Cosas Importantes

Mentalidad para el examen:

1ª ROSARIO LEELO TODO, entiende lo que te pide, subraya los IDs, clases, nombres de las etiquetas y los requisitos que nos piden.

2º Enlazar bien el fichero .js con el <script src=”examen.js”></script>

3º Haz las funcionalidad a la vez y ve comprobando que funciona, sino funciona ve buscando el error fila por fila del código, no te quedes parada que pierdes mucho tiempo, y seguro que es una tonteria.

4º CONSOLE.LOG es nuestro mejor amigo:

- Comprobamos que verifica que selecciona el elemento HTML correcto
- Comprobar el valor de las variables
- Saber si una función se ejecuta correctamente.
- Ver el resultado de las condiciones

5º COMPRUEBA QUE GUARDAS, Y HACEMOS LAS PRUEBAS EN LE NAVEGADOR.

Conceptos claves y Errores más comunes:

1º Seleccionar elementos de HTML (DOM)

- document.getElementById('id_del_elemento'): Para seleccionar UN elemento por su id único. **Este es tu método principal y más fiable.**

(este es para traerme una cosa que tiene la etiqueta con el nombre exacto, solo hay una)

- document.getElementsByClassName('nombre_de_la_clase'): Devuelve una **HTMLCollection** (parecida a un array, es una lista "viva") de TODOS los elementos que tienen esa clase.
    - Para acceder a uno: document.getElementsByClassName('miClase')[0] (el primero).
    - Para iterar:

```
const elementos = document.getElementsByClassName('miClase');

    for (let i = 0; i < elementos.length; i++) {
```

```
    // elementos[i] es cada elemento
}
```

(Este es para traerme TODAS las cosas que tenga la clase)

- document.getElementsByTagName('nombre_de_etiqueta'): Devuelve una **HTMLCollection** de TODOS los elementos con esa etiqueta (ej. 'input', 'td', 'div').
    - Se accede e itera igual que getElementsByClassName.

(Este es para traerme TODAS las cosas de la etiqueta)

- **Navegación (si no tienes IDs/clases directas):**
    - `elemento.parentElement`: El padre directo.
    - `elemento.children`: Colección HTML de los hijos directos.
    - `elemento.firstElementChild`: Primer hijo elemento.
    - `elemento.lastElementChild`: Último hijo elemento.
    - `elemento.nextElementSibling`: Siguiente hermano.
    - `elemento.previousElementSibling`: Hermano anterior.

Errores Comunes:

- Escribir mal el id, la clase o el nombre de etiqueta.
- Tratar una HTMLCollection (de getElementsByClassName o getElementsByTagName) como si fuera un solo elemento:

```
// MAL: Esto da error o undefined si hay más de un input o ninguno
// const valor = document.getElementsByTagName('input').value;
// BIEN (si quieres el primero y sabes que existe):
    const inputs = document.getElementsByTagName('input');
    if (inputs.length > 0) {
       const valor = inputs[0].value;
    }
```

- No guardar el elemento en una variable si lo vas a usar varias veces.

---

2. Leer y Modificar Valores de Inputs y Contenido de Elementos

- **Leer valor de un input:** elementoInput.value (donde elementoInput fue obtenido, por ejemplo, con getElementById).
- **Modificar contenido de un div/p/span, etc.:**
    - elementoDiv.textContent = 'Nuevo texto'; (Solo texto, más seguro)
    - elementoDiv.innerHTML = 'Texto en negrita'; (Permite HTML, cuidado con contenido del usuario)
- **Modificar atributos (ej. src de una imagen):** elementoImagen.src = 'nueva_imagen.jpg';
- **Explicación:** Igual que antes, pero recuerda que primero debes tener el elementoX correctamente seleccionado (probablemente con getElementById).
- **Errores Comunes:**
    - Usar .textContent o .innerHTML para leer el valor de un input.
    - Usar .value en un div o p.
    - Olvidar que el valor de <input type="number"> es un string y necesita parseInt() o parseFloat().

---

3. Manejar Eventos (Clicks, Mouseover, etc.)

- **Cómo se hace:** elemento.addEventListener('tipo_de_evento', funcionQueSeEjecuta);
    - elemento obtenido generalmente con getElementById.
    - tipo_de_evento: 'click', 'mouseover', 'mouseout', 'submit', 'change' (para inputs, selects).
- **Explicación:** "Oye, 'botónConIdEnviar', cuando alguien te haga 'click', entonces ejecuta esta 'listaDeInstrucciones'".
- **Errores Comunes:**
    - Poner paréntesis a la función al asignarla: boton.addEventListener('click', miFuncion()); (MAL). Debe ser miFuncion.
    - En formularios (<form>), si el botón es type="submit" (o el único botón), el evento 'submit' se dispara en el <form> mismo.

