Bài 1: Phân tích height và width của hình ảnh

Hiểu rõ sự khác biệt giữa các cách thiết lập kích thước hình ảnh trong HTML thông qua **attribute** và **CSS inline style**, đồng thời biết cách truy xuất giá trị bằng JavaScript.

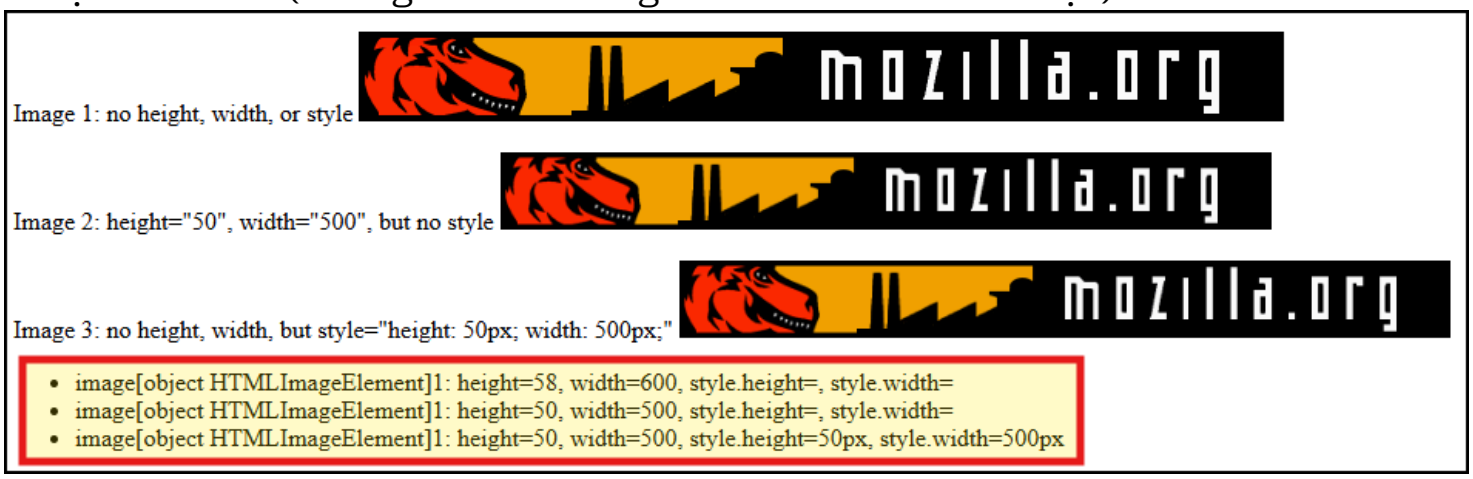
Yêu cầu:

1. Tạo 3 hình ảnh giống nhau nhưng có **cấu hình** khác nhau:
   - Image 1: Không khai báo height, width hoặc style.
   - Image 2: Dùng height="50" và width="500" thông qua attribute trong HTML.
   - Image 3: Dùng inline style style="height: 50px; width: 500px;".

Tất cả hình ảnh dùng cùng một nguồn: *https://www.mozilla.org/images/mozilla-banner.gif*

2. Tạo một vùng hiển thị (div) để in kết quả phân tích.

3. Bằng JavaScript:
   - Truy xuất lần lượt 3 hình ảnh bằng **getElementById**.
   - Tạo 1 mảng chứa 3 đối tượng hình ảnh.
   - Duyệt qua mảng và thu thập các thông tin sau của mỗi hình ảnh:
     - height
     - width
     - style.height
     - style.width
   - In thông tin từng hình ảnh dưới dạng danh sách (<ul> với các <li>) vào vùng hiển thị đã tạo ở bước 2. (khung màu đỏ trong ảnh bên dưới - minh họa)



Tips:

Sử dụng `.height`, `.width` để lấy kích thước hiển thị thực tế của ảnh.
Dùng `.style.height`, `.style.width` để lấy giá trị trong inline style.

