

16/04/2025 - Forms and Input Fields

Types of input elements:

The screenshot shows a Udemy video player interface. The title of the video is "<input> Element Types". Below the title, there is a list of input element types with their descriptions. The "password" type is highlighted with a red box. A tooltip "Captured screenshot" appears over the "password" entry. The "date" type is also highlighted with a red box. A tooltip "However, it can only accept email addresses." appears over the "date" entry. The video player has a progress bar showing 3:25 / 5:05 and various control buttons at the bottom.

Input Type	Description
text	Single-line text field
email	Single line email
password	Single line password with hidden characters
number	Single line numbers
textarea	Multi-line text field
select	Dropdown list of options
date	Date picker However, it can only accept email addresses.

The screenshot shows a second Udemy video player interface, likely from the same video as the first one. The title is "<input> Element Types". Below the title, there is a list of input element types with their descriptions. The "checkbox" type is highlighted with a red box. The "radio" type is also highlighted with a red box. The "submit" type is highlighted with a red box. A tooltip "Captured screenshot" appears over the "submit" entry. The "color" type is highlighted with a red box. A tooltip "We have checkbox fields which are checkbox for multiple options." appears over the "color" entry. The video player has a progress bar showing 3:25 / 5:05 and various control buttons at the bottom.

Input Type	Description
checkbox	Checkbox field with multiple options
radio	Radio boxes with single option
file	File upload field
submit	Submit button Captured screenshot
range	Slider input to select from a range
color	Color picker We have checkbox fields which are checkbox for multiple options.

form element:

The screenshot shows a Udemy video player interface. At the top, it says "Modern HTML & CSS From The Beginning 2.0 (2024 Revamp)". The video title is "<form> Element". Below the title, there is some sample HTML code:

```
<form action="/api/process" method="POST">
    <input type="text" name="company" id="company" />
</form>
```

Below the code, there is a list of three facts:

- ✓ HTML can only display forms, not process them
- ✓ The <form> tag needs to have <input> tags
- ✓ The action is the server-side endpoint and the method is the HTTP method used to process it.

A small note next to the third fact says: "And when you submit that form, it'll actually send it to their servers and it will process it."

The video player has a progress bar showing 2:37 / 5:05. The bottom right corner shows the Udemy logo.

Form tag attributes :

```
<form>
    <div>
        <label for="name">Name</label>
        <input
            type="text" / "password"/ "email"
            id="name"
            placeholder="Enter Name"
            values = "Jascar"
            name="name"
            required
            minlength="3"
            maxlength="25">
            disabled
    </div>
```

Dropdown :

Select attribute :

Sample code :

```
<form>
    <label for="product">Select the Product</label>
    <br>
    <select name="product" id="product" multiple size= "2">
        <option value="iphone">iphone</option>
        <option value="ipad">ipad</option>
        <option value="Macbook">Macbook</option>
        <option value="macbook-air">Macbook-Air</option>
        <option value="iwatch">iwatch</option>
        <option value="airpods">airpods</option>
    </select>
    <br>
    <br>
    <input type="submit" value="Submit">
</form>
```

17/04/2025

Multimedia Tags & Tables

AUDIO -

```
<audio>
  <source src=".mp3" type = "audio/mp3" autoplay controls>
</audio>
```

(Or)

```
<audio src= ".mp3" type= "audio/mp3 autoplay controls">
</audio>
```

Audio loop -

```
<audio src= ".mp3" type= "audio/mp3 autoplay controls
  loop>
</audio>
```

VIDEO -

```
<video height="220" width="300" src=".8820216-
uhd_3840_2160_25fps.mp4" type="video/mp4" controls ></video>
(Or)
```

VIDEO with figure -

```
<figure>
  <video src="src=.8820216-uhd_3840_2160_25fps.mp4
  "type="video/mp4" controls ></video>

  <figurecaption>Video 1</figurecaption>
</figure>
```

TABLE :

```
<div>
  <table>
    <colgroup>
      <col style="background-color:tomato;">
```

```

<col style="background-color: burlywood;">
<col style="background-color: burlywood;">
</colgroup>

<thead>

<tr>

<th>Events</th>
<th>Date</th>
<th>Time </th>

</tr>

<tbody>
<tr>
<td>Campfire</td>
<td>17-April 2025</td>
<td>6pm IST</td>

</tr>

<tr>
<td>DJ Night</td>
<td>19-April-2025</td>
<td>8pm IST</td>
</tr>
<tr>
<td>Campfire </td>
<td>20-April-2025</td>
<td>6pm IST</td>
</tr>
</tbody>
</thead>
</table>
</div>

```

Iframe Element -

```
<iframe
src="https://www.youtube.com/embed/KDIR5KnxEwU?
si=n0BiOtBjO2ibDuGZ"
frameborder="0"
width="560" height="315"
></iframe>
<br><br>
<h4>Location :</h4>

<iframe
width="560" height="315"
src="https://www.google.com/maps/embed?pb=!1m18!
1m12!1m3!1d54955117.16386726!2d-161.5781907699605!
3d35.99350529000485!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!
3m3!1m2!1s0x54eab584e432360b%3A0x1c3bb99243deb742!
2sUnited%20States!5e1!3m2!1sen!2sin!4v1744889996648!5m2!
1sen!2sin" ></iframe>
```

18/04/2025 & 19/04/2025

CSS Selectors

Types of CSS :

The screenshot shows a Udemy video player interface. The title 'Implementing CSS' is at the top. Below it, three methods are listed: 'Inline CSS – Add CSS to the HTML tag' (marked with a red X), 'Internal CSS – Add CSS within <style> tags' (marked with a red X), and 'External CSS – CSS in its own file' (marked with a green checkmark). A small note at the bottom says 'And there's really three ways to do it.' The video player has a progress bar at the bottom.

Box Model :

The screenshot shows a Udemy video player interface. The title 'Box Model' is at the top. Below it, four components are listed with green checkmarks: 'Content – Text, images, etc', 'Padding – Space between content and border', 'Border – Separates the padding & margin', and 'Margin – Space outside of border'. To the right, a diagram illustrates the box model structure with nested boxes for content, padding, border, and margin. A note at the bottom says 'And if you inspect the web page with the dev tools and you go to the elements tab and you hover over'. The video player has a progress bar at the bottom.

Box model properties :

The screenshot shows a slide from a Udemy course titled "Modern HTML & CSS From The Beginning 2.0 (2024 Revamp)". The main title of the slide is "Box Model Properties". Below the title, there is a bulleted list of properties:

- ✓ width & height, max-width, max-height, min-width, min-height
- ✓ padding, padding-top, padding-right, padding-bottom, padding-left
- ✓ border, border-style, border-color, border-width
- ✓ margin, margin-top, margin-right, margin-bottom, margin-left
- ✓ box-sizing

At the bottom left of the slide, there is a note: "If you set a max width of 600 pixels, it's the widest it's going to go is 600 pixels." On the right side of the slide, there is a "53" icon and the Udemy logo.

Padding :

The screenshot shows a browser developer tools interface. On the left, the "Elements" panel displays the following CSS code:

```
html { font-family: 'Poppins', sans-serif; }

.box {
    background-color: darkblue;
    color: white;
    /* padding: 20px; */
    width: 300px;

    /* padding-top: 20px;
    padding-right: 40px;
    padding-bottom: 100px;
    padding-left: 60px; */

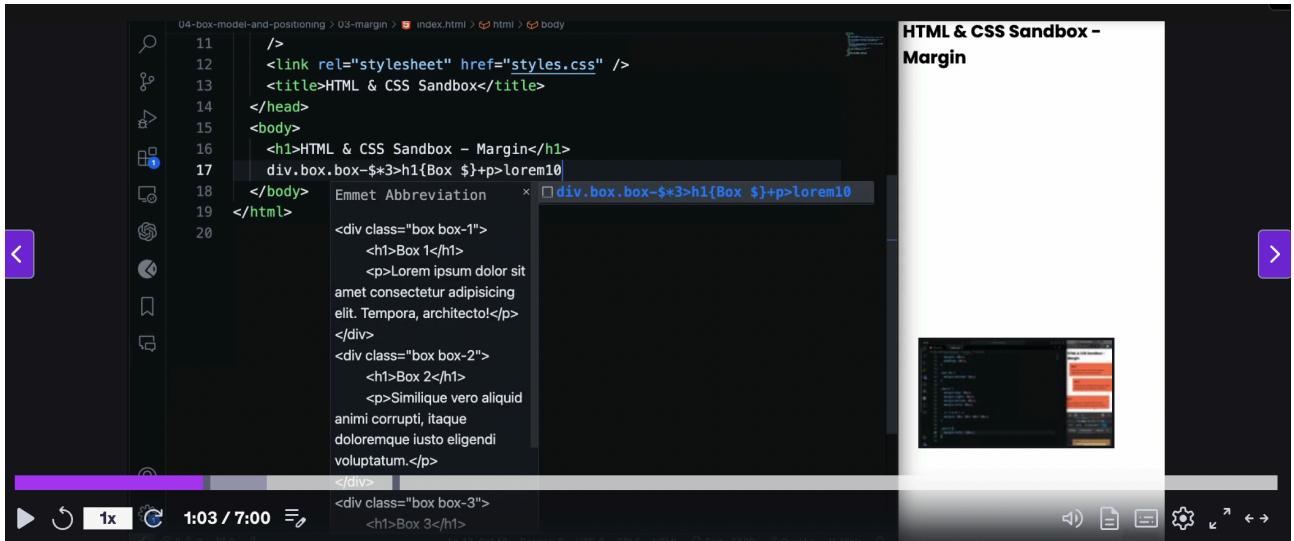
    /* T-R-B-L */
    /* padding: 20px 40px 100px 60px; */

    /* T-LR-B */
    /* padding: 20px 40px; */
    /* T-LR-B */
    padding: 20px 40px 30px;
```

On the right, the "Elements" panel shows a visual representation of the box model for an element with the class "box". The box has a dark blue background and white text. It is 300px wide and 215.812px tall. The box model components are labeled as follows:

- margin: 30px (outermost)
- border: 40px (second outermost)
- padding: 20px (innermost)
- content area: 300x215.812 (the actual content dimensions)

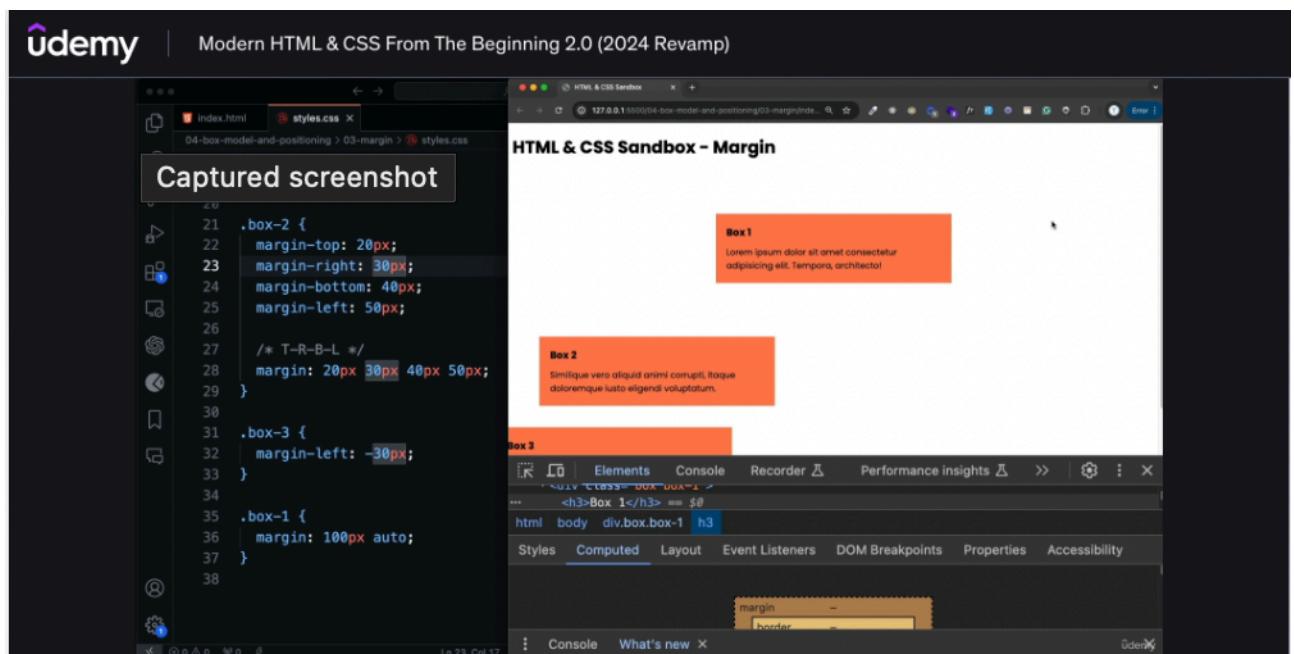
Shortcut to create multiple :