- Olvidar event.preventDefault() en el manejador del submit para que la página no se recargue.

```
const miFormulario = document.getElementById('idDelForm');
if (miFormulario) {
  miFormulario.addEventListener('submit', function(event) {
    event.preventDefault(); // ¡IMPORTANTE!
    // ...lógica de validación y envío aquí...
  });
}
```

Si es un botón normal (type="button") o usas el evento 'click' en un botón type="submit", preventDefault también puede ser necesario para evitar el envío si la validación falla.

4º Validaciones de Formularios (Muy Típico)

- **Campo vacío:** if (nombreInput.value.trim() === '') { ...error... }
- **Longitud mínima:** if (passwordInput.value.length < 8) { ...error... }
- **Contiene carácter:** if (!emailInput.value.includes('@')) { ...error... }
- **Fecha mayor de edad (18 años):**

```
const fechaNacInput = document.getElementById('fechaNacimiento');
const fechaNac = new Date(fechaNacInput.value);
const hoy = new Date();
let edad = hoy.getFullYear() - fechaNac.getFullYear();
const mes = hoy.getMonth() - fechaNac.getMonth();
if (mes < 0 || (mes === 0 && hoy.getDate() < fechaNac.getDate())) {
  edad--;
}
if (isNaN(fechaNac.getTime()) || edad < 18) { // Comprobar si la fecha es válida Y la edad
```

```
  // ...error...
}
```

Fecha posterior a la actual:

```
const fechaEventoInput = document.getElementById('fechaEvento');
const fechaEvento = new Date(fechaEventoInput.value);
const hoy = new Date();
hoy.setHours(0,0,0,0); // Comparar solo fechas
fechaEvento.setHours(0,0,0,0);
if (isNaN(fechaEvento.getTime()) || fechaEvento <= hoy) {
  // ...error...
}
```

Número y conversión:

```
const cantidadInput = document.getElementById('cantidad');
const cantidad = parseInt(cantidadInput.value);
if (isNaN(cantidad) || cantidad <= 20) { // Ejemplo validación
  // ...error...
}
```

- **Errores Comunes:**
  - No usar .trim().
  - Comparar números como strings (¡parseInt()!).
  - Lógica de fechas incorrecta o no validar si la fecha es Invalid Date.

---

5. Mostrar/Ocultar Elementos y Mensajes de Error

- **Con display:**
  - elemento.style.display = 'none'; (Ocultar)
  - elemento.style.display = 'block'; (Mostrar div, p)
- **Con clases CSS (si el HTML las define):**
  - elemento.classList.add('claseError');

- elemento.classList.remove('claseExito');
- elemento.className = 'nombreClase'; (Sobrescribe todas las clases, usar con cuidado. classList es mejor si solo quieres añadir/quitar una).

- **Mostrar mensaje de error:**

```
const errorDiv = document.getElementById('errorMessage');

if (errorDiv) { // Siempre comprobar que existe

  errorDiv.textContent = 'El campo nombre no puede estar vacío.';

  // Opción A: Si la clase ya lo hace visible

  errorDiv.className = 'errorMessageClaseVisible'; // Asumiendo que errorMessageClaseVisible está definida en CSS

  // Opción B: Mostrarlo explícitamente

  // errorDiv.style.display = 'block';

}
```

- **Limpiar errores:**

```
// AL INICIO DE LA VALIDACIÓN
if (errorDiv) {
  errorDiv.textContent = '';
  errorDiv.className = ''; // O la clase que lo oculta/estilo base
  // errorDiv.style.display = 'none';
}
const sorpresaP = document.getElementById('sorpresa');
if (sorpresaP) {
  sorpresaP.style.display = 'none';
}
let esValido = true;
// ...validaciones...
// AL FINAL
if (esValido && sorpresaP) {
  sorpresaP.style.display = 'block';
}
```

Errores Comunes: Olvidar mostrar el error o no limpiar errores antiguos.

