

## Aula 11

# Entropia

► **Unidade**

**Segurança digital: utilizando  
matemática para programar  
senhas seguras**

# O que vamos aprender?



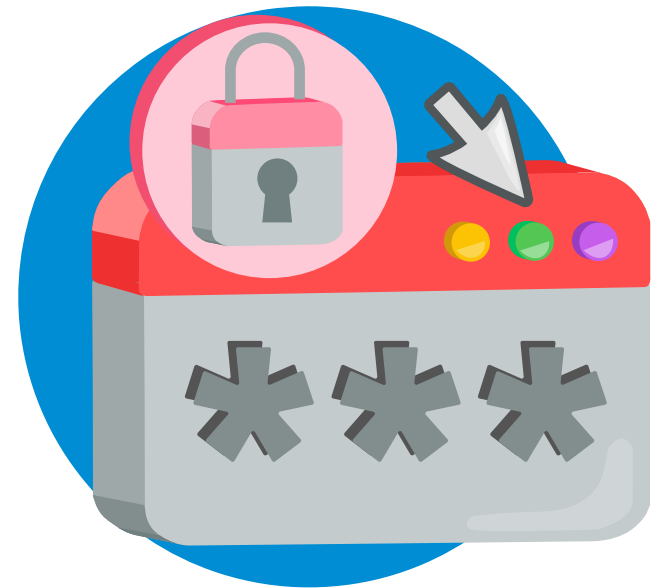
Entender o conceito de entropia de senhas.



Implementar a fórmula da entropia no código JavaScript na função **Math.log2()** ;.



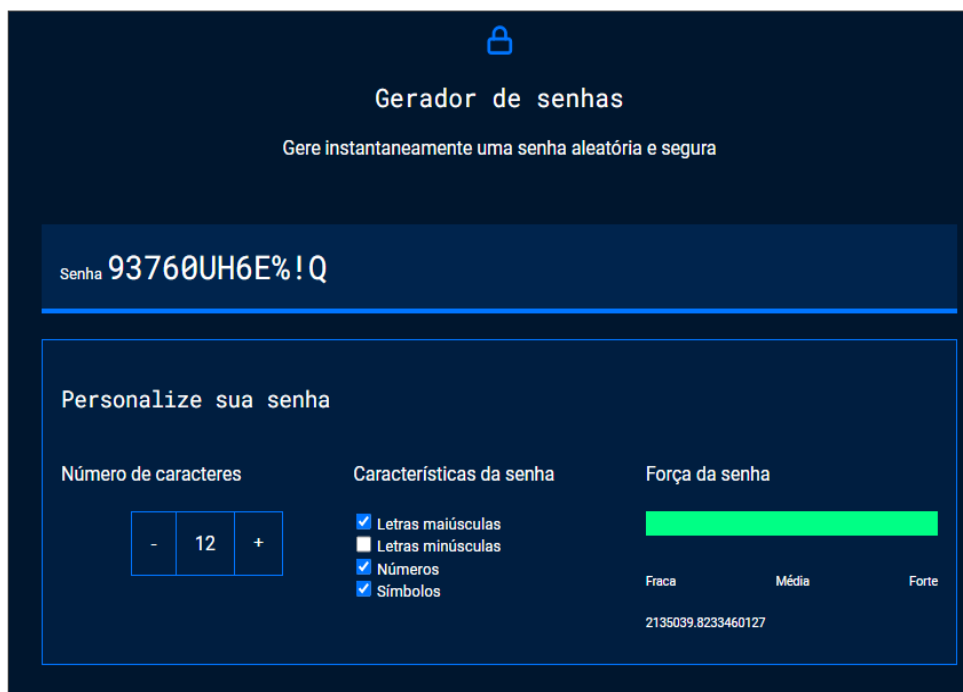
Aumentar a incerteza das senhas para que senhas mais fortes sejam identificadas.



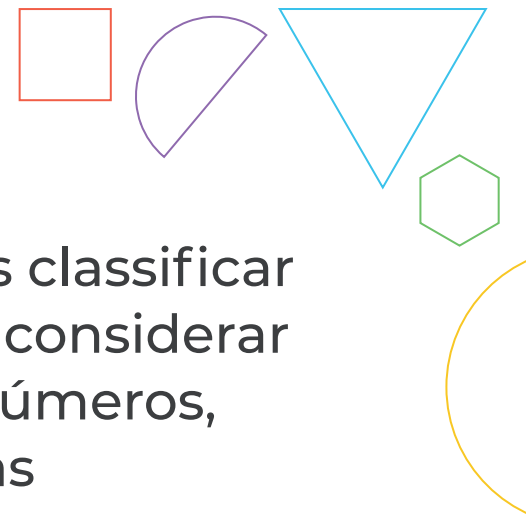
CLIQUE E ACOMPANHE A AULA NA ALURA

# Classificando uma senha

Na aula anterior, implementamos a funcionalidade de medição de força da senha e refinamos o código. Nesta aula, entenderemos o conceito de entropia de senhas, medindo o quão previsível ela é e aumentando a incerteza das senhas para que senhas mais fortes sejam identificadas. Faremos as implementações necessárias utilizando JavaScript e os arquivos já criados em HTML e CSS.



