Home

Sample

CCPS 530

Lab 3

Mattew Hackenbrook

500953949

What is Lorem Ipsum?

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla *vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis* vitae, justo. Nullam dictum felis eu pede **mollis pretium**. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi.

An image of a professor who does not exist

Preschool Race Scoreboard

Name	Speed(km/h)	Height(cm)
Tom	4.7	128
Bob	2.1	163

Suspected Cheaters

- Tom
- Godzilla

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for https://matthackenbrook.github.io/

Checker Input

Show ☐ source ☐ outline ☐ image report Options...

Check by address

https://matthackenbrook.github.io/

Check

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

Message Filtering

1. **Error** **Element** `img` **is missing required attribute** `src`.

From line 61, column 21; to line 61, column 61

```
<img class="img-fluid" alt="fake person">↩
```

Attributes for element `img`:

- Global attributes
- `alt` — Replacement text for use when images are not available
- `src` — Address of the resource
- `srcset` — Images to use in different situations, e.g., high-resolution displays, small monitors, etc.
- `sizes` — Image sizes for different page layouts
- `crossorigin` — How the element handles crossorigin requests
- `usemap` — Name of image map to use
- `ismap` — Whether the image is a server-side image map
- `width` — Horizontal dimension
- `height` — Vertical dimension
- `referrerpolicy` — Referrer policy for fetches initiated by the element
- `decoding` — Decoding hint to use when processing this image for presentation
- `loading` — Used when determining loading deferral
- `fetchpriority` — Sets the priority for fetches initiated by the element

NOTE: this is expected as src is being populated dynamically from javascript.

W3C CSS Validator results for https://matthackenbrook.github.io/ (CSS level 3 + SVG)

Congratulations! No Error Found.

This document validates as CSS level 3 + SVG !

To show your readers that you've taken the care to create an interoperable Web page, you may display this icon on any page that


```
<p>
    <a href="https://jigsaw.w3.org/css-validator/check/referer">
        <img style="border:0;width:88px;height:31px"
            src="https://jigsaw.w3.org/css-validator/images/vcss"
            alt="Valid CSS!" />
    </a>
</p>
```


```
<p>
    <a href="https://jigsaw.w3.org/css-validator/check/referer">
        <img style="border:0;width:88px;height:31px"
            src="https://jigsaw.w3.org/css-validator/images/vcss-blue"
            alt="Valid CSS!" />
    </a>
</p>
```

A) I modified my old work to add the changing image, as I already had an image that met the specifications, I modified it to fetch different images on a schedule of 20 seconds. The tag used is a <figure> which contains an <img> and <figcaption> which does the work of centring the text under an image and is industry standard. This is all done under a div which has a bootstrap row class to fit in the grid that the rest of the site abides by.

B) JQuery simplifies everything immensely. Rather than being tied up working on syntax and requirements, I can simply write $.get("url" function(){}); and a get request is done very easily. It is also very easy to navigate, as all I needed to do was add an ID to my figure (which was not required because it was the only figure in the doc, but good practice) and I was able to access the img and figcaption that were under it without adding ids to them as well.

C) I tested on firefox and chrome as stated before if it works on them it works everywhere.

D) I spent about 4 hours on this lab. Mostly misunderstanding the requirements, as I initially believed that we were being asked to get binary data and convert it to an image with an AJAX call, which doesn't make much sense to me, and would not work. Once I realized that AJAX and the changing image were separate requirements I was able to complete the lab in a half hour, as I got the list of files in my directory and looped through them (which was difficult to test and I needed to set up a local site on IIS to do so without errors), creating a caption dynamically based on the title of the images.