TASK

CSS – Styles, Selectors, and Box Model

Visit our website

Copyright © 2024 HyperionDev. All rights reserved.

Introduction

WELCOME TO THE CSS – STYLES, SELECTORS, AND BOX MODEL TASK!

You have likely by now wanted to be able to change the colour/font of text, or you may have wanted to change the size of an image or centre it. In this task, you will learn how to style your HTML elements. This task will introduce you to the basics of Cascading Style Sheets (CSS) and how to format various HTML elements using “in-line” and “internal” as well as “external” approaches. We’ll also place close attention on the relevance of CSS selectors. We will also explore the box model, which is a fundamental design and layout concept we use when deciding how to arrange the features on our web page.

INTRODUCTION TO CSS

Cascading Style Sheets (CSS) is a language that is used to change the presentation and look of a particular document that has been written in a markup language, such as HTML.

INLINE STYLE

HTML **elements** are described using **attributes** and **properties**. You can style a web page by changing the properties of the elements that make up that web page. For example, any text that you add to a web page has several properties that you can change. These include the following properties:

- **font-family**: Specifies the font for the element (Arial, Times New Roman, etc).
- **font-style**: Sets whether a font should be styled with a standard, *italic,* or **oblique** face from its font-family.
- **font-size**: Sets the size of the font that can be specified as a fixed size in various units, a percentage, or as a predefined keyword.

An example of using the style attribute to change the font of an element is shown below:

```
<p style="font-family: 'Arial'; font-style: italic; font-size: 46px;">
      This is the paragraph where I describe myself.
</p>
```

Copyright © 2024 HyperionDev. All rights reserved.

Like all other attributes, the style attribute goes inside the element's beginning tag, right after the tag name. After specifying that you are changing the **style** attribute, you type **=**, and then, within double quotes, list the properties you want to change and after a colon, specify the value for that property:

<p style="font-family: Arial; font-style: italic; font-size: 46px;">

Attribute	Property	Value
style	**font-family**	**Arial**
	font-style	**Italic**
	font-size	**46px**

When you style an element individually by changing that element's properties, it is known as **inline styling**. Inline styling allows you to specify the style of an individual element in the line where that element is declared. What if you wanted to apply similar styles to all elements of a certain type, though? For example, what if you wanted to change the font of all paragraphs on your web page? You can do this by creating a CSS rule.

INTERNAL CSS

The example below shows how you can define a CSS rule in the **head** part of your HTML template. This is called **internal CSS**. The example below shows a CSS rule that will cause all paragraphs to be in the colour red and be of the font-family Arial. If the browser can't find Arial, then it will look for Helvetica:

```
<head>
      <style>
            p {
                color: red;
                font-family: Arial, Helvetica;
            }
      </style>
</head>
<p style="font-family: 'Arial'; font-style: italic; font-size: 46px;">
      This is the paragraph where I describe myself.
</p>
```

Copyright © 2024 HyperionDev. All rights reserved.

CSS follows the following syntax:

A style sheet consists of a selector and a declaration.

- The **selector** indicates which element you want to style. In the example above we are selecting the **p** element.
- The **declaration** block contains one or more declarations separated by semicolons. Examples of a declaration, as shown above, is:
  - **color: red;**
  - **font-family: Arial, Helvetica;**

  Declaration blocks should always be surrounded by curly braces.
- Each declaration includes a **property** and a **value**, separated by a colon.

See what other properties can be modified using CSS **here**.

You could use a combination of internal CSS (declared in the head of your HTML document) and inline style. How would the style rules apply? Essentially, the closer to the element the style is, the higher the precedence. For example, if you had the internal CSS rule shown in the code above on your page but you wanted one paragraph to be styled differently from the rest, you would simply use inline style for that one paragraph and that would *overwrite* the rule specified by the internal CSS.

EXTERNAL CSS

If your website consists of many HTML files, you are likely to want to be able to apply the same style rules to all the web pages. To accomplish this you would need to use external CSS instead of internal CSS. To do this, **create a separate file** with the extension .css. Within this file write all the style rules that you would like to specify. You can then link this external CSS file to all the HTML files in which you would like the style rules applied. To link an external CSS file (called examplesCSS.css in this example) to a specific HTML file, do the following:

```
<link href = "examplesCSS.css" rel = "stylesheet" type = "text/css"/>
```

In the <head> part of your HTML create a reference to your CSS file so that the styles can be used in your web page. Here, “href” refers to the name and path of your CSS file. In the example above, the file *exampleCSS.css* is in the same folder as the HTML page; “rel” says what sort of relation the file is to the HTML, i.e. the stylesheet; and finally, “type” tells the browser what sort of file it is and how to interpret it. In modern browsers, the “type” attribute is not required anymore for

HyperionDev Copyright © 2024 HyperionDev. All rights reserved.

linking a CSS file to HTML, however, if you need to support very old browsers, consider including the “type” attribute for additional clarity.