A screenshot of the Visual Studio Code (VS Code) interface. The left side shows the code editor with Emmet Abbreviations expanded. The right side shows the browser preview window titled "HTML & CSS Sandbox - Margin". The browser displays three orange boxes labeled Box 1, Box 2, and Box 3, each containing placeholder text. The status bar at the bottom shows the zoom level (1x), time (1:03 / 7:00), and other icons.

```
04-box-model-and-positioning > 03-margin > index.html > styles.css > body
11  />
12  <link rel="stylesheet" href="styles.css" />
13  <title>HTML & CSS Sandbox</title>
14  </head>
15  <body>
16  <h1>HTML & CSS Sandbox - Margin</h1>
17  div.box.box-$*3>h1{Box $}+p>lorem10
18  </body> Emmet Abbreviation x div.box.box-$*3>h1{Box $}+p>lorem10
19  </html>
20
21  <div class="box box-1">
22  <h1>Box 1</h1>
23  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Tempora, architecto!
24  </p>
25  <div class="box box-2">
26  <h1>Box 2</h1>
27  <p>Similique vero aliquid animi corrupti, itaque doloremque iusto eligendi voluptatum.</p>
28  </div>
29  <div class="box box-3">
30  <h1>Box 3</h1>
31
32
33
34
35
36
37
38
```

Margin :



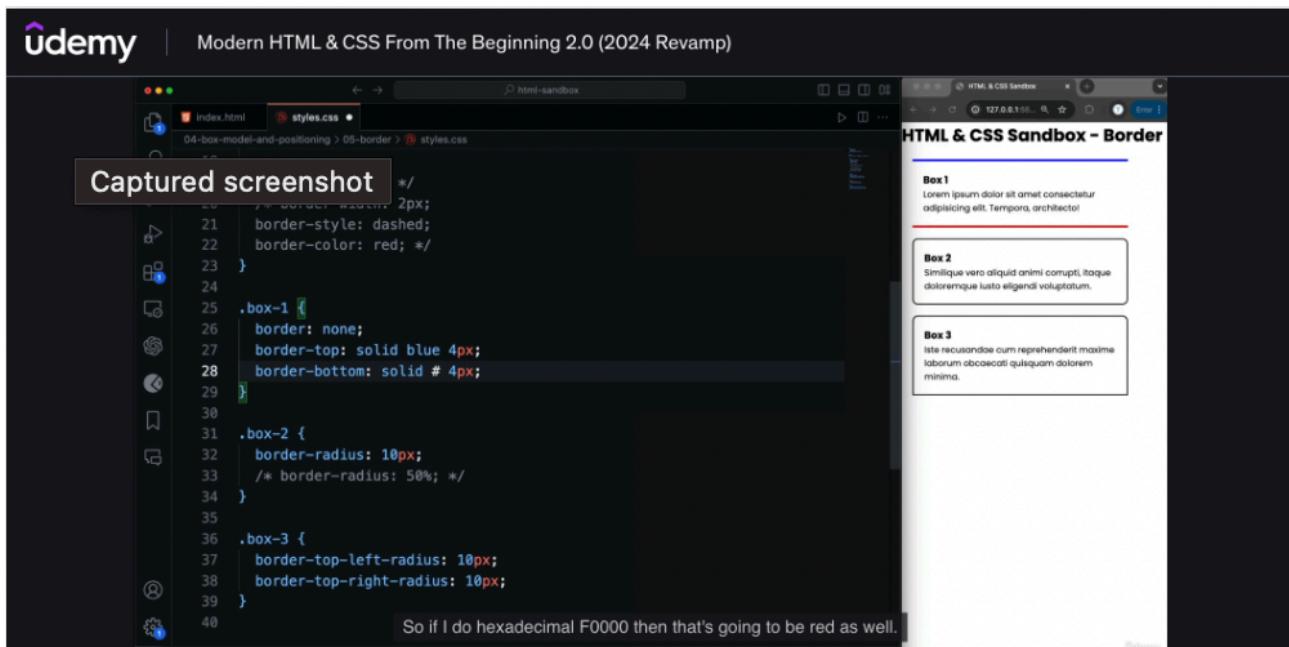
A screenshot of a browser window displaying a "Captured screenshot" from an Udemy course titled "Modern HTML & CSS From The Beginning 2.0 (2024 Revamp)". The browser has two tabs open: "index.html" and "styles.css". The "index.html" tab shows a CSS file with rules for ".box-1", ".box-2", and ".box-3". The "styles.css" tab shows the browser developer tools' Elements panel with the "Styles" tab selected. It highlights the "margin" property for ".box-1" and shows the computed value of "100px auto". The browser preview window on the right shows three orange boxes labeled Box 1, Box 2, and Box 3, each with its own margin properties applied.

```
.box-2 {
  margin-top: 20px;
  margin-right: 30px;
  margin-bottom: 40px;
  margin-left: 50px;
  /* T-R-B-L */
  margin: 20px 30px 40px 50px;
}

.box-3 {
  margin-left: -30px;
}

.box-1 {
  margin: 100px auto;
```

border:



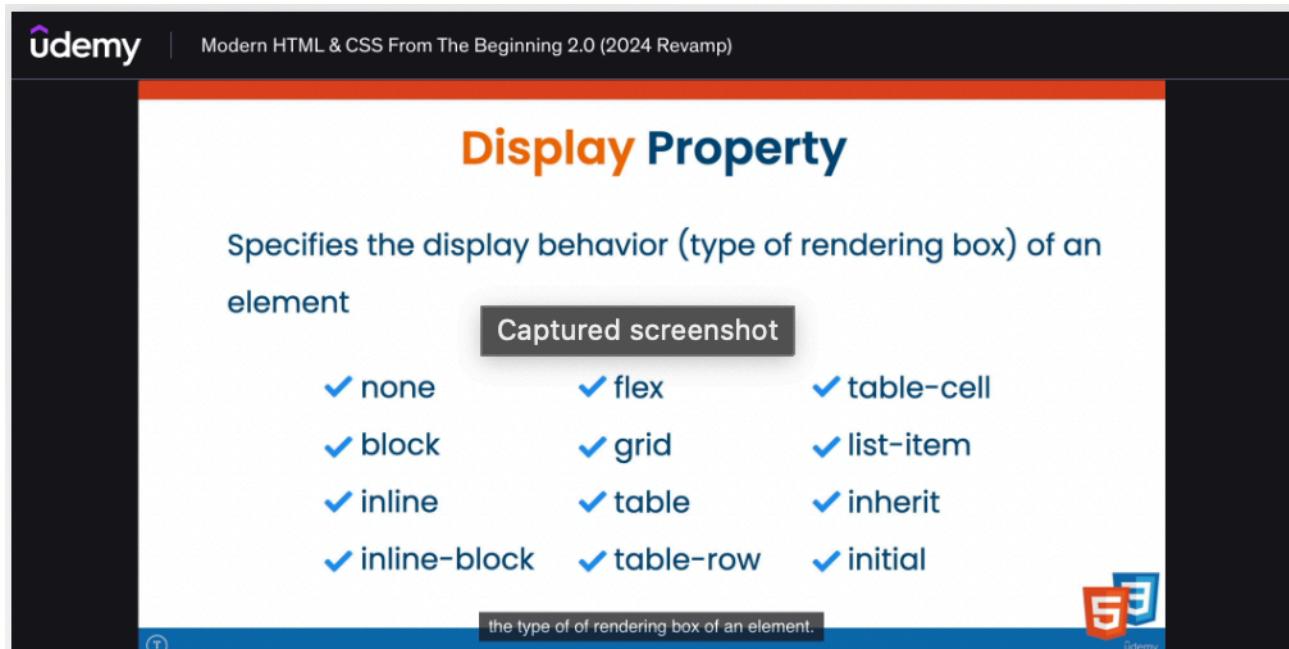
The screenshot shows a browser window with the Udemy logo and the title "Modern HTML & CSS From The Beginning 2.0 (2024 Revamp)". The main content area displays a code editor with a file named "index.html" and a "styles.css" file. The "index.html" file contains three

elements with class names "box-1", "box-2", and "box-3". The "styles.css" file contains the following CSS code:

```
21 border: 2px solid red;
22 border-style: dashed;
23 border-color: red; */
24
25 .box-1 {
26   border: none;
27   border-top: solid blue 4px;
28   border-bottom: solid # 4px;
29 }
30
31 .box-2 {
32   border-radius: 10px;
33   /* border-radius: 50%; */
34 }
35
36 .box-3 {
37   border-top-left-radius: 10px;
38   border-top-right-radius: 10px;
39 }
```

A tooltip at the bottom of the code editor says: "So if I do hexdecimal F0000 then that's going to be red as well." To the right of the code editor is a preview window titled "HTML & CSS Sandbox - Border" showing three boxes labeled "Box 1", "Box 2", and "Box 3" with different border styles.

Display Property:



The screenshot shows a browser window with the Udemy logo and the title "Modern HTML & CSS From The Beginning 2.0 (2024 Revamp)". The main content area has a large orange header with the text "Display Property". Below the header, the text "Specifies the display behavior (type of rendering box) of an element" is displayed. A "Captured screenshot" button is visible. Below this, a table lists nine values with checkmarks:

✓ none	✓ flex	✓ table-cell
✓ block	✓ grid	✓ list-item
✓ inline	✓ table	✓ inherit
✓ inline-block	✓ table-row	✓ initial

A tooltip at the bottom of the table says: "the type of of rendering box of an element." In the bottom right corner, there is a "HTML5" logo.

Position Property:

Modern HTML & CSS From The Beginning 2.0 (2024 Revamp)

Position Property

Captured screenshot

Specifies how to position elements on the page relative to the rest of the layout

- ✓ static
- ✓ relative
- ✓ absolute
- ✓ fixed
- ✓ sticky

So the first the first value is static and that's the default.

Modern HTML & CSS From The Beginning 2.0 (2024 Revamp)

```
index.html styles.css
04-box-model-and-positioning > 07-positioning > styles.css
  ↳ top: 0;
    right: ;
    z-index: 2;
  }
  .box-4 {
    position: sticky;
    top: 200px;
  }
  .box-5 {
    position: relative;
    width: 600px;
    height: 600px;
    background: coral;
    z-index: 1;
  }
  .box-6 {
    position: absolute;
    bottom: 20px;
    right: 20px;
  }
```

If I set this to 100, then obviously box five is going to go on top, but I'll keep that at one.

