CSS-Basics

By G.S.Farooq

Topics:

Selectors

Box Model

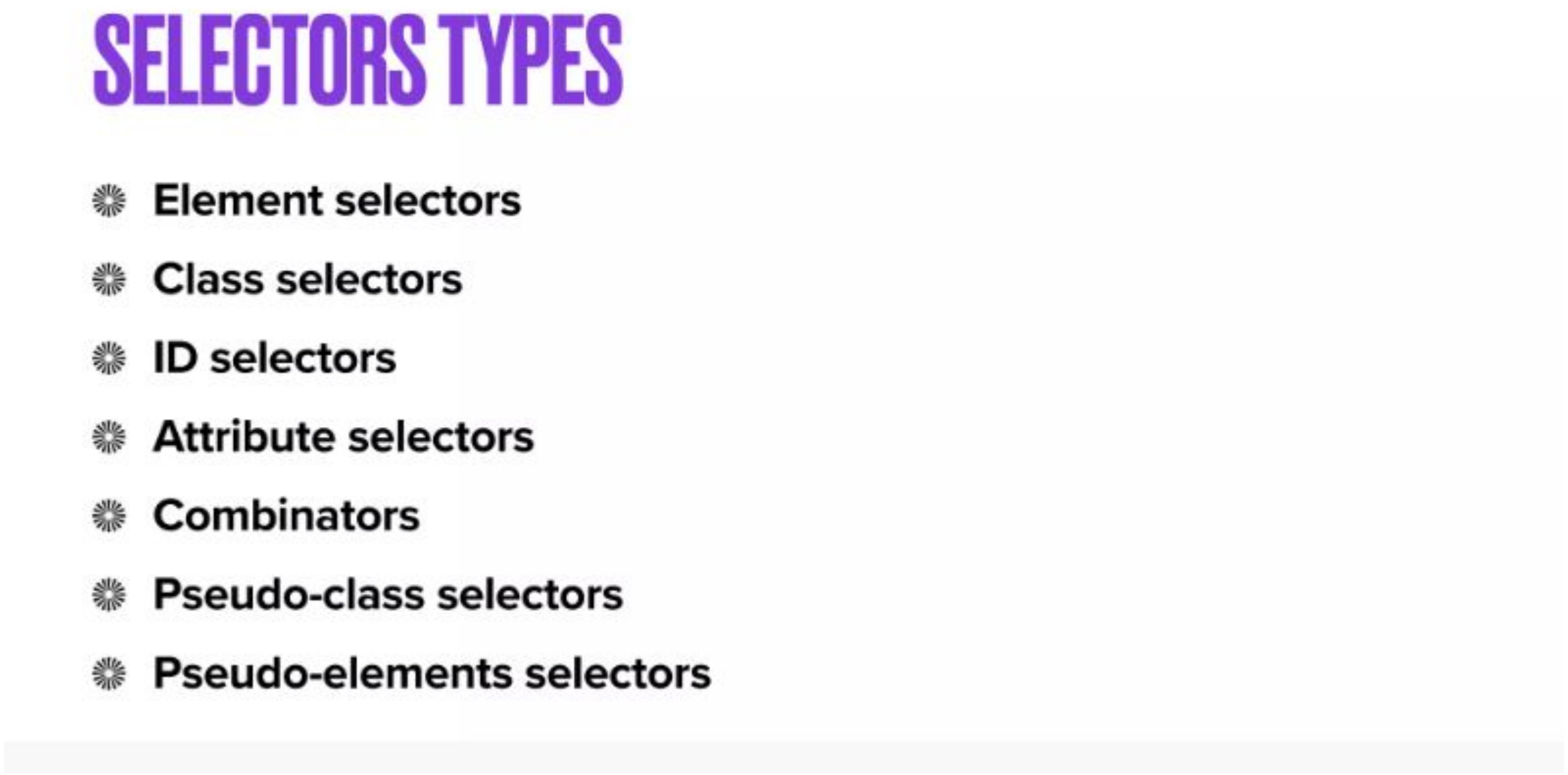
Borders

Positions

Text and Typography

Specificity

CSS-Selectors

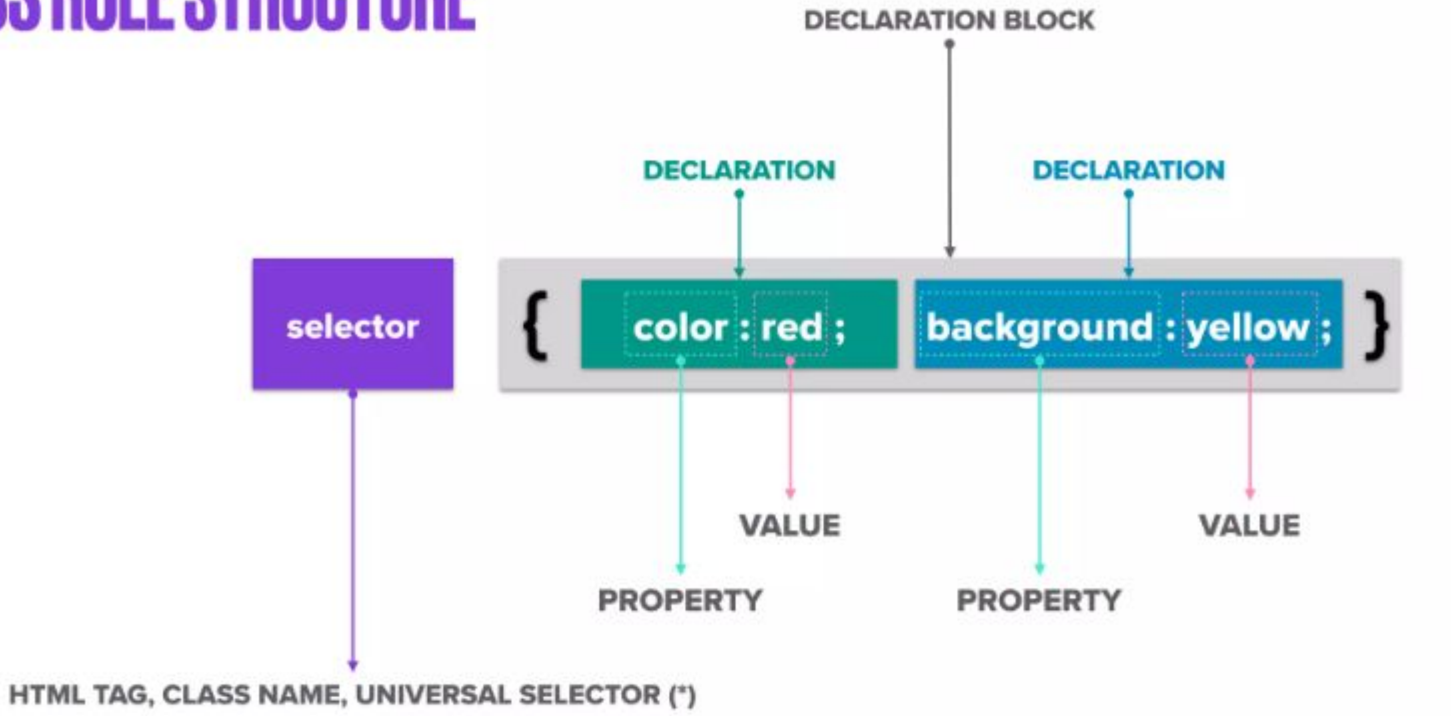


SELECTORS TYPES

- Element selectors
- Class selectors
- ID selectors
- Attribute selectors
- Combinators
- Pseudo-class selectors
- Pseudo-elements selectors

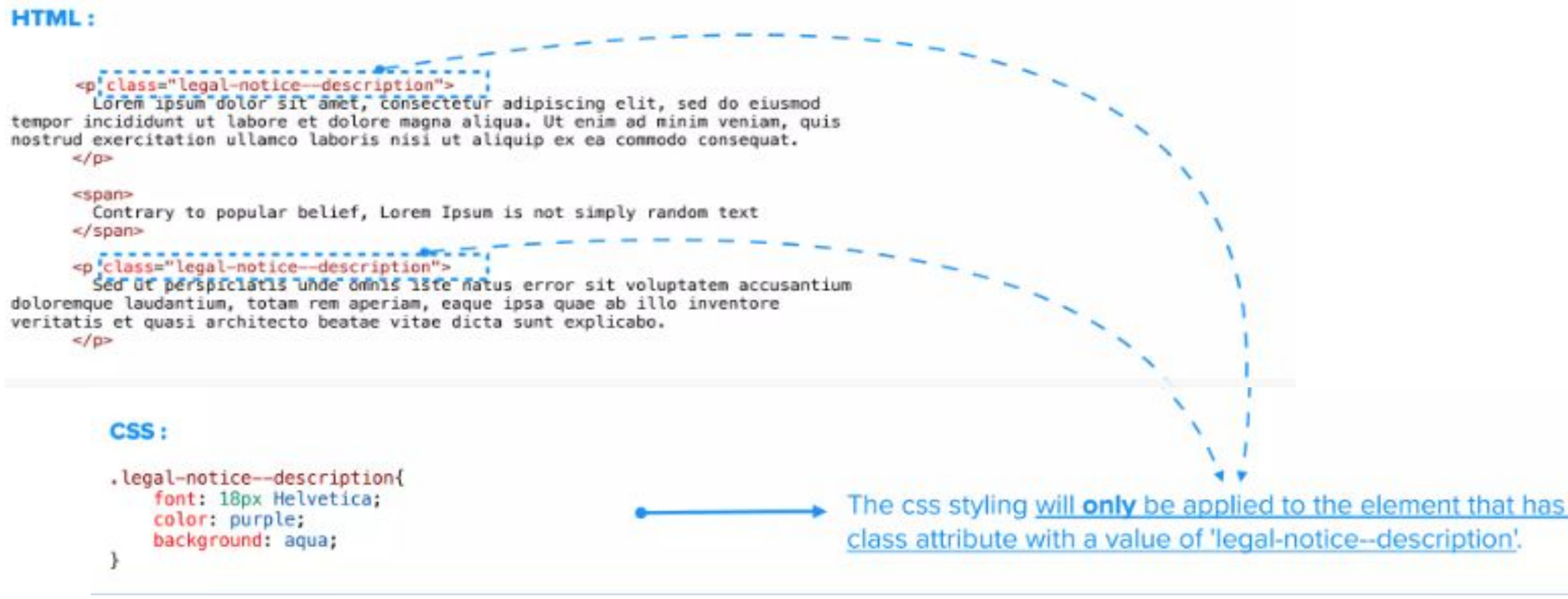
Save slide

CSS RULE STRUCTURE


DECLARATION BLOCK
DECLARATION
DECLARATION
selector
{ color : red ; background : yellow ; }
VALUE
VALUE
PROPERTY
PROPERTY
HTML TAG, CLASS NAME, UNIVERSAL SELECTOR (*)

CLASS SELECTORS

.class-name { property: value; property: value; }



HTML :

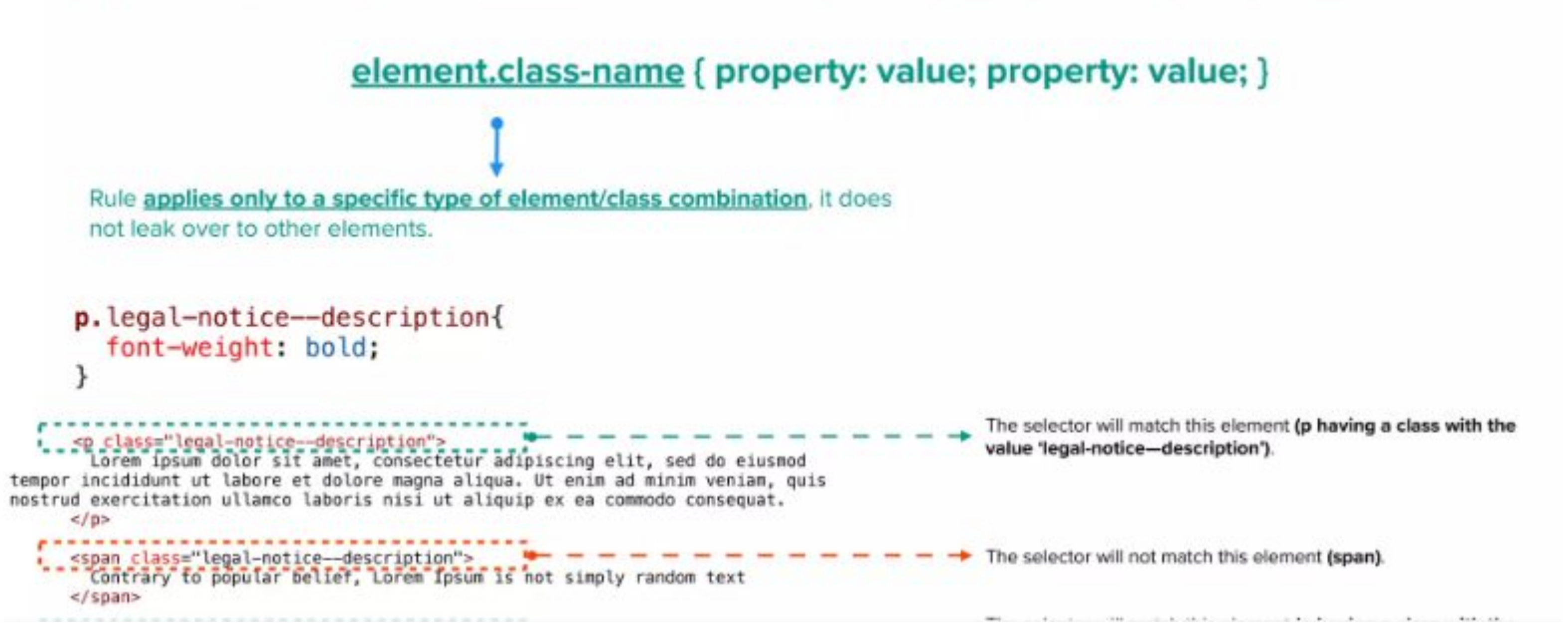
```
      <p class="legal-notice--description">
        Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
      </p>

      <span>
        Contrary to popular belief, Lorem Ipsum is not simply random text
      </span>

      <p class="legal-notice--description">
        Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium
doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore
veritatis et quasi architecto beatae vitae dicta sunt explicabo.
      </p>
```

CSS :

```
.legal-notice--description{
    font: 18px Helvetica;
    color: purple;
    background: aqua;
}
```

The css styling will **only** be applied to the element that has
class attribute with a value of 'legal-notice--description'.

COMBINING ELEMENT AND CLASS SELECTORS



element.class-name { property: value; property: value; }

Rule **applies only to a specific type of element/class combination**, it does not leak over to other elements.

```
p.legal-notice--description{
  font-weight: bold;
}
```

```
  <p class="legal-notice--description">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
  </p>
```

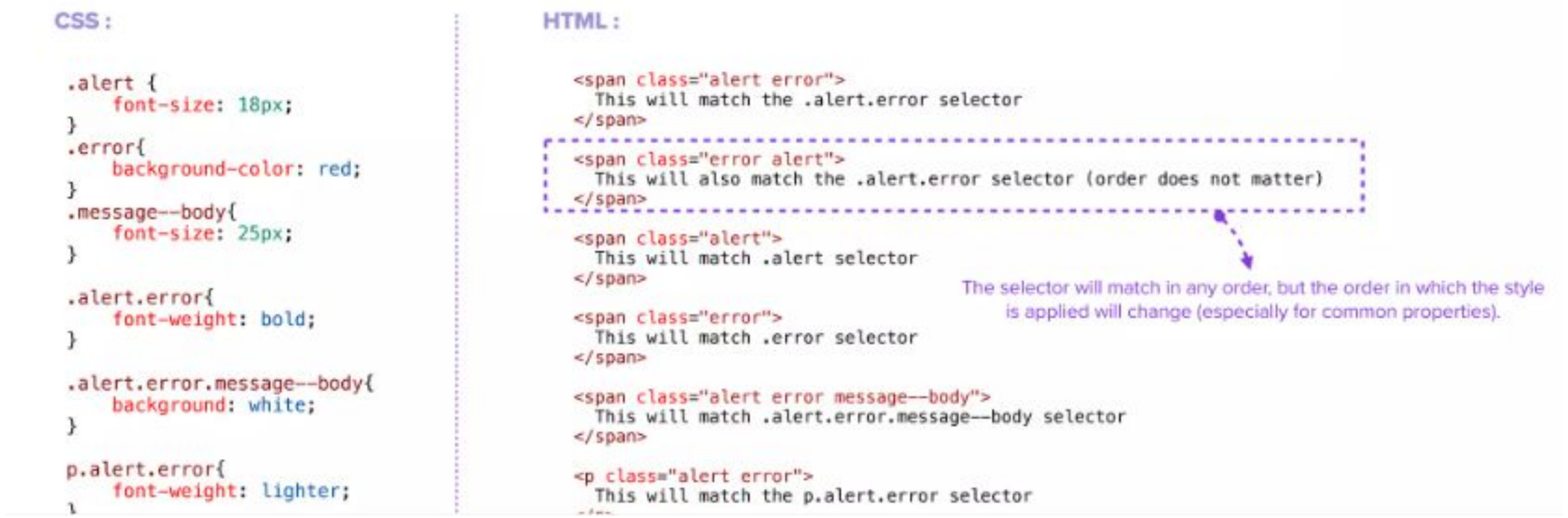
The selector will match this element **(p having a class with the value 'legal-notice—description')**.

```
  <span class="legal-notice--description">
    Contrary to popular belief, Lorem Ipsum is not simply random text
  </span>
```

The selector will not match this element **(span)**.