INTERNAL, EXTERNAL, OR INLINE: WHICH APPROACH IS THE BEST?

If we were to include the CSS shown in the image below in our CSS file, the result would be the same as if it were in the <style> tags on the HTML page. Is it better to use internal or external CSS? Generally, it is ***better to use external CSS wherever possible***. Why? *Readability* is an important factor. Imagine trying to read through a whole bunch of different languages at once (CSS, HTML, JavaScript) all in one file. By separating them – it’s much easier to follow what’s happening, especially when you are building a fancy website where plenty of different styles are being used.

```
p {
      color: red;
      font-family: Arial, Helvetica;
}

body {
      text-align: center;
}
```

Another important reason to separate CSS from HTML files is to improve the *maintainability* of your website. If only external CSS is used for a website, you could update the look and feel of the website easily by simply replacing the external CSS file. Using external CSS also makes it easier to *debug* errors since all the CSS is in one place. You may find, though, that it is necessary to use a combination of external, internal, and inline styles.

Extra resource

Check out this excellent **extra Resource** by W3C on **Web Style Sheets CSS tips & tricks: Centering things**.

Copyright © 2024 HyperionDev. All rights reserved.

CSS SELECTORS

A CSS selector attaches to the HTML elements on our page allowing for customised styling. Let’s take a closer look at some of the most common CSS selectors you’re likely to encounter and use.

Element selector

The element selector is the most basic type of CSS selector. It allows you to specify the precise HTML element you wish to style. This selector pinpoints an element tag and applies the same style to each element with that specific tag name. Notice that this selector **does not have a unique name** – it is simply named according to its tag name.

```
<head>
   <style>
      p {
            text-align: center;
            color: blue;
      }
   </style>
</head>
<body>
  <p>This style will be applied on every paragraph.</p>
  <p id="para1">Me too!</p>
  <p>And me!</p>
</body>
```

ID selector

An **id** selector calls an HTML element by its unique id name. It’s unique because you **can't give the same id name** to any other HTML element in the same web page. The id selector is an attribute inserted at the beginning of an HTML tag. Notice that inside the double quotation, the value of **id** is given. Also notice that the **id** selector is called using hash (**#**), followed by id name (head1) and (para).

```
<style>
    #head1 { color: green }
    #para { color: red; background: yellow }
    p { color: blue }
</style>
```

HyperionDev Copyright © 2024 HyperionDev. All rights reserved.

```
<body>
    <h3 id="head1">Okay, let’s make this h3 tag green.</h3>
    <p id="para">Now let’s make use of our #para declarations.</p>
    <p>This paragraph does not have an id.</p>
</body>
```

The result will look something like this:

Okay, let’s make this h3 tag green.
Now let’s make use of our #para declarations.
This paragraph does not have an id.

Class selector

The class selector differs slightly from an id selector. Whereas an id selector pinpoints an individual HTML element, a **class** selector aims to change **all** HTML elements associated with that class. You’ll also notice that a class selector begins with a ‘**.**’ (dot).

```
<head>
   <style>
      p.right_align {
        text-align: right;
        color: blue;
        font-weight: bold;
      }
   </style>
</head>
<body>
<h1 class="right_align">This h1 text will not change because it does not
contain the p tag.</h1>
<p class="right_align">This paragraph will change to blue, bold, and
align to the right.</p>
</body>
```

The result will look something like this:

This h1 text will not change because it does not contain the p tag.

This paragraph will change to blue, bold, and align to the right.

Copyright © 2024 HyperionDev. All rights reserved.

Each selector gives us a greater degree of customisation. You may not see it just yet, but when you're working with multiple files and style elements, these handy selectors will save you that pain of manually adding styling for each line of code you write.

These selectors are the building blocks for more complex styling choices. Okay, let's have a look at the illustrations below for a clearer view of selectors and other CSS styling:

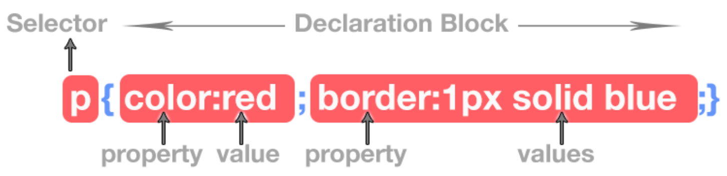


Image source: ***https://tutorial.techaltum.com/cssselectors.html***

There's no need to memorise the image below, but you may find it helpful to learn in order to recognise the difference between a declaration, id, class, and HTML element:

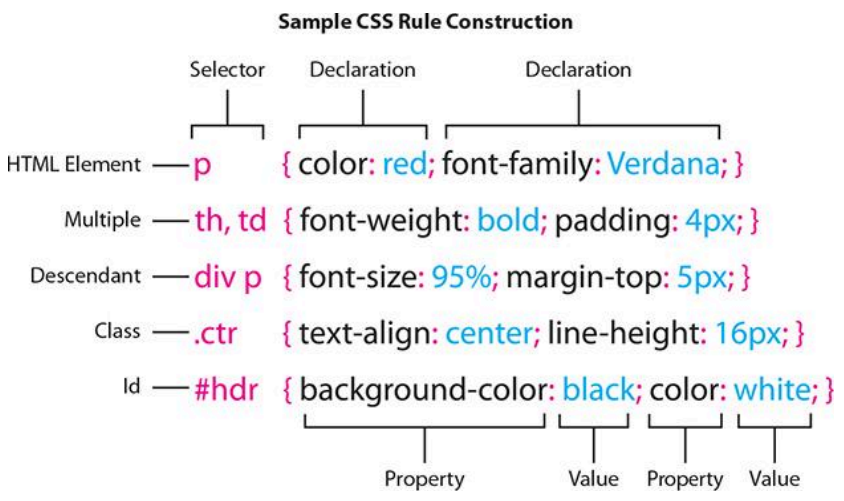

Sample CSS Rule Construction
Selector
Declaration
Declaration
HTML Element — p { color: red; font-family: Verdana; }
Multiple — th, td { font-weight: bold; padding: 4px; }
Descendant — div p { font-size: 95%; margin-top: 5px; }
Class — .ctr { text-align: center; line-height: 16px; }
Id — #hdr { background-color: black; color: white; }
Property
Value
Property
Value

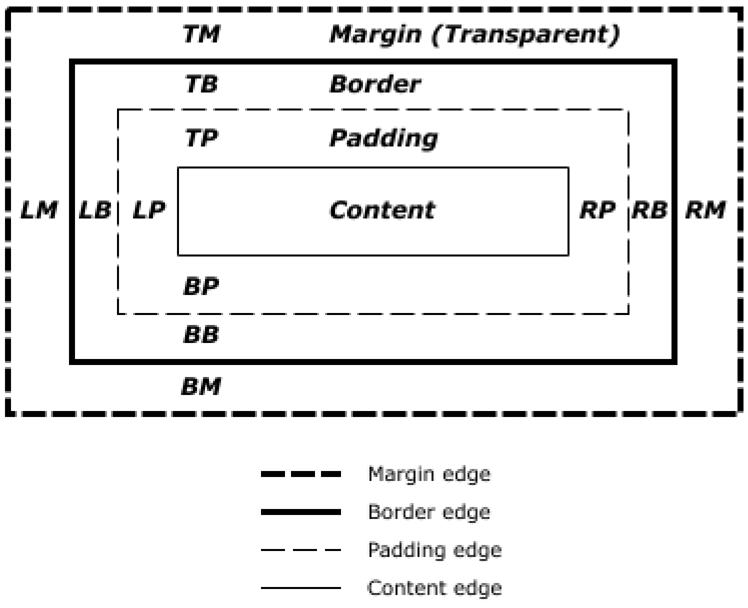
Image source:
https://www.adobepress.com/articles/article.asp?p=2999389&seqNum=5

THE BOX MODEL

An important fundamental concept to understand in CSS is what is called the **box model**. The browser creates a rectangle for each element in the HTML document. The box model describes how the padding, border, and margin are added to the

Copyright © 2024 HyperionDev. All rights reserved.

content to create this rectangle. The following diagram is a depiction of the box model for an element.



Each box has a content area (e.g. text, image, etc.) and optional surrounding padding, border, and margin areas; the size of each area is specified by properties defined below.

The margin, border, and padding can be broken down into top, right, bottom, and left segments (e.g. in the diagram, "LM" for left margin, "RP" for right padding, "TB" for the top border, etc.).

The perimeter of each of the four areas (content, padding, border, and margin) is called an "edge", so each box has four edges:

Content edge or inner edge
The content edge surrounds the rectangle given by the width and height of the box, which often depends on the element's rendered content. The four content edges define the box's content box.

Padding edge
The padding edge surrounds the box padding. If the padding has zero width, the padding edge is the same as the content edge. The four padding edges define the box's padding box.

Border edge

HyperionDev
Copyright © 2024 HyperionDev. All rights reserved.

The border edge surrounds the box's border. If the border has zero width, the border edge is the same as the padding edge. The four border edges define the box's border box.

Margin edge or outer edge
The margin edge surrounds the box margin. If the margin has zero width, the margin edge is the same as the border edge. The four margin edges define the box's margin box.

Each edge may be broken down into a top, right, bottom, and left edge.

Let's put this into practice! Create a folder named **"boxModel"**, and in this folder create an HTML file named "**index. Html**". Copy the following code into the index.html file.