Bài 2: border styles - Thay đổi độ dày đường viền

Hiểu và thực hành thao tác với CSS property thông qua DOM bằng JavaScript – cụ thể là border-width.

Yêu cầu:

1. HTML:

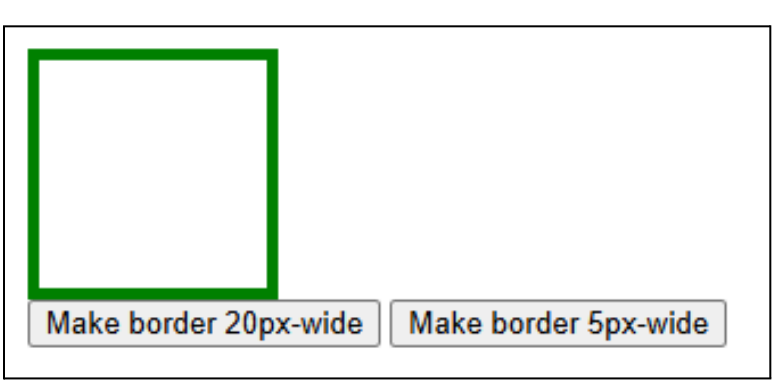
- Tạo một ** ** có **id="box"** dùng để hiển thị khối hình vuông.
- Tạo một thẻ **<form>** có **name="FormName"** chứa hai button:
  - button 1: Khi bấm sẽ đặt độ dày đường viền là 20px.
  - button 2: Khi bấm sẽ đặt độ dày đường viền là 5px.

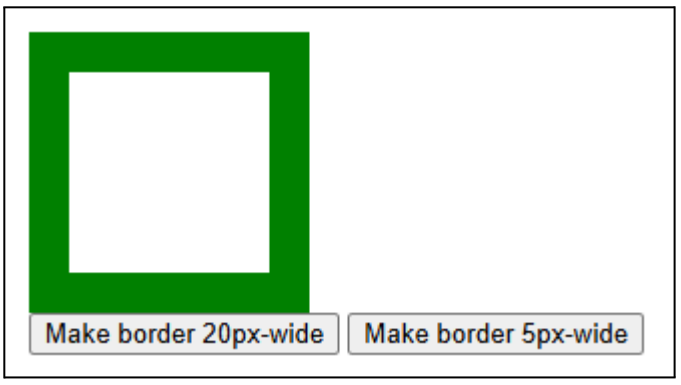
2. CSS:

- Định nghĩa **#box** có các property sau:
  - border: 5px solid green;
  - width: 100px;
  - height: 100px;

3. JavaScript:

- Viết một hàm tên **setBorderWidth**(width) nhận vào một tham số số nguyên.
  - Hàm sẽ thay đổi border-width của **#box** sang giá trị width truyền vào.
- Gắn sự kiện click cho hai nút:
  - Nút thứ nhất gọi **setBorderWidth**(20).
  - Nút thứ hai gọi **setBorderWidth**(5).





Tips:

Khi sử dụng addEventListener, phải đảm bảo không để nút gửi form (mặc định).

```
<form action=""></form>
```

Có thể cần dùng event.preventDefault() nếu cần.

```
Event.preventDefault(): void

If invoked when the cancelable attribute value is true, and while
executing a listener for the event with passive set to false, signals to
the operation that caused event to be dispatched that it needs to be
canceled.
```

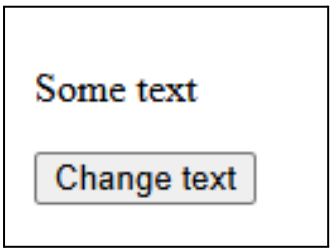
Bài 3: Manipulating styles - Thay đổi kiểu dáng đoạn văn

Biết cách truy cập và thay đổi trực tiếp các `CSS property` của một phần tử thông qua `style object` trong JavaScript.

Yêu cầu:

1. HTML:

- Tạo một `p` có `id="pid"` với nội dung văn bản bất kỳ (ví dụ: "Some text").
- Tạo một `form` chứa một `button` duy nhất với nhãn: **Change text**.
    - Nút này **không được** gửi form, chỉ thực hiện hành động JavaScript khi bấm.



