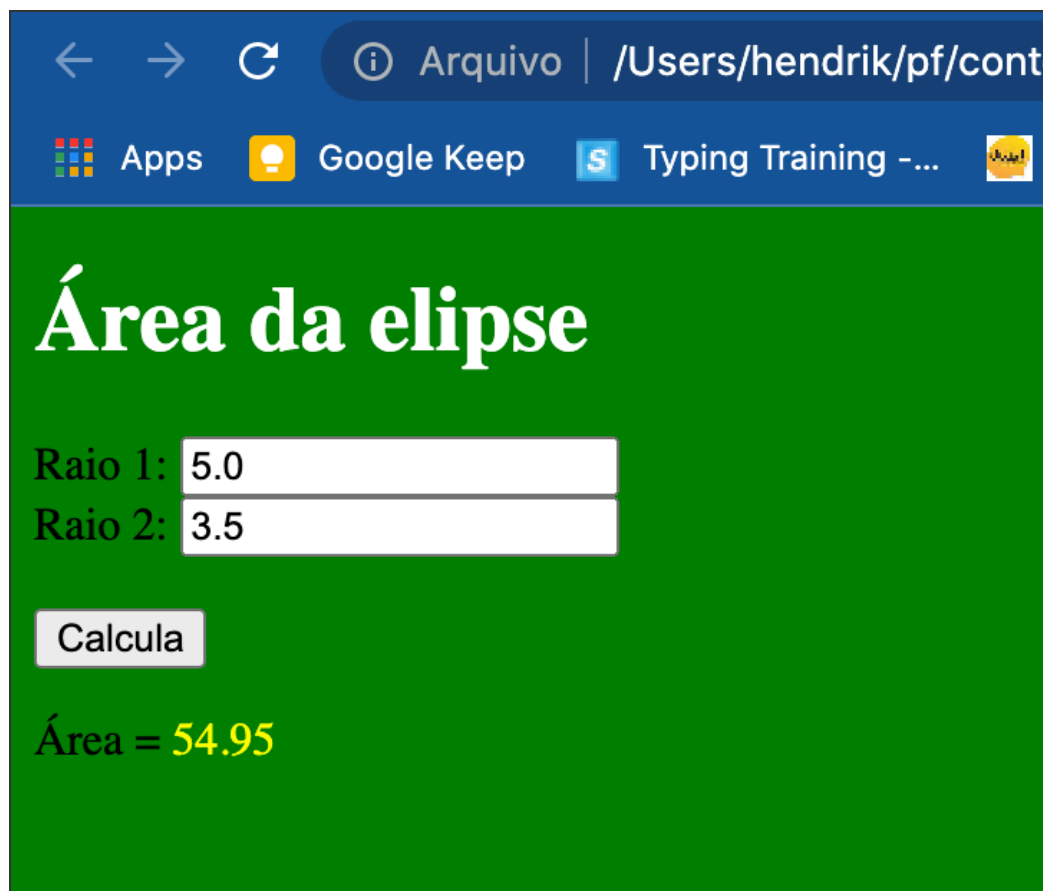
```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script type = "text/javascript" src="entradahtml2.js"></script>
5     <link rel="stylesheet" type="text/css" href="entradahtml2.css"/>
6   </head>
7   <body>
8
9   <h1>Área da elipse</h1>
10  Raio 1: <input id=input1 type="text"><br>
11  Raio 2: <input id=input2 type="text"><br><br>
12  <button onclick="calcula()">Calcula</button>
13  <p>Área = <span id="output"></span></p>
14
15  </body>
16 </html>
```

Arquivo HTML 

Arquivo JS 

Arquivo CSS 

Também é possível utilizar diversos recursos de estilo providos pela linguagem CSS para deixar a interface mais bonitinha. No código, um arquivo de estilo foi criado e referenciado (linha 5). Nesse arquivo, novas cores foram atribuídas à interface.



The screenshot shows a web browser window with a dark blue header bar. The address bar shows the path "/Users/hendrik/pf/cont...". The browser's tab bar includes "Apps", "Google Keep", "Typing Training -...", and a "Calcula" button. The main content area has a green background. At the top, the title "Área da elipse" is displayed in large white font. Below the title, there are two input fields: "Raio 1:" with the value "5.0" and "Raio 2:" with the value "3.5". Below these fields is a button labeled "Calcula". At the bottom, the result "Área = 54.95" is shown in yellow text.

Editando e visualizando o resultado HTML/CSS/JS

Em editores como o JSFiddle, é possível codificar HTML, Javascript e CSS em um ambiente integrado e, assim, visualizar e testar o resultado das alterações quase que instantaneamente.

The screenshot displays the JSFiddle web editor interface. On the left, the 'Fiddle meta' sidebar shows an 'Untitled fiddle' with 'No description', a 'Private fiddle' toggle, and 'Groups' and 'Resources' sections. The main editor is divided into three panes: HTML, CSS, and JavaScript. The HTML pane contains code for an ellipse area calculator with two input fields for radii and a 'Calcula' button. The CSS pane defines styles for the body, h1, and the output area. The JavaScript pane contains a function 'fareaEli' to calculate the area of an ellipse. The rightmost pane shows a live preview of the application, which has a green background and displays the calculated area as 6.28. At the bottom, there is a console area with a 'Clear console' button and a 'Minimize' button.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script type="text/javascript" src="entradahtml2.js"></script>
5     <link rel="stylesheet" type="text/css" href="entradahtml2.css"/>
6   </head>
7   <body>
8     <h1>Área da elipse</h1>
9     Raio 1: <input id="input1" type="number"><br>
10    Raio 2: <input id="input2" type="number"><br><br>
11    <button onclick="calcula()">Calcula</button>
12    <p>Área = <span id="output"></span></p>
13  </body>
14 </html>
```

```
1 body {
2   color: black;
3   background-color: green;
4 }
5 h1 {
6   color: white;
7 }
8 #output {
9   color: yellow;
```

```
1 const fareaEli = (x, y, pi=3.14) => pi*x*y
2
3 const calcula = () => {
4   const r1 = document.getElementById("input1").value
5   const r2 = document.getElementById("input2").value
6   const resultado = fareaEli(r1,r2)
7   document.getElementById("output").innerHTML = resultado
8 }
```

Área da elipse

Raio 1: 1
Raio 2: 2

Calcula

Área = 6.28