Fixed

Box 5

Absolute

Box Shadow:

The screenshot shows a Udemy video player interface. The title 'Box Shadow' is displayed in large orange and blue letters. Below the title, a CSS code snippet is shown: `box-shadow: 5px 6px 8px 6px #000;`. This code is annotated with labels: 'HORIZONTAL OFFSET' (above the first two values), 'BLUR RADIUS' (above the third value), 'COLOR' (above the fourth value), 'VERTICAL OFFSET' (below the first two values), and 'SPREAD RADIUS' (below the third value). A list of three bullet points follows:

- ✓ Horizontal & Vertical Offset - How far from the element each way
- ✓ Blur Radius (Optional) - How blurry the shadow is
- ✓ Spread Radius So now we're going to look at the box shadow property, which should grow or shrink allows us to add a drop shadow to our

 The video player controls at the bottom include play, volume, and progress indicators.

The screenshot shows a developer environment with a code editor and a browser preview window. The code editor displays a file named 'styles.css' with the following content:

```
04-box-model-and-positioning > 08-box-shadow > styles.css
...
11 .box {
12   width: 200px;
13   height: 200px;
14   display: flex;
15   align-items: center;
16   justify-content: center;
17   border: 1px solid #ccc;
18   border-radius: 10px;
}
...
21 .box-1 {
22   box-shadow: 10px 10px 10px 2px #000;
}
...
25 .box-2 {
26   box-shadow: 5px 4px 10px 2px rgba(0, 0, 0, 0.5);
}
...
29 .box-3 {
30   box-shadow: 5px 4px 10px 2px rgba(0, 0, 0, 0.5), 0
10px 5px red;
}
```

The browser preview window shows three boxes labeled 'Box 1', 'Box 2', and 'Box 3'. Box 1 has a black shadow. Box 2 has a semi-transparent black shadow. Box 3 has a red shadow. A callout box highlights the red shadow of Box 3 with the text: 'So now, not only do we have that, this one, this shadow, but now we have the red shadow here below.' The browser tabs show 'Index.html' and 'HTML & CSS Sandbox'.

23/04/2025

Grid and Flexboxs

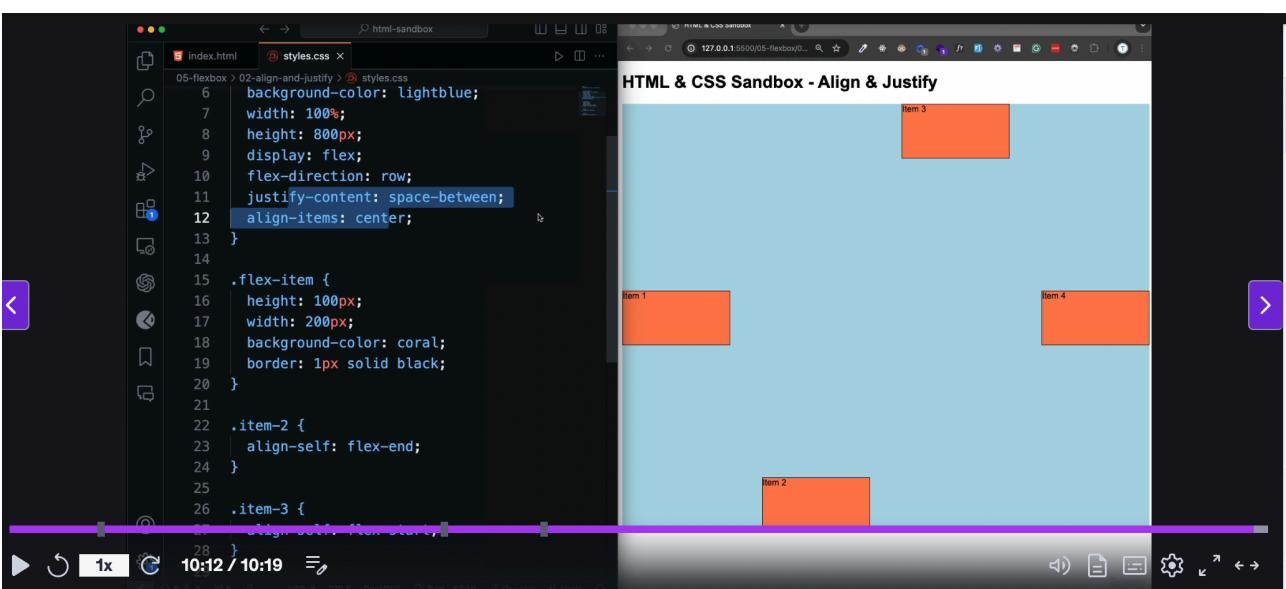
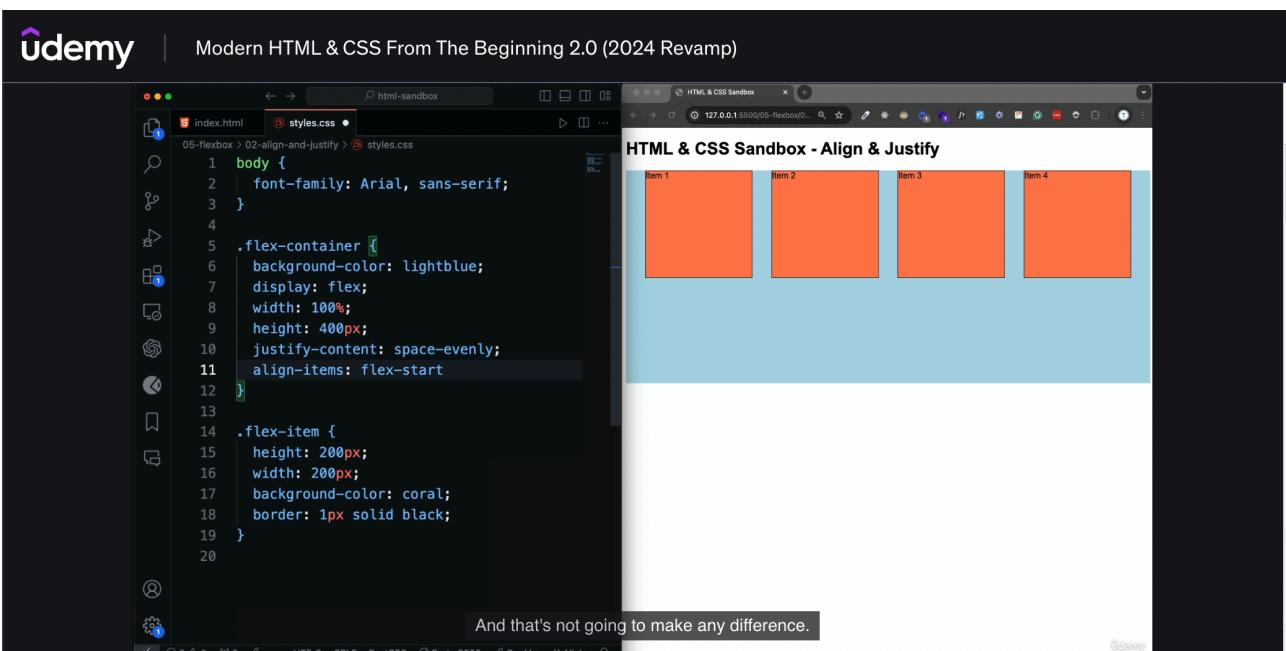
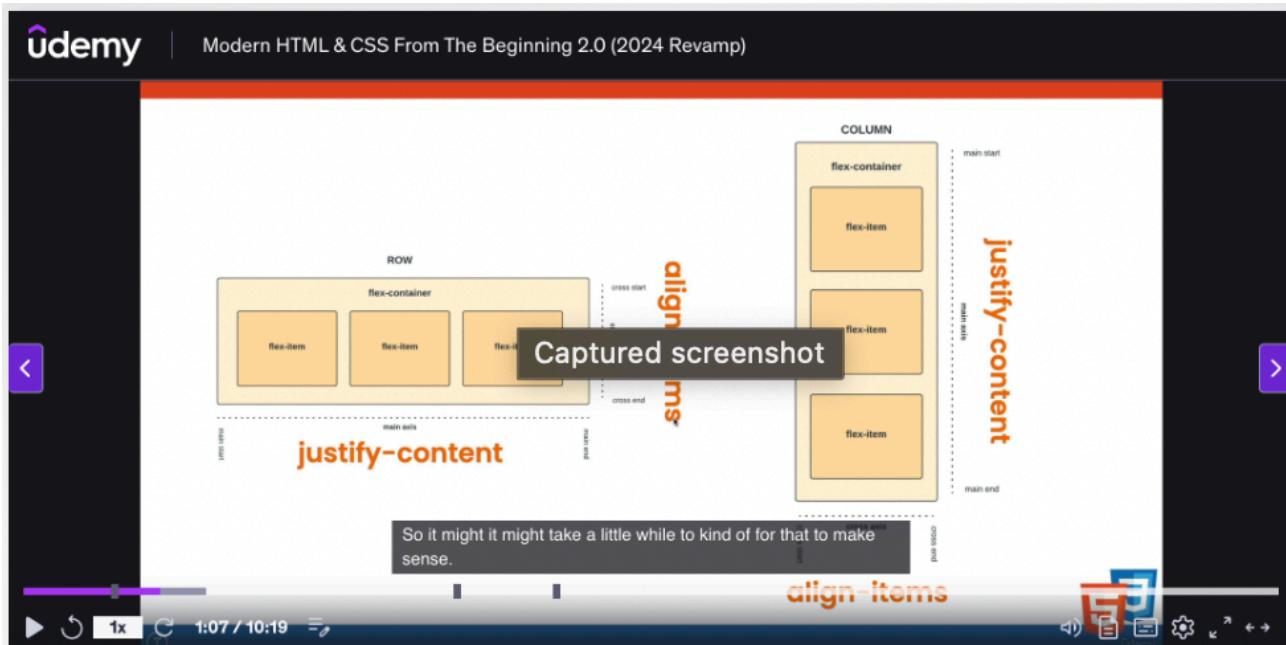
Flexbox Row

The diagram illustrates a **flex-container** with three **flex-item**s arranged horizontally. The horizontal axis is labeled **main axis**, and the vertical axis is labeled **cross axis**. The **main start** is at the top, and the **main end** is at the bottom. The **cross start** is on the left, and the **cross end** is on the right. A callout box states: "And by default it's going to be a row which is horizontal a column is vertical."

Flexbox column

The diagram illustrates a **flex-container** with three **flex-item**s arranged vertically. The vertical axis is labeled **main axis**, and the horizontal axis is labeled **cross axis**. The **main start** is at the top, and the **main end** is at the bottom. The **cross start** is on the left, and the **cross end** is on the right. A callout box states: "And the only thing you have to do to make it a column is set a property called Flex direction."

Align Items :



A screenshot of a code editor showing a CSS file named styles.css. The code defines a class .flex-item with properties like background: coral; border: 1px solid black; padding: 20px; and flex: 1;. It also defines three item classes: .item-1, .item-2, and .item-3, each with specific flex-grow values (2, 1, and 2 respectively) and flex-basis values (200px). To the right of the editor is a preview window showing three orange rectangular items labeled 'Item 1', 'Item 2', and 'Item 3' arranged horizontally.

```
11 .flex-item {  
12   background: coral;  
13   border: 1px solid black;  
14   padding: 20px;  
15   /* flex: 1 0 0; */  
16   flex: 1;  
17 }  
18  
19 .item-1,  
20 .item-2 {  
21   /* flex-shrink: 2; */  
22   /* flex-grow: 1; */  
23 }  
24  
25 .item-3 {  
26   /* flex-grow: 2; */  
27   /* flex-shrink: 0; */  
28   /* flex-basis: 200px; */  
29 }  
30
```

In fact, let's say we wanted item one to be kind of like the the main area.

Flex Order:

A screenshot of a code editor showing a CSS file named styles.css. The code defines six item classes: .item-1 through .item-6, each with a different order value (2, 1, 3, 6, 5, and 4 respectively). To the right of the editor is a browser window titled "HTML & CSS Sandbox - Flex Order" showing a row of six items labeled "Item 2", "Item 1", "Item 3", "Item 6", "Item 5", and "Item 4". The items are arranged in the order defined by their flex-order values.

```
15 }  
16  
17 .item-1 {  
18   order: 2;  
19 }  
20  
21 .item-2 {  
22   order: 1;  
23 }  
24  
25 .item-3 {  
26   order: 3;  
27 }  
28  
29 .item-4 {  
30   order: 6;  
31 }  
32  
33 .item-5 {  
34   order: 5;  
35 }
```

Item four has six, five has five, and six has four.