2. JavaScript:

- Viết một hàm có tên `changeText` để thực hiện các thay đổi sau đối với `p#pid`:
    - Đặt màu chữ (`color`) là `"blue"`.
    - Đặt cỡ chữ (`fontSize`) là `"18pt"`.
- Gắn `event listener` để gọi hàm `changeText()` khi người dùng bấm nút.

Khi người dùng bấm nút, đoạn văn sẽ đổi sang chữ màu xanh dương và kích thước chữ to hơn.

Tips:

- Sử dụng `getElementById` để truy cập phần tử `p`.
- Truy cập và thay đổi thuộc tính `style.color`, `style.fontSize` của phần tử.
- Sử dụng `querySelector("button")` để gắn sự kiện `click`.

Bài 4: Using StyleSheets - Truy xuất và liệt kê các CSS rule selectors từ stylesheet

Hiểu và thao tác với **document.styleSheets** để truy cập danh sách các **stylesheet** đã được nạp vào tài liệu **HTML**, đồng thời biết cách duyệt và in ra các **CSS selector** từ các **CSS rule**.

Yêu cầu:

1. CSS:

Tạo một stylesheet riêng (dùng `<style>` hoặc `<link>` trong `<head>`) chứa ít nhất 3 rule sau:

- Một rule áp dụng cho `body` với `background-color: darkblue`.
- Một rule áp dụng cho tất cả `p`:
  - `font-family: Arial`
  - `font-size: 10pt`
  - `margin-left: 0.125in`
- Một rule cho phần tử có `id="lumpy"` với `display: none`.

2. JavaScript:

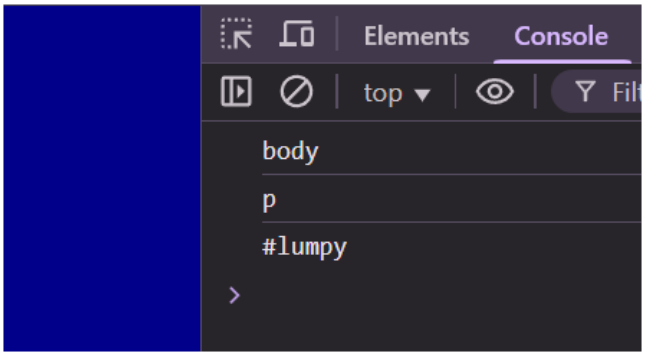
- Sử dụng `document.styleSheets` để lấy tất cả các stylesheet đã được nạp.

```
(property) DocumentOrShadowRoot.styleSheets: StyleSheetList
Retrieves a collection of styleSheet objects representing the style sheets that correspond to each instance of a link or style object in the document.
```

- Duyệt qua từng stylesheet và từng `cssRule` bên trong nó.

```
(property) CSSStyleSheet.cssRules: CSSRuleList
MDN Reference
cssRules) {
```

- In ra `selectorText` của mỗi rule (dùng `console.log()`).



Tips:

- Biến `document.styleSheets` trả về một danh sách các stylesheet.
- `styleSheet.cssRules` chứa danh sách các rule trong stylesheet đó.
- Mỗi rule có thể có `selectorText` thể hiện selector ban đầu.

Bài 5: Event Propagation - Chặn sự ảnh hưởng của sự kiện trong DOM

Hiểu và áp dụng cơ chế **bubbling phase** trong sự kiện DOM, và biết cách sử dụng `event.stopPropagation()` để **ngăn sự kiện ảnh hưởng lên các phần tử cha**.

Yêu cầu:

1. HTML:

- Tạo một **`table`** có **`id="t-daddy"`** với **2 dòng (tr)**:
  - Dòng thứ nhất (**`tr`**) có **`id="tbl1"`** chứa một **`td`** có nội dung **"one"**, **`id="c1"`**.
  - Dòng thứ hai chứa một **`td`** có nội dung **"two"**, **`id="c2"`**.