---

6. Crear Elementos HTML Dinámicamente (Tablas, Listas)

- **Pasos:**
    1. document.createElement('tagName'): Crea el elemento.
    2. Configura: .textContent, .src, .id, .className, etc.
    3. padre.appendChild(hijo): Añade al DOM (donde padre es un elemento ya existente, ej. un tbody obtenido por getElementById o getElementsByTagName('tbody')[0]).
- **Ejemplo tabla (obtener tbody)**

```
const miTabla = document.getElementById('idDeLaTabla');
let tbody;
if (miTabla) {
  const tbodies = miTabla.getElementsByTagName('tbody');
  if (tbodies.length > 0) {
    tbody = tbodies[0];
  } else { // Si no hay tbody, lo creo (opcional, depende del HTML base)
    tbody = document.createElement('tbody');
    miTabla.appendChild(tbody);
  }
}

if (tbody) { // Asegurarse de que tbody existe antes de usarlo
  tbody.innerHTML = ''; // Limpiar contenido previo
  // ... bucle para crear filas (tr) y celdas (td) ...
  // const nuevaFila = document.createElement('tr');
  // const celda = document.createElement('td');
  // celda.textContent = 'dato';
  // nuevaFila.appendChild(celda);
  // tbody.appendChild(nuevaFila);
}
```

- **Errores Comunes:** Crear el elemento pero olvidar appendChild. No limpiar el contenedor si se repinta.

6.A. Crear y Añadir Listas Dinámicamente (<ul> o <ol>)

A menudo necesitarás mostrar una serie de elementos como una lista.

- **Escenario Típico:** Mostrar los ingredientes de una receta, los libros comprados, etc.
- **HTML Base (Ejemplo):**

```
<div id="contenedorLista">
        <h2>Ingredientes:</h2>
         <!-- La lista se insertará aquí por JS -->
        </div>
```

- **JavaScript para Crear y Añadir la Lista:**

```
// Datos de ejemplo
const ingredientes = ["Manzanas", "Azúcar", "Harina", "Canela", "Huevo"];

// 1. Seleccionar el contenedor donde irá la lista
const contenedorListaDiv = document.getElementById('contenedorLista');

if (contenedorListaDiv) { // Siempre verificar que el contenedor existe
  // Opcional: Limpiar el contenedor si ya había algo
  // contenedorListaDiv.innerHTML = '<h2>Ingredientes:</h2>'; // Si quieres mantener el h2
  // o si la lista es el único contenido y la repintas:
  // contenedorListaDiv.innerHTML = '';

  // 2. Crear el elemento de la lista (ul o ol)
  const listaUl = document.createElement('ul'); // Para lista desordenada
  // const listaOl = document.createElement('ol'); // Para lista ordenada

  // 3. Recorrer los datos y crear cada elemento de la lista (li)
  for (let i = 0; i < ingredientes.length; i++) {
    const ingredienteActual = ingredientes[i];

    const itemLi = document.createElement('li');
    itemLi.textContent = ingredienteActual; // Poner el texto en el <li>
```

```
    // 4. Añadir el <li> al <ul> (o <ol>)
    listaUl.appendChild(itemLi);
  }

  // 5. Añadir la lista completa (ul o ol) al contenedor en el DOM
  contenedorListaDiv.appendChild(listaUl);
} else {
  console.error("El contenedor con id 'contenedorLista' no fue
encontrado.");
}
```

- **Explicación:**
  1. "Busca la caja ('contenedorListaDiv') donde pondremos las cosas".
  2. "Crea una nueva bolsa de lista ('listaUl')".
  3. "Para cada ingrediente de nuestra receta:
     - Coge una etiqueta de papel ('itemLi').
     - Escribe el nombre del ingrediente en la etiqueta ('textContent').
     - Mete la etiqueta en la bolsa de lista ('appendChild' al ul)."
  4. "Mete la bolsa de lista completa en la caja grande del HTML ('appendChild' al div)".
- **Errores Comunes:**
  - Olvidar appendChild para los li al ul/ol, o para el ul/ol al contenedor div.
  - No limpiar el contenedor si la función se llama múltiples veces y quieres reemplazar la lista anterior.
  - Intentar añadir texto directamente al ul sin crear li (no es semánticamente correcto).

6.B. Crear y Añadir Tablas Dinámicamente (<table>, <thead>, <tbody>, <tr>, <td>, <th>)

Las tablas son fundamentales para mostrar datos estructurados (incidencias, productos, etc.).