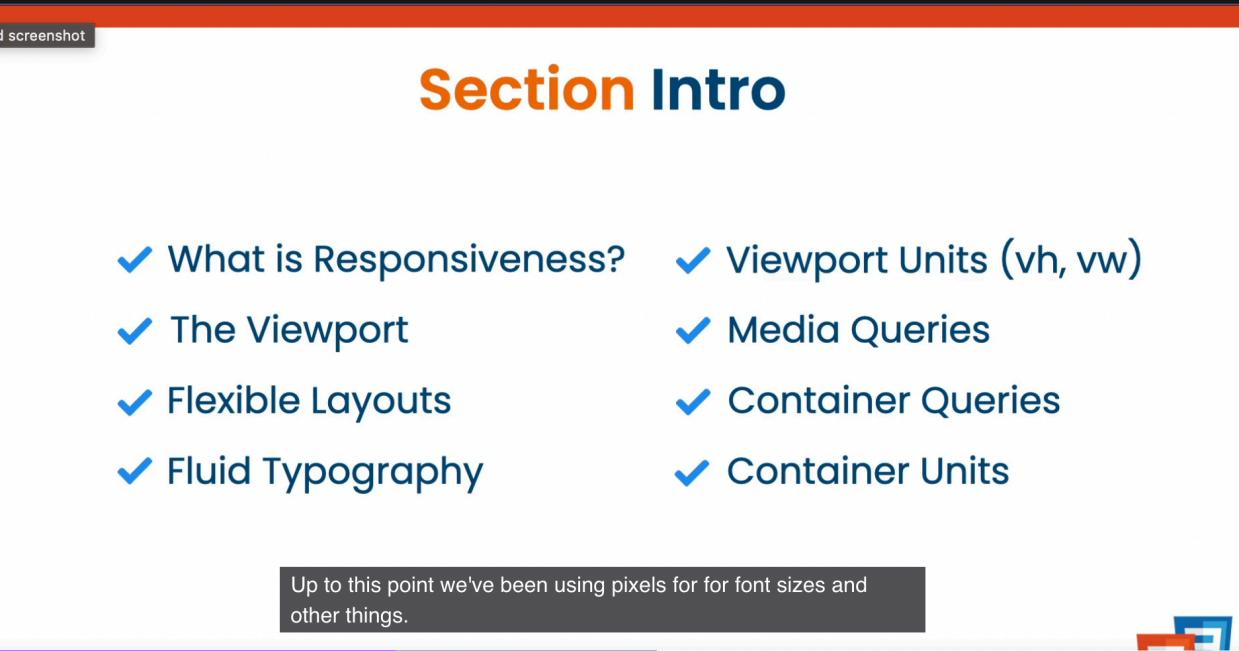
24/05/2025 - Responsive design

Captured screenshot

Section Intro

- ✓ What is Responsiveness?
- ✓ The Viewport
- ✓ Flexible Layouts
- ✓ Fluid Typography
- ✓ Viewport Units (vh, vw)
- ✓ Media Queries
- ✓ Container Queries
- ✓ Container Units

Up to this point we've been using pixels for font sizes and other things.

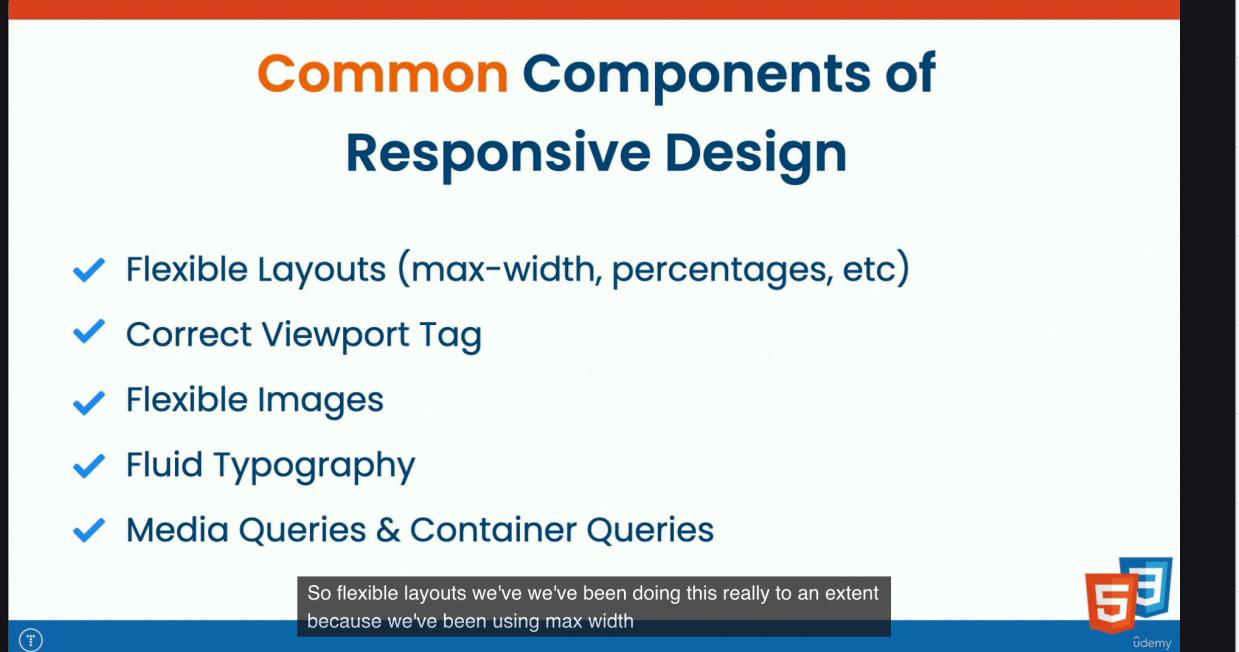


A screenshot of a code editor window titled "Section Intro". The code editor shows several lines of CSS-like code. A tooltip box is overlaid on the editor, containing the text: "Up to this point we've been using pixels for font sizes and other things." The background of the slide features a dark orange header bar at the top and bottom.

Common Components of Responsive Design

- ✓ Flexible Layouts (max-width, percentages, etc)
- ✓ Correct Viewport Tag
- ✓ Flexible Images
- ✓ Fluid Typography
- ✓ Media Queries & Container Queries

So flexible layouts we've been doing this really to an extent because we've been using max width



A screenshot of a code editor window titled "Common Components of Responsive Design". The code editor shows several lines of CSS-like code. A tooltip box is overlaid on the editor, containing the text: "So flexible layouts we've been doing this really to an extent because we've been using max width". The background of the slide features a dark orange header bar at the top and bottom.

Absolute

- ✓ Pixels (px)
- ✓ Centimeters (cm)
- ✓ Millimeters (mm)
- ✓ Inches (in)
- ✓ Points (pt)
- ✓ Picas (pc)

Relative

- ✓ Percentages (%)
- ✓ em & rem
- ✓ Viewport Height/Width (vh/vw)
- ✓ Viewport Max (vmax)
- ✓ Viewport Min (vmin)

So obviously choose the most flexible unit first!

Rem:

index.html x styles.css x

```

18 padding: 20px;
19 border: 1px solid black;
20 /* font-size: 20px; */
21 }
22 p {
23   font-size: 1rem;
24   margin: 1rem 0;
25 }
26
27
28 h1 {
29   font-size: 2rem;
30   padding: 0.3rem 0;
31 }
32

```

06-responsive-media-queries > 02-rem-em-units > styles.css

padding: 20px;
border: 1px solid black;
/* font-size: 20px; */

p {
 font-size: 1rem;
 margin: 1rem 0;

h1 {
 font-size: 2rem;
 padding: 0.3rem 0;

10:14 / 12:53

Em :

A screenshot of a browser's developer tools element inspector. The left panel shows a portion of a CSS file with the following code:

```
19 border: 1px solid black;
20 /* font-size: 20px; */
21 }
22
23 p {
24   font-size: 1rem;
25   margin: 1rem 0;
26 }
27
28 h1 {
29   font-size: 2rem;
30   padding: 0.3rem 0;
31 }
32
33 ul {
34   font-size: 1.5em;
35 }
```

The right panel displays the element tree and properties for a selected element. The 'font-size' property for the `ul` element is highlighted in orange, indicating it is being inspected. The value is listed as `1.5em`. The element itself is a list item with the text "Poppins, sans-serif".

Vh & vw:

A screenshot of a browser's developer tools element inspector. The left panel shows a portion of a CSS file with the following code:

```
5 }
6
7 body {
8   font-family: 'Poppins', sans-serif;
9 }
10
11 .container {
Captured screenshot background: coral;
13   height: 50vh;
14   width: 50vw;
15 }
16
```

The right panel displays the element tree and properties for a selected element. The `height` and `width` properties for the `.container` element are highlighted in orange, indicating they are being inspected. Both are set to `50vh` and `50vw` respectively. The element itself is a red box containing the text "Lorem ipsum dolor sit amet, consectetur adipisciing elit, expedita temporibus illum eius omnis doloremque corrupti delectus itaque voluptatum recusandae! Deserunt maxime nihil quibusdam quis adipisci iste, reprehenderit occassum quod? Eos!"

Media Query Syntax :

Media Queries



```
@media screen and (max-width: 600px) {  
    /* CSS rules */  
}
```

- ✓ **@media** rule defines different style rules
- ✓ **Media features** - min/max-height, min/max-width, orientation
- ✓ **Media types** - screen, print, speech, etc



Captured screenshot

Typical Breakpoints

- ✓ **576px.** Smartphones
- ✓ **768px.** Tablets
- ✓ **992px.** Desktop
- ✓ **1024px.** Landscape
- ✓ **1200px.** Desktop/Widescreen



udemy

Media queries syntax:

The screenshot shows a code editor with an open file named 'index.html' containing an HTML structure, and a file named 'styles.css' containing the following CSS code:

```
06-responsive-media-queries > 05-media-queries > styles.css
30 .smartphone {
31   background: lightcoral;
32 }
33 }
34
35 @media (orientation: landscape) {
36   .landscape {
37     background: lightgoldenrodyellow;
38   }
39 }
40
41 @media (max-height: 500px) {
42   .height {
43     background: lightseagreen;
44   }
45 }
46
47 @media (min-width: 768px) and (max-width: 991px) {
48   body {
49     background: #333;
50   }
51 }
```

To the right of the code editor is a browser window showing a color-coded device simulation. The simulation includes a red bar at the top labeled 'Widescreen', followed by a blue bar labeled 'Normal', a green bar labeled 'Tablet', a red bar labeled 'Smartphone', a light blue bar labeled 'Landscape', and a light green bar labeled 'Height'. The Udemy logo is visible in the bottom right corner of the browser window.