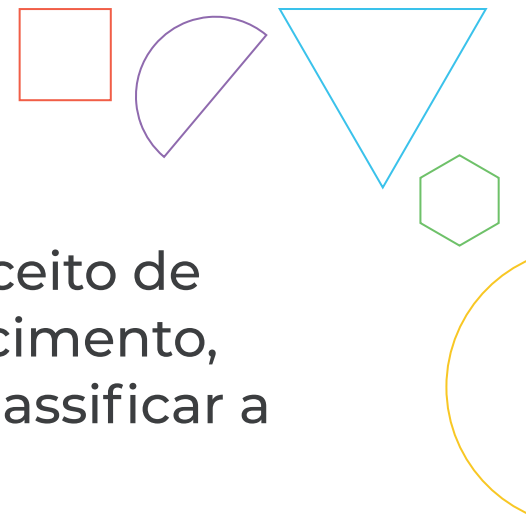
The screenshot shows a web application titled "Gerador de senhas" (Password Generator) with a subtitle "Gere instantaneamente uma senha aleatória e segura" (Generate instantly a random and secure password). The interface is dark-themed. At the top, there is a blue padlock icon. Below the title, a text box displays the generated password "93760UH6E%!Q". Underneath, a section titled "Personalize sua senha" (Customize your password) contains three columns: "Número de caracteres" (Number of characters) with a numeric input set to 12, "Características da senha" (Password characteristics) with four checked checkboxes for "Letras maiúsculas" (Uppercase letters), "Letras minúsculas" (Lowercase letters), "Números" (Numbers), and "Símbolos" (Symbols), and "Força da senha" (Password strength) which features a green progress bar and a strength indicator showing "Frac" (Weak), "Média" (Medium), and "Forte" (Strong). Below the strength indicator, a sample password "2135039.8233460127" is displayed.



Nosso projeto está quase finalizado! Sabemos que podemos classificar nossa senha com base no número de caracteres. Porém, ao considerar uma senha composta por letras minúsculas, maiúsculas e números, o mesmo critério permanece, não levando em conta todas as características de senha de que dispomos.

Portanto, é interessante pensar em uma forma que leve em consideração não apenas o número de caracteres da senha (números e letras), mas também símbolos e outros caracteres para aumentar a força da senha.

Nossa aplicação, até o momento, não consegue atender a essa demanda. Ou seja, estamos fazendo algo diferente do que normalmente é solicitado no momento de criar um cadastro.



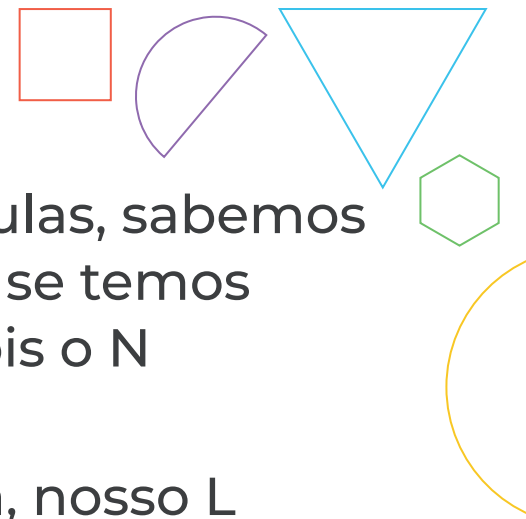
Para resolver essa questão, precisamos compreender o conceito de entropia. Ele pode ser aplicado em diversas áreas do conhecimento, especialmente na termodinâmica, e podemos usá-lo para classificar a força de nossa senha.

A ideia consiste em identificar se a nossa senha é muito previsível ou muito imprevisível. Quanto maior o número da entropia, maior a imprevisibilidade. Ou seja, mais difícil será para uma pessoa ou um computador decifrá-la.

Para entendermos melhor a entropia, existe uma equação matemática que nos auxilia nesse processo:

$$H = L \times \log_2(N)$$

Nesta equação, H é o resultado da entropia. A variável L é o comprimento da senha, ou seja, a quantidade de caracteres da senha. E  $\log_2(N)$  é o tamanho do conjunto de caracteres. Vale lembrar que N, nesse caso, é o número de possibilidades que existem no nosso alfabeto.



Por exemplo: se temos caracteres apenas em letras minúsculas, sabemos que existe um determinado conjunto de caracteres. Porém, se temos letras minúsculas e maiúsculas, nosso alfabeto aumenta, pois o N aumenta e suas possibilidades também.