- **Escenario Típico:** Mostrar un listado de incidencias con su ID, estado, asunto y prioridad.
- **HTML Base (Ejemplo):**

```
<div id="contenedorTablaIncidencias">
  <!-- La tabla se insertará aquí por JS -->
</div>
```

O si la tabla ya existe pero vacía, y solo quieres rellenar el <tbody>:

```
<table id="tablaIncidencias">
  <thead>
    <tr>
      <th>ID</th>
      <th>Estado</th>
      <th>Asunto</th>
      <th>Prioridad</th>
      <!-- Podría haber una columna de Acciones aquí -->
    </tr>
  </thead>
  <tbody>
    <!-- Las filas de datos se insertarán aquí por JS -->
  </tbody>
</table>
<div id="mensajeSinResultados" style="display:none;">No hay incidencias para mostrar.</div>
```

JavaScript para Crear y Añadir la Tabla (o Rellenar <tbody>):

```
// Datos de ejemplo (array de objetos)
const incidencias = [
  { id: 1, estado: 'Abierta', asunto: 'Error login', prioridad: 'Alta' },
  { id: 8, estado: 'Cerrada', asunto: 'Interfaz usuario', prioridad: 'Baja' },
  { id: 3, estado: 'Pendiente', asunto: 'Fallo API', prioridad: 'Media' }
];

// Seleccionar el div contenedor o la tabla existente
const divContenedor = document.getElementById('contenedorTablaIncidencias');
// Si creas la tabla desde cero
const tablaExistente = document.getElementById('tablaIncidencias'); // Si la tabla ya está en el HTML

// Función para pintar/rellenar la tabla
function pintarTablaIncidencias(datos) {
  let tbody;

  if (tablaExistente) { // Si la tabla ya existe en el HTML
    const tbodies = tablaExistente.getElementsByTagName('tbody');
    if (tbodies.length > 0) {
      tbody = tbodies[0];
    } else {
      tbody = document.createElement('tbody');
      tablaExistente.appendChild(tbody);
    }
  } else if (divContenedor) { // Si creamos la tabla desde cero
    divContenedor.innerHTML = ''; // Limpiar el div por si acaso
    const nuevaTabla = document.createElement('table');
```

```
    nuevaTabla.id = 'miNuevaTabla'; // Opcional: darle un ID o clase
    // nuevaTabla.border = "1"; // Estilo básico, mejor por CSS

    // Crear cabecera (thead y th)
    const thead = document.createElement('thead');
    const filaCabecera = document.createElement('tr');
    const cabeceras = ['ID', 'Estado', 'Asunto', 'Prioridad', 'Acciones']; // Incluye
Acciones

    for (let i = 0; i < cabeceras.length; i++) {
      const th = document.createElement('th');
      th.textContent = cabeceras[i];
      filaCabecera.appendChild(th);
    }
    thead.appendChild(filaCabecera);
    nuevaTabla.appendChild(thead);

    // Crear cuerpo (tbody)
    tbody = document.createElement('tbody');
    nuevaTabla.appendChild(tbody);

    // Añadir la nueva tabla al div contenedor
    divContenedor.appendChild(nuevaTabla);
  } else {
    console.error("No se encontró ni 'tablaIncidencias' ni
'contenedorTablaIncidencias'");
    return; // Salir si no hay donde pintar
  }
```

```
  // Siempre limpiar el tbody antes de añadir nuevas filas
  tbody.innerHTML = '';

  const mensajeDiv = document.getElementById('mensajeSinResultados'); // Para el mensaje de "no hay resultados"
  if (datos.length === 0) {
    if (mensajeDiv) mensajeDiv.style.display = 'block'; // Mostrar mensaje
    // La tabla (tbody) quedará vacía
    return;
  } else {
    if (mensajeDiv) mensajeDiv.style.display = 'none'; // Ocultar mensaje
  }


  // Recorrer los datos y crear las filas (tr) y celdas (td)
  for (let i = 0; i < datos.length; i++) {
    const incidenciaActual = datos[i];
    const filaTr = document.createElement('tr');

    // Añadir listener para mouseover (Ejemplo del examen)
    filaTr.addEventListener('mouseover', function() {
      this.style.backgroundColor = '#ffffcc'; // 'this' es la filaTr
    });
    filaTr.addEventListener('mouseout', function() {
      this.style.backgroundColor = ''; // Quitar el fondo
    });

    // Celda ID
```

```
        const celdaId = document.createElement('td');
        celdaId.textContent = incidenciaActual.id;
        filaTr.appendChild(celdaId);

        // Celda Estado
        const celdaEstado = document.createElement('td');
        celdaEstado.textContent = incidenciaActual.estado;
        filaTr.appendChild(celdaEstado);