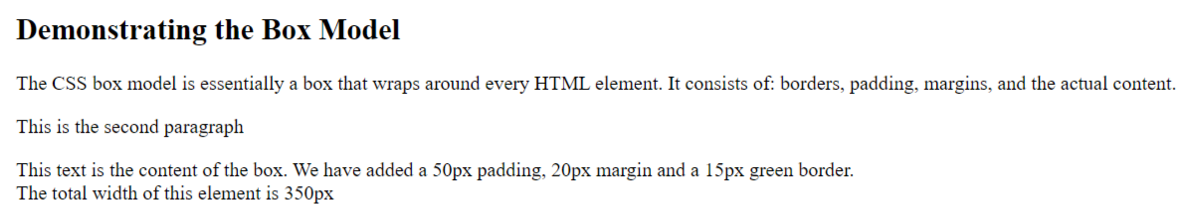
```html
<!DOCTYPE html>
  <head>
    <title>Static Template</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h2>Demonstrating the Box Model</h2>

    <p id="firstParagraph">
      The CSS box model is essentially a box that wraps around every HTML
      element. It consists of: borders, padding, margins, and the actual
      content.
    </p>

    <p class="secondParagraph">This is the second paragraph </p>

    <div class="div1">
      This text is the content of the box. We have added a 50px padding, 20px
      margin and a 15px green border.
    </div>
    <div class="div2">The total width of this element is 350px</div>
  </body>
</html>
```

Copyright © 2024 HyperionDev. All rights reserved.

Now in the same folder create a file named "**style.css**". We perform the link to this external stylesheet by using the following line of code in the **head section** of the "**index.html**" file.

```
<link rel="stylesheet" href="style.css" />
```

With no CSS code in the CSS file, when index.html is opened in the browser, it should look like this.


Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

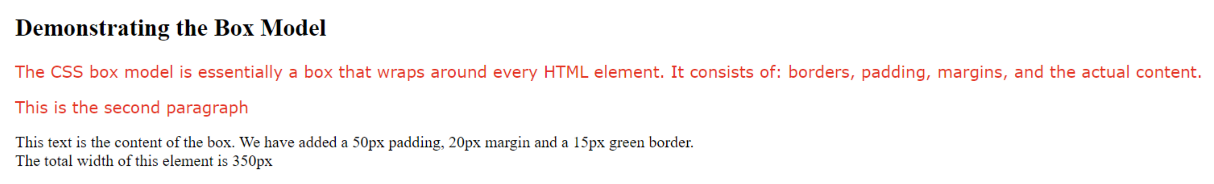
This is the second paragraph

This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.
The total width of this element is 350px

Let's change the CSS properties of all of the paragraph tags in the HTML file. We will use an element selector, "p", and then change the font to "Verdana" and the font colour to "red". You can copy and paste this into the CSS file.

```
p {
  font-family: Verdana;
  color: red;
}
```

After saving and refreshing the browser, both paragraphs should change accordingly:


Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

This is the second paragraph

This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.
The total width of this element is 350px

Now let's style the first div element. (The <div> HTML element is the generic container for flow content. It has no effect on the content or layout until styled in some way using CSS; it is also known as a content division element.)

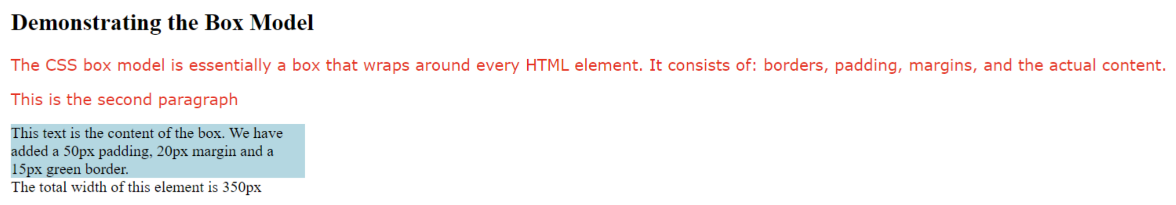
Let's set the background colour and width of the element first. We will use the class selector for the first div element with the class name div1, and change the background colour to **lightblue**, and the width of the element to 300px. Copy and paste the following into the CSS file, save, and look at the changes in the rendered web page after reloading the browser.

HyperionDev

Copyright © 2024 HyperionDev. All rights reserved.

```
.div1 {
  background-color: lightblue;
  width: 300px;
}
```

The result should be as follows:



Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

This is the second paragraph

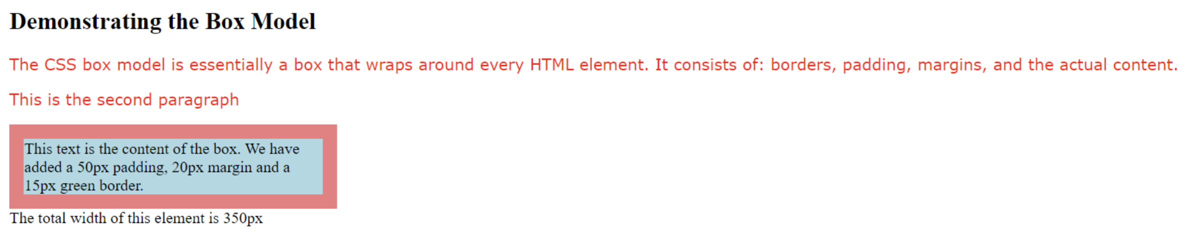
This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.