Container queries:

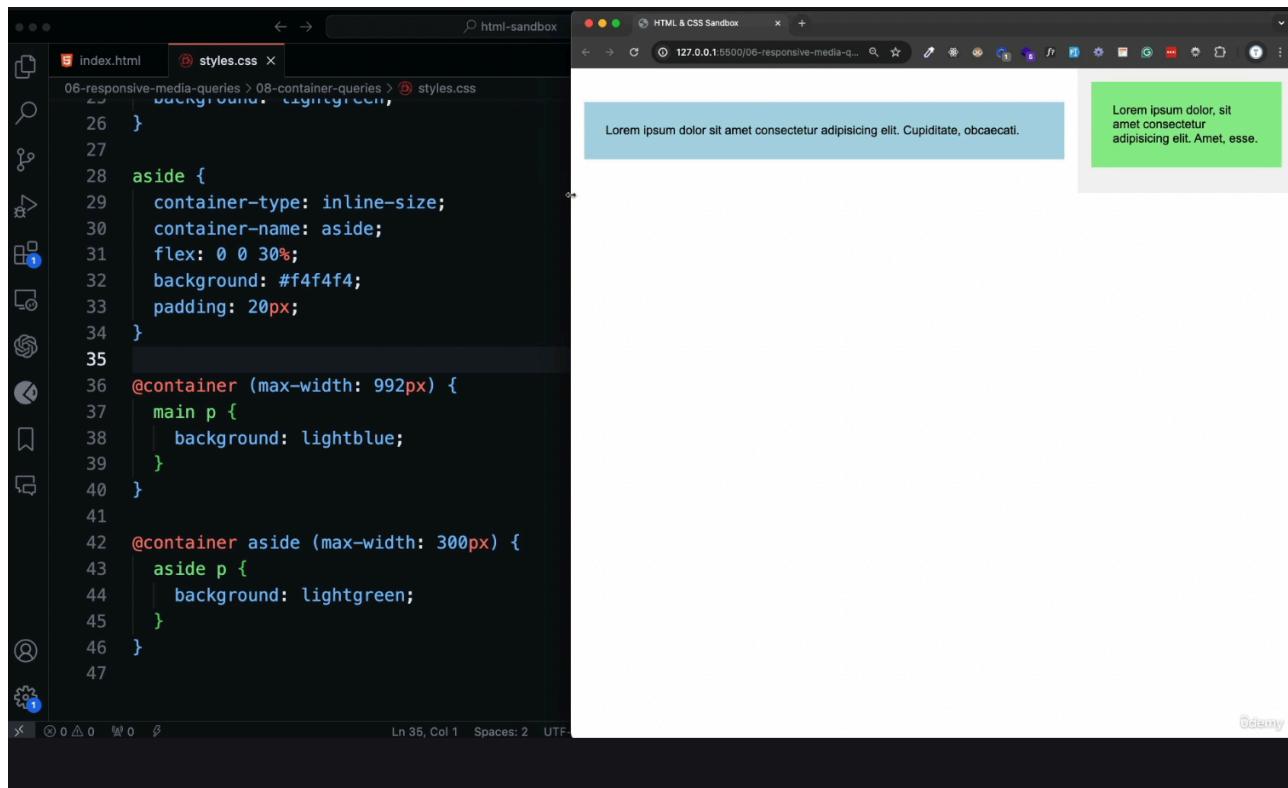
The screenshot shows a web page with the title "Container Queries" in large orange and blue text. Below the title is a code snippet:

```
● ● ●
@container (max-width: 600px) {
  /* CSS rules */
}
```

Below the code, there is a list of features with checkmarks:

- ✓ **@container** rule defines different style rules
- ✓ **container-type** property creates a container (inline-size, size)
- ✓ **container-name** is used to name containers

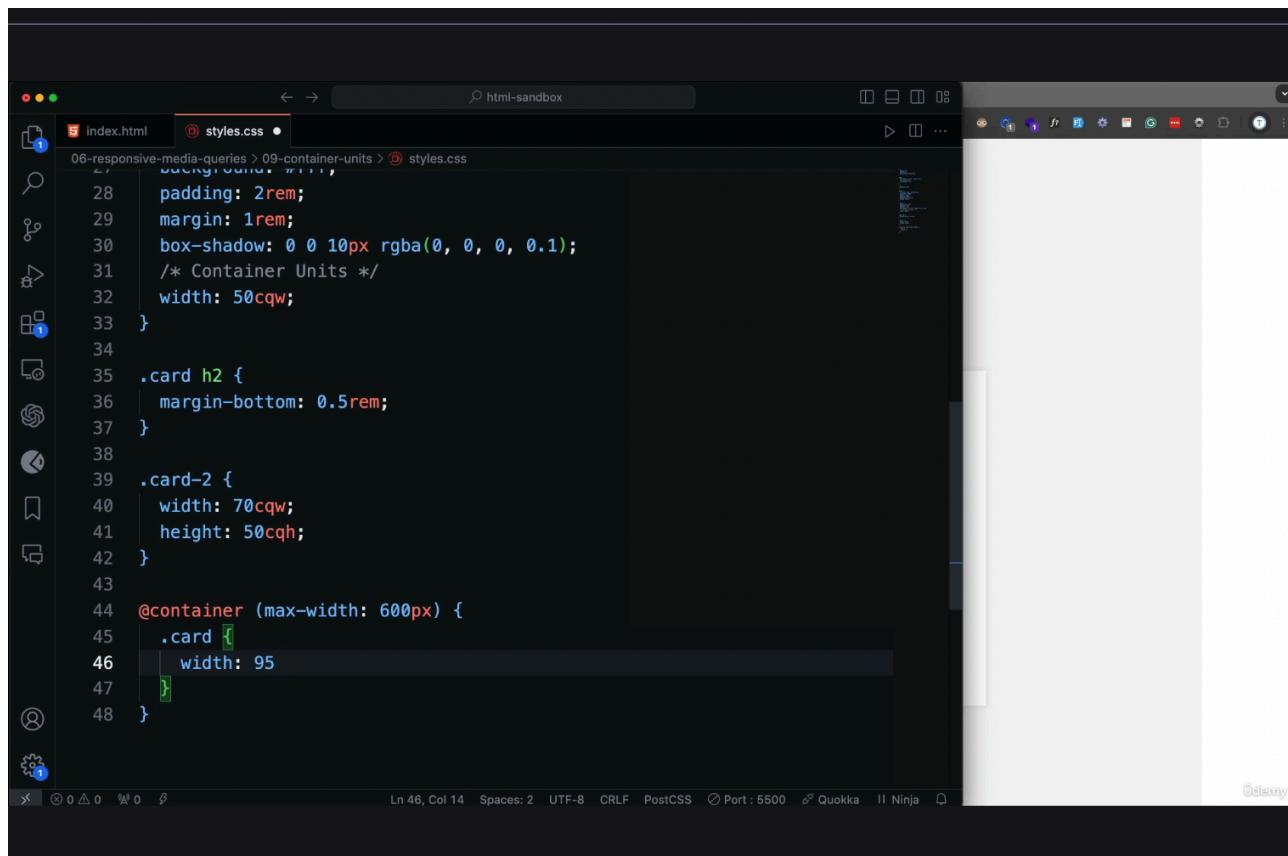
In the top right corner of the page, it says "Section 4: More HTML Element". In the bottom right corner, there is a Udemy logo with the number "53" and the word "Udemy".



```
index.html styles.css
06-responsive-media-queries > 08-container-queries > styles.css
26 }
27
28 aside {
29   container-type: inline-size;
30   container-name: aside;
31   flex: 0 0 30%;
32   background: #f4f4f4;
33   padding: 20px;
34 }
35
36 @container (max-width: 992px) {
37   main p {
38     background: lightblue;
39   }
40 }
41
42 @container aside (max-width: 300px) {
43   aside p {
44     background: lightgreen;
45   }
46 }
47
```

Ln 35, Col 1 Spaces: 2 UTF-8

cqw & cqh: (Container Query width & Container Query height)



```
index.html styles.css
06-responsive-media-queries > 09-container-units > styles.css
28   background: white;
29   padding: 2rem;
30   margin: 1rem;
31   box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
32   /* Container Units */
33   width: 50cqw;
34
35 .card h2 {
36   margin-bottom: 0.5rem;
37 }
38
39 .card-2 {
40   width: 70cqw;
41   height: 50cqh;
42 }
43
44 @container (max-width: 600px) {
45   .card {
46     width: 95
47   }
48 }
```

Ln 46, Col 14 Spaces: 2 UTF-8 CRLF PostCSS Port : 5500 Quokka Ninja

custom properties:

The screenshot shows a code editor on the left and a browser preview on the right. The code editor displays a CSS file named styles.css with the following content:

```
23 .container {  
24   display: flex;  
25   max-width: 600px;  
26   margin: 30px auto;  
27   gap: 20px;  
28 }  
29  
30 .box {  
31   flex: 1;  
32   padding: 20px;  
33   border-radius: 10px;  
34   box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
35 }  
36  
37 .box-1 {  
38   background: var(--primary-color);  
39 }  
40  
41 .box-2 {  
42   background: var(--secondary-color);  
43 }  
44  
45 .box-3 {  
46 }
```

The browser preview shows three boxes with different backgrounds: primary (blue), secondary (orange), and a third one with a light gray background. Each box contains placeholder text: "Lorem ipsum dolor".

Filters :

The screenshot shows a video player interface with the title "95. Filters" at the top. The main content area features the heading "filter Property" in large, bold letters. Below the heading is a code snippet:

```
img {  
  filter: grayscale(100%);  
}
```

Below the code, there is a list of filter effects with checkmarks:

<p>✓ grayscale</p> <p>✓ blur</p> <p>✓ brightness</p> <p>✓ contrast</p> <p>✓ drop shadow</p>	<p>✓ hue-rotate</p> <p>✓ invert</p> <p>✓ opacity</p> <p>✓ saturate</p> <p>✓ sepia</p>
--	--

At the bottom of the video player, there are navigation controls (play, stop, volume, etc.) and a progress bar showing "0:30 / 7:03".

95. Filters

```
index.html styles.css
07-various-css-features > 02-filters > styles.css
1 .container {
2   max-width: 900px;
3   margin: 70px auto;
4 }
5
6 img {
7   max-width: 100%;
8   /* -webkit-filter: grayscale(100%); */
9   -moz-filter: grayscale(100%);
10  -o-filter: grayscale(100%);
11  -ms-filter: grayscale(100%);
12  filter: grayscale(100%); */
13  /* filter: blur(5px); */
14  /* filter: brightness(200%); */
15  /* filter: contrast(50%); */
16  /* filter: drop-shadow(10px 10px 5px red); */
17  /* filter: hue-rotate(180deg); */
18  /* filter: invert(100%); */
19  /* filter: opacity(50%); */
20  /* filter: saturate(200%); */
21  filter: sepia(100%); */
22 }
23
```

HTML & CSS Sandbox - Filters



1x 6:07 / 7:03

Nesting:

99. Nesting

No Nesting

```
.header {
  background: lightcoral;
  padding: 0 20px;
}

.header h1 {
  font-size: 4rem;
}

.header p {
  font-size: 2rem;
  margin: 25px 0px 25px;
}
```

Nesting

```
.header {
  background: lightcoral;
  padding: 0 20px;
}

h1 {
  font-size: 4rem;
}

p {
  font-size: 2rem;
  margin: 25px 0px 25px;
}
```

1x 0:19 / 12:26

Code to insert the logo of famous brands :

```
<i class="fa-brands fa-facebook"></i>
```

