Comandos e recursos fundamentais JavaScript

Variáveis e Tipos de Dados:
- `var`, `let`, `const`: Declaração de variáveis com diferentes escopos e características de atribuição.
- `Number`: Tipo de dado numérico que pode representar números inteiros ou de ponto flutuante.
- `String`: Tipo de dado que representa texto.
- `Boolean`: Tipo de dado que representa valores verdadeiros (`true`) ou falsos (`false`).
- `Object`: Tipo de dado que pode conter propriedades e métodos.
- `Array`: Tipo de dado que armazena uma coleção ordenada de valores.
- `null` e `undefined`: Valores especiais que representam a ausência de valor.

Operadores:
- Aritméticos: `+`, `-`, `*`, `/`, `%` (módulo) realizam operações matemáticas.
- Comparação: `==`, `===`, `!=`, `!==`, `>`, `<`, `>=`, `<=` comparam valores e retornam um valor booleano.
- Lógicos: `&&` (E), `||` (OU), `!` (NÃO) realizam operações lógicas em valores booleanos.
- Atribuição: `=`, `+=`, `-=` etc. atribuem valores a variáveis.
- Ternário: Operador ternário `? :` permite uma forma concisa de expressar uma condição.

Estruturas de Controle:
- `if`, `else if`, `else`: Estruturas condicionais para executar código com base em condições.
- `switch`: Estrutura de seleção múltipla para escolher entre várias opções.
- `for`, `while`, `do...while`: Estruturas de repetição para controlar o fluxo do programa.

Funções:
- `function`: Declaração de função para definir blocos de código reutilizáveis.
- `return`: Utilizado para retornar um valor de uma função.
- Arrow functions: Funções de seta (`() => {}`) fornecem uma sintaxe mais concisa para definir funções.

Arrays e Manipulação de Strings:
- Métodos como `push()`, `pop()`, `shift()`, `unshift()`, `join()`, `split()`, `slice()`, `splice()` para manipular arrays e strings.

Objetos e Propriedades:
- Objetos são coleções de pares chave-valor.
- Acesso a propriedades com notação de ponto (`objeto.propriedade`) ou notação de colchetes (`objeto['propriedade']`).

Escopo e Closure:
- `var` tem escopo de função; `let` e `const` têm escopo de bloco.
- Closures permitem que funções mantenham acesso a variáveis de escopo externo.

Promises e Assincronia:
- `Promise`: Um objeto que representa um valor que pode estar disponível agora, no futuro ou nunca.
- `async` e `await`: Palavras-chave usadas para lidar com código assíncrono de forma mais síncrona e legível.

Tratamento de Exceções:
- `try`, `catch`, `throw`, `finally`: Usados para tratar exceções e erros no código.

Console:
- `console.log()`: Comando para imprimir mensagens de registro no console.
- `console.error()`: Comando para imprimir mensagens de erro no console.
- `console.warn()`: Comando para imprimir mensagens de aviso no console.

Eventos:
- `addEventListener()`: Método usado para anexar funções a eventos HTML, permitindo interatividade com o usuário.

Manipulação do DOM (Document Object Model):
- `getElementById()`, `querySelector()`: Métodos para selecionar elementos HTML no documento.
- `innerHTML`, `textContent`: Propriedades usadas para manipular conteúdo HTML de elementos.
- `createElement()`, `appendChild()`: Métodos para criar e adicionar elementos HTML ao DOM.

JSON (JavaScript Object Notation):
- `JSON.stringify()`: Método para serializar objetos JavaScript em formato JSON.
- `JSON.parse()`: Método para desserializar dados JSON em objetos JavaScript.

Módulos e Importações (ES6+):
- `import`, `export`: Recursos para modularizar o código e separá-lo em diferentes arquivos.

Temporizadores:
- `setTimeout()`, `setInterval()`: Funções para executar código após um atraso ou em intervalos regulares.

Expressões Regulares (Regex):
- `RegExp()`: Objeto para criar expressões regulares.
- Métodos como `test()`, `match()`, `replace()`: Usados para trabalhar com expressões regulares.

Outros Recursos:
- `typeof`: Operador para verificar o tipo de dado de uma variável.
- `instanceof`: Operador para verificar se um objeto é uma instância de uma classe ou construtor.
- `Math`: Objeto para realizar operações matemáticas.
- `Date`: Objeto para trabalhar com datas e horas.
- `localStorage`, `sessionStorage`: Mecanismos de armazenamento local no navegador.
- `fetch()`: API para fazer requisições HTTP.

Palavras-chave JavaScript

1. Array: Usada para criar e manipular arrays (listas de valores).
2. Async/await: Usada para trabalhar com código assíncrono de forma mais síncrona e legível.
3. Callback: Usada para passar uma função como argumento para outra função, geralmente para controle assíncrono.
4. Class: Usada para definir classes e criar objetos a partir delas.
5. Constructor: Usada para criar e inicializar objetos de uma classe.
6. DOM (Document Object Model): Usada para interagir com elementos HTML e CSS em uma página web.
7. Event: Usada para lidar com eventos, como cliques do mouse e pressionamentos de tecla.
8. Function: Usada para definir funções reutilizáveis.
9. JSON (JavaScript Object Notation): Usada para serializar e desserializar dados no formato JSON.
10. Local Storage: Usada para armazenar dados no navegador do usuário.
11. Method: Usada para definir funções associadas a objetos.
12. Object: Usada para criar objetos que podem conter propriedades e métodos.
13. Promise: Usada para representar um valor que pode estar disponível agora, no futuro ou nunca.
14. Prototype: Usada para criar herança em JavaScript.
15. Regex (Expressão Regular): Usada para buscar e manipular padrões de texto.
16. SetTimeout/SetInterval: Usada para criar atrasos ou executar funções repetidamente em intervalos.
17. String: Usada para manipular texto.
18. This: Usada para se referir ao objeto atual em um contexto.
19. Try/Catch: Usada para tratar exceções e erros em código.
20. Typeof: Usada para verificar o tipo de dado de uma variável.
21. Var/Let/Const: Usadas para declarar variáveis.
22. While/For/Do...While: Usadas para criar loops.
23. Window: Representa a janela do navegador e oferece acesso a informações sobre a janela, como localização e histórico de navegação.
24. XMLHttpRequest: Usada para fazer solicitações HTTP assíncronas, permitindo a atualização de conteúdo da página sem recarregá-la (embora seja menos comum agora, com fetch() sendo mais preferível).
25. Map/Filter/Reduce/ForEach: Métodos de arrays usados para manipulação funcional de arrays.
26. Arrow Function: Usada para definir funções de seta com uma sintaxe mais concisa.
27. Await: Usada junto com async para esperar que uma Promise seja resolvida.
28. Export/Import: Usadas para modularizar código em vários arquivos (introduzido no ECMAScript 2015 - ES6).
29. Null/Undefined: Usadas para representar a ausência de valor.
30. Spread Operator (...): Usado para espalhar elementos de um array ou propriedades de um objeto.
31. Destructuring Assignment: Usada para extrair valores de objetos e arrays de forma concisa.
32. Symbol: Usada para criar valores únicos que podem ser usados como chaves de propriedades de objetos.
33. Proxy: Usada para interceptar operações em objetos, permitindo a personalização do comportamento do objeto.
34. WeakMap/WeakSet: Estruturas de dados que permitem que os valores sejam coletados automaticamente quando não há mais referências a eles.
35. BigInt: Usada para representar números inteiros arbitrariamente grandes.
36. Optional Chaining (?.): Usado para evitar erros quando se tenta acessar propriedades de objetos aninhados que podem ser undefined ou null.