The total width of this element is 350px

You will now notice that the background colour has changed and the container in which the text is situated now has a width of 300px. Let's add a border to the same div container. We will make the border width 15px, the border type solid and the colour lightcoral.

```
.div1 {
  background-color: lightblue;
  width: 300px;
  border: 15px solid lightcoral;
}
```

We should see the following change as this is applied:



Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

This is the second paragraph

This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.

The total width of this element is 350px

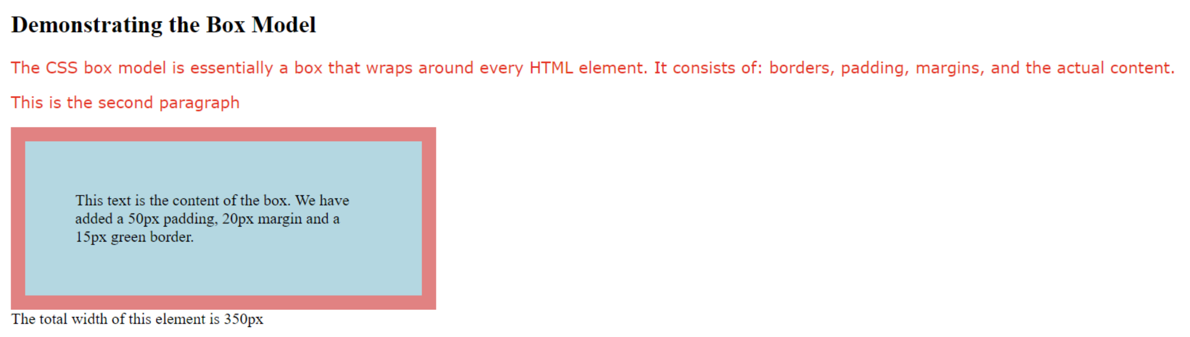
You will notice that there is no spacing between the border and the text (we need to add padding) and the border and the div element below it (with the content that reads "The total width of this element is 350px"). Let's add padding of 50 pixels around the text:

```
.div1 {
  background-color: lightblue;
  width: 300px;
  padding: 50px;
```

HyperionDev Copyright © 2024 HyperionDev. All rights reserved.

```
  border: 15px solid lightcoral;
}
```

Have a look a the result:


Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

This is the second paragraph

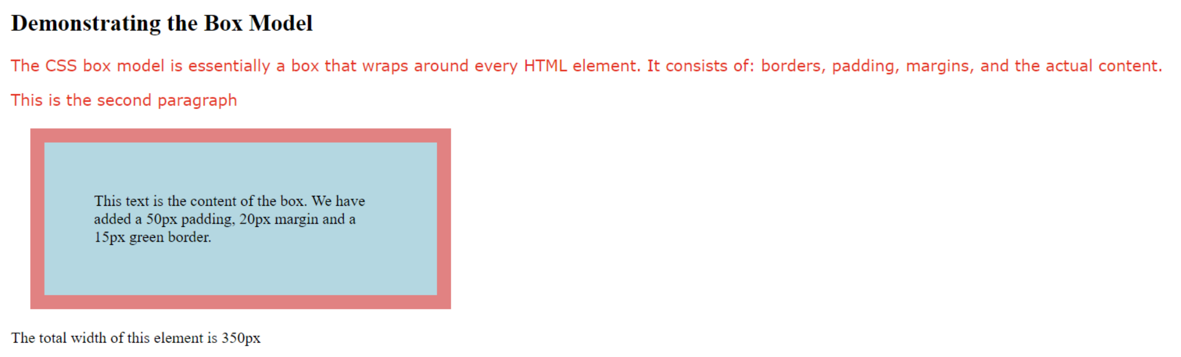
This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.

The total width of this element is 350px

Now let's add a margin around the border of 20 pixels:

```
.div1 {
  background-color: lightblue;
  width: 300px;
  padding: 50px;
  border: 15px solid lightcoral;
  margin: 20px;
}
```

The result is spacing around the element:


Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

This is the second paragraph

This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.