2. CSS:

- Đặt đường viền đỏ (**`1px solid red`**) cho **`#t-daddy`**.
- Đặt nền màu hồng (**`background-color: pink`**) cho **`#c1`**.

3. JavaScript:

- Viết hàm **`stopEvent(event)`** thực hiện:
  - Thay đổi nội dung của **`#c2`** thành **`"hello"`**.
  - Gọi **`event.stopPropagation()`** để **ngăn sự kiện ảnh hưởng lên table**.

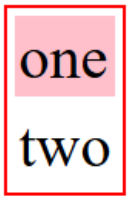
```
// stopPropagation ngăn t-daddy nhận event "click".
event.stopPropagation();
```

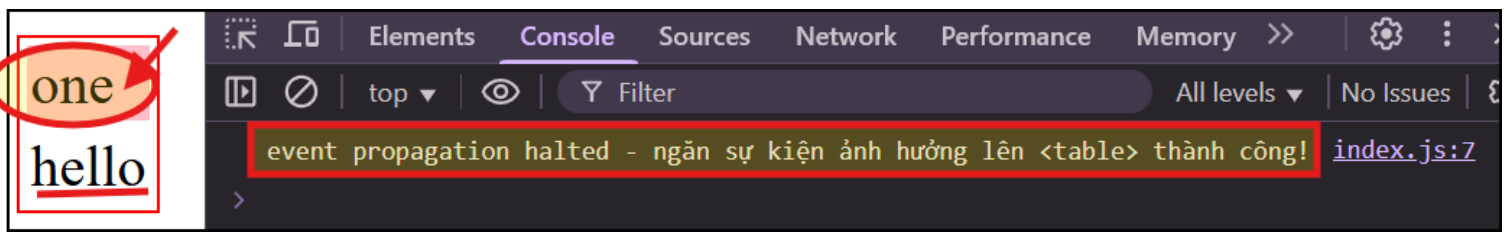
  - In ***"event propagation halted - ngăn sự kiện ảnh hưởng lên <table> thành công!"*** ra console.

- Gắn `event listener` vào:
  - **`#tbl1`** để gọi **`stopEvent()`** khi bấm vào dòng đầu tiên.
    `#t-daddy` để in **`"t-daddy clicked"`** ra console nếu sự kiện không bị chặn.

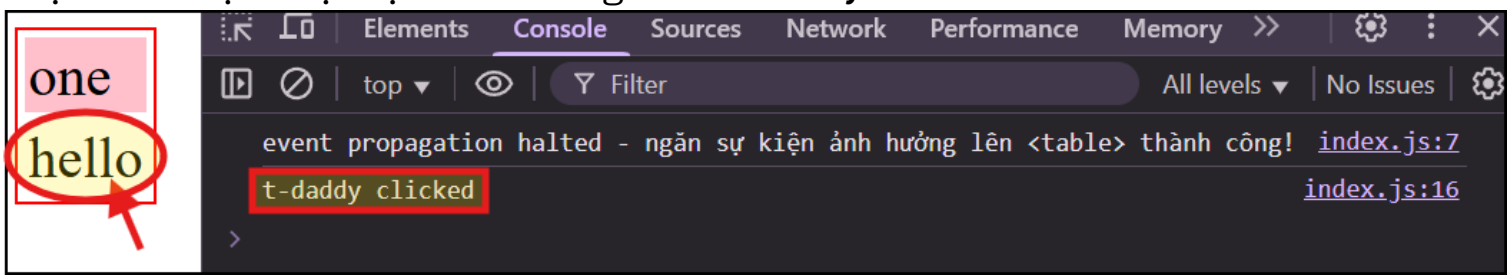
Test-Case:

Khi người dùng **bấm vào dòng đầu tiên (nút one -** one **)**, chỉ nội dung **`#c2`** bị đổi thành **`"hello"`** và không có dòng **`"t-daddy clicked"`** nào xuất hiện trên **`console`**.