MULTIPLE CLASSES SELECTORS

.class-name-1.class-name-2..... { property: value; property: value; }



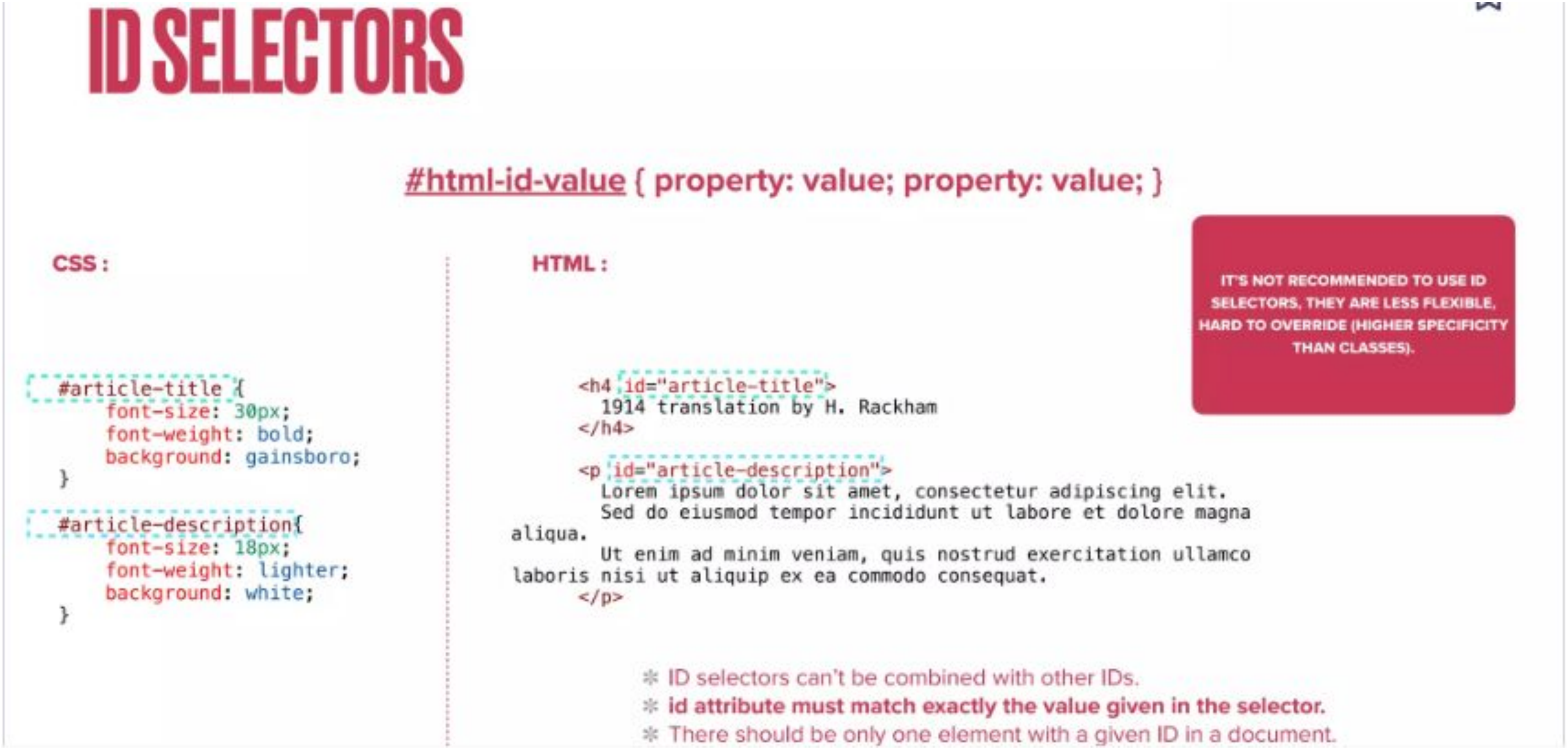
CSS :

```
.alert {
    font-size: 18px;
}
.error{
    background-color: red;
}
.message--body{
    font-size: 25px;
}

.alert.error{
    font-weight: bold;
}

.alert.error.message--body{
    background: white;
}

p.alert.error{
    font-weight: lighter;
}
```

HTML :

```
<span class="alert error">
  This will match the .alert.error selector
</span>

<span class="error alert">
  This will also match the .alert.error selector (order does not matter)
</span>

<span class="alert">
  This will match .alert selector
</span>

<span class="error">
  This will match .error selector
</span>

<span class="alert error message--body">
  This will match .alert.error.message--body selector
</span>

<p class="alert error">
  This will match the p.alert.error selector
```

The selector will match in any order, but the order in which the style is applied will change (especially for common properties).



ID SELECTORS

#html-id-value { property: value; property: value; }

IT'S NOT RECOMMENDED TO USE ID SELECTORS, THEY ARE LESS FLEXIBLE, HARD TO OVERRIDE (HIGHER SPECIFICITY THAN CLASSES).

CSS :

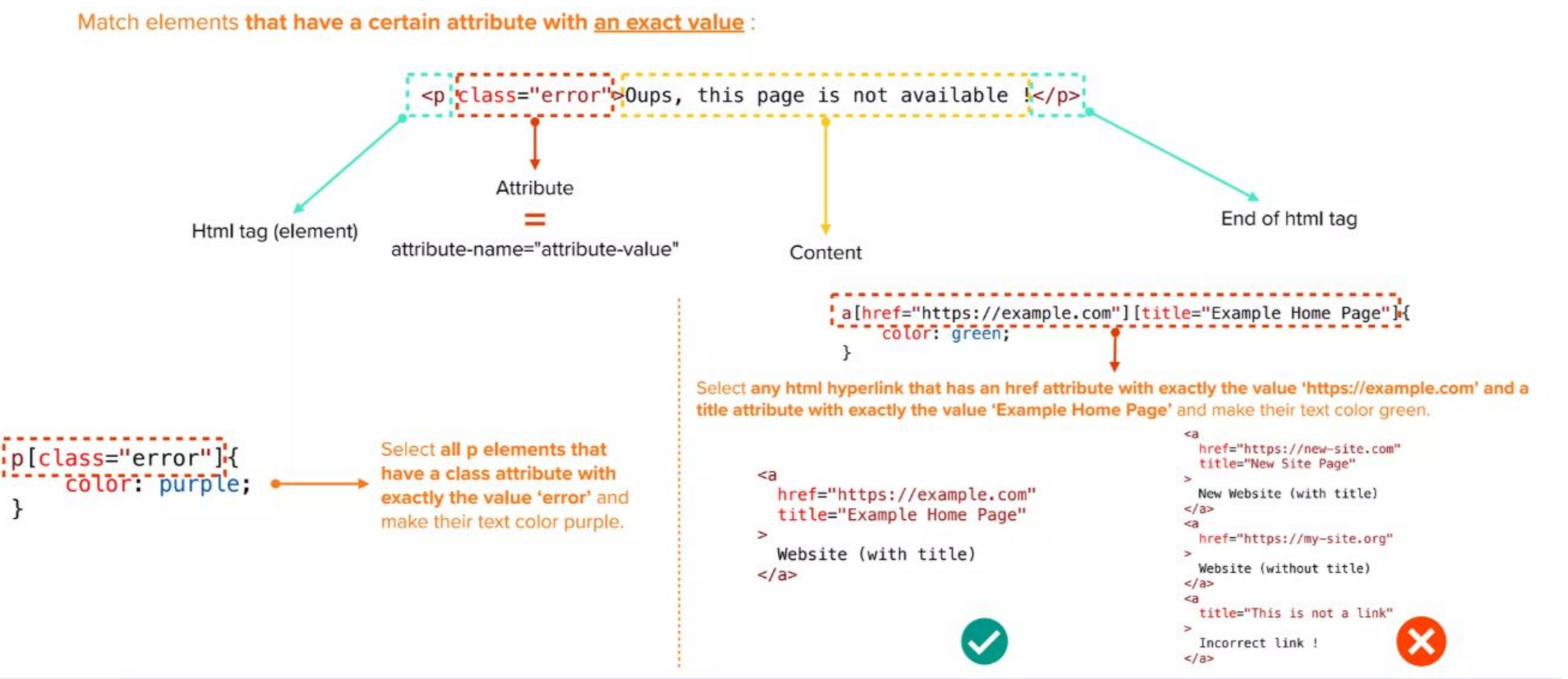
```
  #article-title {
        font-size: 30px;
        font-weight: bold;
        background: gainsboro;
  }

  #article-description{
        font-size: 18px;
        font-weight: lighter;
        background: white;
  }
```

HTML :

```
      <h4 id="article-title">
        1914 translation by H. Rackham
      </h4>

      <p id="article-description">
        Lorem ipsum dolor sit amet, consectetur adipiscing elit.
        Sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua.
        Ut enim ad minim veniam, quis nostrud exercitation ullamco
laboris nisi ut aliquip ex ea commodo consequat.
      </p>
```

- ✻ ID selectors can't be combined with other IDs.
- ✻ **id attribute must match exactly the value given in the selector.**
- ✻ There should be only one element with a given ID in a document.

Attribute selector



Match elements **that have a certain attribute with an exact value** :

```
<p class="error">Oups, this page is not available !</p>
```

```
p[class="error"]{
    color: purple;
}
```

Select **all p elements that have a class attribute with exactly the value 'error'** and make their text color purple.

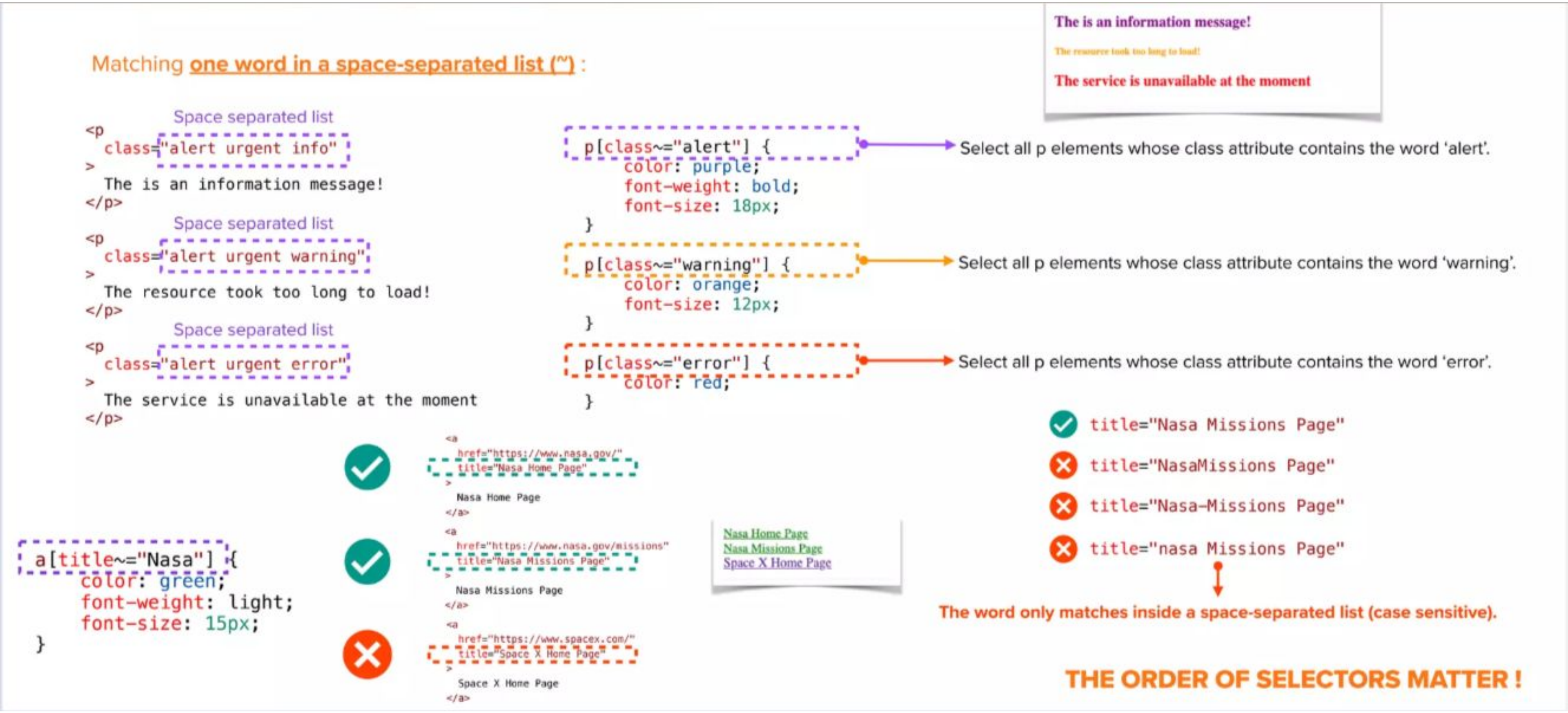
```
a[href="https://example.com"][title="Example Home Page"]{
    color: green;
}
```

Select **any html hyperlink that has an href attribute with exactly the value 'https://example.com' and a title attribute with exactly the value 'Example Home Page'** and make their text color green.

```
<a
  href="https://example.com"
  title="Example Home Page"
>
  Website (with title)
</a>
```

```
<a
  href="https://new-site.com"
  title="New Site Page"
>
  New Website (with title)
</a>
<a
  href="https://my-site.org"
>
  Website (without title)
</a>
<a
  title="This is not a link"
>
  Incorrect link !
</a>
```

One word in a space-separated list(~):



Matching **one word in a space-separated list (~)** :

```
<p
  class="alert urgent info"
>
  The is an information message!
</p>
```

```
<p
  class="alert urgent warning"
>
  The resource took too long to load!
</p>
```

```
<p
  class="alert urgent error"
>
  The service is unavailable at the moment
</p>
```

```
p[class~="alert"] {
    color: purple;
    font-weight: bold;
    font-size: 18px;
}
```

Select all p elements whose class attribute contains the word 'alert'.

```
p[class~="warning"] {
    color: orange;
    font-size: 12px;
}
```

Select all p elements whose class attribute contains the word 'warning'.

```
p[class~="error"] {
    color: red;
}
```

Select all p elements whose class attribute contains the word 'error'.

The is an information message!

The resource took too long to load!

The service is unavailable at the moment

```
a[title~="Nasa"] {
    color: green;
    font-weight: light;
    font-size: 15px;
}
```

✓
```
<a
  href="https://www.nasa.gov/"
  title="Nasa Home Page"
>
  Nasa Home Page
</a>
```

✓
```
<a
  href="https://www.nasa.gov/missions"
  title="Nasa Missions Page"
>
  Nasa Missions Page
</a>
```

✗
```
<a
  href="https://www.spacex.com/"
  title="Space X Home Page"
>
  Space X Home Page
</a>
```

Nasa Home Page
Nasa Missions Page
Space X Home Page

✓ `title="Nasa Missions Page"`

✗ `title="NasaMissions Page"`

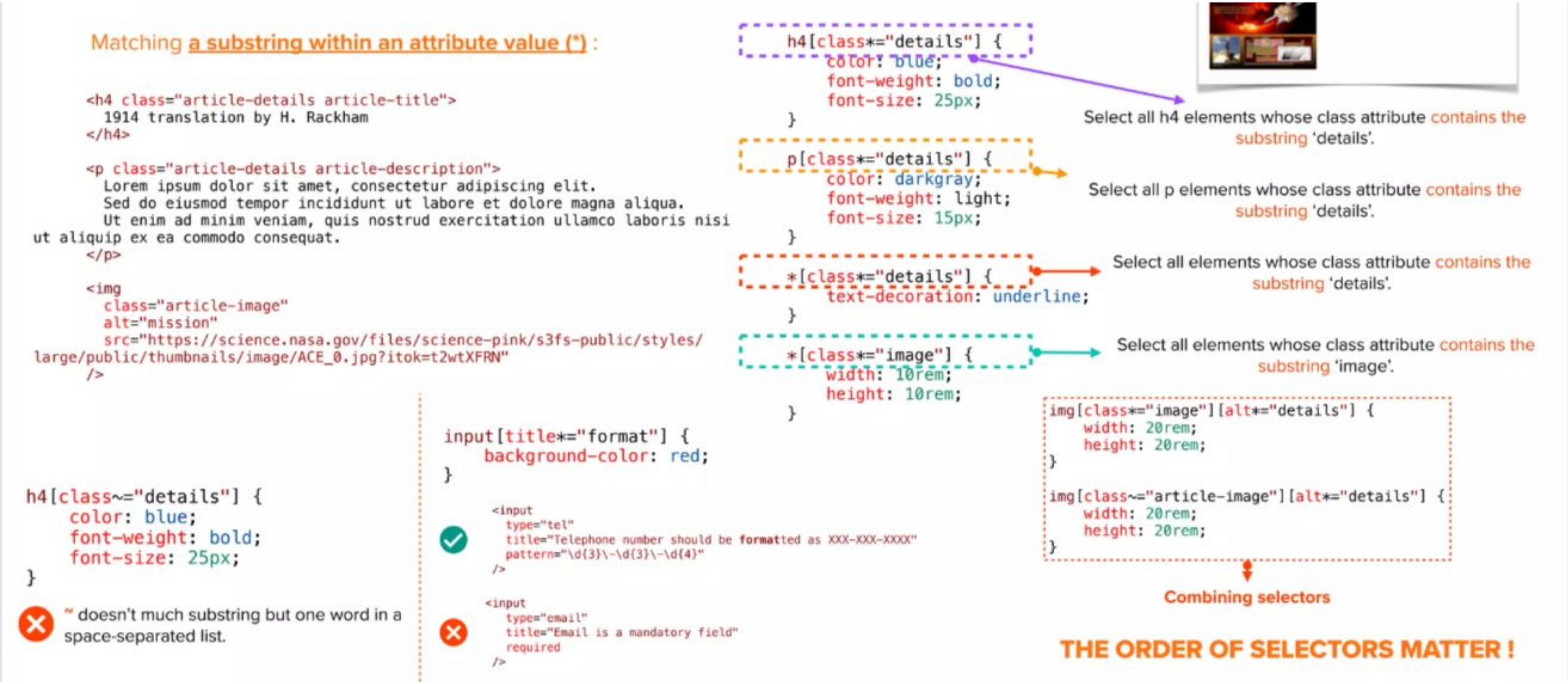
✗ `title="Nasa-Missions Page"`

✗ `title="nasa Missions Page"`

The word only matches inside a space-separated list (case sensitive).

