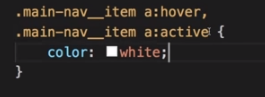
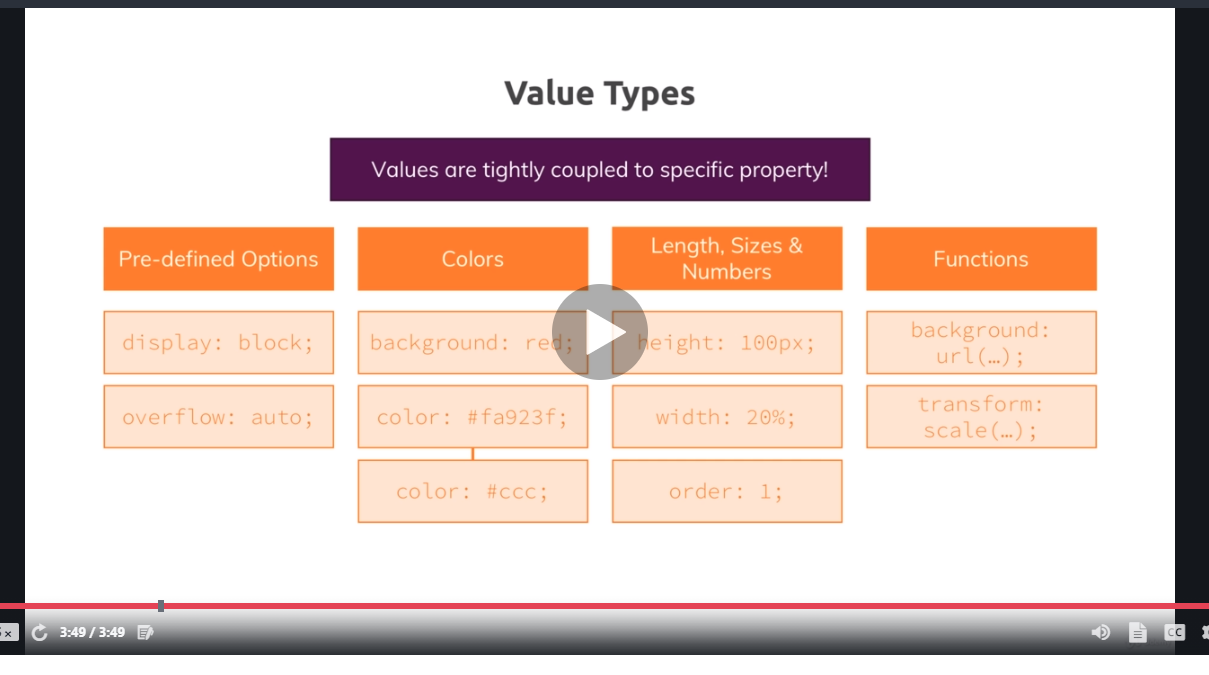