        // Celda Asunto
        const celdaAsunto = document.createElement('td');
        celdaAsunto.textContent = incidenciaActual.asunto;
        filaTr.appendChild(celdaAsunto);

        // Celda Prioridad
        const celdaPrioridad = document.createElement('td');
        // Si 'prioridad' viene de un objeto anidado como en algunos exámenes:
        // celdaPrioridad.textContent = incidenciaActual.detalles ? incidenciaActual.detalles.prioridad : 'N/A';
        celdaPrioridad.textContent = incidenciaActual.prioridad; // Para el ejemplo simple
        filaTr.appendChild(celdaPrioridad);

        // Celda Acciones (Ejemplo de una imagen 'ver.jpg')
        const celdaAcciones = document.createElement('td');
        const imgVer = document.createElement('img');
        imgVer.src = 'ver.jpg'; // Asegúrate que la ruta es correcta
        imgVer.alt = 'Detalle de la incidencia';
        imgVer.title = 'Detalle de la incidencia'; // Tooltip
```

```
    imgVer.style.width = '20px'; // Como pide el examen
    imgVer.style.height = '20px';

    // Guardar el ID de la incidencia en la imagen para usarlo en el click
    // Esto es una forma, otra es usar una función que reciba el ID
    imgVer.dataset.incidenciaId = incidenciaActual.id;
    // O pasar toda la incidencia si es necesario (más pesado si son muchos datos)
    // imgVer.incidenciaData = incidenciaActual; // No estándar, mejor dataset o closure

    imgVer.addEventListener('click', function() {
      // 'this' aquí es la imagen (imgVer)
      const idIncidenciaClicada = this.dataset.incidenciaId; // Recuperar el ID
      // O si guardaste el objeto: const data = this.incidenciaData;
      console.log("Ver detalle de incidencia ID:", idIncidenciaClicada);
      // Aquí iría la lógica de window.open() y pasar el ID o los datos
      // Ejemplo: window.open('detalle.html?id=' + idIncidenciaClicada, '_blank', 'width=500,height=500');

      // Lógica del examen: Cambiar estado a "Cerrada" en la tabla dinámicamente
      // Necesitaríamos encontrar la celda del estado en esta fila.
      // 'this.parentElement' es la celda <td>, 'this.parentElement.parentElement' es la fila <tr>
      const filaActual = this.parentElement.parentElement;
      const celdasDeLaFila = filaActual.getElementsByTagName('td');
      if (celdasDeLaFila.length > 1) { // Asumiendo que la celda de estado es la segunda (índice 1)
        celdasDeLaFila[1].textContent = 'Cerrada';
```

```
        }
        // También deberías actualizar el estado en tu array de datos original
        // const incidenciaOriginal = datos.find(inc => inc.id == idIncidenciaClicada);
        // if (incidenciaOriginal) incidenciaOriginal.estado = 'Cerrada';
      });
      celdaAcciones.appendChild(imgVer);
      filaTr.appendChild(celdaAcciones);



      // Añadir la fila completa al tbody
      tbody.appendChild(filaTr);
    }
}

// Llamada inicial para pintar la tabla con todos los datos (o después de una búsqueda)
// Se llamaría desde el addEventListener del botón "Buscar"
// pintarTablaIncidencias(incidencias);

// Ejemplo de cómo se usaría con un botón de búsqueda
const btnBuscar = document.getElementById('botonBuscar'); // Asume que tienes este botón
if (btnBuscar) {
  btnBuscar.addEventListener('click', function() {
    // Aquí recogerías los valores de los filtros (idInput, estadoSelect, etc.)
    // const idFiltro = document.getElementById('idFiltroInput').value;
    // const estadoFiltro = document.getElementById('estadoFiltroSelect').value;

    // Filtrarías el array 'incidencias' original
```

```
      // const resultadosFiltrados = incidencias.filter(inc => {
      //   let cumple = true;
      //   if (idFiltro && inc.id != idFiltro) cumple = false;
      //   if (estadoFiltro && estadoFiltro !== "Todas" && inc.estado !== estadoFiltro) cumple = false;
      //   return cumple;
      // });
      // pintarTablaIncidencias(resultadosFiltrados);

      // Para este ejemplo, simplemente la volvemos a pintar con todos los datos:
      pintarTablaIncidencias(incidencias);
    });
} else {
    // Si no hay botón de búsqueda, pintar directamente
    pintarTablaIncidencias(incidencias);
}
```