THE ORDER OF SELECTORS MATTER !

Substring within an attribute value(*):



Matching **a substring within an attribute value (*)** :

```
<h4 class="article-details article-title">
  1914 translation by H. Rackham
</h4>

<p class="article-details article-description">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
  Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi
ut aliquip ex ea commodo consequat.
</p>

<img
  class="article-image"
  alt="mission"
  src="https://science.nasa.gov/files/science-pink/s3fs-public/styles/
large/public/thumbnails/image/ACE_0.jpg?itok=t2wtXFRN"
/>
```

```
h4[class*="details"] {
    color: blue;
    font-weight: bold;
    font-size: 25px;
}
```

Select all h4 elements whose class attribute contains the substring 'details'.

```
p[class*="details"] {
    color: darkgray;
    font-weight: light;
    font-size: 15px;
}
```

Select all p elements whose class attribute contains the substring 'details'.

```
*[class*="details"] {
    text-decoration: underline;
}
```

Select all elements whose class attribute contains the substring 'details'.

```
*[class*="image"] {
    width: 10rem;
    height: 10rem;
}
```

Select all elements whose class attribute contains the substring 'image'.

```
img[class*="image"][alt*="details"] {
    width: 20rem;
    height: 20rem;
}

img[class~="article-image"][alt*="details"] {
    width: 20rem;
    height: 20rem;
}
```

Combining selectors

THE ORDER OF SELECTORS MATTER !

```
h4[class~="details"] {
    color: blue;
    font-weight: bold;
    font-size: 25px;
}
```

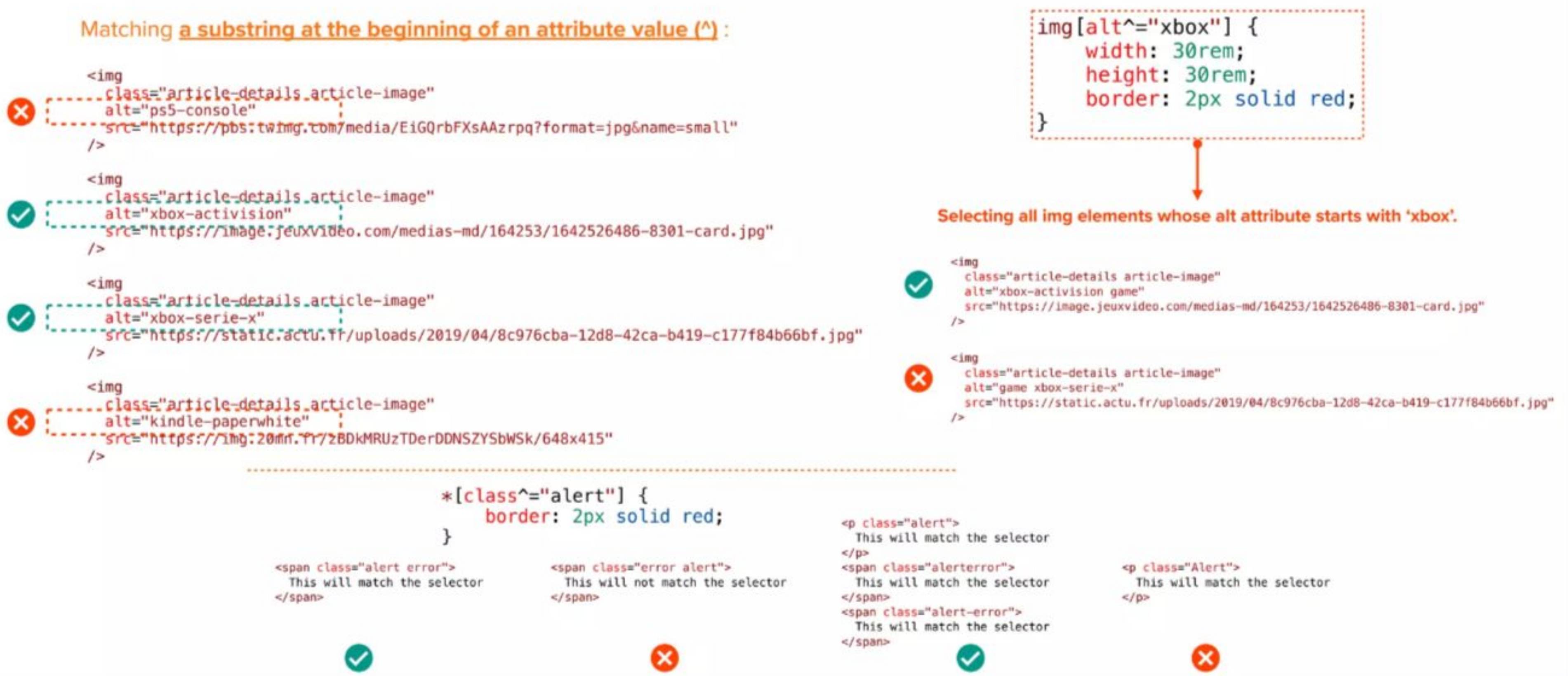
✗ ~ doesn't much substring but one word in a space-separated list.

```
input[title*="format"] {
    background-color: red;
}
```

✓
```
<input
  type="tel"
  title="Telephone number should be formatted as XXX-XXX-XXXX"
  pattern="\d{3}\-\d{3}\-\d{4}"
/>
```

✗
```
<input
  type="email"
  title="Email is a mandatory field"
  required
/>
```

Substring at the beginning of an attribute value (^):



Matching a substring at the beginning of an attribute value (^) :

✗
```
<img
  class="article-details article-image"
  alt="ps5-console"
  src="https://pbs.twimg.com/media/EiGQrbFXsAAzrpq?format=jpg&name=small"
/>
```

✓
```
<img
  class="article-details article-image"
  alt="xbox-activision"
  src="https://image.jeuxvideo.com/medias-md/164253/1642526486-8301-card.jpg"
/>
```

✓
```
<img
  class="article-details article-image"
  alt="xbox-serie-x"
  src="https://static.actu.fr/uploads/2019/04/8c976cba-12d8-42ca-b419-c177f84b66bf.jpg"
/>
```

✗
```
<img
  class="article-details article-image"
  alt="kindle-paperwhite"
  src="https://img.20mn.fr/zBDkMRUzTDerDDNSZYSbWSk/648x415"
/>
```

```
img[alt^="xbox"] {
    width: 30rem;
    height: 30rem;
    border: 2px solid red;
}
```

Selecting all img elements whose alt attribute starts with 'xbox'.

✓
```
<img
  class="article-details article-image"
  alt="xbox-activision game"
  src="https://image.jeuxvideo.com/medias-md/164253/1642526486-8301-card.jpg"
/>
```

✗
```
<img
  class="article-details article-image"
  alt="game xbox-serie-x"
  src="https://static.actu.fr/uploads/2019/04/8c976cba-12d8-42ca-b419-c177f84b66bf.jpg"
/>
```

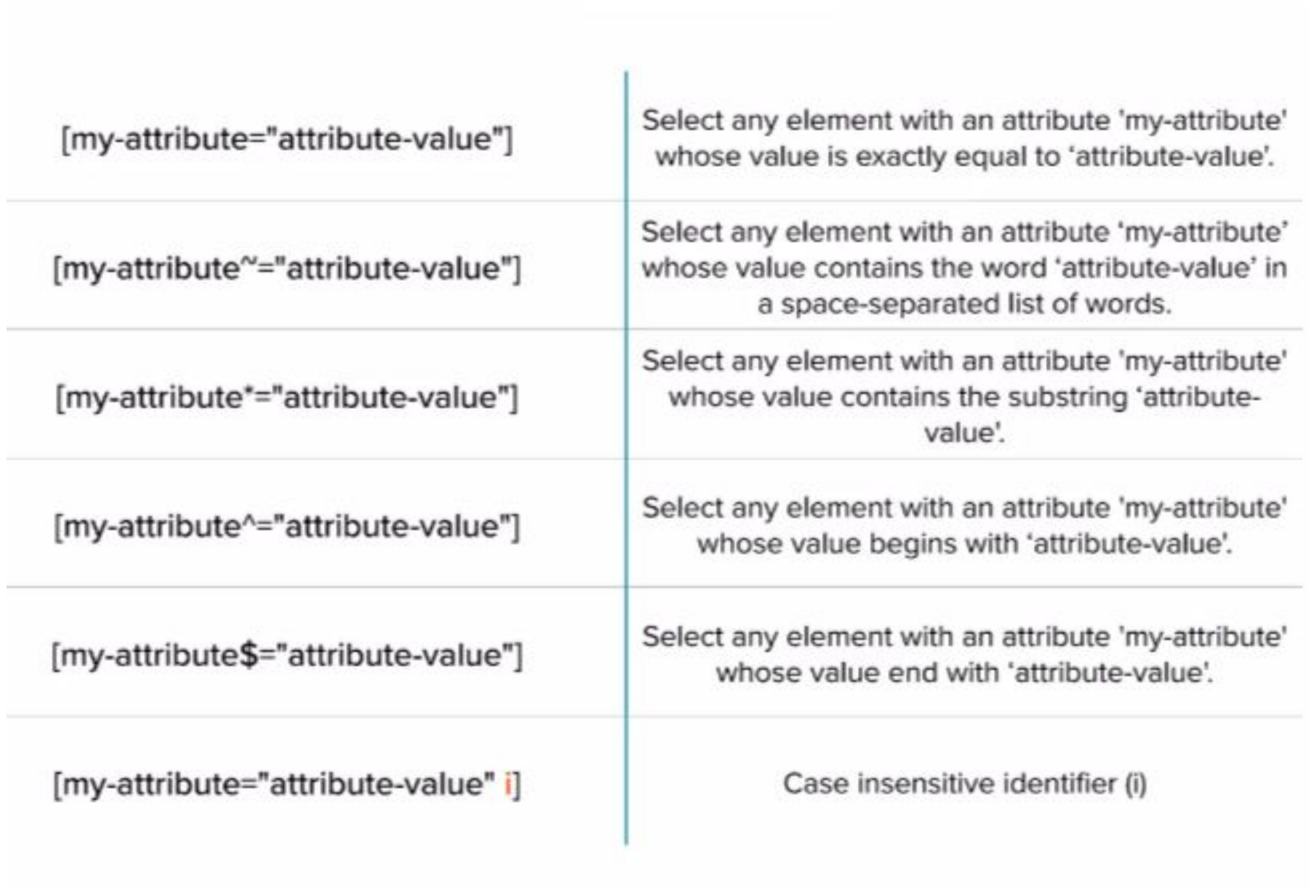
```
*[class^="alert"] {
    border: 2px solid red;
}
```

```
<span class="alert error">
  This will match the selector
</span>
```
✓

```
<span class="error alert">
  This will not match the selector
</span>
```
✗

```
<p class="alert">
  This will match the selector
</p>
<span class="alerterror">
  This will match the selector
</span>
<span class="alert-error">
  This will match the selector
</span>
```
✓

```
<p class="Alert">
  This will match the selector
</p>
```
✗



[my-attribute="attribute-value"]	Select any element with an attribute 'my-attribute' whose value is exactly equal to ‘attribute-value’.
[my-attribute~="attribute-value"]	Select any element with an attribute ‘my-attribute’ whose value contains the word ‘attribute-value’ in a space-separated list of words.
[my-attribute*="attribute-value"]	Select any element with an attribute 'my-attribute' whose value contains the substring ‘attribute-value’.
[my-attribute^="attribute-value"]	Select any element with an attribute 'my-attribute' whose value begins with ‘attribute-value’.
[my-attribute$="attribute-value"]	Select any element with an attribute 'my-attribute' whose value end with ‘attribute-value’.
[my-attribute="attribute-value" i]	Case insensitive identifier (i)

DESCENDANT COMBINATOR

selector1 selector2 { property: value; }

```
div span {
  color: red;
}
```

Set the text color to red for any span element descending from div

```
<div>
  <h4>
   1914 translation by H. Rackham
  </h4>
  <span>
✓  1. This will match
  </span>
  <span>
    <span>
✓    2. This will match
    </span>
  </span>
</div>
```

```
<div>
  <p>
✓  <span>3.This will match</span>
  </p>
</div>
```

```
.article-details span {
    color: red;
}
```

Selector can be class names

```
<div className="article-details">
  <span>
    <span>
✓    1. This will match
    </span>
  </span>
  <p>
    <span>
      <span>
✓      2. This will match
      </span>
    </span>
  </p>
</div>
```

CHILD COMBINATOR

(stricter than the descendant combinator)

selector1 > selector2 { property: value; }

```
div > span {
  color: red;
}
```

Set the text color to red for any span element descending from div

Elements matched by the second selector **must be the immediate children** of the elements matched by the first selector :

```
<div>
  <h4>
    1914 translation by H. Rackham
  </h4>
  <span>
    1. This will match
  </span>
  <span>
    <span>
      2. This will match
    </span>
  </span>
</div>
```

✓

```
<div>
  <p>
    <span>3.This will not match</span>
  </p>
</div>
```

✗

```
.article-details > span {
    color: red;
}
```

Selector can be class names

```
<div>
  <h4>
    1914 translation by H. Rackham
  </h4>
  <span>
    1. This will not match
  </span>
  <div className="article-details">
    <span>
      <span>
        2. This will match
      </span>
    </span>
  </div>
</div>
```

✗ ✓

```
<div>
  <p>
    <span>3.This will not match</span>
  </p>
</div>
```

✗

ADJACENT SIBLING COMBINATOR

Select **an element that immediately follows another element with the same parent** :

former_element + target_element { property: value; }

```
h1 + p {
  margin-top: 0;
}
```

Select any p element that **immediately follows an h1** element that shares a parent with the p element.

Remove the top margin from a paragraph immediately following an h1 .

```
<div>
  <h1>
    1914 translation by H. Rackham
  </h1>
  <p>
    This paragraph will match
  </p>
  <p>
    This will not match
  </p>
</div>
```

This p isn't immediately after h1.

GENERAL SIBLING COMBINATOR

Select an **element that follows another element when both elements share the same parent** :

former_element ~ target_element { property: value; }

```
h2 ~ ol {
    font-style: italic;
}
```

The two elements do not need to be adjacent sibling.

Italicize any ol element that follows an h2 element and also share a parent with h2.

```
<div>
  <h2>
    1914 translation by H. Rackham
  </h2>
  <p>
    Contrary to popular belief, Lorem Ipsum is not simply random text.
  </p>
  <ol>
    <li>1. This will match</li>
    <li>2. This will match</li>
    <li>3. This will match</li>
  </ol>
  <p>
    Contrary to popular belief, Lorem Ipsum is not simply random text.
  </p>
  <ol>
    <li>4. This will match</li>
    <li>5. This will match</li>
    <li>6. This will match</li>
  </ol>
  <div>
    <ol>
      <li>1. This will not match</li>
      <li>2. This will not match</li>
      <li>3. This will not match</li>
    </ol>
  </div>
</div>
```

Follows h2 + shares the same parent

Follows h2 + shares the same parent

Follows h2 but doesn't shares the same parent

NEGATION PSEUDO-CLASS

element:not(selector) { property: value; }

```
.moreinfo:not(li) {
    font-style: italic;
}
```

Select all elements with a class whose value contains the word 'moreinfo' as long as they are not li elements

```
*.article-link:not(li):not(p){
    color: red;
}
```

Select all elements with class 'article-link' that are neither list items nor paragraphs.

