

TEMAS DE ACTIVIDAD DE COMPROBACIÓN UNO

Programación Web en el Entorno Cliente

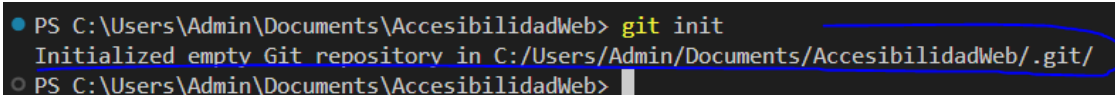
Instrucciones Generales:

Instrucciones Generales:

1. De los tres casos, deberá realizar el indicado, siguiendo las instrucciones del caso correspondiente y cumpliendo con lo solicitado en el documento "Actividad de comprobación N1.docx".
2. El trabajo se hace de forma individual, puede utilizar como referencia cualquier recurso dado y/o ejercicio realizado en clase o en las prácticas de clase (presencial o virtual).
3. Prohibido el uso de alguna IA u otro sitio de internet.
4. Los teléfonos celulares deberán estar guardados.
5. Cualquier duda sobre la forma del ejercicio consúltela con el profesor.
6. No hable con ningún compañero/a ya que esto invalida la actividad de los involucrados.
7. La Actividad de comprobación N°1, tiene DOS intentos o posibilidades, en caso de ser necesaria, la segunda posibilidad quedaría para el día martes 19 de noviembre 2024.
8. En caso de que la calificación final de las dos oportunidades sea NL (No Logrado), reprobaría el módulo y NO podrá continuar en el programa.
9. Si alguna persona comete fraude de alguna de las formas descritas o de otra forma, su actividad será invalidada y no tendrá derecho a la segunda oportunidad, por lo que quedará reprobado/a automáticamente.
10. Al cumplirse el tiempo establecido, suba su trabajo a www.github.com comparte el link junto con la carpeta comprimida con el trabajo realizado y la hoja de evaluación debidamente llena y guardada con su nombre, al correo gcotocalderon@ina.cr.

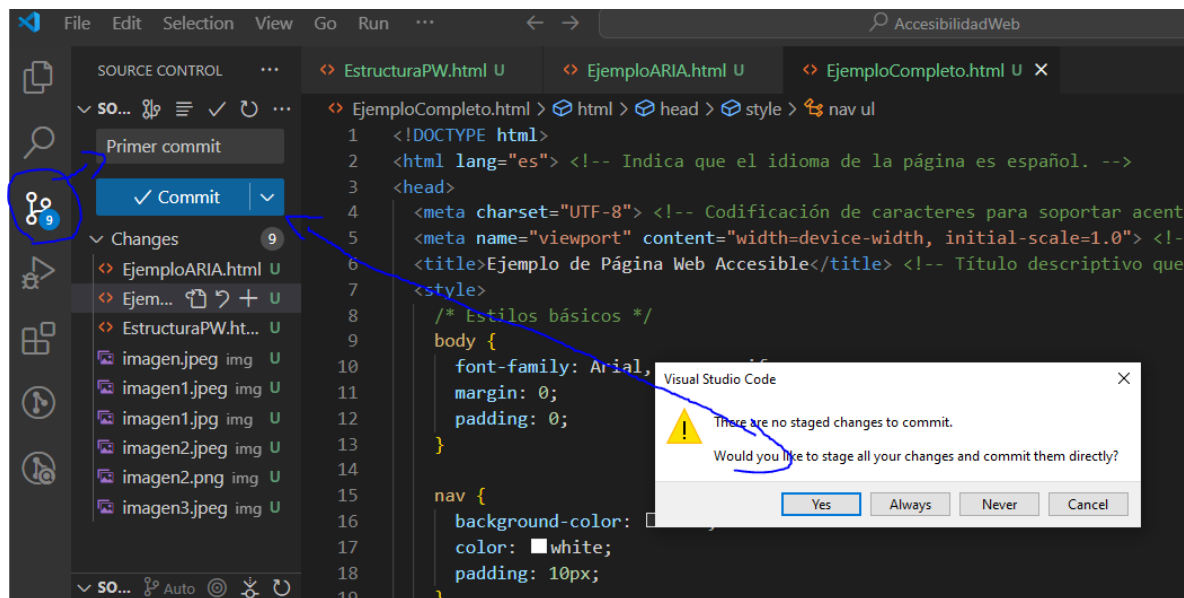
- Pasos para subir a github desde VS-Code

