```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>querySelector and querySelector example 1</title>
  <script src="action.js"></script>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <button onclick="displayListOfCheckedItems();">Show Checked items</button>
  <button onclick="reset();">Reset list</button>
  <br>
<ul id="fruits">
   <li>
     <input type="checkbox" name="fruit" value="apples">Apples
   </li>
   <li>
     <input type="checkbox" name="fruit" value="oranges">
     Oranges
   </li>
   <li>
     <input type="checkbox" name="fruit" value="bananas">
     Bananas
   </li>
   <li>
     <input type="checkbox" name="fruit" value="grapes">
     Grapes
   </li>
</ul>
</body>
```

```
</html>


.checked {
   border: 2px dashed #000;
   background-color: green;
   color:yellow;
}
```

Este código combina **HTML** y **JavaScript** para crear una página interactiva en la que el usuario puede seleccionar frutas de una lista, ver las seleccionadas, y restablecer las selecciones. Vamos a analizarlo por partes:

HTML

El HTML define la estructura de la página:

1. **Estructura básica de la página:**
2. <!DOCTYPE html>
3. <html lang="en">

Define que el documento está en inglés (lang="en") y es de tipo HTML5.

4. **Encabezado (<head>):**
5. <meta charset="utf-8">
6. <meta name="viewport" content="width=device-width">
7. <title>querySelector and querySelector example 1</title>
8. <script src="action.js"></script>
9. <link rel="stylesheet" href="style.css">
    - **Codificación**: utf-8 permite caracteres especiales.
    - **Responsive design**: viewport optimiza para diferentes dispositivos.
    - Incluye un archivo JavaScript (action.js) y una hoja de estilos (style.css), aunque su contenido no está en el código mostrado.
10. **Cuerpo (<body>):** Define los elementos interactivos:
    - Dos botones:

- <button onclick="displayListOfCheckedItems();">Show Checked items</button>
- <button onclick="reset();">Reset list</button>
    - El primer botón llama a la función displayListOfCheckedItems() para mostrar los ítems seleccionados.
    - El segundo llama a reset() para reiniciar la lista.
- Una lista no ordenada (<ul>) con id="fruits" que contiene ítems (<li>), cada uno con un checkbox y el nombre de una fruta:
- <ul id="fruits">
-
- <input type="checkbox" name="fruit" value="apples">Apples
-
- ...
-

JavaScript

El JavaScript define dos funciones principales:

1. **displayListOfCheckedItems**:
2. function displayListOfCheckedItems() {
3. var listOfSelectedValues = "";
4. var list = document.querySelectorAll("#fruits input:checked");
5. list.forEach(function(elm) {
6. listOfSelectedValues += elm.value + " ";
7. var liParent = elm.parentNode;
8. liParent.classList.add("checked");
9. });
10. document.body.append("You selected: " + listOfSelectedValues);
11. }

- **Propósito:** Obtiene los elementos seleccionados, muestra sus valores y marca los ítems seleccionados con un estilo especial.
- **Pasos:**
    1. Obtiene todos los input seleccionados dentro de la lista #fruits usando querySelectorAll("#fruits input:checked").
    2. Itera sobre cada elemento seleccionado (list.forEach).

3. Para cada elemento:
    - Agrega su value (el valor de la fruta) a listOfSelectedValues.
    - Encuentra su elemento padre (liParent) con .parentNode.
    - Añade la clase CSS checked al elemento padre (liParent.classList.add("checked")).
4. Muestra los valores seleccionados en la página.

12. **reset**:
13. function reset() {
14. var list = document.querySelectorAll("#fruits input");
15. list.forEach(function(elm) {
16. elm.checked = false; // Desmarca el checkbox
17. var liParent = elm.parentNode;
18. liParent.classList.remove("checked"); // Elimina la clase CSS
19. });
20. }

- **Propósito:** Restablece la lista desmarcando todos los checkboxes y quitando cualquier estilo aplicado.
- **Pasos:**
    1. Obtiene todos los input de la lista #fruits.
    2. Itera sobre cada elemento.
    3. Para cada elemento:
        - Lo desmarca (elm.checked = false).
        - Elimina la clase CSS checked del padre.