~~p:not(:not(p))~~

Cannot be nested

HYPERLINK PSEUDO-CLASSES

:link	Matches links that have not yet been visited.
:visited	Matches links that have been visited.

```css
a:link{
    color: blue;
}

a:visited{
    color: red;
}
```

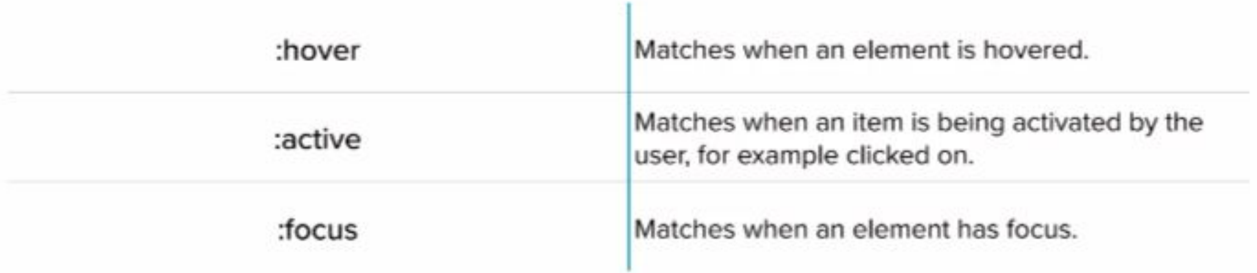
```css
a.article-details-link:link, a[href^="http"]:link {
    color: slateblue;
}

a.article-details-link:visited, a[href^="http"]:visited {
    color: maroon;
}
```

```css
a#article-link-id:link {
    color: yellow;
}

a#article-link-id:visited {
    color: gray;
}
```

Class name selectors

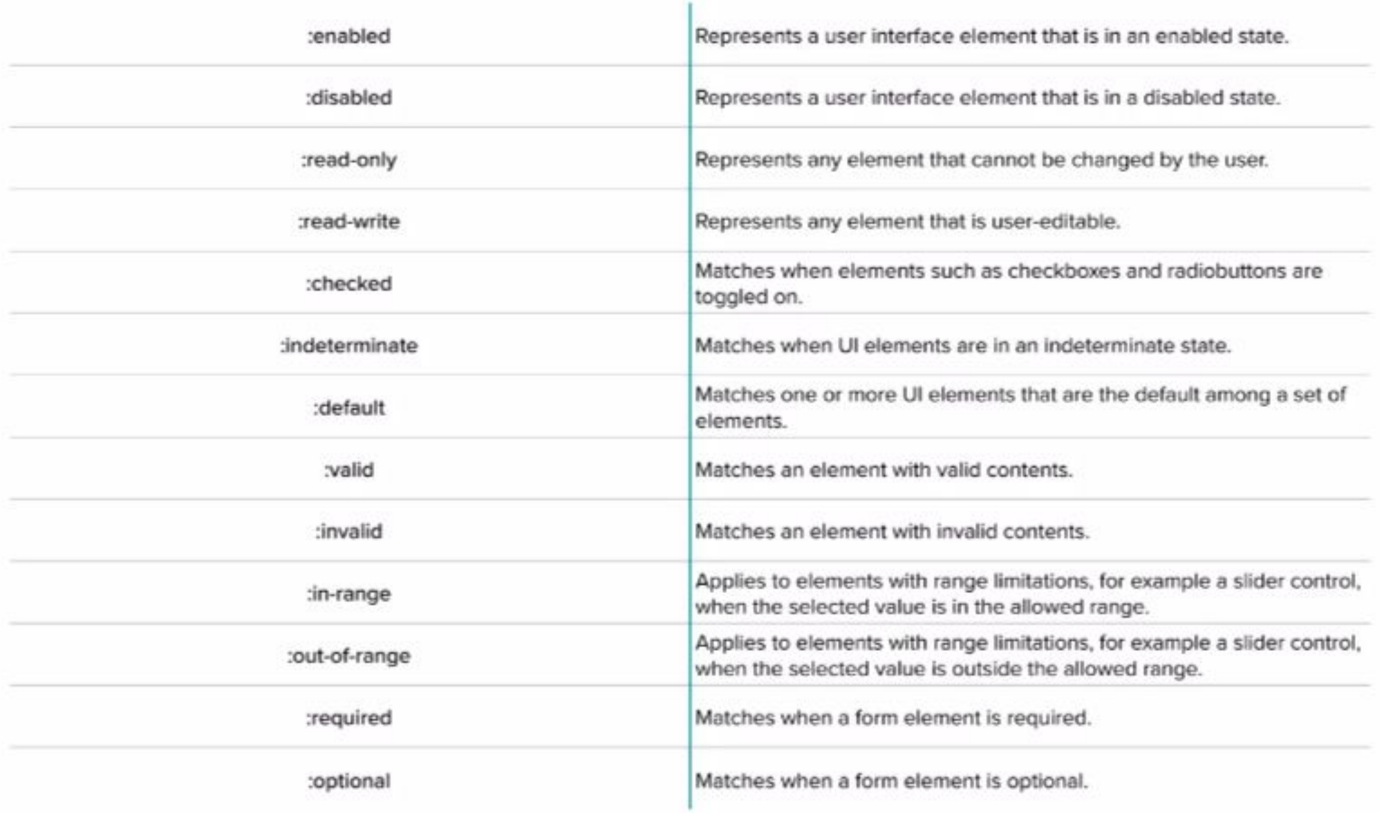
Element ID selectors

USER ACTION PSEUDO-CLASSES



:hover	Matches when an element is hovered.
:active	Matches when an item is being activated by the user, for example clicked on.
:focus	Matches when an element has focus.

UI-STATE PSEUDO-CLASSES



:enabled	Represents a user interface element that is in an enabled state.
:disabled	Represents a user interface element that is in a disabled state.
:read-only	Represents any element that cannot be changed by the user.
:read-write	Represents any element that is user-editable.
:checked	Matches when elements such as checkboxes and radiobuttons are toggled on.
:indeterminate	Matches when UI elements are in an indeterminate state.
:default	Matches one or more UI elements that are the default among a set of elements.
:valid	Matches an element with valid contents.
:invalid	Matches an element with invalid contents.
:in-range	Applies to elements with range limitations, for example a slider control, when the selected value is in the allowed range.
:out-of-range	Applies to elements with range limitations, for example a slider control, when the selected value is outside the allowed range.
:required	Matches when a form element is required.
:optional	Matches when a form element is optional.

BEFORE & AFTER PSEUDO ELEMENTS SELECTORS

::before	Inserts something before the content of each selected element(s).
	
	`p::before {`
	`  content: "Read this: ";`
	`}`

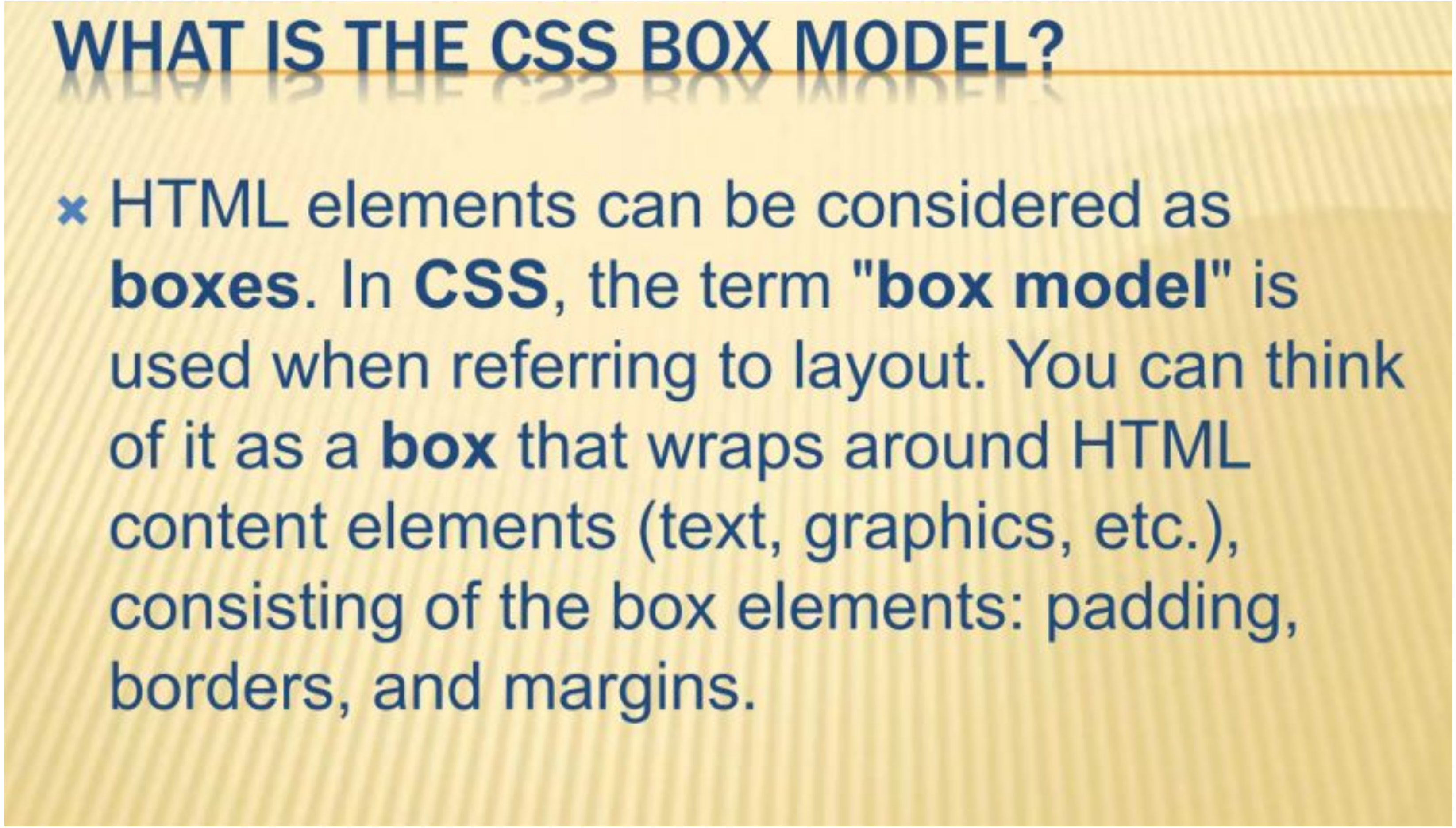
::after	Inserts something after the content of each selected element(s).
	
	`p::after {`
	`  content: " - Remember this";`
	`}`

FIRST LETTER PSEUDO ELEMENT SELECTOR

```css
p::first-letter{
    color: red;
}

p:first-of-type::first-letter{
    font-size: 200%;
}
```

FIRST LINE PSEUDO ELEMENT SELECTOR

```
p::first-line{
    font-size: 150%;
    color: purple;
}
```



WHAT IS THE CSS BOX MODEL?

- HTML elements can be considered as **boxes**. In **CSS**, the term "**box model**" is used when referring to layout. You can think of it as a **box** that wraps around HTML content elements (text, graphics, etc.), consisting of the box elements: padding, borders, and margins.

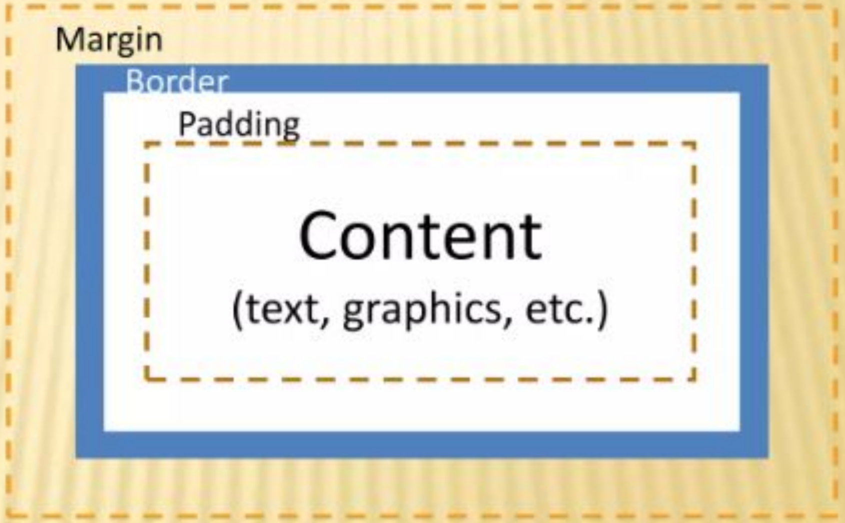
LET'S LOOK AT BOX MODEL PROPERTIES

Content = text, images

Padding = transparent space around content and within border

Border = a varying-thickness line around the padding space

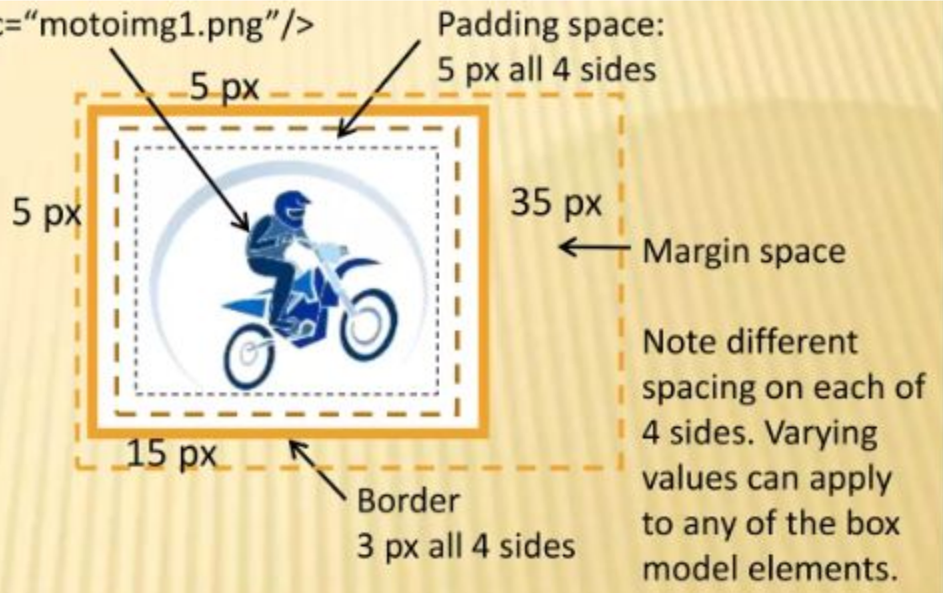
Margin = transparent space outside the border



All box model properties can be precisely defined with values for top, right, bottom, and left sides and are contained within the HTML <div> tag.

With the CSS box model, you can apply the same spacing value to all sides, or different spacing to each side.

Example:
padding: 3px; **(All 4 sides the same)**
border: 3px; **(All 4 sides the same)**
margin: 5px 35px 15px 5px; **(Different)**

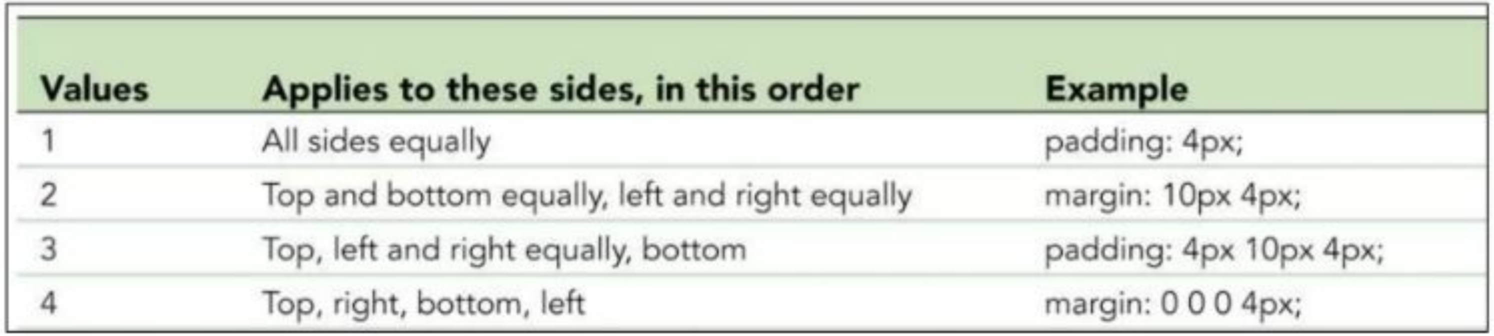


<img class="**motoImg**" src="motoimg1.png"/>
Padding space:
5 px all 4 sides
5 px
5 px
35 px
Margin space
15 px
Border
3 px all 4 sides
Note different spacing on each of 4 sides. Varying values can apply to any of the box model elements.

