HTML Table Rowspan Colspan Issue

In your HTML code, the reason why "Watson" doesn't span across two columns is related to how the `rowspan` and `colspan` attributes interact with each other. Let's break it down step by step.

Table Structure Explanation:

The HTML you've written represents a table with 3 rows and 3 columns (visually). Let's go over each part:

1. **First Row (`<tr>`)**:
   Contains three `<td>` elements:
   - "Priya"
   - "Sharma"
   - "24"

   This row is fine, no issues here.

2. **Second Row (`<tr>`)**:
   - "Arun"
   - "Singh"
   - The third cell in this row uses `rowspan="2"`, meaning the value `32` will stretch over both the second and third rows vertically.

3. **Third Row (`<tr>`)**:
   - "Sam"
   - The second cell here has `colspan="2"`, meaning "Watson" is supposed to span across 2 columns horizontally.

The Issue: `rowspan` and `colspan` Interaction

The issue arises because the cell with `rowspan="2"` (the cell with "32") already occupies the third column, and thus when you try to apply `colspan="2"` to "Watson", it doesn't get the desired space.

Here's what happens step by step:

- In the **second row**, "Arun" occupies the first column, "Singh" occupies the second column, and the cell with "32" spans vertically into both the second and third rows, occupying the third column in both rows.
- In the **third row**, "Sam" occupies the first column, and "Watson" is meant to take up 2 columns (`colspan="2"`). However, since "32" from the previous row is already taking up the third column (due to `rowspan="2"`), "Watson" only spans across the first and second columns, and there is no space left for it to span across a third column.

Summary:

Yes, `rowspan` takes precedence over `colspan` in this case because the cell with `rowspan="2"` already occupies the third column. This causes "Watson" to span only the first and second columns, leaving no room for a true two-column span as intended by the `colspan="2"`.

Solution:

To fix this, you need to adjust the table structure, ensuring that there's no conflict between `rowspan` and `colspan`. Here's one possible solution:

```html
<table>
  <tbody>
    <tr>
      <td>Priya</td>
      <td>Sharma</td>
      <td>24</td>
    </tr>
    <tr>
      <td>Arun</td>
      <td>Singh</td>
      <td rowspan="2">32</td> <!-- Keep rowspan -->
    </tr>
    <tr>
      <td>Sam</td>
      <td>Watson</td> <!-- Remove colspan -->
    </tr>
  </tbody>
</table>
```

By removing `colspan="2"` from the third row, "Watson" takes up its correct place in the second column, avoiding the conflict with the `rowspan`.

Alternatively, you can adjust the logic of how many columns you expect in each row if you want a different layout altogether.

This behavior happens due to how browsers interpret and render the HTML table structure, specifically when handling the interaction between the `rowspan` and `colspan` attributes. Understanding the layout mechanism will make this clearer.

How HTML Tables Work:

HTML tables are built in a grid-like format, where each cell (`<td>`) is aligned into rows and columns. The browser renders this table by first calculating how many rows and columns exist based on the number of cells (`<td>`) within each row (`<tr>`).

Key Concepts:

1. `rowspan`: This attribute allows a cell to span vertically across multiple rows. For example, `rowspan="2"` means the cell will cover the current row plus one additional row below it.

2. `colspan`: This attribute allows a cell to span horizontally across multiple columns. For example, `colspan="2"` means the cell will take up space across two columns instead of just one.

Browser Rendering Process:

When a browser processes a table, it tries to "lay out" the grid of cells based on both the number of rows and columns each cell is meant to occupy. Here's how it handles this for your case:

1. **Normal Layout**:
   Each row is expected to have three columns of data because the table's first row contains three `<td>` elements, implying three columns.

2. **Rowspan Handling**:
   When the browser reaches the second row and encounters the `rowspan="2"` attribute for the "32" cell, it understands that this cell should not only occupy the third column in the second row, but also continue to occupy that column in the third row.
   - The browser marks the third column as "taken" by the `rowspan="2"` cell, reserving that space for both the second and third rows.

3. **Colspan Handling**:
   In the third row, when the browser encounters the `colspan="2"` for "Watson," it tries to make that cell span two columns.
   - However, the cell with "32" from the previous row (due to `rowspan`) already occupies the third column. So, the browser only finds two available columns: the first and second columns.
   - As a result, the `colspan="2"` only spans the first and second columns, because the third column is already taken by the `rowspan` cell.

Why Does This Happen?

The main reason this behavior happens is because `rowspan` affects the vertical space before the browser processes `colspan` horizontally. When the browser starts processing the third row, the third column is already occupied by the cell with the `rowspan`, so there's no room for "Watson" to take up the intended two columns.

This is a normal behavior based on the browser's rendering order:

1. **Rowspan first**: The browser allocates space for any cells with `rowspan` before rendering the next row.
2. **Colspan next**: When it processes a cell with `colspan`, it only looks for available columns horizontally. If any of those columns are already occupied by a cell with `rowspan`, the `colspan` cell can't extend into them.

In your case, the third column is already occupied by the "32" cell (due to the `rowspan`), so "Watson" can only span across two available columns (columns 1 and 2).

Visualizing the Problem

Row 1	Row 2	Row 3
Priya	Sharma	24
Arun	Singh	**32** (Rowspan 2)
Sam	**Watson (Colspan 2)**	

- In **Row 2**, "32" uses `rowspan="2"`, meaning it spans across the third column for both **Row 2** and **Row 3**.
- In **Row 3**, "Watson" is meant to span two columns, but it can't take up the third column because "32" is already occupying that space.

Summary:

- `rowspan` **has precedence** over `colspan` because the browser reserves the row and column space for `rowspan` first.
- The browser calculates space row by row, and once a column is reserved for a `rowspan`, it's no longer available for any other cells trying to span into that column (even with `colspan`).

This is why "Watson" can't span across the third column—it's already taken by the cell with "32" due to the `rowspan="2"` rule.