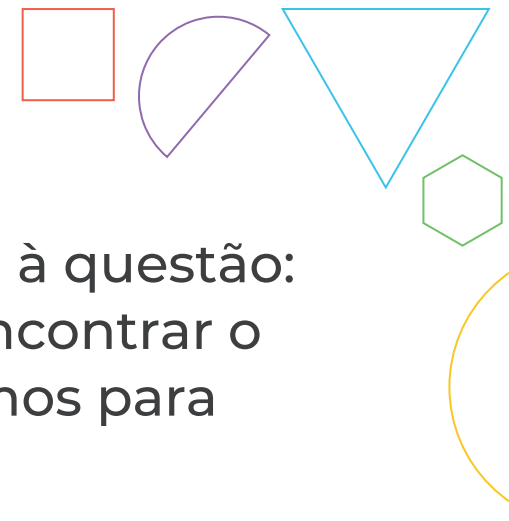
Na prática, ao rodar a aplicação, temos 12 caracteres, ou seja, nosso L é igual a 12, pois consideramos o conjunto do alfabeto somente como as letras maiúsculas por padrão. Temos, portanto, de A a Z, todas em maiúsculas. Logo, nosso alfabeto resulta nesse valor e, ao calcularmos essa equação, obteremos um resultado de 56,4 bits. Observe:

$$H = 12 \times \log_2(26)$$

$$H = 56,4 \text{ bits}$$

O logaritmo é o inverso da exponencial. Ou seja, se temos, por exemplo, 2 elevado a b igual a 8, precisamos saber quantas vezes devemos multiplicar o 2 para obter 8. Sabemos que 2 vezes 2 vezes 2 é igual a 8, ou seja, o valor de b é 3.

Portanto, sabemos que o log vai calcular o número de possibilidades, isto é, quantas vezes precisamos repetir o 2 até encontrar o resultado final. Logo, o log faz exatamente essa operação inversa.

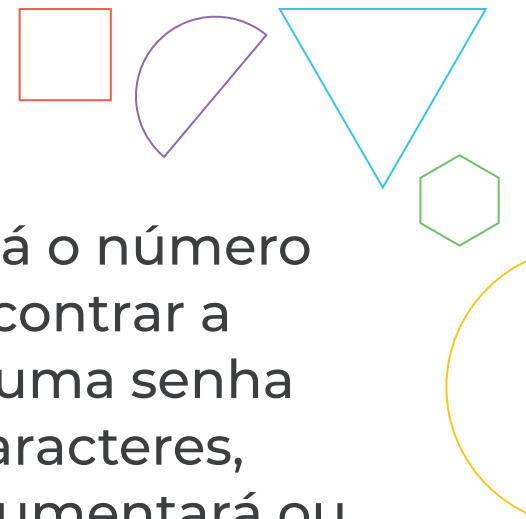


Consequentemente, podemos pensar que o log responderá à questão: quantos números binários precisamos ter para conseguir encontrar o resultado da senha? De quantas tentativas de 0 e 1 precisamos para conseguir encontrar a solução?

Se sabemos que 2 elevado a 3 é igual a 8, isso indica que temos 3 bits, pois o logaritmo ofereceu essa informação e podemos testar todas as possibilidades. Assim, sabemos que existem 8 possibilidades até encontrar todas elas.

Então, vamos supor que a senha seja uma delas. Para descobrir essa senha, o computador pode encontrá-la de imediato na primeira tentativa, ou ele terá que testar todas as possibilidades até completar as 8. Isso é possível caso a entropia seja igual a 3.

Entretanto, nossa entropia não é igual a 3. No nosso caso, a nossa entropia é 56,4, e 2 elevado a 56,4 resultará em um número gigante, de aproximadamente 95 quatrilhões de tentativas. **Ou seja, quanto tempo o computador levará para testar todas essas possibilidades?**



Observe que quanto maior for o valor da entropia, maior será o número de possibilidades que o computador precisará testar até encontrar a combinação da nossa senha, e é justamente isso que torna uma senha mais segura. Se aumentarmos o alfabeto ou o número de caracteres, esse valor também será influenciado e, por consequência, aumentará ou diminuirá. Agora, vamos implementar esse processo na prática.

Começaremos programando no arquivo *main.js*, na função **classificaSenha()**. A ideia não é mais usar o conceito de **tamanhoSenha**, mas sim considerar todo o contexto usando a entropia. Então, nessa função, podemos criar uma variável chamada entropia, que receberá um valor. Ela receberá **tamanhoSenha**, que é justamente o valor que estamos testando, multiplicado por **Math.log2()**, correspondente ao tamanho do alfabeto.