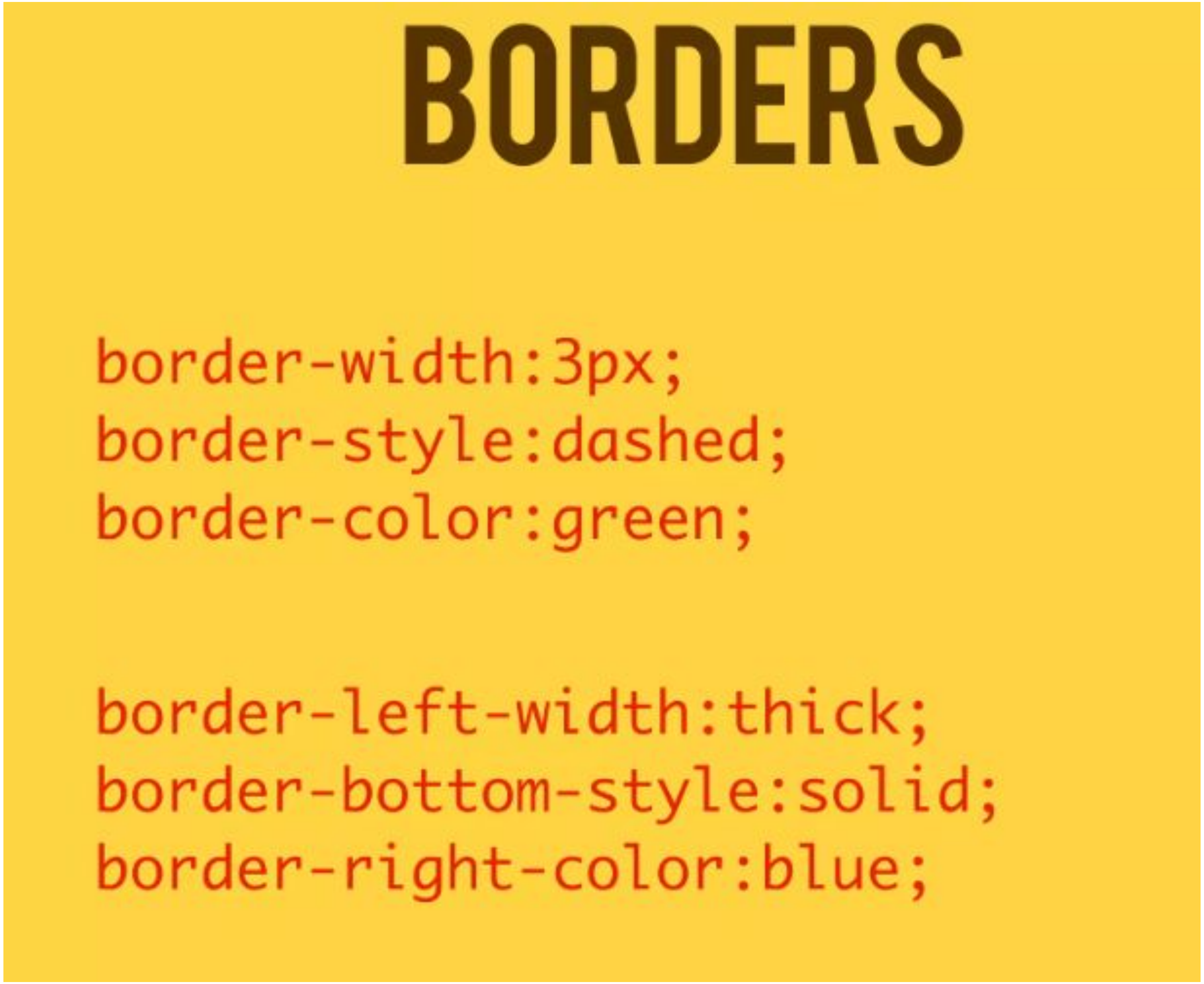
CSS Class

```
.motoImg {
        padding: 5px;
        border: 3px;
        margin: 5px 35px 15px 5px;
}
```

Using the `padding` and `margin` Shorthand Properties

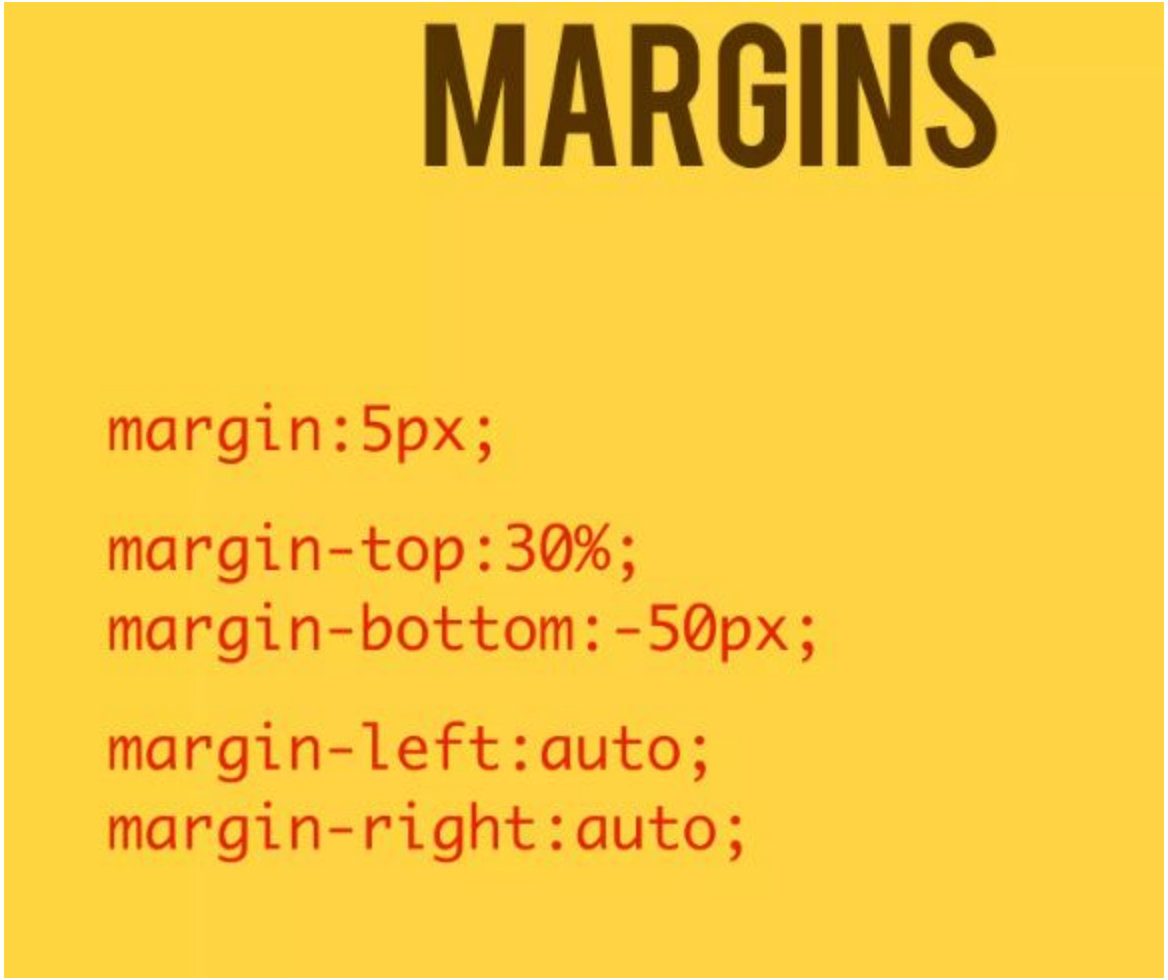


Values	Applies to these sides, in this order	Example
1	All sides equally	padding: 4px;
2	Top and bottom equally, left and right equally	margin: 10px 4px;
3	Top, left and right equally, bottom	padding: 4px 10px 4px;
4	Top, right, bottom, left	margin: 0 0 0 4px;



BORDERS

```
border-width:3px;
border-style:dashed;
border-color:green;

border-left-width:thick;
border-bottom-style:solid;
border-right-color:blue;
```

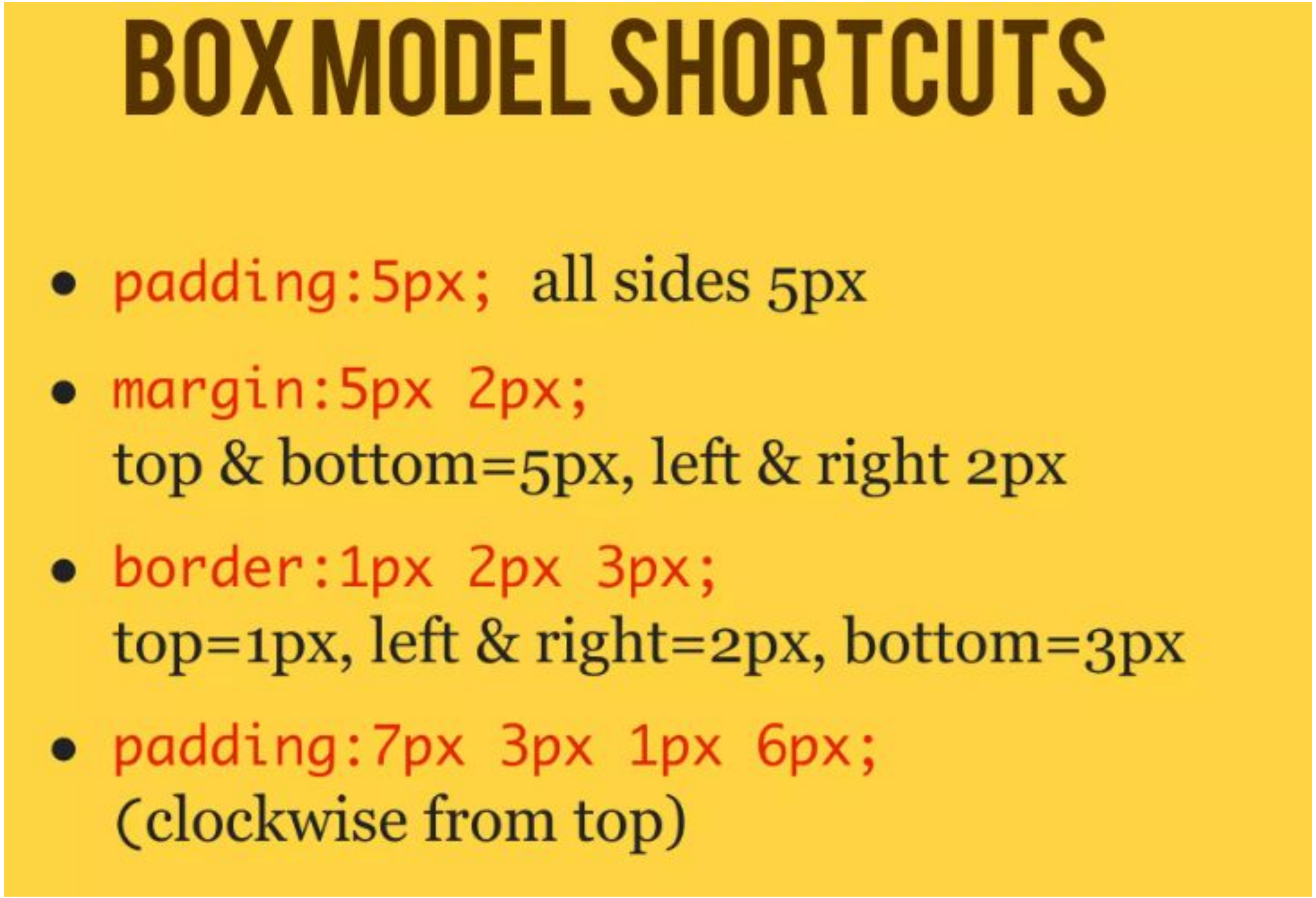


MARGINS

```
margin:5px;

margin-top:30%;
margin-bottom:-50px;

margin-left:auto;
margin-right:auto;
```

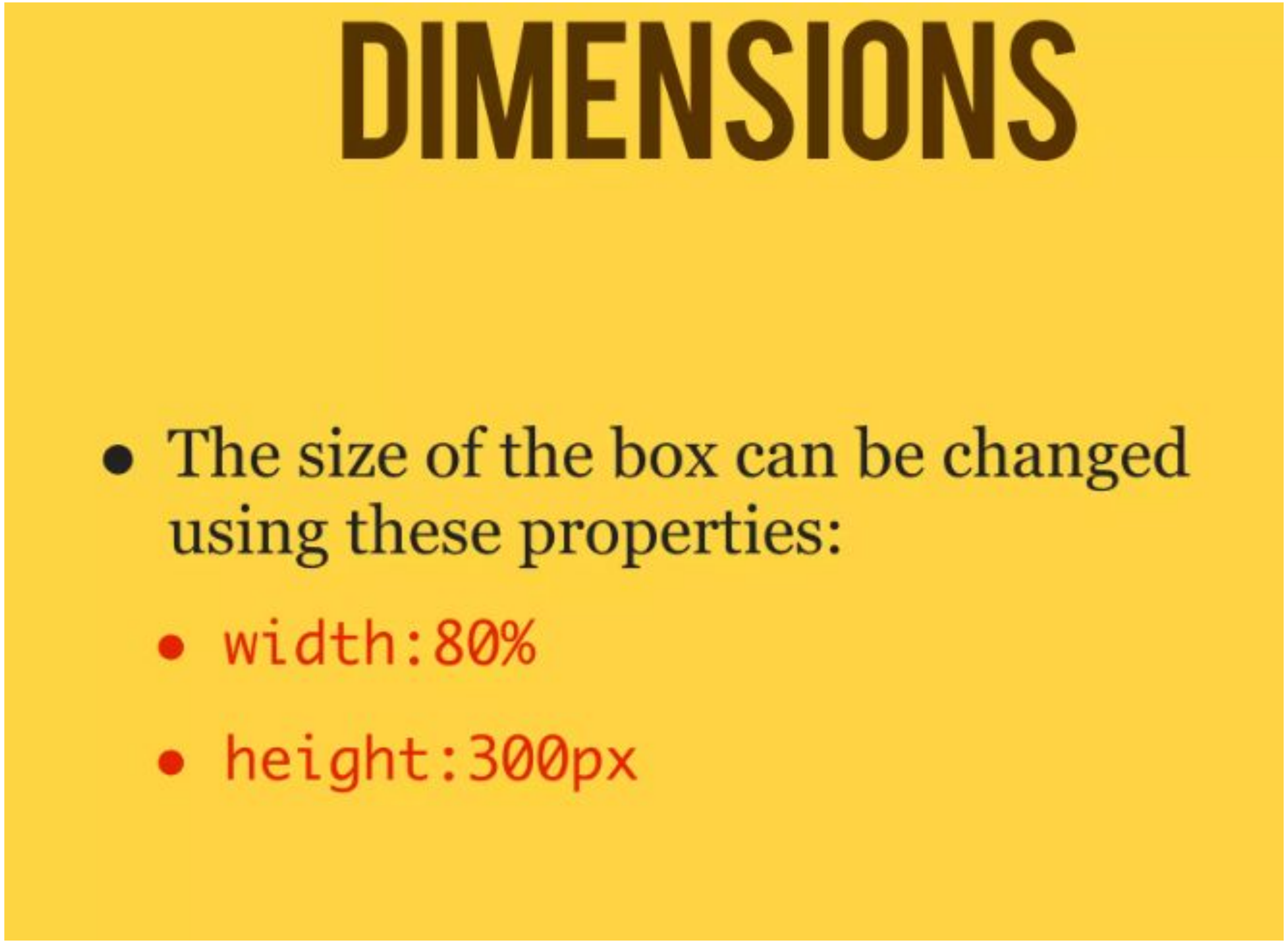
PADDING

```
padding:5px;

padding-top:10%;
padding-bottom:2em;

padding-left:30px;
padding-right:2.5em;
```



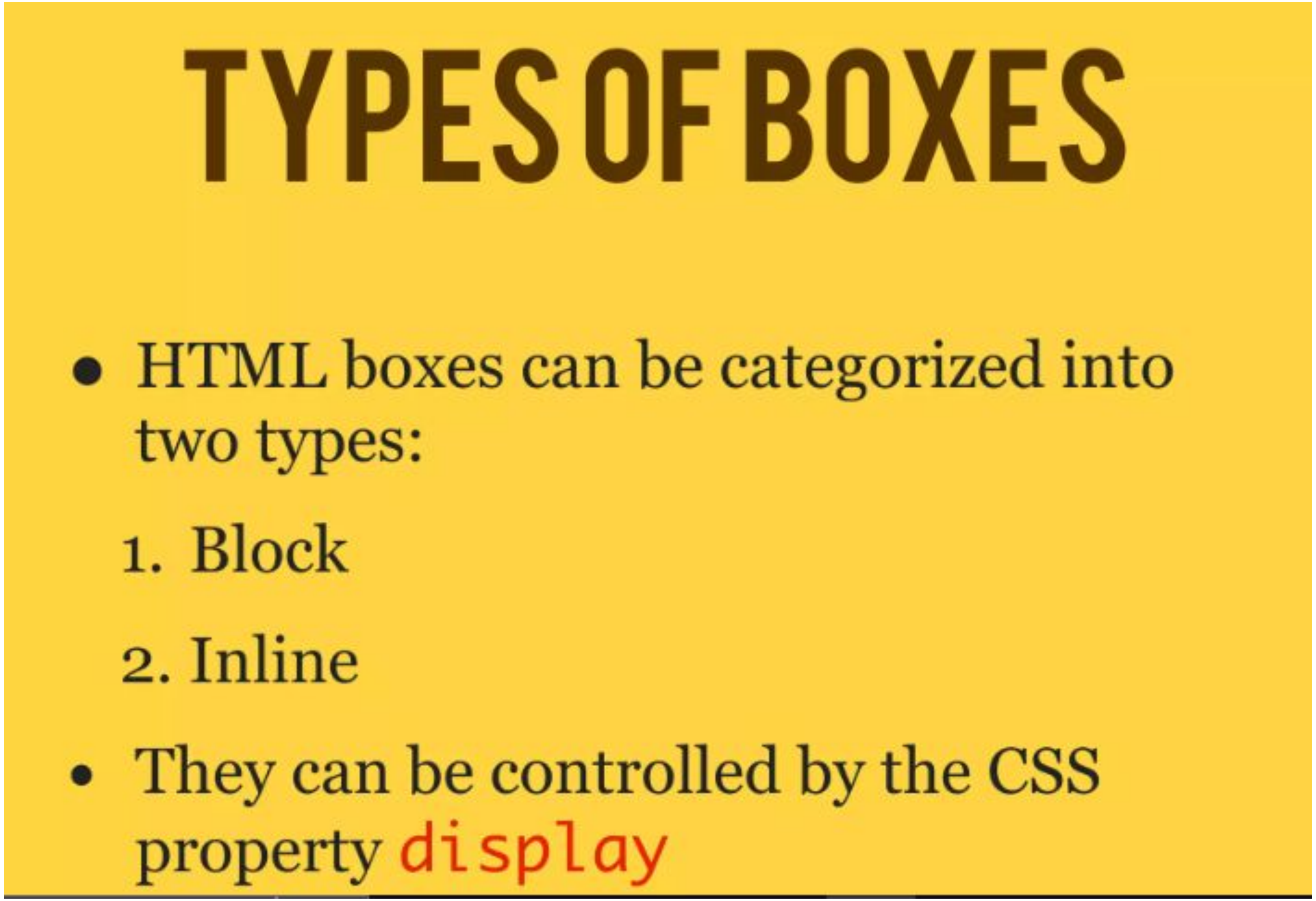
BOX MODEL SHORTCUTS

- `padding:5px;` all sides 5px
- `margin:5px 2px;`
  top & bottom=5px, left & right 2px
- `border:1px 2px 3px;`
  top=1px, left & right=2px, bottom=3px
- `padding:7px 3px 1px 6px;`
  (clockwise from top)