24/05/2025 - Pseudo elements & grid properties

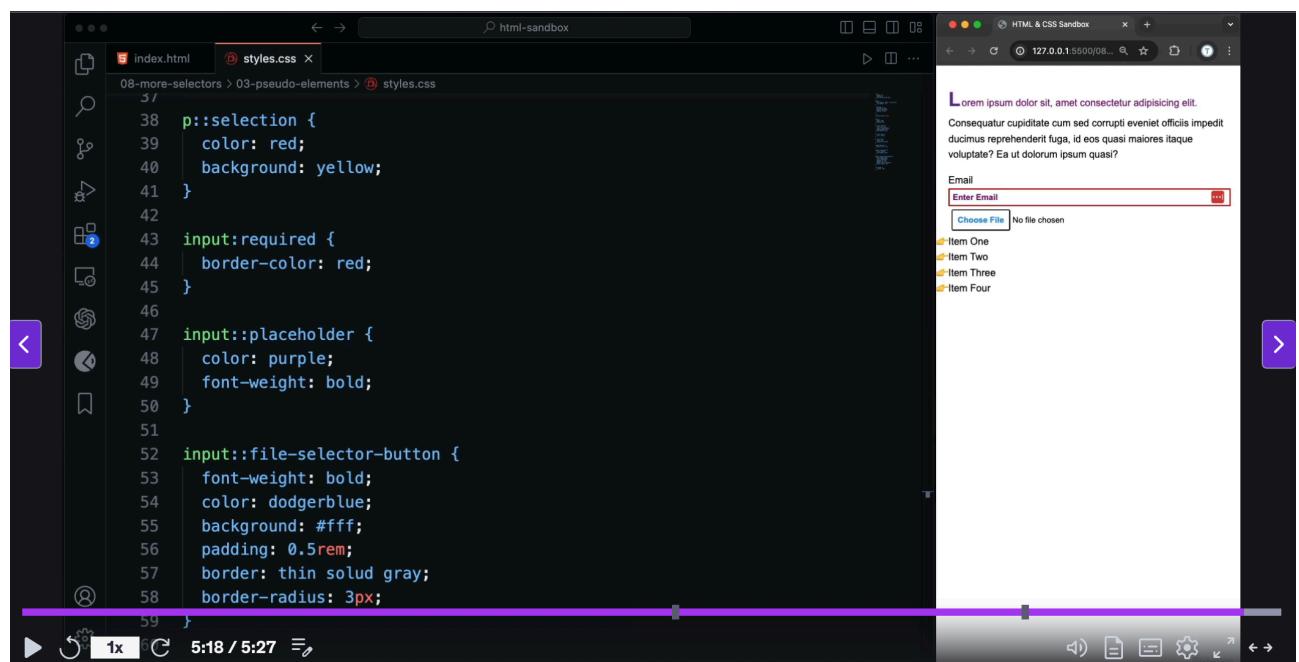
```
08-more-selectors > 03-pseudo-elements > styles.css
31   font-weight: bold;
32 }
33
34 p::first-line {
35   color: purple;
36 }
37
38 p::selection {
39   color: red;
40   background: yellow;
41 }
42
43
```

Pseudo class:

Edited screenshot

```
input:required {
  border-color: red;
}
```

Extra features :



Pseudo class - first, last and nth child:



Captured screenshot styles.css

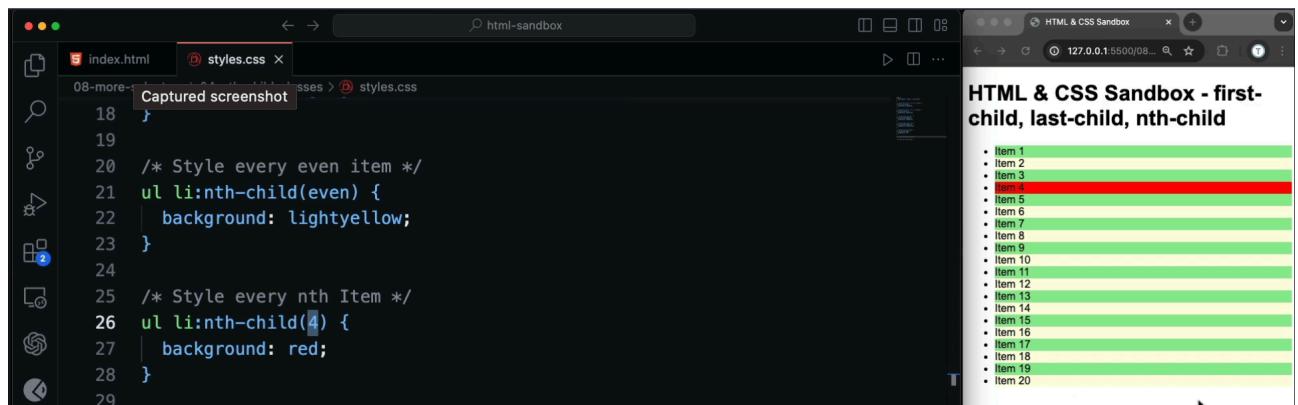
```
08-more-selectors > 04-nth-child-classes > styles.css
1 body {
2   font-family: 'Arial', sans-serif;
3 }
4
5 /* Style the first li of it's parent */
6 ul li:first-child {
7   background: lightblue;
8 }
9
10 /* Style the last li of it's parent */
11 ul li:last-child {
12   background: lightblue;
13 }
14
```

HTML & CSS Sandbox - first-child, last-child, nth-child

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5
- Item 6
- Item 7
- Item 8
- Item 9
- Item 10
- Item 11
- Item 12
- Item 13
- Item 14
- Item 15
- Item 16
- Item 17
- Item 18
- Item 19
- Item 20

```
/* Style every odd item */
ul li:nth-child(odd) {
  background: lightgreen;
}

/* Style every even item */
ul li:nth-child(even) {
  background: lightyellow;
}
```



Captured screenshot styles.css

```
08-more-selectors > 04-nth-child-classes > styles.css
18 }
19
20 /* Style every even item */
21 ul li:nth-child(even) {
22   background: lightyellow;
23 }
24
25 /* Style every nth Item */
26 ul li:nth-child(4) {
27   background: red;
28 }
29
```

HTML & CSS Sandbox - first-child, last-child, nth-child

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5
- Item 6
- Item 7
- Item 8
- Item 9
- Item 10
- Item 11
- Item 12
- Item 13
- Item 14
- Item 15
- Item 16
- Item 17
- Item 18
- Item 19
- Item 20

```
/* Style every 4th item */
ul li:nth-child(4n) {
    background: orange;
}
```

A screenshot of a browser window showing a list of 20 items. Every fourth item (Item 4, Item 8, Item 12, Item 16, Item 20) has an orange background, while the others have a light green background.

Captured screenshot

HTML & CSS Sandbox - nth-of-type

```
22 ul li:nth-of-type(6) {
23     font-weight: bold;
24     color: green;
25 }
26
27 /* Style the even items */
28 ul li:nth-of-type(even) {
29     font-weight: bold;
30     background: lightgreen;
31 }
32
33 /* Style the odd items */
34 ul li:nth-of-type(odd) {
35     font-weight: bold;
36     background: lightgreen;
37 }
38
```

A screenshot of a code editor and a browser side-by-side. The code editor shows CSS rules for styling list items based on their type (even or odd). The browser shows a list of 20 items where the even-numbered items are bolded and have a light green background, while the odd-numbered items are regular weight and have a light green background.

Pseudo by type

Captured screenshot

Header

Hello
World

```
46 }
47
48 /* Using pseudo classes with pseudo elements */
49 .container p:first-of-type::first-letter {
50     font-weight: bold;
51     font-size: 2rem;
52     color: red;
53 }
54
```

A screenshot of a code editor and a browser side-by-side. The code editor shows a rule for the first letter of the first paragraph in a container. The browser shows a header with the word "Hello" in large, bold, red font, and the word "World" below it in smaller black font.

Captured screenshot

::before & ::after

```
p::before {
    content: '';
}

p::after {
    content: '';
}
```

A diagram illustrating the ::before and ::after pseudo-elements. It shows a central yellow rectangular box labeled "[content]" with a pointer pointing to it. Above it is a blue trapezoidal box labeled ":before" and below it is a green trapezoidal box labeled ":after". Both the :before and :after boxes are labeled with arrows pointing towards the [content] box. Above the entire assembly is a larger green trapezoidal box labeled "<element>".

A screenshot of a browser developer tools interface. On the left, the styles.css file shows CSS code for a form field. On the right, the browser window displays a red background with a white input field containing "Hello World" and a "Click Here" button.

```
index.html styles.css
35 margin-right: 5px;
36 }
37
38 label.is-required::after {
39   content: '*';
40   color: red;
41   margin-left: 5px;
42   font-size: 1.2rem;
43 }
44
45 h2::after {
46   content: '';
47   position: absolute;
48   top: 0;
49   right: 0;
50   width: 300px;
51   height: 200px;
52   background: red;
53 }
54
```

A screenshot of a browser developer tools interface. On the left, the styles.css file shows CSS code for a container class. On the right, the browser window displays a red header with "Hello World" and "Goodbye World" in blue, and a green paragraph below.

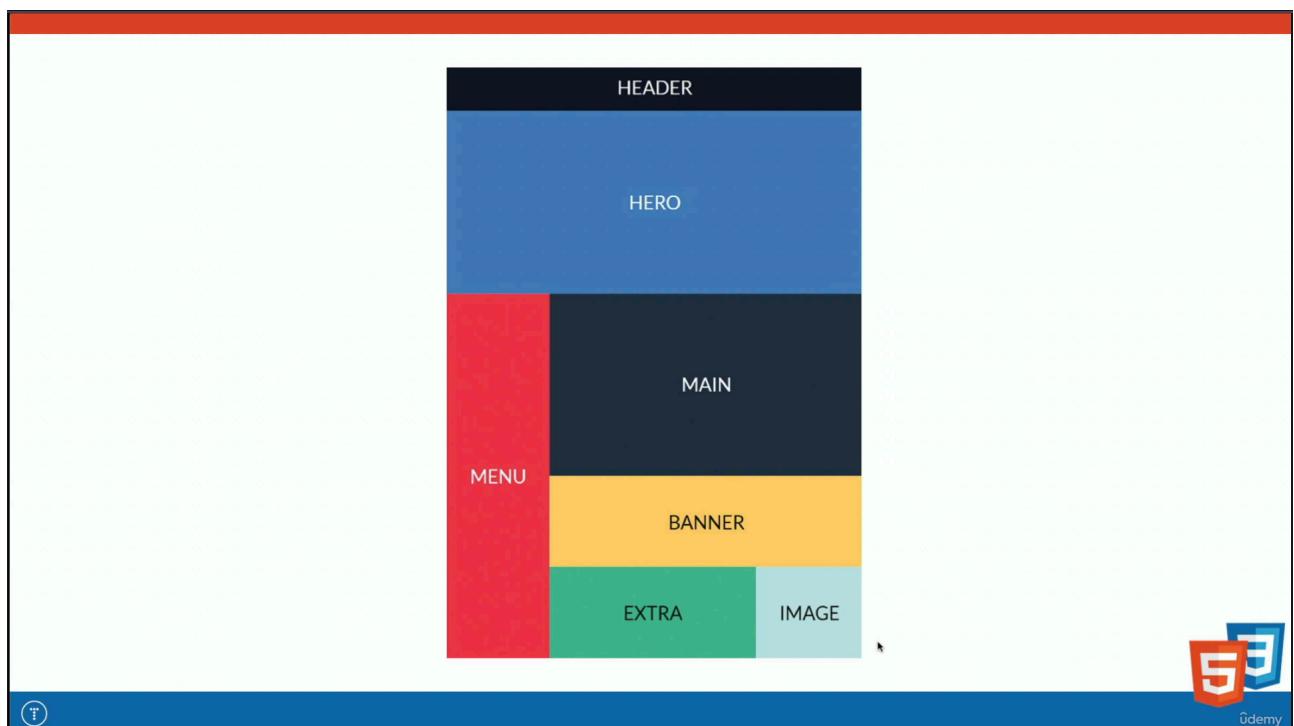
```
.container {
17 .container h2,
18 .container p {
19   color: red;
20 }
21
22 .container :is(h1, h2, p) {
23   color: blue;
24 }
25
26 a:is(:hover, :focus) {
27   background-color: pink;
28 }
```

```
a:is(:hover, :focus) {
  background-color: pink;
}

.container :where(h1, h2, p) {
  color: green;
}
```

```
22 .contai Captured screenshot p) {  
23 | color: blue;  
24 }  
25  
26 a:is(:hover, :focus) {  
27 | background-color: pink;  
28 }  
29  
30 .container :where(h1, h2, p) {  
31 | color: green;  
32 } /*  
33  
34 .container:has(h1 span) {  
35 | color: red;  
36 }  
37
```

Grid :



CSS Grid Properties

- ✓ display: grid
- ✓ grid-template-rows
- ✓ grid-template-columns
- ✓ grid-gap (gap)
- ✓ grid-column
- ✓ grid-row
- ✓ grid-area
- ✓ justify-items
- ✓ align-items
- ✓ justify-content
- ✓ align-content



The screenshot shows a developer setup with two main windows. On the left is a code editor with the file 'index.html' open, displaying:

```
09-css-grid > 01-grid-columns > styles.css
11 .item {
12   background-color: coral;
13   padding: 1rem;
14   border: 1px solid #333;
15   text-align: center;
16 }
17
18 .item-grid {
19   display: grid;
20   grid-template-columns: 600px 1fr 1fr;
21   /* grid-row-gap: 20px; */
22   /* grid-column-gap: 20px; */
23   /* grid-gap: 20px; */
24   gap: 20px;
25 }
```

On the right is a browser window titled 'HTML & CSS Sandbox - Grid Columns' showing a 3x3 grid of items:

Item 1	Item 2	Item 3
Item 4	Item 5	Item 6
Item 7	Item 8	Item 9

The browser's developer tools are visible at the bottom, with the 'Elements' tab selected and the 'Styles' panel showing the rule `:hover .cls { }`.