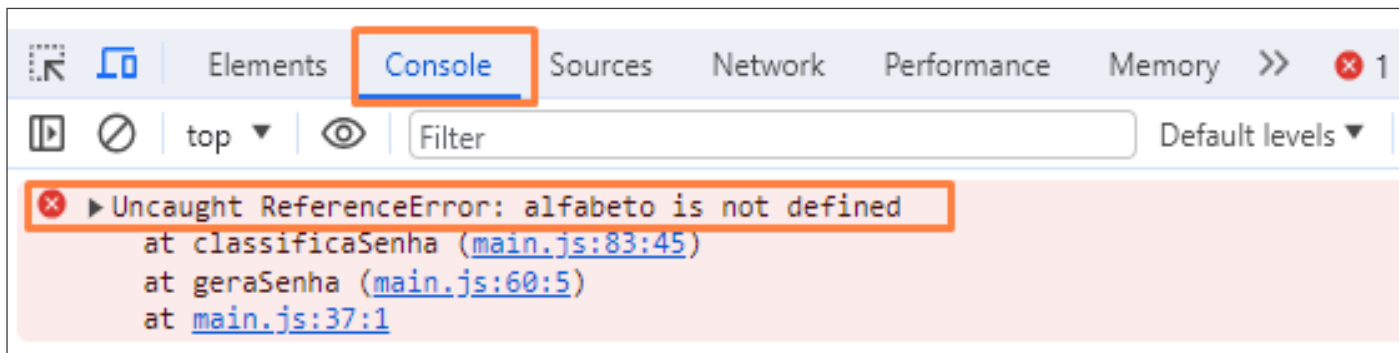
No nosso código já existe **alfabeto.length**, que é exatamente a informação de que precisamos. Portanto, passamos **alfabeto.length** para **Math.log2()**.



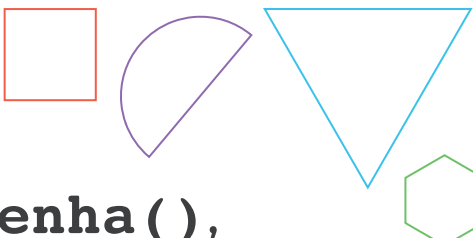
Ainda, para exibir essa informação, vamos adicionar um **console.log()** do valor da entropia na linha abaixo. Calcularemos seu valor primeiro, antes de fazer as alterações na força da senha. Seu código ficará como mostrado a seguir:

```
function classificaSenha() {  
  let entropia = tamanhoSenha * Math.log2(alfabeto.length);  
  console.log(entropia);  
}
```

Feito isso, vamos acessar nossa página e abrir o console com a tecla de atalho *F12*.



Notamos que há um erro indicando que o alfabeto não está definido.



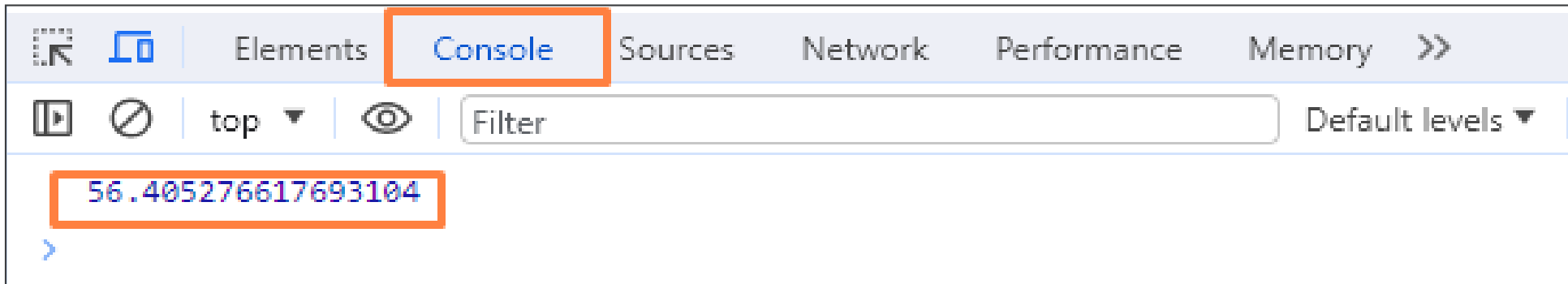
Se analisarmos o código, percebemos que, na função **geraSenha()**, declaramos a variável **alfabeto**. Portanto, ela está alocada apenas dentro de uma função. Assim, não conseguimos acessá-la por meio de outra função, a menos que seja passada a referência do **alfabeto.length** para um parâmetro de entrada dentro da função **classificaSenha()**. Ou seja, além da função **geraSenha()**, o alfabeto também existe na função **classificaSenha()**, então, passamos essa referência, que é um parâmetro de entrada. Observe:

```
function geraSenha() {  
    let alfabeto = '';  
    // código omitido  
    classificaSenha(alfabeto.length);  
}
```