DIMENSIONS

- The size of the box can be changed using these properties:
  - `width:80%`
  - `height:300px`

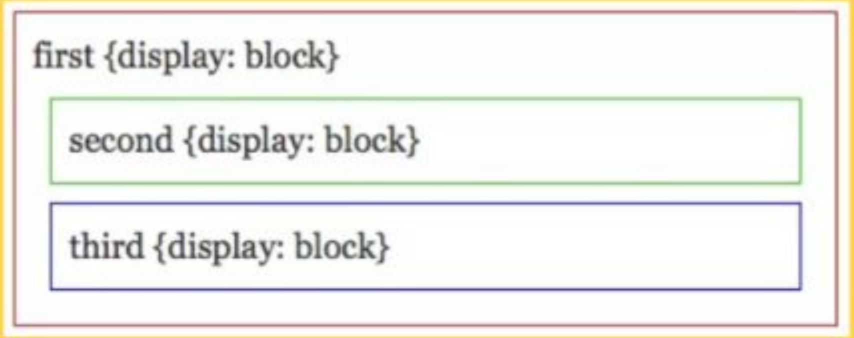


TYPES OF BOXES

- HTML boxes can be categorized into two types:
  1. Block
  2. Inline
- They can be controlled by the CSS property `display`

BLOCK BOX

- Occupies the whole width of the container element
- Has whitespace before and after it
- Dimensions are controllable



first {display: block}

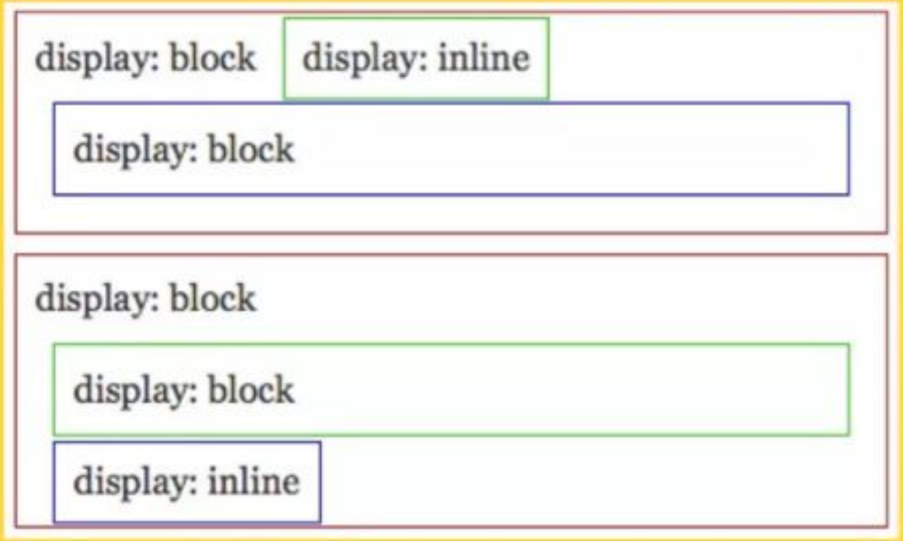
second {display: block}

third {display: block}

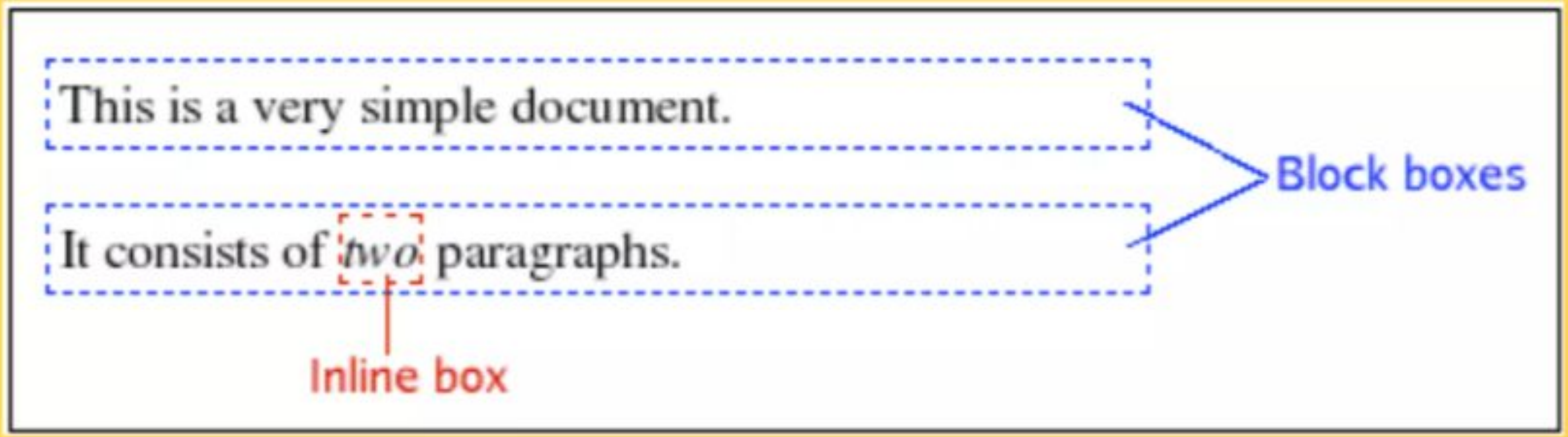
`<p> <h1> to <h6>`
`<div> <ul> <ol> <li>`

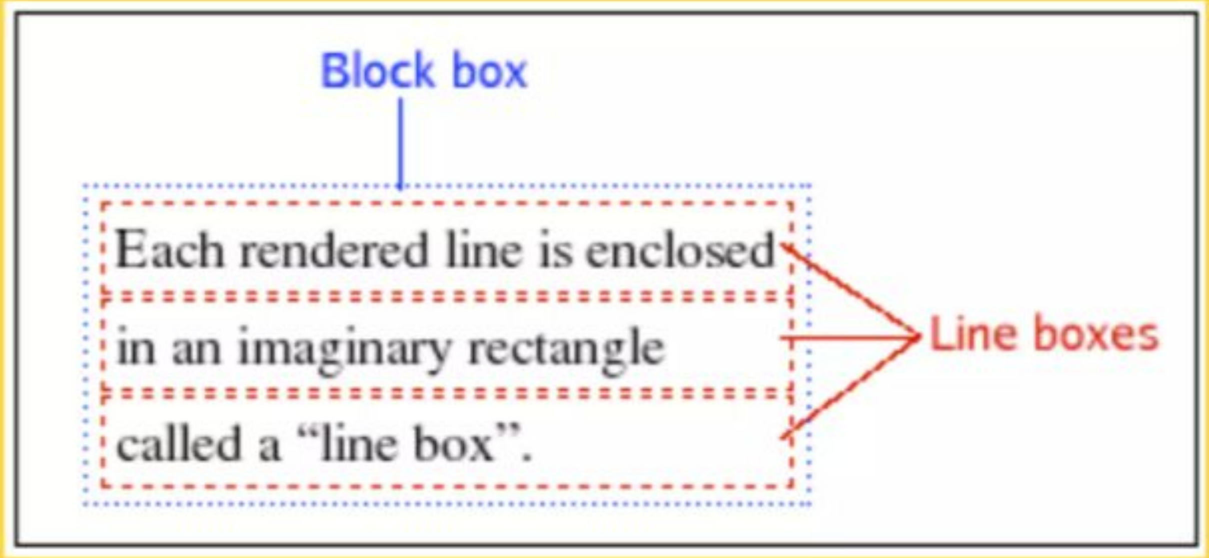
INLINE BOX

- Only as wide as its content
- Flows with text lines
- Dimensions aren't easily controllable

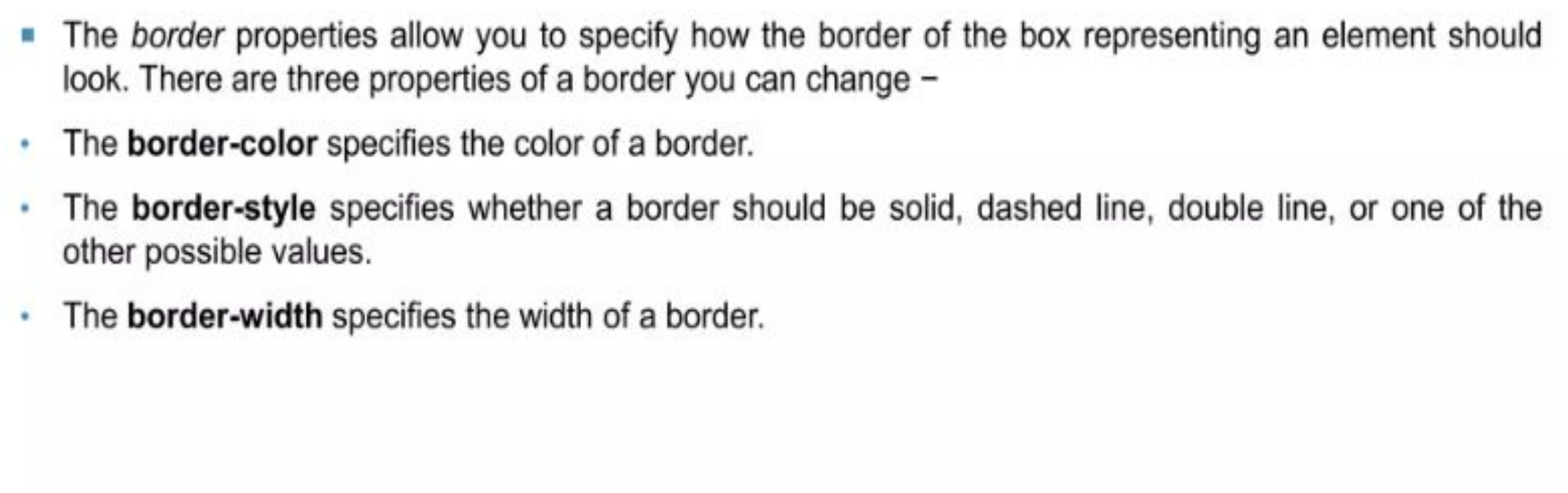


<a> <span>
<strong> <em> <img>


This is a very simple document.
It consists of *two* paragraphs.
Block boxes
Inline box


Block box
Each rendered line is enclosed
in an imaginary rectangle
called a "line box".
Line boxes

Css Borders

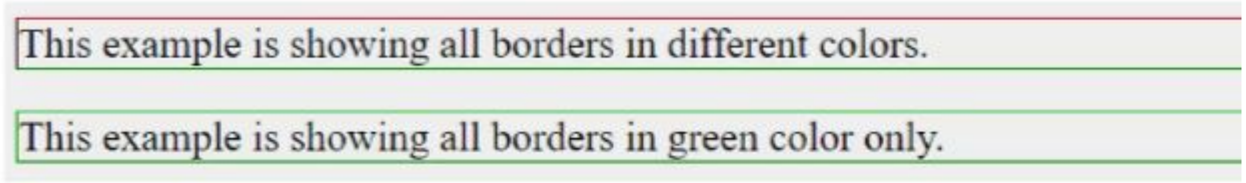


- The *border* properties allow you to specify how the border of the box representing an element should look. There are three properties of a border you can change −
- The **border-color** specifies the color of a border.
- The **border-style** specifies whether a border should be solid, dashed line, double line, or one of the other possible values.
- The **border-width** specifies the width of a border.

Border-color-property

- The border-color property allows you to change the color of the border surrounding an element. You can individually change the color of the bottom, left, top and right sides of an element's border using the properties −
- **border-bottom-color** changes the color of bottom border.
- **border-top-color** changes the color of top border.
- **border-left-color** changes the color of left border.
- **border-right-color** changes the color of right border.

```
<html>
  <head>
    <style type = "text/css">
      p.example1 {
        border:1px solid;
        border-bottom-color:#009900; /* Green */
        border-top-color:#FF0000;    /* Red */
        border-left-color:#330000;   /* Black */
        border-right-color:#0000CC;  /* Blue */
      }
      p.example2 {
        border:1px solid;
        border-color:#009900;        /* Green */
      }
    </style>
  </head>
  <body>
    <p class = "example1">
      This example is showing all borders in different colors.
    </p>
    <p class = "example2">
      This example is showing all borders in green color only.
    </p>
  </body>
</html>
```

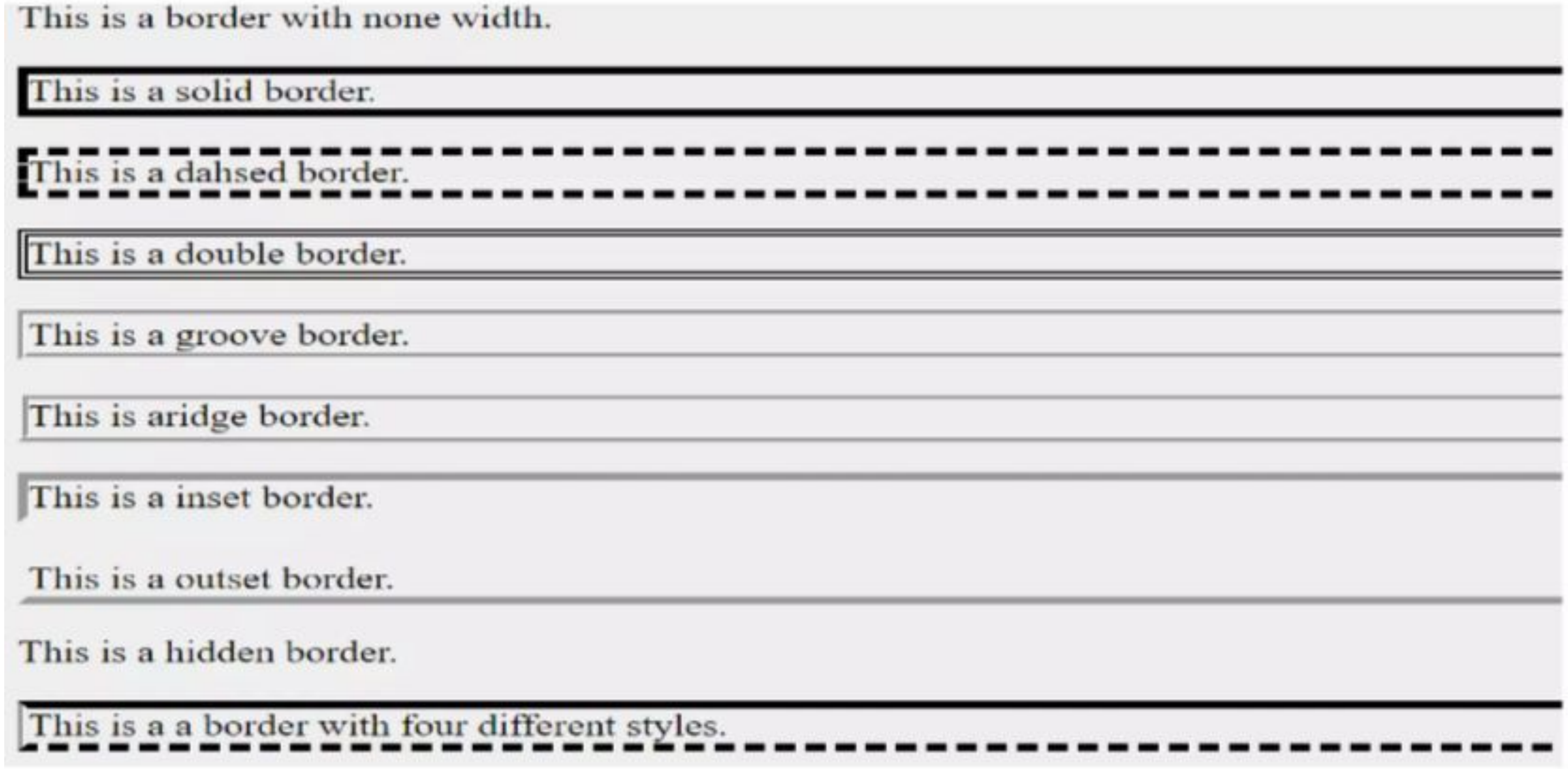


This example is showing all borders in different colors.

This example is showing all borders in green color only.

Border style property

- The border-style property allows you to select one of the following styles of border –
  - **none** – No border. (Equivalent of border-width:0;)
  - **solid** – Border is a single solid line.
  - **dotted** – Border is a series of dots.
  - **dashed** – Border is a series of short lines.
  - **double** – Border is two solid lines.
  - **groove** – Border looks as though it is carved into the page.
  - **ridge** – Border looks the opposite of groove.
  - **inset** – Border makes the box look like it is embedded in the page.
  - **outset** – Border makes the box look like it is coming out of the canvas.
  - **hidden** – Same as none, except in terms of border-conflict resolution for table elements.
- You can individually change the style of the bottom, left, top, and right borders of an element using the following properties –
  - **border-bottom-style** changes the style of bottom border.
  - **border-top-style** changes the style of top border.
  - **border-left-style** changes the style of left border.
  - **border-right-style** changes the style of right border.