Estilos (CSS)

Aunque no está incluido, el estilo checked probablemente define cómo se ven los ítems seleccionados. Por ejemplo:

```
.checked {
  background-color: yellow;
  font-weight: bold;
}
```

Comportamiento esperado

1. El usuario selecciona una o más frutas.

2. Hace clic en **"Show Checked items"**, y el navegador:
    - Muestra un mensaje con las frutas seleccionadas (You selected: apples oranges).
    - Cambia el estilo visual de las frutas seleccionadas (por la clase checked).
3. Si hace clic en **"Reset list"**, se:
    - Desmarcan todas las frutas.
    - Restablece su estilo visual original.

¿Te gustaría que modifiquemos o ampliemos algo?

El padre de los elementos seleccionados por document.querySelectorAll("#fruits input:checked") depende de cómo está estructurado el HTML.

Análisis

1. **Qué selecciona document.querySelectorAll("#fruits input:checked"):**
   Busca todos los elementos <input> que están dentro del elemento con id="fruits" y que tienen el estado checked (seleccionados).

En este caso, seleccionará los <input> que cumplen estas condiciones dentro del elemento <ul id="fruits">.

2. **¿Quién es el padre de cada elemento seleccionado?**
   Para cada <input> seleccionado, su padre inmediato (parentNode) es el elemento <li> que lo contiene.

Por ejemplo, en el siguiente fragmento:

```
<li>
  <input type="checkbox" name="fruit" value="apples">Apples
</li>
```

    - El <input> tiene como padre inmediato el <li>.

3. **Uso en el código:** En el JavaScript, cuando recorremos la lista:
4. var list = document.querySelectorAll("#fruits input:checked");
5. list.forEach(function(elm) {
6. var liParent = elm.parentNode; // Padre del input es el li
7. liParent.classList.add("checked"); // Se agrega la clase al
8. });

Aquí, elm.parentNode apunta al <li> que contiene el <input>.

Conclusión

El **padre inmediato** de los elementos seleccionados (#fruits input:checked) es el <li> correspondiente.

Por ejemplo, para:

  <input type="checkbox" name="fruit" value="apples">Apples

El padre de <input type="checkbox" ...> es el <li> que lo envuelve.

Si necesitas verificar algo más relacionado, ¡avísame! 😊

La razón por la que classList.remove("checked") se aplica sobre liParent y no sobre elm.checked radica en la **diferencia entre la clase CSS y el estado del checkbox (checked)**. Vamos a explicarlo paso a paso:

---

1. classList.remove("checked")

- **Qué hace:**
  El método classList.remove("checked") elimina la clase CSS llamada "checked" de un elemento DOM.
- **Sobre qué elemento se aplica:**
  En el código, liParent es el elemento <li> que contiene el checkbox (<input>).
- var liParent = elm.parentNode; // El padre del input es el
- liParent.classList.remove("checked");
- **Por qué sobre liParent:**
  La clase "checked" está diseñada para aplicarse visualmente al elemento <li>, no al <input>. Esto es lógico porque queremos destacar visualmente toda la fila del checkbox seleccionado, no solo el checkbox en sí.

---

2. elm.checked

- **Qué hace:**
  elm.checked es una propiedad booleana del elemento <input> que indica si el checkbox está marcado o no.
- **Por qué no se usa aquí para clases CSS:**
  elm.checked = false; se usa para cambiar el estado del checkbox (marcado o desmarcado). No tiene nada que ver con las clases CSS. Por lo tanto, aplicar classList.remove("checked") sobre elm no tiene sentido, ya que el checkbox no tiene la clase "checked"; esa clase pertenece al <li>.

---

3. Diferencia funcional

- **classList.remove("checked"):** Elimina la clase visual del elemento <li> para que deje de tener el estilo especial que indica selección.
- **elm.checked = false;:** Desmarca el checkbox, quitando su estado de selección.

Ejemplo práctico

Supongamos que el checkbox de "Apples" está marcado y se quiere reiniciar:

<li class="checked">

<input type="checkbox" name="fruit" value="apples" checked>Apples

Durante la ejecución de reset():

1. Se elimina el estado de marcado del checkbox:
2. elm.checked = false; // Desmarca el checkbox

Resultado:

<li class="checked">

<input type="checkbox" name="fruit" value="apples">Apples

3. Se elimina la clase "checked" del <li>:
4. liParent.classList.remove("checked"); // Elimina la clase CSS

Resultado:

<input type="checkbox" name="fruit" value="apples">Apples

Conclusión

Se aplica classList.remove("checked") sobre liParent porque el propósito es quitar el estilo aplicado al <li>. En cambio, elm.checked controla solo el estado lógico del checkbox, no sus clases CSS ni su apariencia visual.