Nếu không gọi **`event.stopPropagation()`** trong **`stopEvent()`**, thì ***`"t-daddy clicked"`*** sẽ được hiển thị do sự kiện ảnh hưởng lên **`#t-daddy`**.



Tips:

- Sử dụng **`addEventListener(..., ..., false)`** để lắng nghe sự kiện ở **bubbling phase**.
- Hiểu rằng **mặc định sự kiện click sẽ ảnh hưởng từ element con lên cha**, trừ khi bị chặn bằng **`stopPropagation()`**.

Bài 6: getComputedStyle - Truy xuất giá trị CSS thực tế

Biết cách sử dụng *`window.getComputedStyle()`* để truy xuất các *`CSS property`* đã được áp dụng từ **stylesheet**, thay vì chỉ thông qua *`inline style`*.

Yêu cầu:

1. HTML:

- Tạo một **`div`** có **`id="d1"`** với nội dung rỗng (hoặc dùng **` `**).
- Tạo một **`form`** chứa:
  - Một **`button`** có nội dung: **getComputedStyle**
  - Ba **`input`**:
    - *`input 1: dùng để hiển thị giá trị height của`* ***`div#d1`****, có* ***`id="t1"`***
    - *`input 2: hiển thị max-width,`* ***`id="t2"`***
    - *`input 3: hiển thị background-color,`* ***`id="t3"`***

2. CSS:

- Gán các `CSS property` sau cho **`#d1`** bằng cách viết trong stylesheet (không dùng inline):
  - `margin-left: 10px;`
  - `background-color: rgb(173 216 230);`
  - `height: 20px;`
  - `max-width: 20px;`

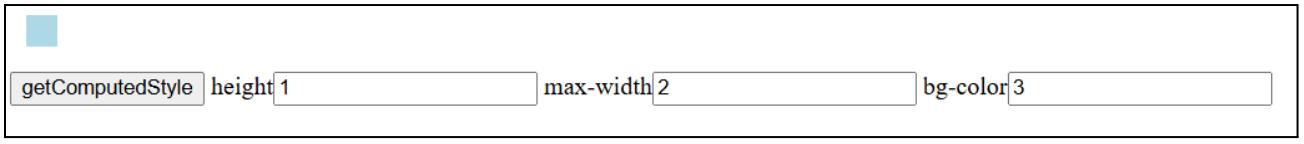
3. JavaScript:

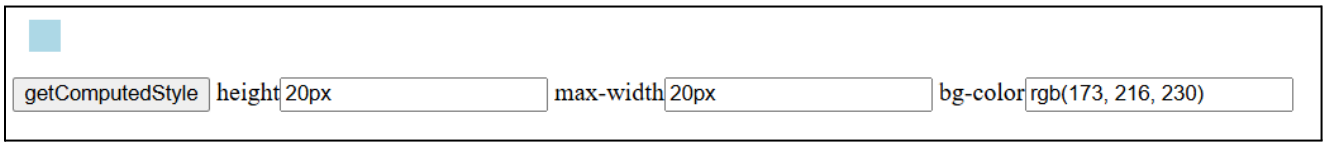
- Viết một hàm tên **`cStyles`** thực hiện:
  - Gọi **`window.getComputedStyle()`** để lấy danh sách tất cả style áp dụng lên **`#d1`**.
  - Dùng **`.getPropertyValue()`** để lấy:
    - *`height`*
    - *`width hoặc max-width`*
    - *`background-color`*
  - Hiển thị từng giá trị này vào 3 `input` tương ứng.
- Gắn sự kiện **`click`** vào nút để gọi hàm **`cStyles`**.

Test-case:

Khi người dùng bấm nút, các **`input`** sẽ được cập nhật để hiển thị:

- Chiều cao thực tế của **`#d1`**
- Chiều rộng thực tế hoặc **`max-width`**
- Màu nền hiện tại





Tips:

- `getComputedStyle(element)` trả về `CSSStyleDeclaration`.
- Dùng `.getPropertyValue("tên-thuộc-tính")` để lấy từng giá trị cụ thể.
- Tránh dùng `.style.propertyName`, vì nó chỉ trả về giá trị trong `inline style`.