This is a border with none width.

This is a solid border.

This is a dahsed border.

This is a double border.

This is a groove border.

This is aridge border.

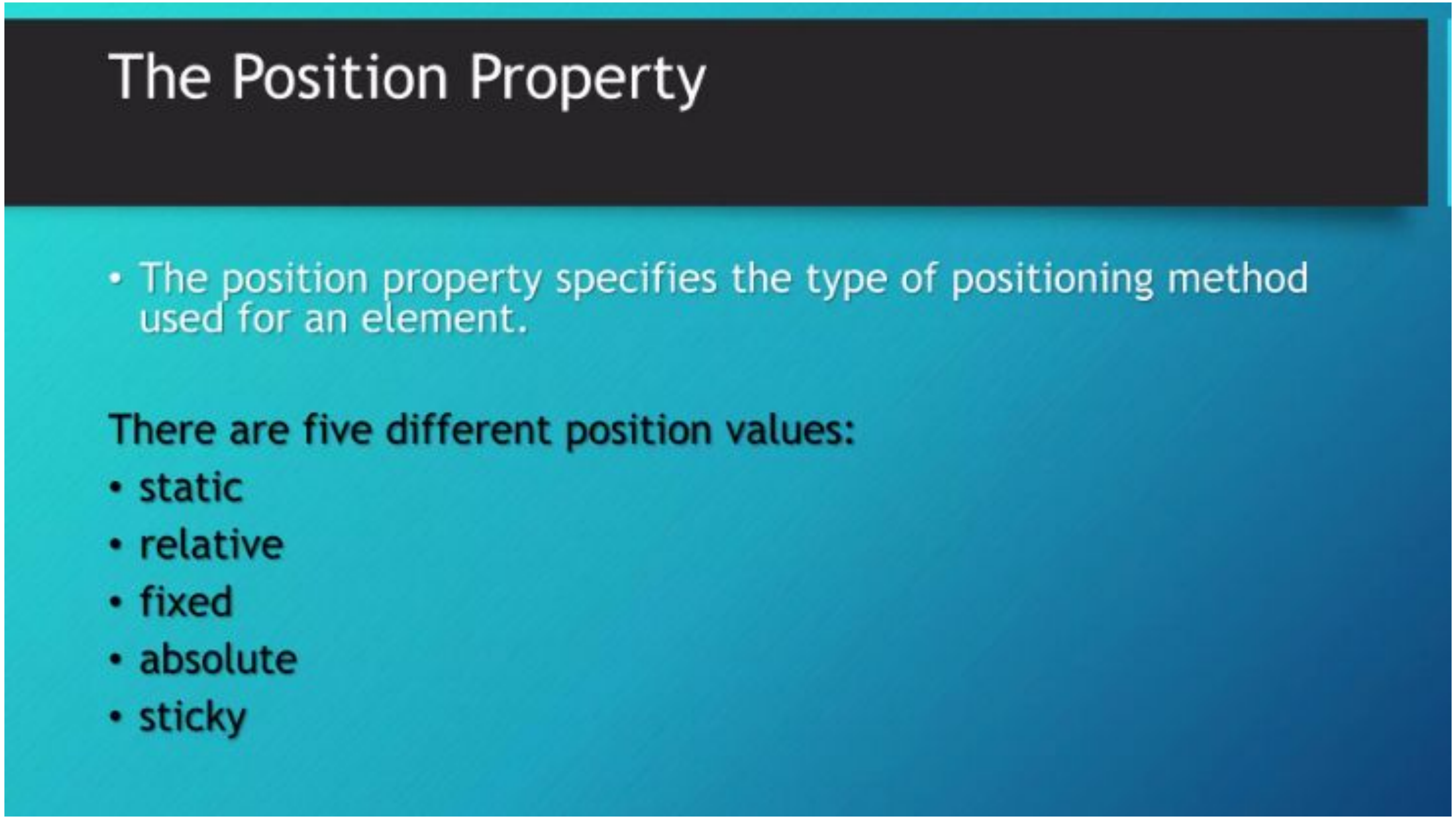
This is a inset border.

This is a outset border.

This is a hidden border.

This is a a border with four different styles.

Positions in css



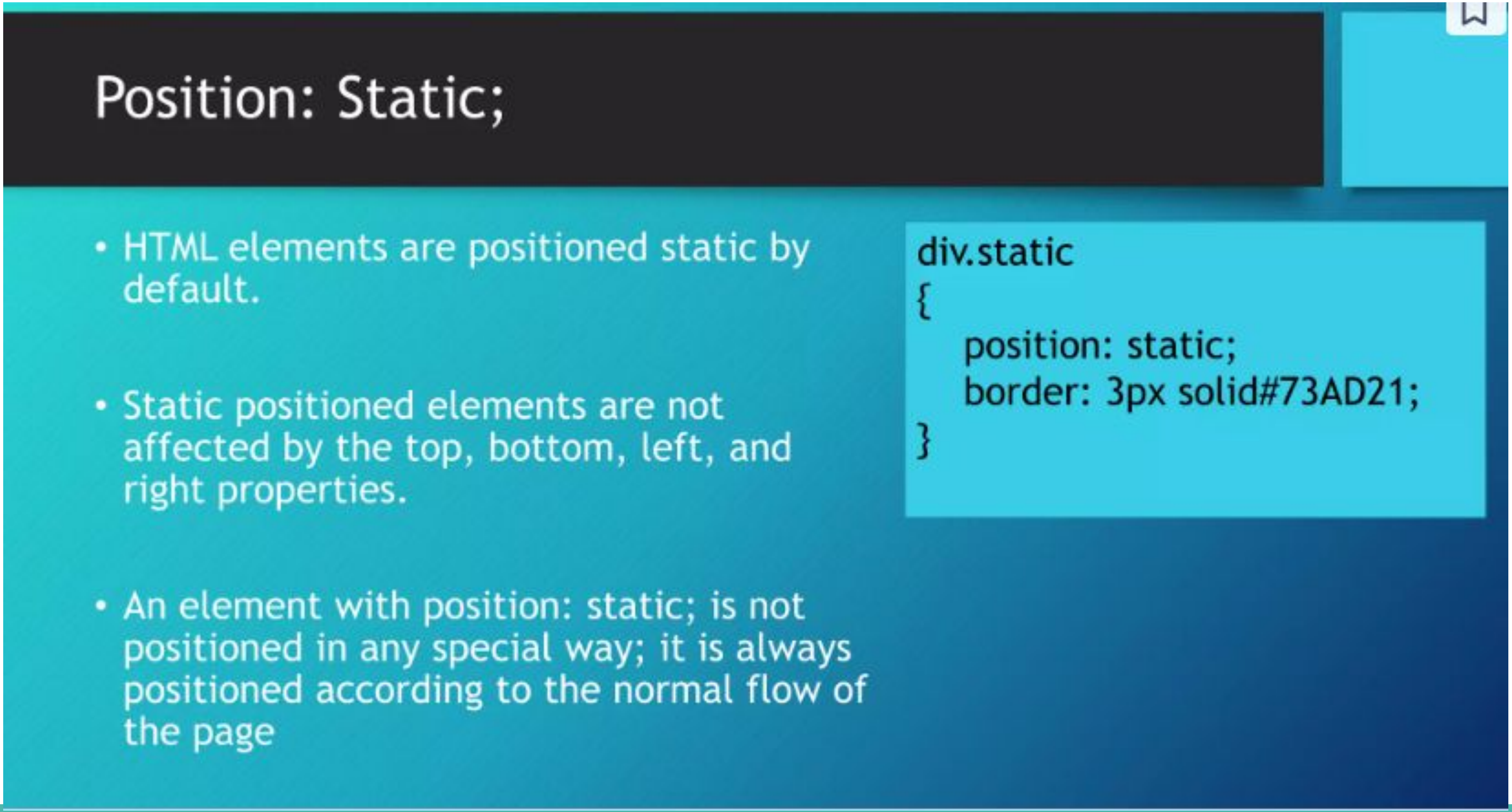
The Position Property

- The position property specifies the type of positioning method used for an element.

There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Static position:

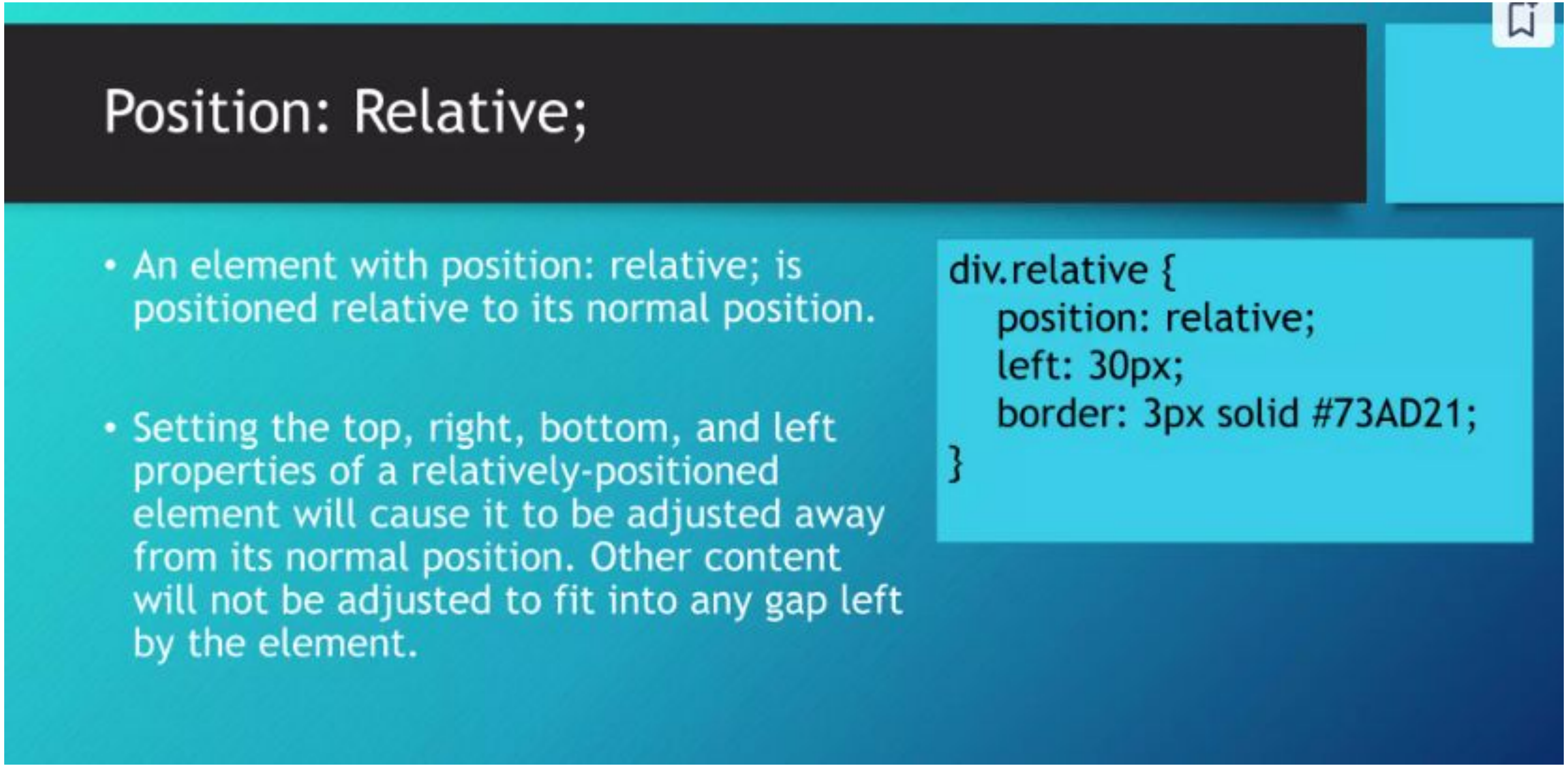


Position: Static;

- HTML elements are positioned static by default.
- Static positioned elements are not affected by the top, bottom, left, and right properties.
- An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page

```
div.static
{
    position: static;
    border: 3px solid#73AD21;
}
```

Position relative:

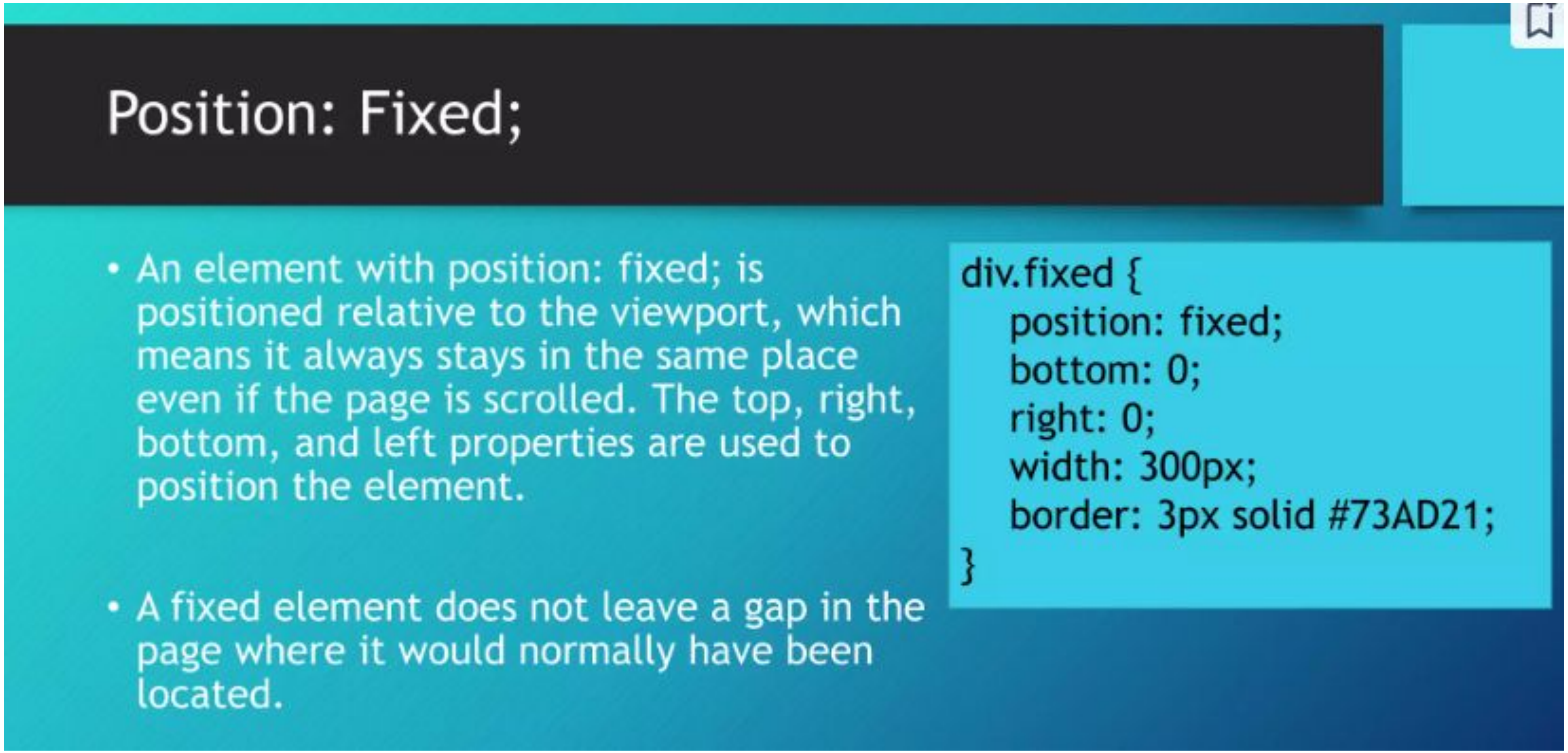


Position: Relative;

- An element with position: relative; is positioned relative to its normal position.
- Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

```
div.relative {
    position: relative;
    left: 30px;
    border: 3px solid #73AD21;
}
```

Position Fixed:

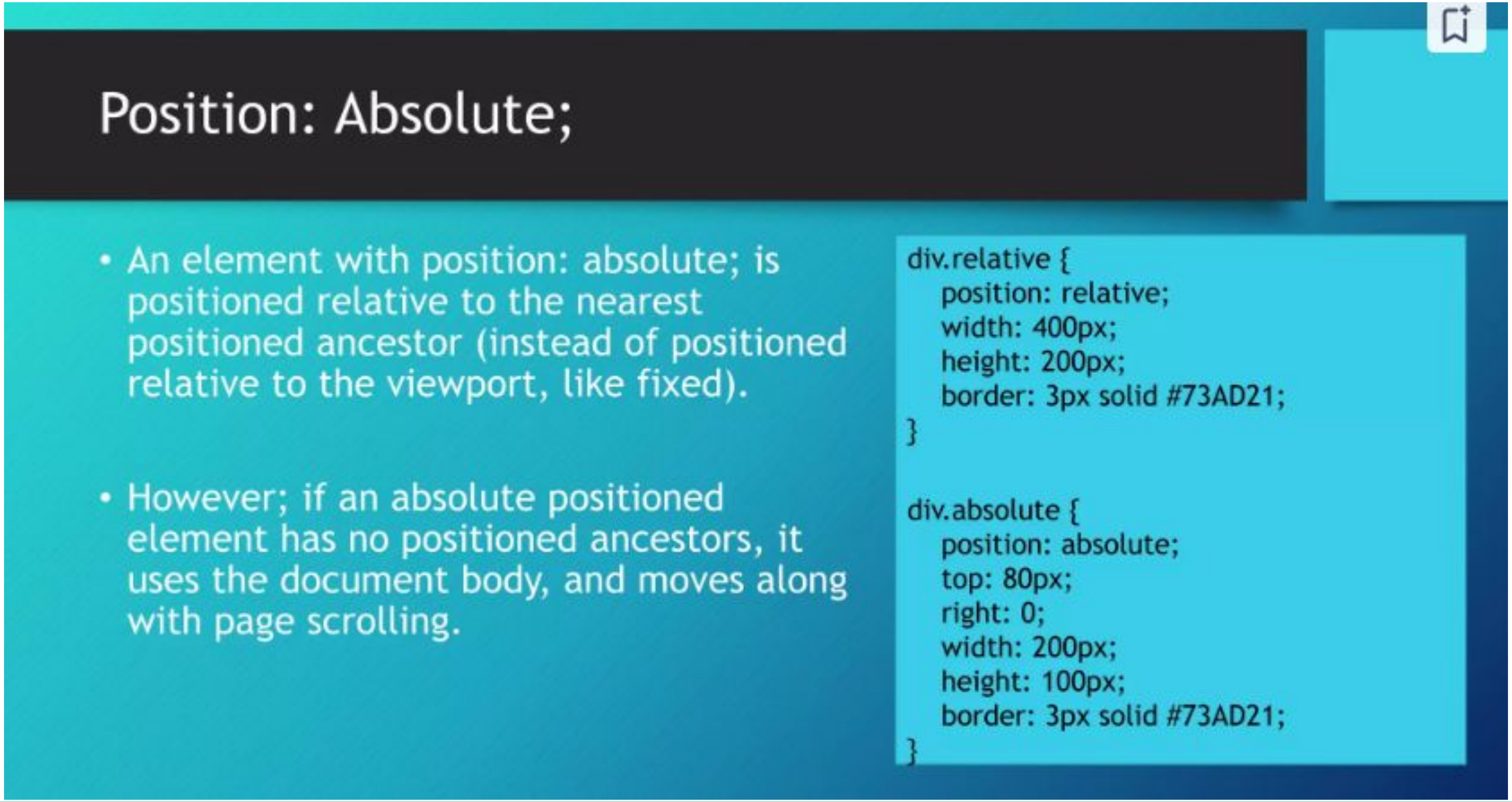


Position: Fixed;

- An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.
- A fixed element does not leave a gap in the page where it would normally have been located.

```
div.fixed {
    position: fixed;
    bottom: 0;
    right: 0;
    width: 300px;
    border: 3px solid #73AD21;
}
```

Position absolute:

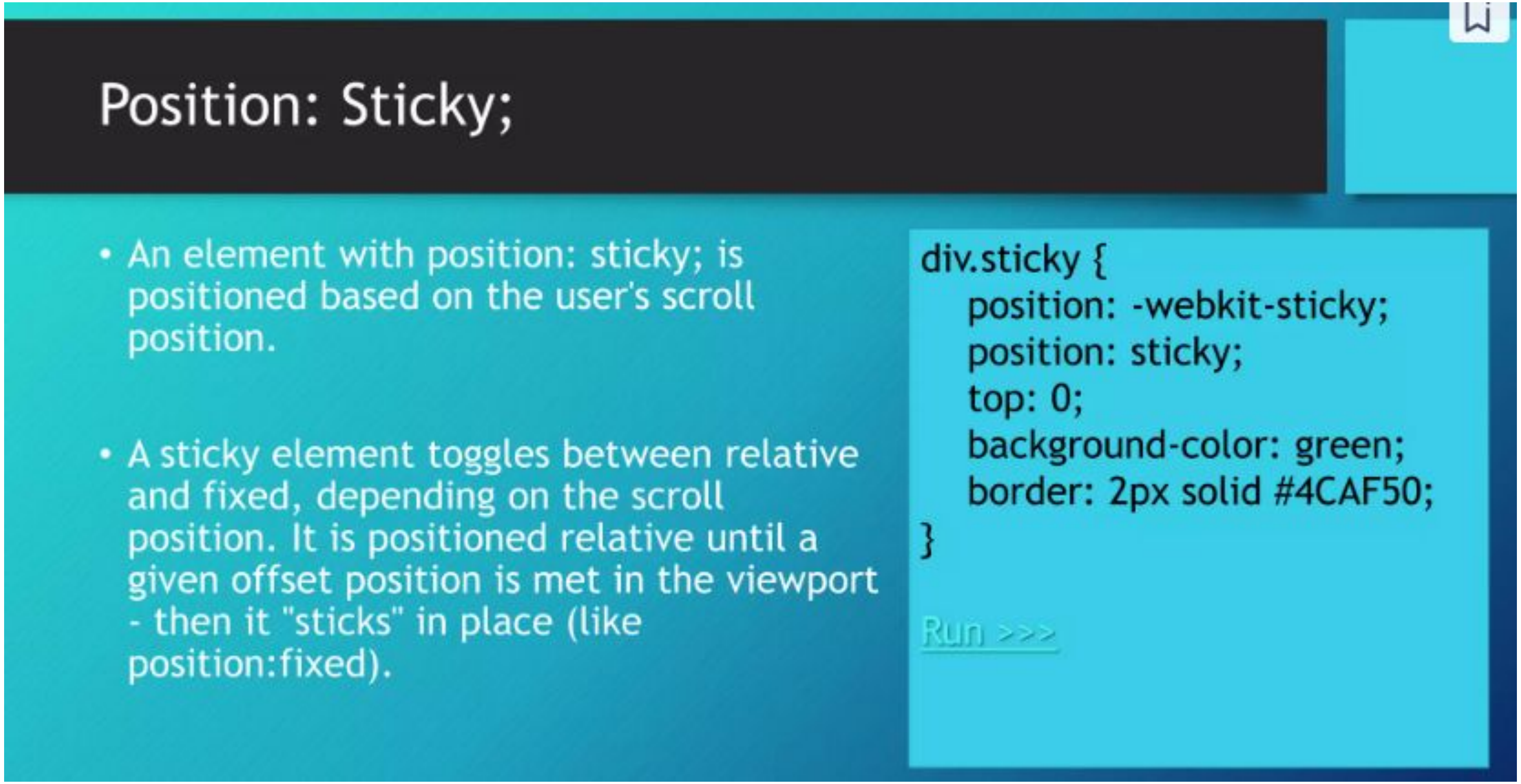


Position: Absolute;

- An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

```
div.relative {
    position: relative;
    width: 400px;
    height: 200px;
    border: 3px solid #73AD21;
}

div.absolute {
    position: absolute;
    top: 80px;
    right: 0;
    width: 200px;
    height: 100px;
    border: 3px solid #73AD21;
}
```

Position sticky:

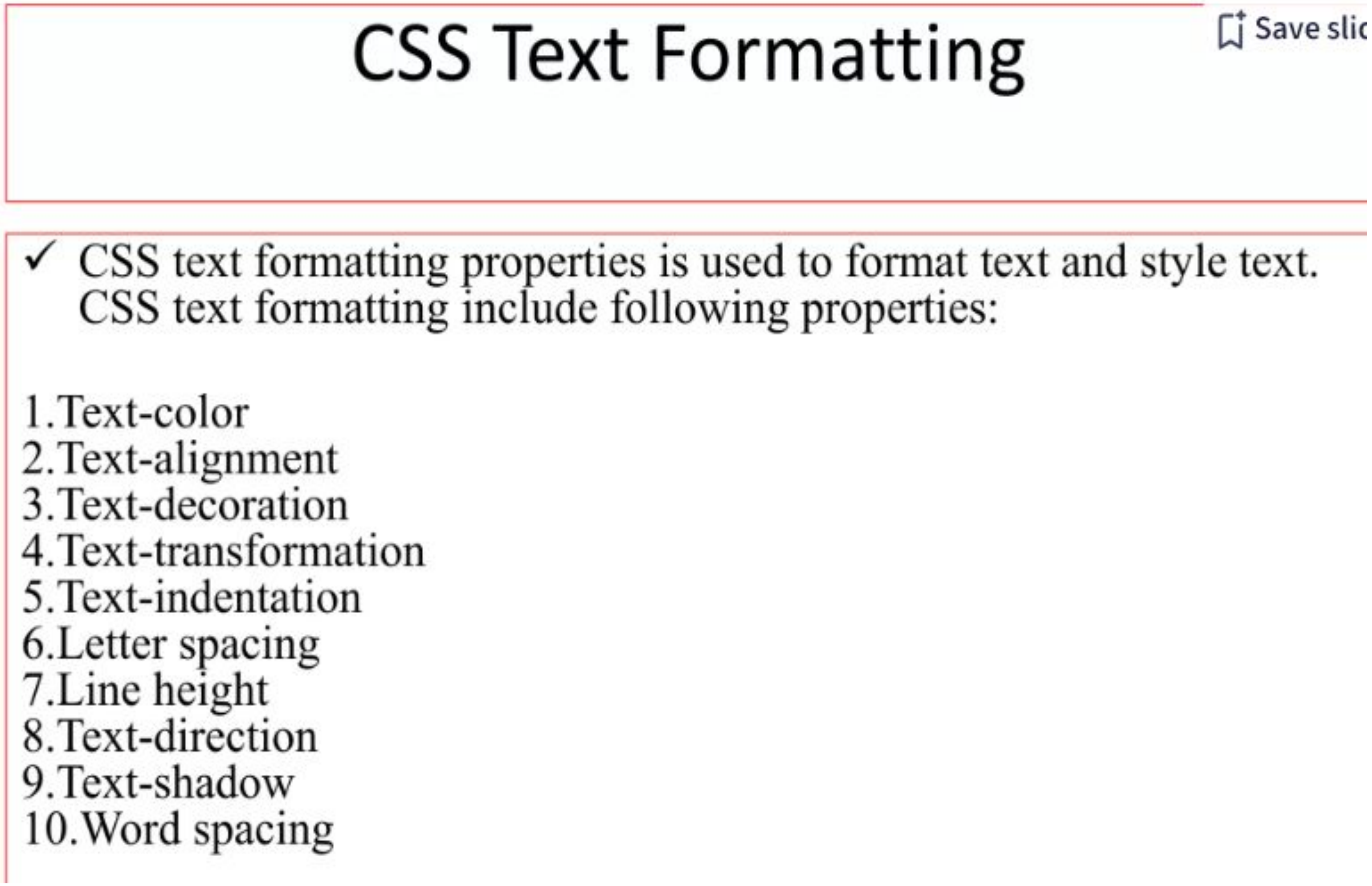


Position: Sticky;

- An element with position: sticky; is positioned based on the user's scroll position.
- A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

```
div.sticky {
    position: -webkit-sticky;
    position: sticky;
    top: 0;
    background-color: green;
    border: 2px solid #4CAF50;
}
```

Run >>>



CSS Text Formatting

✓ CSS text formatting properties is used to format text and style text. CSS text formatting include following properties:

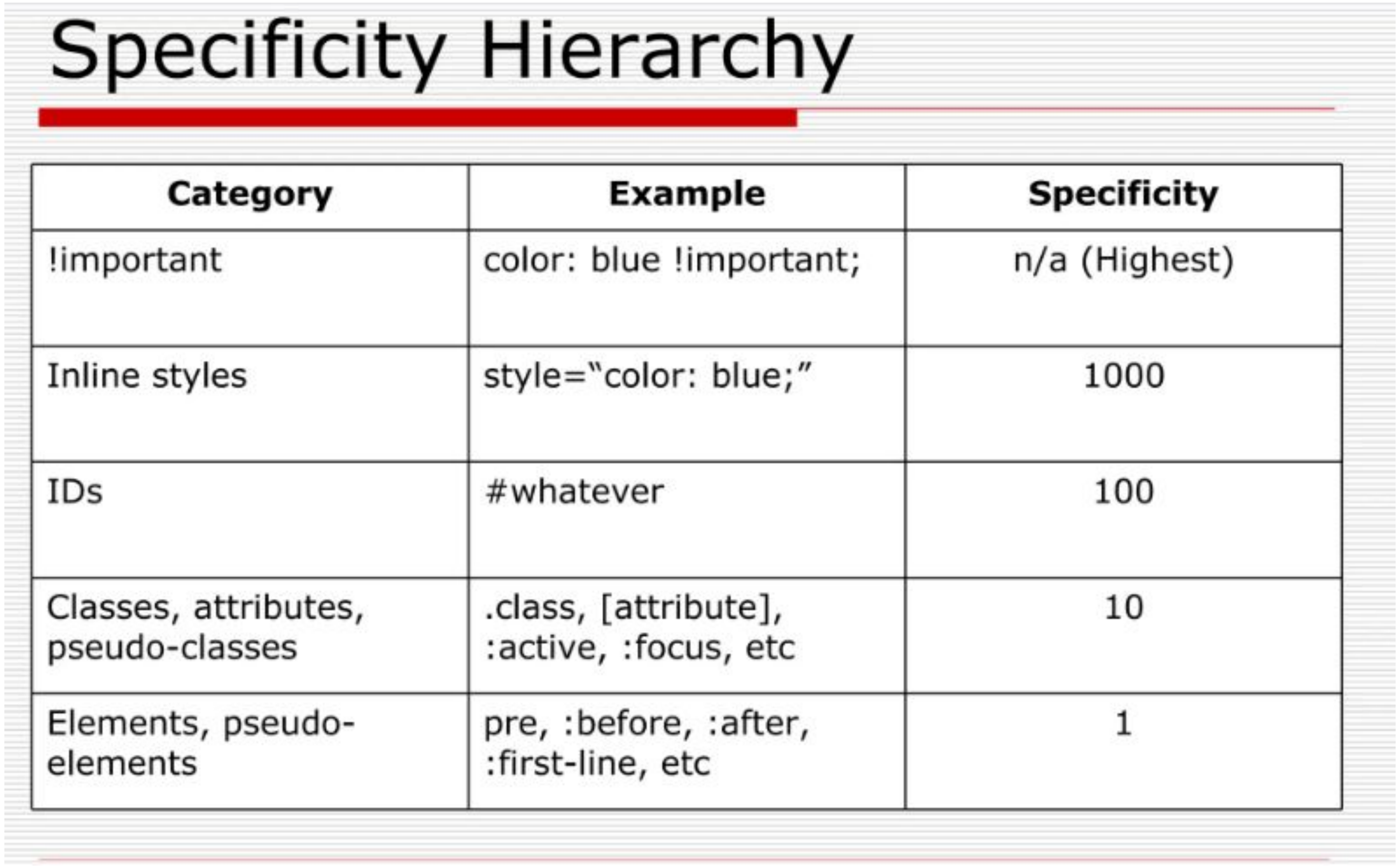
1. Text-color
2. Text-alignment
3. Text-decoration
4. Text-transformation
5. Text-indentation
6. Letter spacing
7. Line height
8. Text-direction
9. Text-shadow
10. Word spacing

Specificity?

What is Specificity?

If there are two or more CSS rules that point to the same element, the selector with the highest specificity value will "win", and its style declaration will be applied to that HTML element.

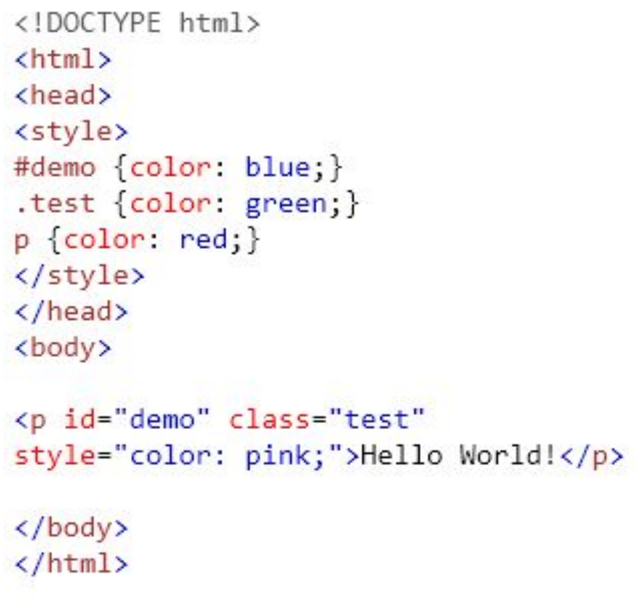
Think of specificity as a score/rank that determines which style declaration is ultimately applied to an element.



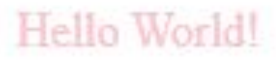
Specificity Hierarchy

Category	Example	Specificity
!important	color: blue !important;	n/a (Highest)
Inline styles	style="color: blue;"	1000
IDs	#whatever	100
Classes, attributes, pseudo-classes	.class, [attribute], :active, :focus, etc	10
Elements, pseudo-elements	pre, :before, :after, :first-line, etc	1

Example:



```html
<!DOCTYPE html>
<html>
<head>
<style>
#demo {color: blue;}
.test {color: green;}
p {color: red;}
</style>
</head>
<body>

<p id="demo" class="test"
style="color: pink;">Hello World!</p>

</body>
</html>
```



```
Hello World!
```

THANK
YOU