Caso a tag <script> </script> seja colocada dentro da tag <head>, ele será carregado antes do seu HTML, por isso é necessário SEMPRE colocar todo o script dentro do seguinte comando:

```
window.onload = function () {
}
```

Caso a tag **<script> </script>** seja colocada dentro da tag **<body>,** ele será carregado ao final do carregamento da página, ou seja, nao atrapalha a renderização da página como um todo.

Além disso, o script, como um todo, assim como um CSS, pode ser chamado por um arquivo externo já existente:

```
<script src=”exercicio1.js”> </script>
```

SELECIONANDO ELEMENTO POR ID:

Para selecionar um elemento da página HTML pelo ID usando java script, se usa o seguinte comando:

```
document.getElementById(“nome do id do elemento”)
```

Para selecionar o que está dentro da ID:

```
document.getElementById(“nome do id do elemento”).innerHTML
```

MAIS FORMAS:

```html
<body>
  <h2 id="pageTitle">Título</h2>
  <p id="paragraph">Dê uma cor para este parágrafo!</p>
  <h4 id="subtitle">Subtítulo</h4>
  <p id="secondParagraph">Segundo parágrafo!</p>

  <script>
    var paragraph = document.getElementById("paragraph");
    paragraph.style.color = "red";
    paragraph.innerHTML = "Lançado recentemente, esse filme já está dando o que falar!"

    let subtitle = document.getElementById("subtitle");
    subtitle.innerHTML = "A seguir, mais informações:"

    let secondParagraph = document.getElementById("secondParagraph");
    secondParagraph.style.color = "red";

    let pageTitle = document.getElementById("pageTitle");
    pageTitle.innerHTML = "O Poço - Original Netflix"
  </script>
```

SELECIONANDO ELEMENTO POR CLASSE:

document.getElementsByClassName(“nome da classe”)[numero do indice]

```html
<body>

  <div class="urso">ruuuu</div>
  <div class="urso">ruuuuu11</div>
  <div class="shrek">faz o urruuu</div>


  <script>

    let urso = document.getElementsByClassName("urso")[1]
    // Aqui ele pegou a class urso que faz ruuu111, pois o [1] corresponde ao segundo índice da classe urso!
    // se eu usasse o [0], ele selecionaria o ruuuu, pois é o primeiro índice da classe urso.
  </script>
</body>
```

SELECIONANDO ELEMENTO POR TAG:

document.getElementsByTagName(“nome da tag”)[numero do indice]

SELECIONANDO PELO QUERY:

O query seleciona apenas O **PRIMEIRO** elemento indicado. Ou seja, se eu tiver vários <p> no html e usar o querySelector(“p”), ele só selecionará o PRIMEIRO <p>. O mesmo vale pra IDs e Classes!!!

Para selecionar por tag: **document.querySelector(“nome da tag”)**

Para selecionar por id: **document.querySelector(“#nomedoid”)**

Para selecionar por classe: **document.querySelector(“.nomedaclasse”)**

Existe também a possibiliade de selecionar TODOS os elementos pelo query, o famoso query selector all:

Para selecionar por tag: **document.querySelectorAll(“nome da tag”)**

Para selecionar por id: **document.querySelectorAll(“#nomedoid”)**

Para selecionar por classe: **document.querySelectorAll(“.nomedaclasse”)**

ACESSANDO UM ELEMENTO PAI DE UM ELEMENTO:

Vamos supor que no seu código, existem vários elementos dentro de outros elementos e voce precisa descobrir qual é o elemento pai daquele elemento. O que você faz? Voce utiliza o parentNode:

```
console.log(
document.querySelector(“exeplo do elemento”).parentNode
)
```

E esse comando retornará no console o elemento pai do “exemplo do elemento” que você usou. Lembrando que este exemplo pode ser tag, classe, id, etc. Caso queira saber qual é o elemento pai do elemento pai do exemplo, é só prosseguir com o código:

```
console.log(
document.querySelector(“exeplo do elemento”).parentNode.parentNode
)
```

Outros recursos:

- parentNode: retorna o elemento pai.
- childNodes: retorna um array com todos os elementos filhos
- firstChild: retorna o primeiro filho
- lastChild: retorna o último filho
- nextSibling: retorna o próximo nó.
- previousSibling: retorna o nó anterior.
- nextElementSibling: retorna o próximo elemento.
- previousElementSibling: retorna o elemento anterior.