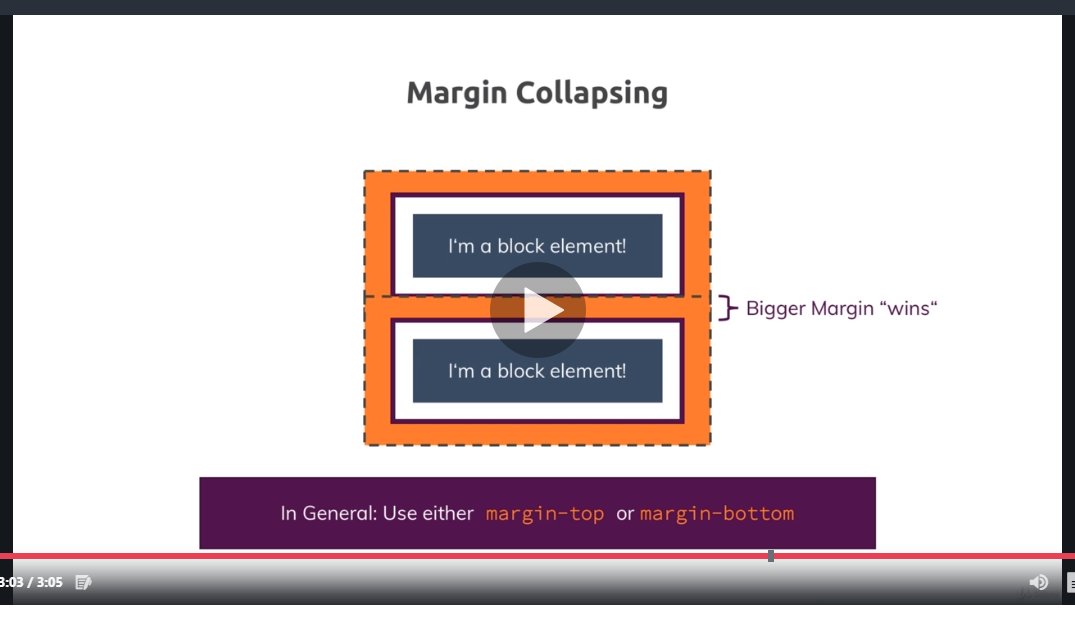
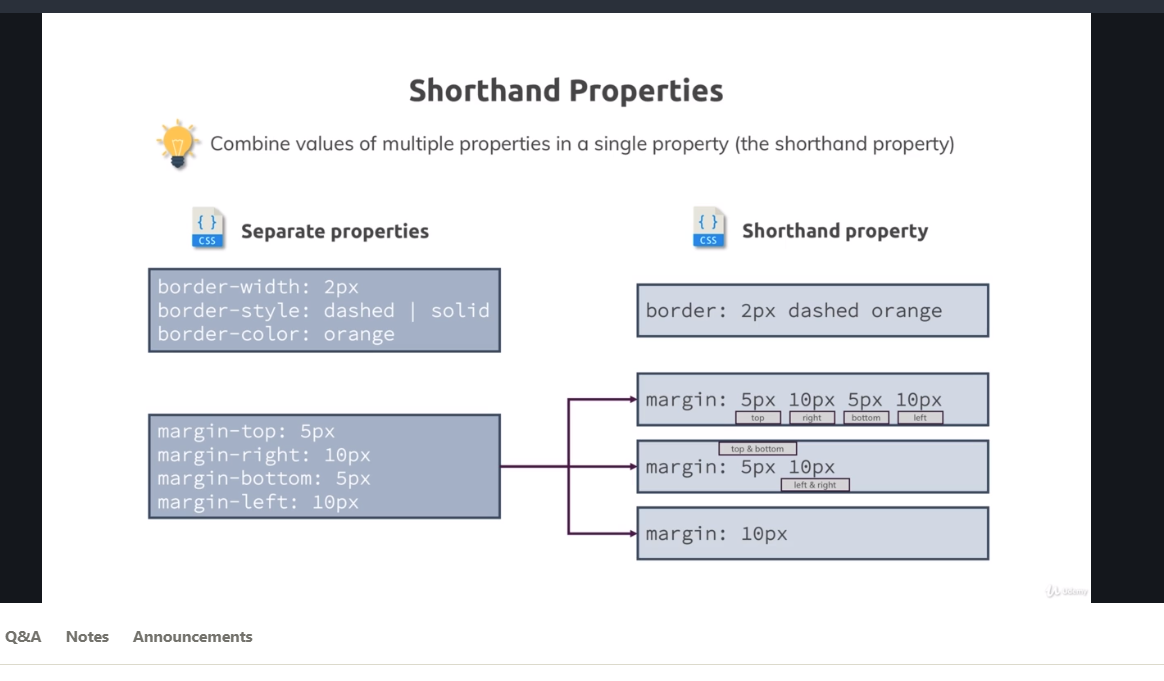