Bài 7: Displaying Event Object Properties - Hiển thị tất cả thuộc tính của đối tượng sự kiện

Hiểu cấu trúc của *event object* trong **DOM** và biết cách liệt kê toàn bộ thuộc tính của nó thông qua vòng lặp *for...in*.

Yêu cầu:

1. HTML:

- Tạo **<h1>,** trong đó, **#eventType** sẽ được JavaScript thay đổi để hiển thị type của sự kiện (load, click, v.v.).

```
<h1>Properties of the DOM <span id="eventType"></span> Event Object</h1>
```

2. CSS:

Tạo bảng có style như sau:

- **table** dùng *border-collapse: collapse;*
- **thead** có *font-weight: bold*
- **td** có *padding 2px 10px*
- Các dòng xen kẽ (**tr**) có class "odd" và "even" để tạo hiệu ứng màu nền:
  - ***.odd*** *{ background-color: #efdfef; }*
  - ***.even*** *{ background-color: #ffffff; }*

3. JavaScript:

- Viết một hàm có tên **showEventProperties(event)** thực hiện:
  - Hiển thị tên sự kiện (**event.type**) vào **#eventType**.
  - Tạo một bảng HTML (**<table>**) gồm:
    - Header (**thead**) với 3 cột: ***#, Property, Value***
    - Body (**tbody**) hiển thị từng thuộc tính trong event:
      - *Dùng vòng* ***for...in*** *để duyệt qua các* ***key*** *trong đối tượng* ***event***.
      - *Dòng chẵn gán class "****even****", dòng lẻ gán class "****odd****".*
      - ***Cột 1****: số thứ tự dòng,* ***Cột 2****: tên thuộc tính,* ***Cột 3****: giá trị tương ứng.*
- Gọi **showEventProperties(event)** trong sự kiện **window.onload**.

Test-Case

Khi trang web được tải:

- Một bảng hiển thị đầy đủ tất cả các thuộc tính của đối tượng onload event sẽ xuất hiện bên dưới tiêu đề.
- Người học có thể thấy sự khác nhau nếu chạy trên các trình duyệt khác nhau.

Tips:

- Dùng **`insertCell(-1)`** để thêm ô vào cuối dòng.
- **`event[p]`** có thể trả về giá trị *`undefined`* hoặc hàm; nên in tất cả như chuỗi (**`.toString()`** nếu cần).
- Cân nhắc thêm các phần tử khác và thử gọi **`showEventProperties()`** với sự kiện khác như **`click`**.

 - Chrome

Properties of the DOM load Event Object

#	Property	Value
1	isTrusted	true
2	type	load
3	target	[object HTMLDocument]
4	currentTarget	[object Window]
5	eventPhase	2
6	bubbles	false
7	cancelable	false
8	defaultPrevented	false
9	composed	false
10	timeStamp	153.29999999981374
11	srcElement	[object HTMLDocument]
12	returnValue	true
13	cancelBubble	false
14	NONE	0
15	CAPTURING_PHASE	1
16	AT_TARGET	2
17	BUBBLING_PHASE	3
18	composedPath	function composedPath() { [native code] }
19	initEvent	function initEvent() { [native code] }
20	preventDefault	function preventDefault() { [native code] }
21	stopImmediatePropagation	function stopImmediatePropagation() { [native code] }
22	stopPropagation	function stopPropagation() { [native code] }