1. En VS-Code, entre en la terminal y digite: git init

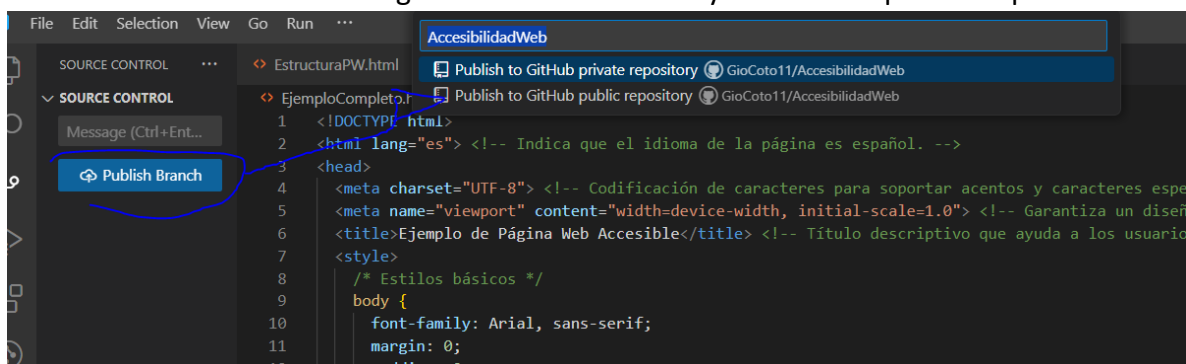


```
PS C:\Users\Admin\Documents\AccesibilidadWeb> git init
Initialized empty Git repository in C:/Users/Admin/Documents/AccesibilidadWeb/.git/
PS C:\Users\Admin\Documents\AccesibilidadWeb>
```

2. Haga clic en segundo botón de VS-Code, en Mensaje escriba el nombre del commit (ejemplo commit 1) y oprima Commit y luego en yes



3. haga clic en Public Branch y decida si es publico o privado.



4. Reciba mensaje de confirmación y haga clic Open GitHub y comparta la URL

11. De no cumplir con estas instrucciones su trabajo NO será calificado, por lo que perdería dicha actividad de comprobación.

Caso 1. Actividad de Comprobación: Gestión de Productos

Instrucciones:

1. Diseña una página web que permita gestionar un inventario de productos.
2. Cada producto debe tener:
 - Nombre.
 - Precio.
 - Cantidad en inventario.
3. Implementa las siguientes funcionalidades:
 - **Agregar productos** con validación de datos.
 - **Listar los productos** en una tabla que incluye el total (precio \times cantidad) por producto.
 - **Editar productos existentes** (nombre, precio y cantidad).
 - **Eliminar productos** del inventario.
4. Los productos deben guardarse en el almacenamiento local para que la información persista al recargar la página.
5. Incluye un contador dinámico que muestre el valor total del inventario.

Estilos:

```
<style>
body { font-family: Arial, sans-serif; margin: 20px; }
table { width: 100%; border-collapse: collapse; margin-top: 20px; }
th, td { border: 1px solid #ccc; padding: 10px; text-align: left; }
th { background-color: #f4f4f4; }
.btn { margin-left: 5px; padding: 5px 10px; cursor: pointer; }
.btn-edit { background-color: #ffc107; border: none; color: #fff; }
.btn-delete { background-color: #dc3545; border: none; color: #fff; }
</style>
```

Tabla html:

```
<!-- Tabla para listar productos -->
<table>
  <thead>
    <tr>
      <th>Nombre</th>
      <th>Precio</th>
      <th>Cantidad</th>
      <th>Total</th>
      <th>Acciones</th>
    </tr>
  </thead>
  <tbody id="listaProductos">
  </tbody>
</table>
```

Criterios de evaluación cumplidos:

1. **Manipulación del DOM:** Se crean elementos dinámicos para la tabla de productos.
2. **Eventos:** El formulario y los botones dinámicos manejan eventos como `submit` y `onclick`.
3. **Validación de datos:** Se valida que el nombre, precio y cantidad sean válidos.
4. **Funciones:** Cada acción principal (agregar, editar, eliminar, guardar en almacenamiento local) está organizada en funciones.
5. **Almacenamiento local:** Los productos se guardan en `localStorage` y se recuperan al recargar la página.
6. **Estructuras JSON:** Los datos de los productos se manejan como objetos en un arreglo JSON.
7. **Operadores matemáticos:** Se calcula el total por producto y el valor total del inventario.
8. **Buenas prácticas:** Código estructurado, limpio y comentado para facilitar su comprensión.