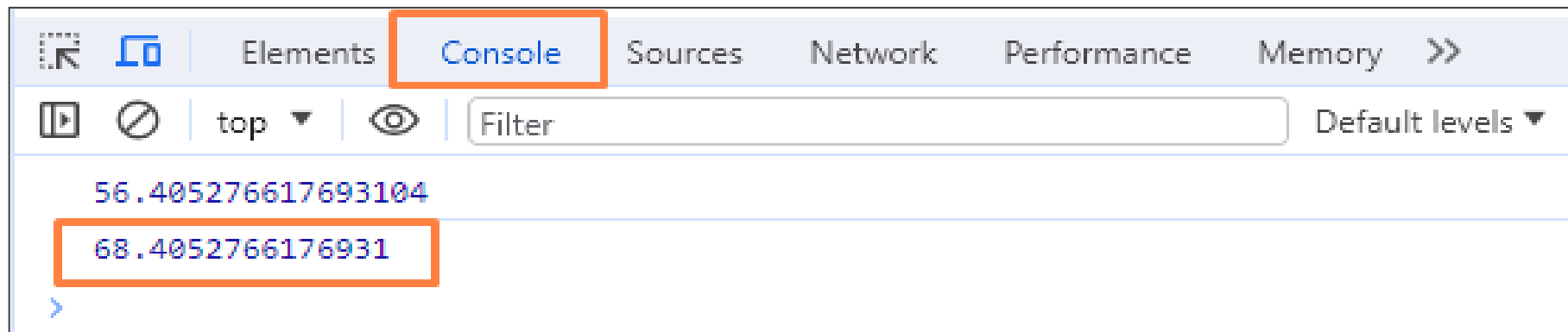
Agora que temos esse parâmetro de entrada, podemos entrar na função **classificaSenha()** e adicionar um valor, alterando o nome dessa variável para, por exemplo, **tamanhoAlfabeto**.

```
function classificaSenha(tamanhoAlfabeto) {  
    // código omitido  
}
```

Esses dois valores, `alfabeto.lenght` e `tamanhoAlfabeto`, serão os mesmos, ou seja, em vez de `Math.log2()` de `alfabeto.length`, teremos `Math.log2()` de `tamanhoAlfabeto`, porque será esse o nome da variável que a função `classificaSenha()` recebe. Agora, podemos voltar ao navegador:

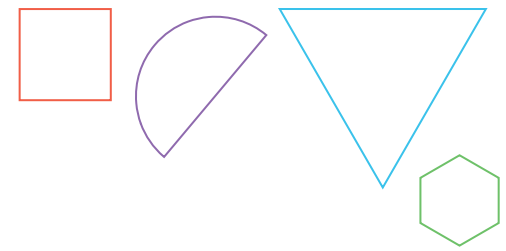


Podemos perceber que o 56,4, que é o valor da nossa entropia, foi calculado corretamente. Na sequência, queremos que, ao clicarmos no checkbox *Letras minúsculas*, o valor da entropia aumente, porque a incerteza aumentará e a aleatoriedade da senha também ficará maior. Verificando o console novamente, poderemos observar que, após selecionarmos o checkbox, o valor da entropia vai de 56 vai para 68, ou seja, o código está funcionando corretamente:





Observe que ao selecionar os checkboxes *Números* e *Símbolos*, o valor também aumentará. Do mesmo modo, ao diminuirmos o número de caracteres, esperamos que o valor da entropia diminua. Agora, conseguimos manipular os eventos e notar que a entropia muda a cada interação. Então, além da dinâmica do tamanho da senha, também estamos considerando as letras maiúsculas, minúsculas, os números e símbolos.



Se voltarmos ao código, nossa ideia é que a variável **entropia** passe a ser utilizada nos parâmetros das condicionais da função **classificaSenha()**. Então, podemos substituir **if (tamanhoSenha > 11)** por **if (entropia > 11)**, **if (entropia > 5 && tamanhoSenha < 12)** por **if (entropia > 5 && entropia < 12)**, e **if (tamanhoSenha <= 5)** por **if (entropia <= 5)**.

Observe:

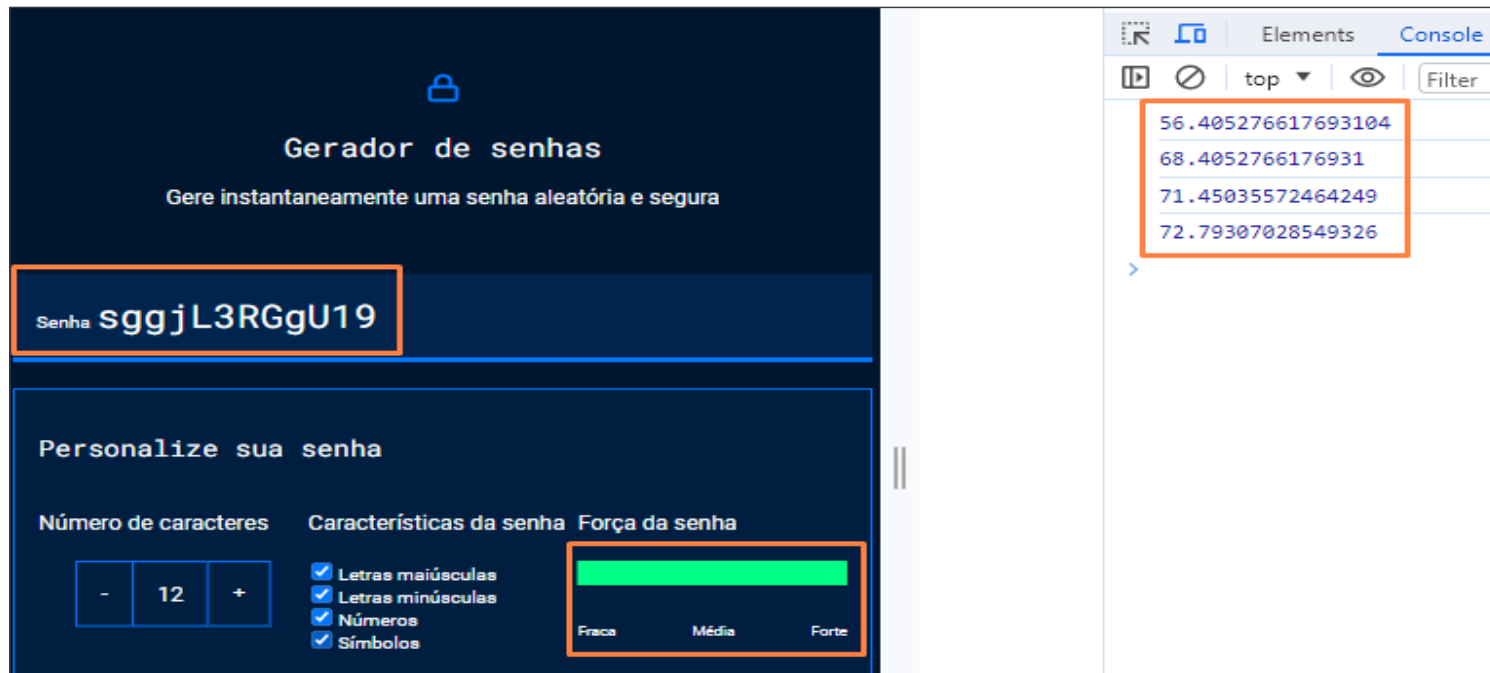
```
function classificaSenha(tamanhoAlfabeto){
  let entropia = tamanhoSenha * Math.log2(tamanhoAlfabeto);
  if (entropia > 11){
    forcaSenha.classList.add('forte');
  } else if (entropia > 5 && entropia < 12 ) {
    forcaSenha.classList.add('media');
  } else if (entropia <= 5){
    forcaSenha.classList.add('fraca');
  }
  console.log(entropia);
}
```



Porém, sabemos que 11 é um valor muito pequeno para a entropia, com base nos nossos cálculos. Nesse caso, precisamos pensar em valores diferentes. Por exemplo: se por padrão queremos que a senha seja forte, a ideia é que 56,4 já seja um valor de entrada, caracterizando uma senha forte. Assim, podemos definir que, se a entropia for maior que 57, teremos uma senha forte. Se for uma entropia menor que 57 e maior que 35, por exemplo, ela será considerada uma entropia média. Por fim, se for menor que 35, é uma senha fraca. Observe como seu código ficará:

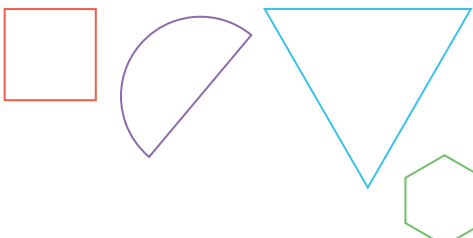
```
function classificaSenha(tamanhoAlfabeto){  
  let entropia = tamanhoSenha * Math.log2(tamanhoAlfabeto);  
  if (entropia > 57){  
    forcaSenha.classList.add('forte');  
  } else if (entropia > 35 && entropia < 57 ) {  
    forcaSenha.classList.add('media');  
    forcaSenha.classList.add('fraca')  
  }  
  console.log(entropia);  
}
```

Ao rodar o código, por padrão, o gerador já começa com uma senha média. Quando aumentamos o número de caracteres, a força da senha também aumenta. Da mesma forma, ao reduzir o número de caracteres, a força da senha diminui. Podemos testar várias possibilidades. Observe:



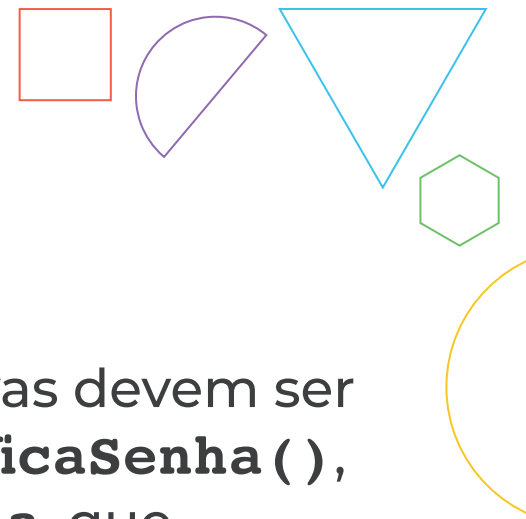
Perceba que uma senha fraca pode ter 4 caracteres com todos os tipos de caracteres. Porém, se a senha possuir 6 caracteres, entre letras maiúsculas e minúsculas, percebemos que mudamos um pouco a dinâmica. Ou seja, não estamos considerando apenas o número de caracteres, mas todas as características de que dispomos, como desejávamos.





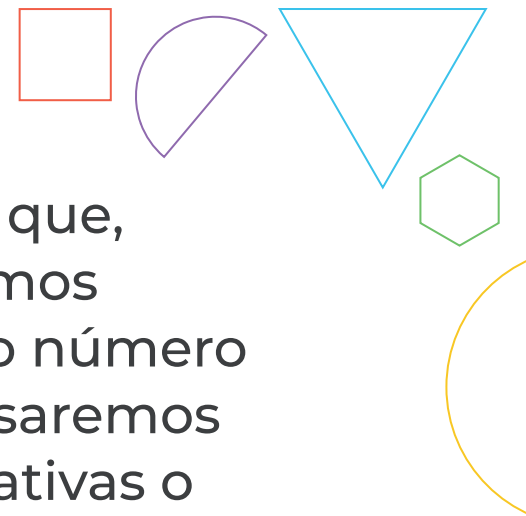
Agora, podemos de fato entender se a nossa senha é forte ou não. A depender, talvez um caractere a mais possa aumentar em semanas o tempo que um computador levará para completar os testes e decifrar nossa senha. Assim, vamos fazer esse cálculo e mostrá-lo no arquivo *index.html*. Ao fim do código, após `<div>` de **parametro-senha-texto**, adicionaremos um parágrafo `<p>` de classe **entropia**. Observe:

```
<div class="parametro-senha">  
  <h4 class="parametro-senha__titulo">Força da senha</h4>  
  <!--Resto do código-->  
  <p class="entropia"></p>  
</div>
```



Queremos escrever em cima de **entropia** quantas tentativas devem ser feitas. Então, no arquivo *main.js*, ao final da função **classificaSenha()**, podemos declarar uma constante chamada **valorEntropia**, que receberá **document.querySelector( '.entropia' )**. Assim, conseguimos acessar esse elemento. Em seguida, podemos mostrar o valor de entropia usando **valorEntropia.textContent**. Observe:

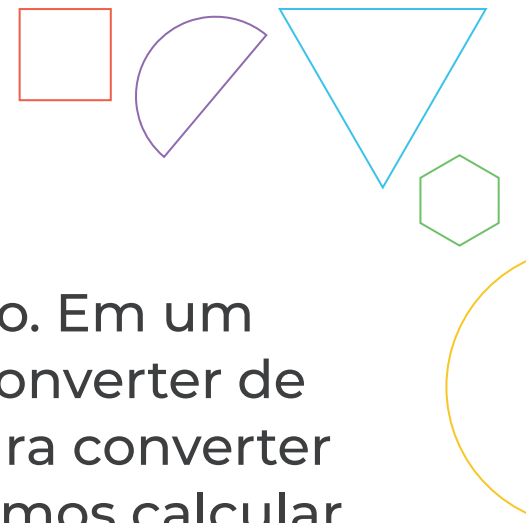
```
}  
const valorEntropia = document.querySelector( '.entropia' );  
valorEntropia.textContent = entropia;  
console.log(entropia);  
}
```



Se visualizarmos a aplicação no navegador, identificaremos que, em vez de mostrar a informação no **console.log()**, podemos mostrá-la para a pessoa usuária. Assim, vamos arredondar o número retornado para conseguirmos entender melhor. Para isso, usaremos **Math.floor(entropia)**. A ideia é pensarmos quantas tentativas o computador precisa realizar para decifrar a senha. Anteriormente, colocamos um 2 elevando a **entropia**. Para elevar a **entropia**, adicionamos dois asteriscos (**2\*\***). Observe:

```
valorEntropia.textContent = 2**Math.floor(entropia);
```