- **Explicación:**
    1. "Decide si vas a usar una mesa ('tablaExistente') que ya está ahí o si vas a construir una nueva ('nuevaTabla') en un espacio vacío ('divContenedor')".
    2. "Si es nueva, ponle un mantel de cabecera ('thead') con los títulos de las columnas ('th')".
    3. "Prepara la parte principal de la mesa donde irán los platos ('tbody'). Límpiala bien ('innerHTML = '')".
    4. "Para cada incidencia (cada plato de comida):
        - Coge una bandeja ('filaTr').
        - Ponle papelitos ('celdaTd') para cada dato: ID, estado, asunto...
        - En el último papelito de 'Acciones', dibuja un botón para 'Ver' ('imgVer'). Enséñale qué hacer si le pican.
        - Coloca la bandeja en la mesa ('appendChild' al tbody)".
- **Errores Comunes:**
    - Olvidar appendChild en cualquier nivel (td a tr, tr a tbody, th a tr de cabecera, thead/tbody a table, table a div).

- No limpiar tbody.innerHTML = ''; antes de repintar, lo que duplica las filas.
- Estructura incorrecta de la tabla (ej. td directamente en table sin tr o tbody).
- Al añadir listeners dentro de un bucle (como el mouseover o el click en la imagen):
  - Asegurarse de que this se refiere al elemento correcto. Las funciones normales (function() {}) suelen hacerlo bien para listeners de evento, donde this es el elemento que disparó el evento.
  - Si necesitas pasar datos específicos de esa iteración del bucle a la función del listener, usar dataset como en el ejemplo de la imagen, o closures (un poco más avanzado).
- Referenciar mal las celdas al intentar modificar dinámicamente (como cambiar el estado a "Cerrada"). filaActual.getElementsByTagName('td')[indice] es una forma robusta.

---

7. Trabajar con Arrays de Objetos (Incidencias, Productos)

- **Definición y recorrido:** Igual que antes (const datos = [{...}]; datos.forEach(...);).
- **Filtrar:** array.filter() sigue igual.
- **Mostrar en tabla:** La lógica de crear tr y td es la misma, pero la obtención del tbody se hace como en el punto 6.
- **Errores Comunes:** Typos en propiedades, lógica de filtro, no convertir strings de input a números.

---

8. Timers (setTimeout, setInterval)

- **setTimeout(funcion, milisegundos):** Ejecuta funcion UNA VEZ.
- **setInterval(funcion, milisegundos):** Ejecuta funcion REPETIDAMENTE. Devuelve ID para clearInterval(id).
- **Errores Comunes:** Olvidar clearInterval, pasar función con ().

---

9. Números Aleatorios

- **Math.random():** Decimal entre 0 (incluido) y 1 (excluido).
- **Entero entre min y max (ambos incluidos):** Math.floor(Math.random() * (max - min + 1)) + min;
- **Errores Comunes:** Fórmula incorrecta, olvidar Math.floor().

---

10. Imágenes (Mostrar, Cambiar Tamaño)

Crear y mostrar:

```
const imgContainer = document.getElementById('sorpresald'); // El div donde irá la img

if (imgContainer) {
  imgContainer.innerHTML = ''; // Limpiar

  const nuevaImagen = document.createElement('img');
  nuevaImagen.src = 'sorpresa1.jpg';
  nuevaImagen.alt = 'Imagen sorpresa';
  nuevaImagen.width = 650;
  nuevaImagen.height = 350;
  imgContainer.appendChild(nuevaImagen);
}
```

- **Errores Comunes:** Ruta incorrecta, no limpiar contenedor, no alt.

---

11. Nuevas Ventanas (window.open, window.close)

- **Abrir:** const nuevaVentana = window.open('url.html', '_blank', 'width=500,height=500');
- **Cerrar:** window.close(); (desde la nueva) o nuevaVentana.close(); (desde la que la abrió).
- **Acceder desde nueva ventana a la original (window.opener):**

```
// En el JS de la nueva ventana (ej. resumen.js)
if (window.opener && !window.opener.closed) {
  const elementoPrincipal = window.opener.document.getElementById('idEnIndexHtml');
  // ... usar elementoPrincipal ...
}
```