Gestión de Productos

Nombre del producto Precio Cantidad

Inventario Total: 0.00

Nombre	Precio	Cantidad	Total	Acciones
--------	--------	----------	-------	----------

Caso 2. Actividad de Comprobación: Gestión de Compras

Instrucciones:

1. Diseña una página web para simular un carrito de compras.
2. Cada producto debe tener:
 - Nombre del producto.
 - Precio unitario.
 - Cantidad.
3. Implementa las siguientes funcionalidades:
 - **Agregar productos al carrito** con validación de datos.
 - **Listar productos en el carrito** mostrando el total de cada producto (precio \times cantidad).
 - **Eliminar productos del carrito.**
 - Mostrar el **total general de la compra.**
 - Al finalizar la compra, guarde el historial en el almacenamiento local y vacíe el carrito.
4. Los productos en el carrito deben persistir utilizando el almacenamiento local

```
<style>
body { font-family: Arial, sans-serif; margin: 20px; }
table { width: 100%; border-collapse: collapse; margin-top: 20px; }
th, td { border: 1px solid #ccc; padding: 10px; text-align: left; }
th { background-color: #fafaf4; }
.btn { margin-left: 5px; padding: 5px 10px; cursor: pointer; }
.btn-delete { background-color: #dc3545; border: none; color: #fff; }
.btn-finalizar { background-color: #28a745; border: none; color: #fff; padding: 10px 20px; }
</style>
```

```
<table>
  <thead>
    <tr>
      <th>Producto</th>
      <th>Precio Unitario</th>
      <th>Cantidad</th>
      <th>Total</th>
      <th>Acciones</th>
    </tr>
  </thead>
  <tbody id="listaCarrito"></tbody>
</table>
```

Criterios de evaluación cumplidos:

1. **Manipulación del DOM:** Se crean elementos dinámicos para listar productos en la tabla.
2. **Eventos:** Maneja eventos como `submit` para agregar productos y `onclick` para eliminar productos o finalizar la compra.
3. **Validación de datos:** Valida que los datos ingresados sean válidos (nombre, precio y cantidad).
4. **Estructuras JSON:** El carrito y el historial se manejan como objetos en arreglos JSON.
5. **Almacenamiento local:** Los datos persisten utilizando `localStorage`.
6. **Operadores matemáticos:** Calcula totales por producto y el total general de la compra.
7. **Funciones:** Organiza cada acción principal (agregar, eliminar, finalizar compra, guardar en `localStorage`) en funciones independientes.
8. **Buenas prácticas:** Código estructurado, legible y comentado línea a línea.

Gestión de Compras

Carrito de Compras

Producto	Precio Unitario	Cantidad	Total	Acciones
----------	-----------------	----------	-------	----------

Total General: 0.00

Caso 3. Actividad de Comprobación: Gestión de Ventas

Instrucciones:

1. Diseña una página web que permita registrar ventas de productos.
2. Cada venta debe incluir:
 - Nombre del producto.
 - Precio unitario.
 - Cantidad vendida.
3. Implementa las siguientes funcionalidades:
 - **Registrar una venta** con validación de datos.
 - **Listar las ventas** en una tabla que muestre el nombre del producto, cantidad, precio total de la venta (precio \times cantidad) y la fecha de la venta.
 - Mostrar el **total global de todas las ventas**.
 - Almacenar las ventas en el almacenamiento local para que persistan.
4. Diseña el proyecto con buenas prácticas, código estructurado y comentarios para facilitar la comprensión.

```
<style>
  body { font-family: Arial, sans-serif; margin: 20px; }
  table { width: 100%; border-collapse: collapse; margin-top: 20px; }
  th, td { border: 1px solid #ccc; padding: 10px; text-align: left; }
  th { background-color: #f4f4f4; }
  .btn-clear { margin-top: 20px; padding: 10px 20px; background-color: #dc3545; color: white; border: none; cursor: pointer; }
</style>
```

```
<table>
  <thead>
    <tr>
      <th>Producto</th>
      <th>Precio Unitario</th>
      <th>Cantidad</th>
      <th>Total de la Venta</th>
      <th>Fecha</th>
    </tr>
  </thead>
  <tbody id="listaVentas"></tbody>
</table>
```