The screenshot shows a developer setup with two main windows. On the left is a code editor with the file 'index.html' open, displaying:

```
09-css-grid > 02-repeat-minmax-functions > styles.css
6 max-width: 1100px;
7 █ Captured screenshot ;
8 background: #f4f4f4;
9 }
10
11 .item {
12   background-color: coral;
13   padding: 1rem;
14   border: 1px solid #333;
15   text-align: center;
16 }
17
18 .item-grid {
19   display: grid;
20   grid-template-columns: repeat(3, minmax(200px, 1fr));
21 }
```

On the right is a browser window showing a 3x1 grid of items:

Item 3
Item 6
Item 9

Minmax in grid

Grids rows

The screenshot shows a code editor on the left and a browser preview on the right. The code editor displays the file `index.html` and `styles.css`. The `index.html` file contains a single line of HTML:

```
<div class="item-grid"></div>
```

The `styles.css` file contains the following CSS:

```
1 font-family: Arial, sans-serif;
2
3 }
4
5 .container {
6   max-width: 1100px;
7   margin: 30px auto;
8 }
9
10 .item-grid {
11   display: grid;
12   grid-template-rows: 200px 2fr 3fr;
13   grid-auto-rows: 100px;
14   background: #f4f4f4;
15 }
16
17 .item {
18   background-color: coral;
19   padding: 1rem;
20   border: 1px solid #333;
21   text-align: center;
22 }
```

The browser preview shows a grid container with six items labeled Item 1 through Item 6. The first item has a height of 200px. The second item spans two rows (2fr) and the third item spans three rows (3fr). Each item has a height of 100px and a white background.

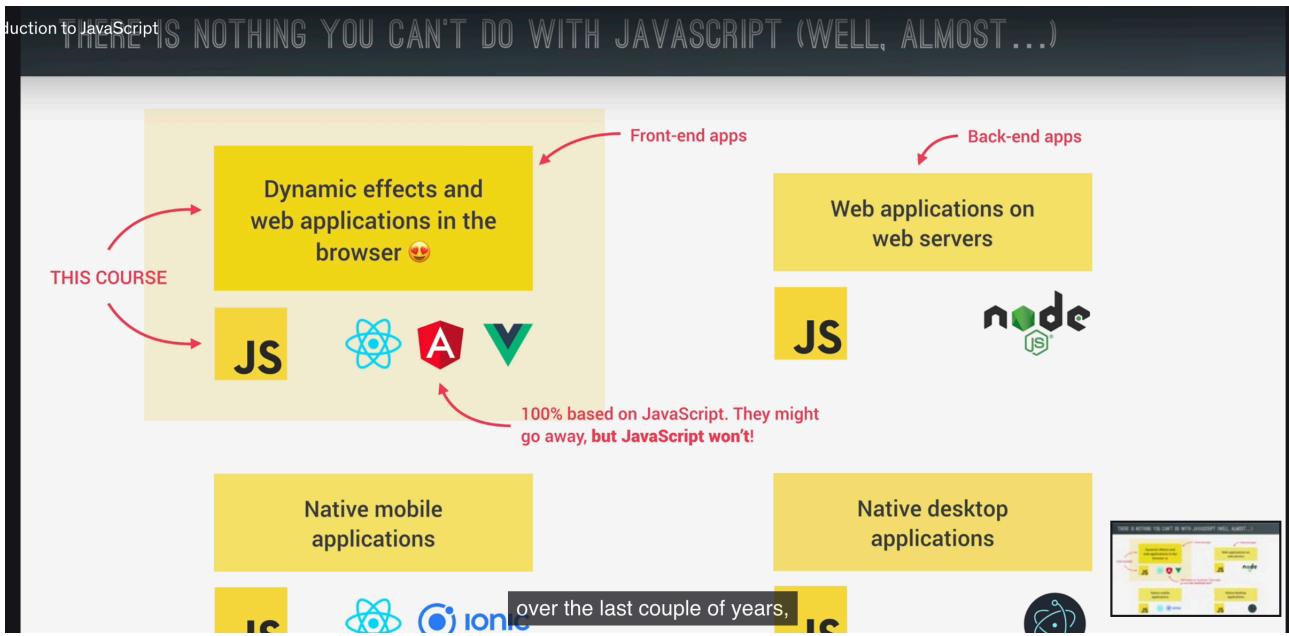
29/04/2025

JavaScript Basics

A screenshot of a browser's developer tools, specifically the Console tab. The console window has a dark background. At the top, it says "Captured screenshot" and "newtab says". Below that is a message: "Pages that you view in this window won't appear in the browser history and they won't leave other traces, like cookies, on the". The console itself shows the following JavaScript code:

```
> alert("Hello World!")
< undefined
> let js = 'amazing'
< undefined
> if (js === 'amazing') alert('JavaScript is FUN!')
< undefined
```

At the bottom of the console window, there is a message: "But for now I hope that you can see the logic".



```
<script>
  let js = 'amazing';
  if (js === 'amazing') alert('Java Script is Fun!');

</script>
```

Variables and values:

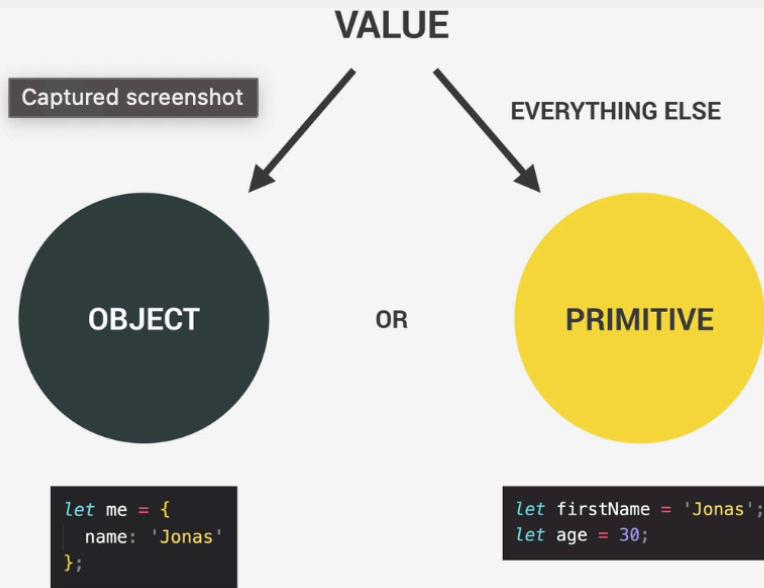
```
starter > JS script.js > ...
1   let js = 'amazing';
2
3   console.log(40 + 8 + 23 - 10);
4
5
6
7
8   let firstName = "Benish";
9   console.log(firstName);
```

Descriptive way writing and constant value examples:

```
let person = "jonas";
let PI = 3.1415;

let myFirstJob = 'Programmer';
let myCurrentJob = 'Teacher';
```

11. Data Types OBJECTS AND PRIMITIVES



THE 7 PRIMITIVE DATA TYPES

Captured screenshot

1. **Number:** Floating point numbers 👉 Used for decimals and integers `let age = 23;`
2. **String:** Sequence of characters 👉 Used for text `let firstName = 'Jonas';`
3. **Boolean:** Logical type that can only be true or false 👉 Used for taking decisions `let fullAge = true;`
4. **Undefined:** Value taken by a variable that is not yet defined ('empty value') `let children;`
5. **Null:** Also means 'empty value'
6. **Symbol (ES2015):** Value that is unique and cannot be changed *[Not useful for now]*
7. **BigInt (ES2020):** Larger integers than the Number type can hold

👉 **JavaScript has dynamic typing:** We do **not** have to manually define the data type of the value stored in a variable. Instead, data types are determined **automatically**.

Value has type, NOT variable!

Type of in js syntax:

The screenshot shows a browser developer tools console window. On the left, a code editor displays a file named 'script.js' with the following content:

```
21 let myFirstJob = "Coder";
22 let myCurrentJob = "Teacher";
23
24 let job1 = "programmer";
25 let job2 = "teacher";
26
27 console.log(myFirstJob);
28 */
29
30 let javascriptIsFun = true;
31 console.log(javascriptIsFun);
32
33 console.log(typeof true);
34 console.log(typeof javascriptIsFun);
35 console.log(typeof 23);
36 console.log(typeof 'Jonas');
37
38 ja
```

The cursor is at line 38, character 1, where 'ja' is typed. On the right, the browser window shows the title 'JavaScript Fundamentals – Part 1'. The developer tools console tab is active, showing the output of the 'typeof' operator:

- number
- string
- > 'Jonas'
- < "Jonas"
- > "Jonas"
- < "Jonas"
- > "jonas"
- ✖ Uncaught SyntaxError: Invalid or unexpected token VM694:1
- ✖ ▶ Uncaught ReferenceError: jonas is not defined VM711:1
- at <anonymous>:1:1

Var, const, let :

The screenshot shows a browser developer tools console window. On the left, a code editor displays a file named 'script.js' with the following content:

```
45
46
47 // Captured screenshot
48 */
49
50 let age = 30;
51 age = 31;
52
53 const birthYear = 1991;
54 // birthYear = 1990;
55 // const job;
56
57 var job = 'programmer';
58 job = 'teacher'
59
60 lastName = 'Schmedtmann';
61 console.log(lastName);
```

The cursor is at line 61, character 1, where 'lastName' is typed. On the right, the browser window shows the title 'Captured screenshot'. The developer tools console tab is active, showing the output of the variable assignments:

- Captured screenshot
- Uncaught ReferenceError: birthYear is not defined at <anonymous>:1:1
- Uncaught ReferenceError: job is not defined at <anonymous>:1:1
- lastName = 'Schmedtmann'
- lastName

Assignment operators and math operators:

```
// Math operators
const now = 2037;
const ageJonas = now - 1991;
const ageSarah = now - 2018;
console.log(ageJonas, ageSarah);

console.log(ageJonas * 2, ageJonas / 10, 2 ** 3);
// 2 ** 3 means 2 to the power of 3 = 2 * 2 * 2

const firstName = 'Jonas';
const lastName = 'Schmedtmann';
console.log(firstName + ' ' + lastName);

// Assignment operators
let x = 10 + 5; // 15
x += 10; // x = x + 10 = 25
x *= 4; // x = x * 4 = 100
x++; // x = x + 1
```

Comparison operator :

```
// Comparison operators
console.log(ageJonas > ageSarah); // >, <, >=, <=
console.log(ageSarah >= 18);

const isFullAge = ageSarah >= 18;
💡
console.log(now - 1991 > now - 2018);
```