 - Firefox

Properties of the DOM load Event Object

#	Property	Value
1	isTrusted	true
2	composedPath	function composedPath() { [native code] }
3	stopPropagation	function stopPropagation() { [native code] }
4	stopImmediatePropagation	function stopImmediatePropagation() { [native code] }
5	preventDefault	function preventDefault() { [native code] }
6	initEvent	function initEvent() { [native code] }
7	type	load
8	target	[object HTMLDocument]
9	srcElement	[object HTMLDocument]
10	currentTarget	[object Window]
11	eventPhase	2
12	bubbles	false
13	cancelable	false
14	returnValue	true
15	defaultPrevented	false
16	composed	false
17	timeStamp	65
18	cancelBubble	false
19	originalTarget	[object HTMLDocument]
20	explicitOriginalTarget	[object HTMLDocument]
21	NONE	0
22	CAPTURING_PHASE	1
23	AT_TARGET	2
24	BUBBLING_PHASE	3
25	ALT_MASK	1
26	CONTROL_MASK	2
27	SHIFT_MASK	4
28	META_MASK	8

Bài 8: Using the DOM table Interface - Thêm dòng và ô vào bảng bằng DOM Table Interface[1]

Thực hành sử dụng các phương thức **`HTMLTableElement.insertRow()`** và **`HTMLTableRowElement.insertCell()`** để tạo thêm dòng và ô trong bảng bằng JavaScript.

Yêu cầu:

1. HTML:

- Tạo một bảng (**`<table>`**) có **`id="table0"`** với cấu trúc ban đầu như sau:
  - Gồm 1 dòng (**`<tr>`**)
  - Trong dòng có 2 ô (**`<td>`**) với nội dung lần lượt là:
    - *`Row 0 Cell 0`*
    - *`Row 0 Cell 1`*

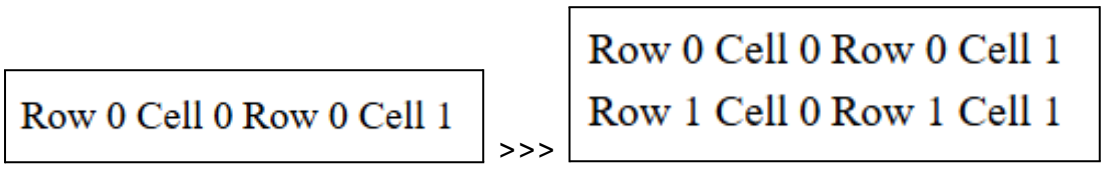
2. JavaScript:

- Viết đoạn mã JavaScript để thực hiện các thao tác sau:
  1. Truy cập vào bảng qua **`getElementById("table0")`**.
  2. Thêm một dòng mới vào **cuối bảng** bằng phương thức **`insertRow(-1)`**.
  3. Thêm **2 ô mới** vào dòng vừa tạo bằng phương thức **`insertCell(-1)`** trong loop.
  4. Mỗi ô có nội dung theo định dạng:
     - *`Row <số thứ tự dòng mới> Cell <chỉ số ô>`*
     - *Gợi ý: dùng `row.rowIndex` để lấy số thứ tự dòng trong bảng.*

Testcase:

Sau khi chạy script:

- Một dòng mới được thêm vào cuối bảng với nội dung hai ô như:



Tips:

- **`insertRow(-1)`** chèn dòng vào cuối **table** (hoặc cuối **`tbody`** nếu có).
- **`insertCell(-1)`** chèn ô vào cuối dòng.
- Dùng **`document.createTextNode()`** để tạo nội dung văn bản trong **`td`**.

---

[1] Ghi chú:

- *Thuộc tính `innerHTML` của một `table` **không bao giờ nên được sử dụng để thay đổi nội dung bảng từng phần**, mặc dù **có thể sử dụng nó để ghi toàn bộ bảng** hoặc **nội dung của một `cell`**.*
- *Nếu sử dụng các phương thức của **DOM Core** như `document.createElement` và `Node.appendChild` để tạo các `row` và `cell`, thì **trình duyệt Internet Explorer yêu cầu các phần tử này phải được gắn vào một `tbody`**, trong khi các trình duyệt khác cho phép gắn trực tiếp vào `table` (trong trường hợp này, các `row` sẽ tự động được thêm vào `tbody` cuối cùng trong `table`).*
- *Có nhiều phương thức tiện lợi khác thuộc về interface `HTMLTableElement` có thể được sử dụng để **tạo mới và thao tác với table**.*