The total width of this element is 350px

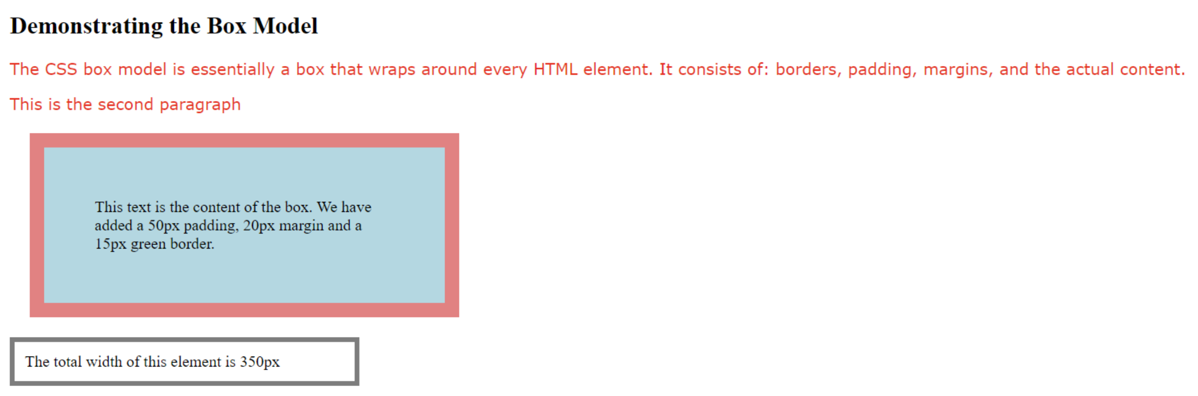
Now let's add styling to our second div element with the class name "div2":

```
.div2 {
  width: 320px;
  padding: 10px;
  border: 5px solid gray;
  margin: 0;
```

HyperionDev Copyright © 2024 HyperionDev. All rights reserved.

```
}
```

The result is as follows:


Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

This is the second paragraph

This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.

The total width of this element is 350px

It is also possible to have different pixel sizes for the left, right, top, and bottom padding and margins by using the following CSS properties:

- padding-top
- padding-right
- padding-bottom
- padding-left
- margin-top
- margin-right
- margin-bottom
- margin-left

You are encouraged to explore these using the files that you created so that you can see how the changes occur in the browser.

CASCADE

As we know, CSS stands for Cascading Style Sheets. You may have wondered why they are called *cascading* style sheets. 'Cascading' refers to how the rules are applied.

If your website contains external, internal, and inline CSS, inline CSS overrides internal CSS rules and external CSS files. Internal CSS overrides external CSS rules. If there are conflicting rules regarding properties, properties override other properties, but entire rules don't override other rules. When several CSS rules match the same element, they are all applied to that element. Only after that are any conflicting properties evaluated to see which individual styles will win over others.

Copyright © 2024 HyperionDev. All rights reserved.

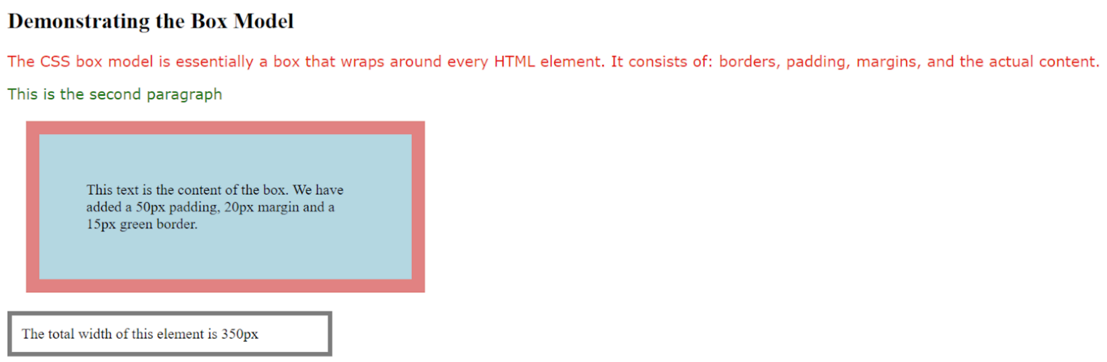
Another important rule to remember is that **the more specific a rule is, the higher its precedence**. For example, in a stylesheet that uses element selectors, class selectors, and ID selectors, *element selectors are the least specific* (because they could match the most elements on a page) whereas ID selectors are the most specific. Therefore, ID selectors will be applied over class selectors and element selectors.

Let's see this in action using the files from the box model demonstration which you have created.

Earlier we used the element selector “p” to style all paragraph elements red. We can override this by being a bit more specific and changing the colour of all paragraphs of a specific class. Let's do this for the “secondParagraph” class and change it to green:

```
.secondParagraph {
  color: green;
}
```

We can now see that the second paragraph colour changed despite the red colour assigned to all paragraphs:


Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

This is the second paragraph

This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.

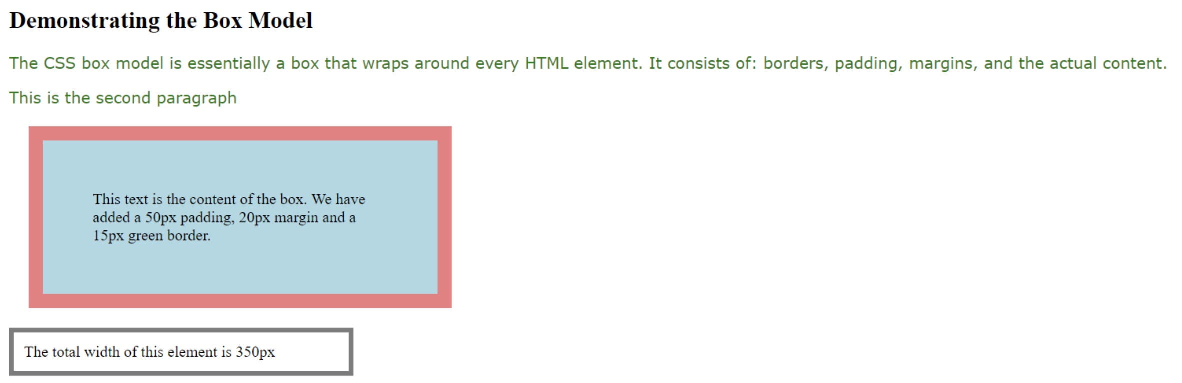
The total width of this element is 350px