Grouping rules can be done with comma(,) separation like









Block-level elements occupy 100% width unlike Inline Elements. Consecutive Inline elements are positioned next to each other. Consecutive Block-level elements are positioned in the next line as they occupy 100% width.

HTML Refresher: Block-level vs Inline Elements

It's not really a CSS topic, though it's related to it: The difference between **block-level** and **inline elements**.

You can read a more detailed article (which also includes a YouTube video about HTML at the top of the page) here: <https://academind.com/learn/html/beginner-s-guide/diving-deeper-into-html#block-level-vs-inline-elements>

*Here's the executive summary:*

**Block-level elements** are rendered as a block and hence take up all the available horizontal space. You can set margin-top and margin-bottom and two block-level elements will render in two different lines.

Some examples are: <div> , <section> , <article> , <nav>  but also <h1> , <h2>  etc and <p> .

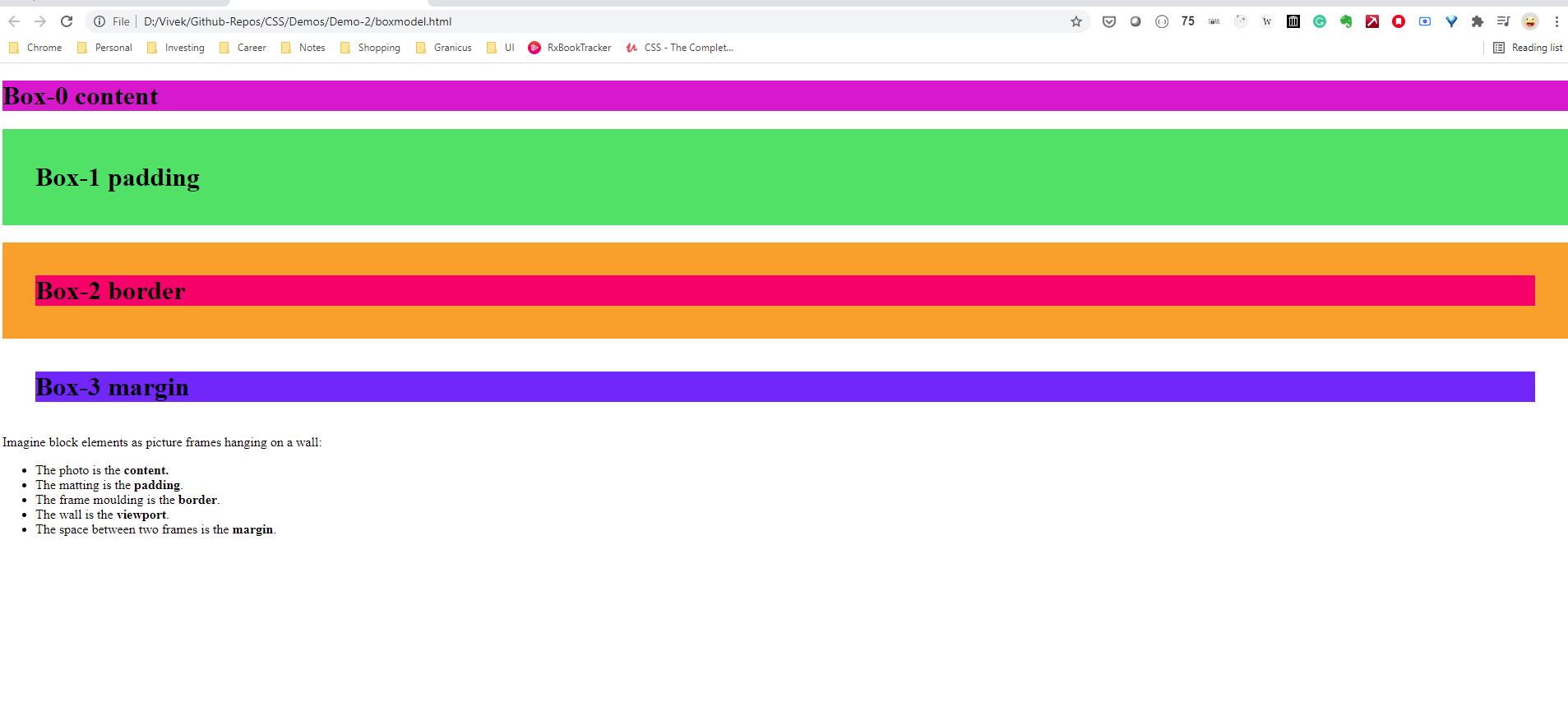
**Inline elements**on the other hand only take up the space they require to fit their content in. Hence two inline-elements will fit into the same line (as long as the combined content doesn't take up the entire space in which case a line break would be added).

They also use the box-model you learned about but margin-top  and margin-bottom  have no effect on the element. padding-top  and padding-bottom  also have a different effect. They don't push the adjacent content away but they will do so with the element border. You can read more about that behavior in the following article: <https://hacks.mozilla.org/2015/03/understanding-inline-box-model/>

Additionally, setting a width  or height  on an inline element also has no effect. The width and height is auto to take as much space as required by the content.

Logically, this makes sense since you don't want your inline elements to destroy your multi-line text-layout. If you want to do so or need both block-level and inline behavior, you can set display: inline-block  to merge behaviors.

Some example elements are: <a> , <span> , <img>



Box sizing: (a) Content-box (b) Border-box

Border-box is set in universal selector because display: block overrides it.

display:none,inline,block,inline-block

display:inline-block is a very useful option because it has advantages of both(a) Block level elements can behave like inline elements where elements can be next to each other and (b) you can set margin, border and padding like you do for block level elements.

display: none vs visibility: hidden

We had a look at display: none;  - this value removes the element to which you apply it from the document flow. This means that the element is not visible and it also doesn't "block its position". Other elements can (and will) take its place instead.

There is an alternative to that though.

If you only want to hide an element but you want to keep its place (i.e. other elements don't fill the empty spot), you can use visibility: hidden;

Here's a visual example:

1. .box-1 {
2. display: none;
3. }
5. .box-2 {
6. display: inline-block;
7. }

Will render:

x

where x  has the class box-2 . The first element just isn't displayed. **It's still part of the DOM though**, you can still access it via JavaScript for example.

Here's an example for visibility: hidden :

1. .box-1 {
2. visibility: hidden;
3. }
5. .box-2 {
6. display: inline-block;
7. }

Will render:

\_x

where \_  simply is an empty spot and x  has the class box-2 .

The element is only invisible, **it's not removed from the document flow and of course also not from the DOM.**