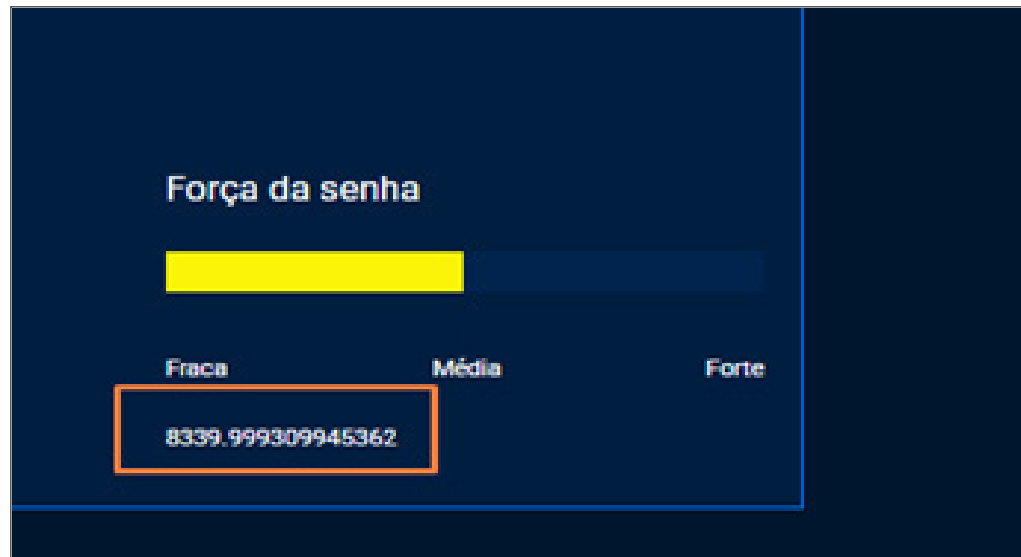
Vamos explorar o número que será exibido na tela. Ele representará a quantidade de tentativas que o computador precisa fazer. É importante notar que esse número de tentativas deverá ser muito grande. Desse modo, para escrever números grandes, como milhões, por exemplo, em vez de usar seis zeros em sequência, podemos adicionar um e6. Isso trará maior simplicidade ao nosso código e evitará repetições desnecessárias.



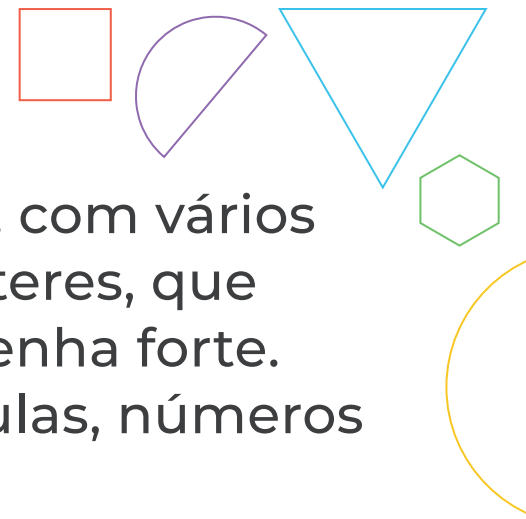
Dessa forma, em um segundo, podemos encontrar a solução. Em um minuto, podemos multiplicar o valor por 60. Se quisermos converter de minutos para horas, basta multiplicamos por 60 de novo. Para converter de horas para dias, multiplicamos por 24. Assim, conseguiremos calcular quantos dias um computador levaria para decifrar a senha, a depender do número de possibilidades. Observe:

```
valorEntropia.textContent = 2**Math.floor(entropia)/  
(100e6*60*60*24);
```

Verificando a aplicação, veremos que o resultado do cálculo está sendo exibido para a pessoa usuária em nossa página:



Assim, para uma senha média, são necessários cerca de 8.339 dias para que ela seja quebrada. Isso é muito tempo. Se considerarmos uma senha mais longa, esse número se tornará ainda maior. À medida que acrescentamos caracteres, a quantidade de tentativas aumentará e, conseqüentemente, o tempo necessário para decifrar a senha também.



Se pensarmos matematicamente, uma senha muito longa, com vários caracteres, será mais forte do que uma senha com 8 caracteres, que muitas vezes encontramos na internet como sendo uma senha forte. Ela geralmente é composta por letras maiúsculas, minúsculas, números e símbolos.

Porém, se criarmos uma senha com 20 caracteres usando apenas letras minúsculas ou maiúsculas, ou seja, somente um conjunto de alfabeto, perceberemos que a entropia dessa senha será muito maior. Dessa forma, se consolidarmos uma senha em uma frase, será muito mais difícil para um computador quebrá-la.

Agora que sabemos como calcular a entropia e conseguimos classificar nossa senha, na próxima aula entenderemos as diferenças entre o que fizemos e o que outros sites fazem para conseguirem gerar senhas.

Até breve!

## ► Desafio

Nesta aula, aprendemos sobre entropia e como utilizá-la para definir a força de uma senha por meio de uma função. Seu desafio será pesquisar e compreender como um computador pode testar um milhão de senhas por segundo. Será que existem processadores ou supercomputadores que testem bilhões de vezes por segundo? Se sim, qual ou quais são? Sabendo dessas informações, busque aumentar ainda mais a entropia de sua senha.



CLIQUE **AQUI** PARA AVALIAR ESTE MATERIAL