Now let's give the first paragraph the same class name as the second paragraph:

```
    <p id="firstParagraph" class="secondParagraph">
      The CSS box model is essentially a box that wraps around every HTML
      element. It consists of: borders, padding, margins, and the actual
      content.
    </p>

    <p class="secondParagraph">This is the second paragraph </p>
```

Copyright © 2024 HyperionDev. All rights reserved.

Both paragraphs will now be green.



Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

This is the second paragraph

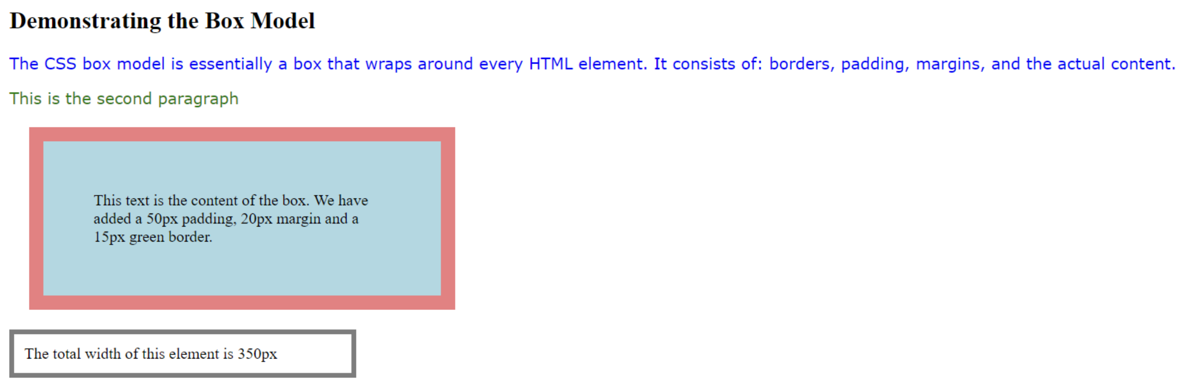
This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.

The total width of this element is 350px

Lastly, we can see that the first paragraph has the id “firstParagraph”. If we use an id selector and change the colour of this id to blue, we will see the class colour get overridden. Add the following to your CSS:

```
#firstParagraph {
  color: blue;
}
```

The result:



Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

This is the second paragraph

This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.

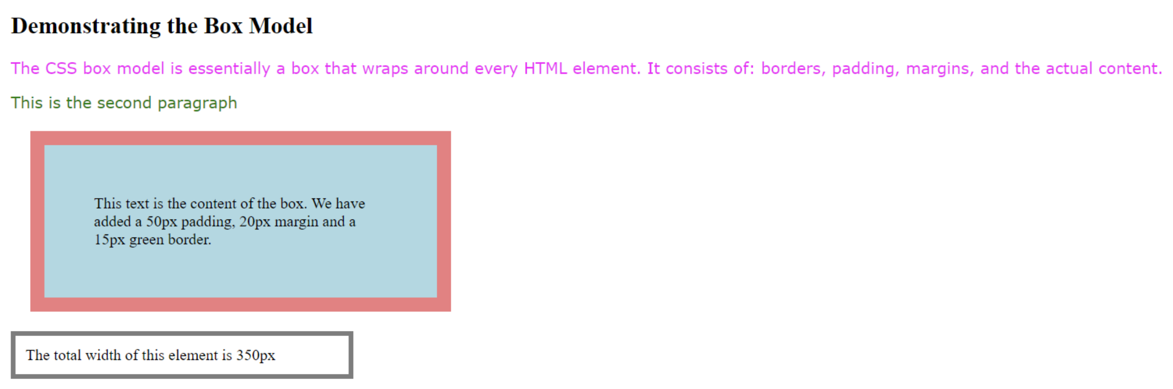
The total width of this element is 350px

Lastly, the most specific type of style is inline styling, which will override any other styles applied to an element. Let's add inline styling to the first paragraph by changing the colour to magenta, and consider the result:

```
    <p id="firstParagraph" class="secondParagraph" style="color: magenta;">
      The CSS box model is essentially a box that wraps around every HTML
      element. It consists of: borders, padding, margins, and the actual
      content.
    </p>
```

The result:

Copyright © 2024 HyperionDev. All rights reserved.


Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.

This is the second paragraph

This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border.