Criterios de evaluación cumplidos:

1. **Manipulación del DOM:** Se crean elementos dinámicos para listar las ventas en la tabla.
2. **Eventos:** Maneja eventos como `submit` para registrar ventas y `onclick` para limpiar el historial de ventas.
3. **Validación de datos:** Valida que los datos ingresados sean válidos (nombre, precio y cantidad).
4. **Estructuras JSON:** Las ventas se manejan como objetos en un arreglo JSON.
5. **Almacenamiento local:** Los datos persisten utilizando `localStorage`.
6. **Operadores matemáticos:** Calcula el total de cada venta (precio \times cantidad) y el total global.
7. **Funciones:** Organiza las funcionalidades principales (registrar venta, limpiar ventas, guardar en almacenamiento local) en funciones independientes.
8. **Buenas prácticas:** Código estructurado, legible y comentado línea a línea.

Gestión de Ventas

Ventas Realizadas

Producto	Precio Unitario	Cantidad	Total de la Venta	Fecha
----------	-----------------	----------	-------------------	-------

Total Global de Ventas: 0.00

Caso 4. Actividad de Comprobación: Gestión de Publicidad

Tema: Gestión de publicidad (Registrar, programar y gestionar anuncios publicitarios).

Instrucciones:

1. Diseña una página web para gestionar anuncios publicitarios.
2. Cada anuncio debe incluir:
 - Nombre del anuncio.
 - Descripción.
 - Fecha de inicio.
 - Fecha de fin.
 - Costo por día de publicación.
3. Implementa las siguientes funcionalidades:
 - **Registrar un anuncio publicitario** con validación de datos.
 - **Listar los anuncios** en una tabla, mostrando la duración en días y el costo total (duración × costo por día).
 - **Eliminar anuncios** de la lista.
 - **Mostrar el costo total global de todas las publicaciones.**
 - Almacenar los anuncios en el almacenamiento local para que persistan.

```
<style>
  body { font-family: Arial, sans-serif; margin: 20px; }
  table { width: 100%; border-collapse: collapse; margin-top: 20px; }
  th, td { border: 1px solid #ccc; padding: 10px; text-align: left; }
  th { background-color: #f4f4f4; }
  .btn-delete { margin-left: 5px; background-color: #dc3545; border: none; color: #fff; padding: 5px 10px; cursor: pointer; }
</style>
```

```
<table>
  <thead>
    <tr>
      <th>Nombre</th>
      <th>Descripción</th>
      <th>Fecha Inicio</th>
      <th>Fecha Fin</th>
      <th>Días Publicados</th>
      <th>Costo Total</th>
      <th>Acciones</th>
    </tr>
  </thead>
  <tbody id="listaAnuncios"></tbody>
</table>
```

Criterios de evaluación cumplidos:

1. **Manipulación del DOM:** Los anuncios se muestran dinámicamente en una tabla con cálculo de métricas.
2. **Eventos:** Se maneja el evento submit para registrar anuncios y onclick para eliminarlos.
3. **Validación de datos:** Se validan los datos ingresados, incluyendo fechas y costos.
4. **Estructuras JSON:** Los anuncios se manejan como objetos en un arreglo JSON.
5. **Almacenamiento local:** Los datos persisten en el almacenamiento local del navegador.
6. **Operadores matemáticos:** Calcula la duración del anuncio en días y el costo total (duración × costo por día).
7. **Funciones:** Cada funcionalidad principal (registrar, eliminar, guardar en localStorage, renderizar) está encapsulada en funciones independientes.
8. **Buenas prácticas:** Código estructurado, limpio y comentado línea a línea.

Gestión de Publicidad

Nombre del anuncio Descripción del anuncio dd/mm/aaaa dd/mm/aaaa Costo por día

Anuncios Publicitarios

Nombre	Descripción	Fecha Inicio	Fecha Fin	Días Publicados	Costo Total	Acciones
--------	-------------	--------------	-----------	-----------------	-------------	----------

Costo Total Global: 0.00