The total width of this element is 350px

CSS VALIDATOR

As a developer, you have no doubt come to realise that with HTML it is very important to follow the syntax rules when developing websites. The same is true of CSS.

You need to follow the rules for formatting your CSS rules precisely or unexpected errors will occur when you try to view your web page in the browser. Examples of common errors include spelling the name of an element incorrectly, not having matching opening and closing braces { }, or leaving out semicolons, "**;**", or colons, "**:**".

You will eventually make mistakes that will violate these rules and that will cause problems when you try to view web pages in the browser. Look, we all make syntax errors – often! Being able to identify and correct these errors becomes easier with time and is an extremely important skill to develop.

To help you identify errors in your CSS, use this helpful **tool**.

A note from the
HyperionDev Team

Please have a look at the following video playlist which explains CSS, **here**.

Copyright © 2024 HyperionDev. All rights reserved.

Instructions

Open all the examples in the directory called “Examples” for this task and read through the comments before attempting these tasks. Also, please consult the **additional reading**. This additional reading is a resource provided by the World Wide Web Consortium (W3C).

Take note:

The task(s) below is/are **auto-graded**. An auto-graded task still counts towards your progression and graduation.

Give it your best attempt and submit it when you are ready.

You will receive a 100% pass grade once you’ve submitted the task.

When you submit the task, you will receive an email with a link to a model answer, as well as an overview of the approach taken to reach this answer. Take some time to review and compare your work against the model answer.

In the same email, you will also receive a link to a survey for this task, which you can use as a self-assessment tool. Please take a moment to complete the survey. This exercise will help solidify your understanding and provide an opportunity for reflection on how to apply these concepts in future projects.

Once you’ve done that, feel free to progress to the next task.

Copyright © 2024 HyperionDev. All rights reserved.

Auto-graded Task 1

In this task, you will create a tribute page. This web page will be about someone you admire in your life. You can find an example of a tribute page **here**.

- Create files called **tribute.html** and **myStyles.css** in this folder. The **tribute.html** file should include:
  - A title or a heading,
  - An image,
  - A caption for the image, and
  - A timeline of the life of the tribute in the form of a list.
- Set out the basic HTML document template and title it as you see fit.
- Use the following eight elements at least once:
  - h1
  - p
  - img
  - h2
  - h3
  - nav (find help **here** and **here**.)
  - header
  - aside
- Depending on the content you have chosen, align some on the left, some on the right, and some in the middle.
- The nav bar should be centred and the elements placed next to each other (horizontally).
- Change the background colour of the entire page. *Hint: Think about where all of the content is kept.*

HyperionDev
Copyright © 2024 HyperionDev. All rights reserved.

- Make all the headings appear in italics, and change their font family to Times New Roman.
- The following should be added to your elements (1 per element)
    - Text-colour
    - Image-size
    - Border
    - Padding
- Before submitting your code, check it with the HTML and CSS validators located **here** and **here**.

Auto-graded Task 2

Let's add to the tribute page you created in the previous task. Remember, you can find an example of a tribute page **here**.

- You'll be working with your **tribute.html** and **myStyles.css files** from your previous task. Now in your **tribute.html** file:
- Apply styling to each of the following elements as described:
    - Ensure your h1 tags have padding of 20 pixels left and right, and 10 pixels top and bottom.
    - Change your h1 tags to brown.
    - Insert a blue border around all images you've used.
    - Use an id selector (id="para1") to change the colour of your first paragraph to 'navy'.
    - Add two sets of 'ul' items to your document and highlight them with CadetBlue.
    - Change your caption(s) for any images to bold, centred, and 20 pixels in size.
    - Give all div elements rounded corners or 5 pixels.

HyperionDev
Copyright © 2024 HyperionDev. All rights reserved.

- Ensure that all h2 tags are italic.
- All h3 tags should have a width of 25%.
- Ensure that all aside tags are indented by 20 pixels.
- Select or create a paragraph and add a 'spring green', mixed border of 5 pixels to it.
- Add a class selector with these properties and values:
    - Font size of 25 pixels
    - Line height of 20 pixels
    - Colour should be 'plum'
    - Font type should be monospace
    - Padding of 4 pixels
    - Left margin of 6 pixels

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.

REFERENCES

CSS reference - CSS: Cascading Style Sheets | MDN. Mozilla.org. Published June 16, 2022. Accessed June 28, 2022. **https://developer.mozilla.org/en-US/docs/Web/CSS/Reference**

Copyright © 2024 HyperionDev. All rights reserved.

Powell K. HTML & CSS for beginners. YouTube. Published online 2022. Accessed June 28, 2022. **https://www.youtube.com/playlist?list=PL4-IK0AVhVjM0xE0K2uZRvsM7LkIhsPT-**

Pinimg.com. Published 2022. Accessed June 29, 2022. **https://i.pinimg.com/originals/ab/6b/61/ab6b61d2f4197fb41a07fa04038836df.png**

Copyright © 2024 HyperionDev. All rights reserved.