Templates of normal and with back text :

```
const jonas = "I'm " + firstName + ', a ' + (year - birthYear) + ' year old ' + job + '!';
console.log(jonas);

const jonasNew = `I'm ${firstName}, a ${year - birthYear} year old ${job}!`;
console.log(jonasNew);

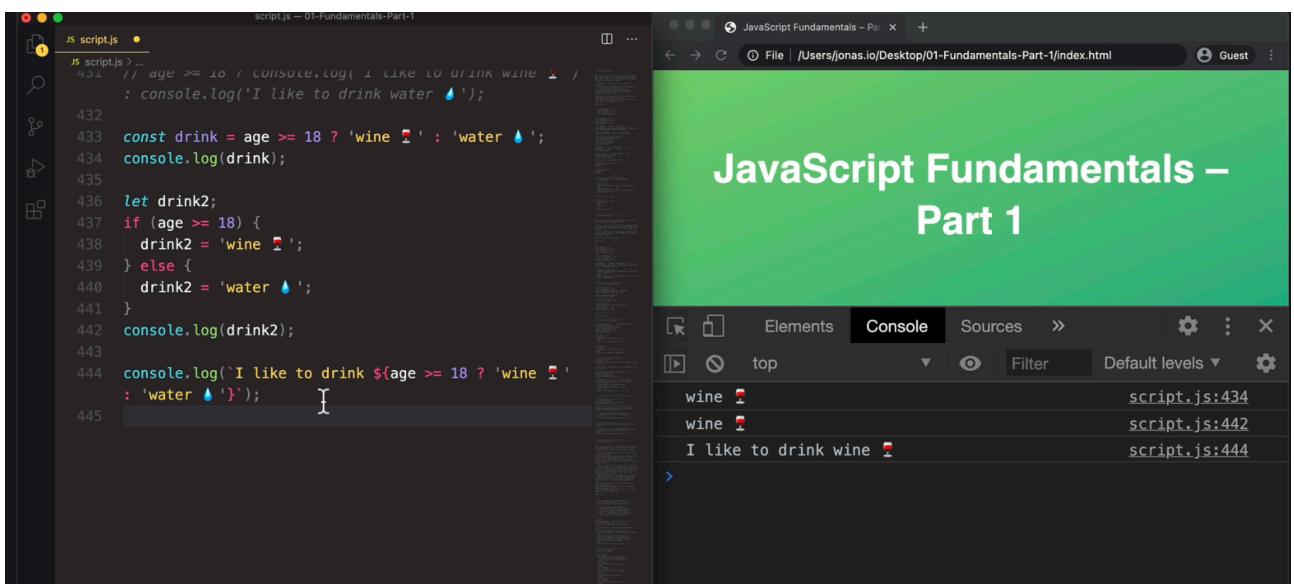
console.log(`Just a regular string...`);
```

If Else Template :

```
const age = 14;
const isOldEnough = age >= 18;

if (age >= 18) {
  console.log(`You are ${age} so you are allowed to drive`);
} else {
  console.log(`Your age of ${age} which does'nt meets the requirement on driving age of ${isOldEnough}, so you are not allowed to drive`);
}
```

Conditional Ternary operator:



The screenshot shows a code editor on the left and a browser window on the right. The code editor displays script.js with the following content:

```
431 // age >= 18 ? console.log('I like to drink wine 🍷') : console.log('I like to drink water 💧');
432
433 const drink = age >= 18 ? 'wine 🍷' : 'water 💧';
434 console.log(drink);
435
436 let drink2;
437 if (age >= 18) {
438   drink2 = 'wine 🍷';
439 } else {
440   drink2 = 'water 💧';
441 }
442 console.log(drink2);
443
444 console.log(`I like to drink ${age >= 18 ? 'wine 🍷' : 'water 💧'}`);
```

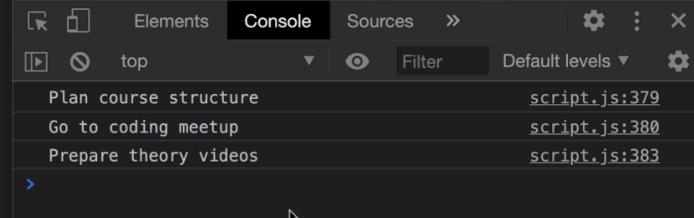
The browser window shows the title "JavaScript Fundamentals – Part 1". The developer tools' Console tab is open, displaying the following log entries:

Log	File	Line
wine 🍷	script.js	434
wine 🍷	script.js	442
I like to drink wine 🍷	script.js	444

Switch and case :

```
switch (day) {  
    case 'monday': // day === 'monday'  
        console.log('Plan course structure');  
        console.log('Go to coding meetup');  
        break;  
    case 'tuesday':  
        console.log('Prepare theory videos');  
        break;  
    case 'wednesday':  
    case 'thursday':  
        console.log('Write code examples');  
        break;  
    case 'friday':  
        console.log('Record videos');  
        break;  
    case 'saturday':  
    case 'sunday':  
        console.log('Enjoy the weekend :D');  
        break;  
    default:  
        console.log('Not a valid day!');  
}
```

JavaScript Fundamentals – Part 1



Plan course structure script.js:379
Go to coding meetup script.js:380
Prepare theory videos script.js:383

02/05/2025

Functions

```
function logger() {  
    console.log('My name is Jascar');  
}  
  
logger();  
  
  
function fruitProcessor(apples, oranges) {  
    console.log(`apples: ${apples}, oranges: ${oranges}`);  
    const juice = `Juice with ${apples} apples and ${oranges} oranges`;  
    return juice;  
}  
  
const appleJuice = fruitProcessor(5, 0);  
console.log(appleJuice);  
  
const appleOrangeJuice = fruitProcessor(2, 4);  
console.log(appleOrangeJuice);
```

FUNCTIONS REVIEW: 3 DIFFERENT FUNCTION TYPES

Function declaration

Function that can be used before it's declared

```
function calcAge(birthYear) {  
  return 2037 - birthYear;  
}
```

Function expression

Essentially a function value stored in a variable

```
const calcAge = function (birthYear) {  
  return 2037 - birthYear;  
};
```

Arrow function

Great for a quick one-line functions. Has no this keyword (more later...)

```
const calcAge = birthYear => 2037 - birthYear;
```

Captured screenshot

And finally we have arrow functions

udemy

FUNCTIONS REVIEW: ANATOMY OF A FUNCTION

Function name

Parameters: placeholders to receive input values. Like local variables of a function

Function body: block of code that we want to reuse. Processes the function's input data

return statement to output a value from the function and terminate execution

```
function calcAge(birthYear, firstName) {  
  const age = 2037 - birthYear;  
  console.log(`${firstName} is ${age} years old`);  
  return age;  
}
```

Variable to save returned value (function output)

Arguments: actual values of function parameters, to input data

Calling, running or invoking the function, using ()

```
const age = calcAge(1991, 'Jonas')
```

udemy

04/05/2025

Arrays and objects

The screenshot shows a code editor window titled "script.js — 02-Fundamentals-Part-2". The code in the editor is as follows:

```
JS script.js > [e] years
164  checkWinner(scoreDolphins, scoreKoalas);
165  */
166
167  const friend1 = 'Michael';
168  const friend2 = 'Steven';
169  const friend3 = 'Peter';
170
171  const friends = ['Michael', 'Steven', 'Peter'];
172  console.log(friends);
173
174  const years = new Array(1991, 1984, 2008, 2020);
175
176  console.log(friends[0]);
177  console.log(friends[2]);
178
179  console.log(friends.length);
180  console.log(friends[friends.length - 1]);
181
182  friends[2] = 'Jay';
183  console.log(friends);
184 // friends = ['Bob', 'Alice']
185
186  const firstName = 'Jonas';
187  const jonas = [firstName, 'Schmedtmann', 2037 - 1991,
188    'teacher', friends];
189  console.log(jonas);
190  console.log(jonas.length);
```

The screenshot shows a browser developer tools console tab labeled "Console". The output in the console is:

```
Part 2
207
208  const friends = ['Michael', 'Steven', 'Peter'];
209  const newLength = friends.push('Jay');
210  console.log(friends);
211  console.log(newLength);
212
213  friends.unshift('John');
214  console.log(friends);
```

The console output shows the following results:

- Line 210: `[("Michael", "Steven", "Peter", "Jay")]` (script.js:210)
- Line 211: `4` (script.js:211)
- Line 214: `[("John", "Michael", "Steven", "Peter", "Jay")]` (script.js:214)

A screenshot of a browser's developer tools console tab. The code being run is:

```

216 console.log(friends);
217 // Remove elements
218 friends.pop(); // Last
219 const popped = friends.pop();
220 console.log(popped);
221 console.log(friends);
222
223

```

The console output shows three log statements:

- Line 216: (4) ["Michael", "Steven", "Peter", "Jay"] script.js:212
- Line 218: 4 script.js:213
- Line 219: (5) ["John", "Michael", "Steven", "Peter", "Jay"] script.js:216
- Line 220: (3) ["John", "Michael", "Steven"] script.js:221

Remove elements in the array:

A screenshot of a browser's developer tools console tab. The code being run is:

```

is (Methods) —
  JS script.js > ...
    216 // Remove elements
    217 friends.pop(); // Last
    218 const popped = friends.pop();
    219 console.log(popped);
    220 console.log(friends);
    221
    222 friends.shift(); // First
    223 console.log(friends);
    224
    225 console.log(friends.indexOf('Steven'));
    226 console.log(friends.indexOf('Bob'));
    227
    228 console.log(friends.includes('Steven'));
    229 console.log(friends.includes('Bob'));
    230
    231
  
```

The console output shows several log statements and a search result:

- Line 217: 4 script.js:213
- Line 219: (5) ["John", "Michael", "Steven", "Peter", "Jay"] script.js:216
- Line 220: Peter script.js:221
- Line 222: (3) ["John", "Michael", "Steven"] script.js:222
- Line 223: (2) ["Michael", "Steven"] script.js:225
- Line 224: 1 script.js:227
- Line 225: -1 script.js:228
- Line 226: true script.js:230
- Line 227: false script.js:231

In the background, a search result for "because well, there is no element called Bob." is visible.

A screenshot of a browser's developer tools console tab. The code being run is:

```

JS script.js — 02-Fundamentals-Part-2
  JS script.js > ...
    216 // Remove elements
    217 friends.pop(); // Last
    218 const popped = friends.pop();
    219 console.log(popped);
    220 console.log(friends);
    221
    222 friends.shift(); // First
    223 console.log(friends);
    224
    225 console.log(friends.indexOf('Steven'));
    226 console.log(friends.indexOf('Bob'));
    227
    228 friends.push(23);
    229 console.log(friends.includes('Steven'));
    230 console.log(friends.includes('Bob'));
    231 console.log(friends.includes(23));
    232
    233 if (friends.includes('Steven')) {
    234   console.log('You have a friend called Steven');
    235 }
    236
    237
    238
    239
  
```

The console output shows several log statements and an if statement execution:

- Line 217: 4 script.js:213
- Line 219: (5) ["John", "Michael", "Steven", "Peter", "Jay"] script.js:216
- Line 220: Peter script.js:221
- Line 222: (3) ["John", "Michael", "Steven"] script.js:222
- Line 223: (2) ["Michael", "Steven"] script.js:225
- Line 224: 1 script.js:227
- Line 225: -1 script.js:228
- Line 226: true script.js:231
- Line 227: false script.js:232
- Line 228: true script.js:233
- Line 234: You have a friend called Steven script.js:236

In the background, a search result for "But for now, I just wanted to show you the most basic ones." is visible.

The screenshot shows a Mac desktop with two windows. The left window is a code editor displaying a file named 'script.js' with the following content:

```

292
293 const jonas = {
294   firstName: 'Jonas',
295   lastName: 'Schmedtmann',
296   age: 2037 - 1991,
297   job: 'teacher',
298   friends: ['Michael', 'Peter', 'Steven']
299 };
300 console.log(jonas);
301
302 console.log(jonas.lastName);
303 console.log(jonas['lastName']);
304
305 const nameKey = 'Name';
306 console.log(jonas['first' + nameKey]);
307 console.log(jonas['last' + nameKey]);

```

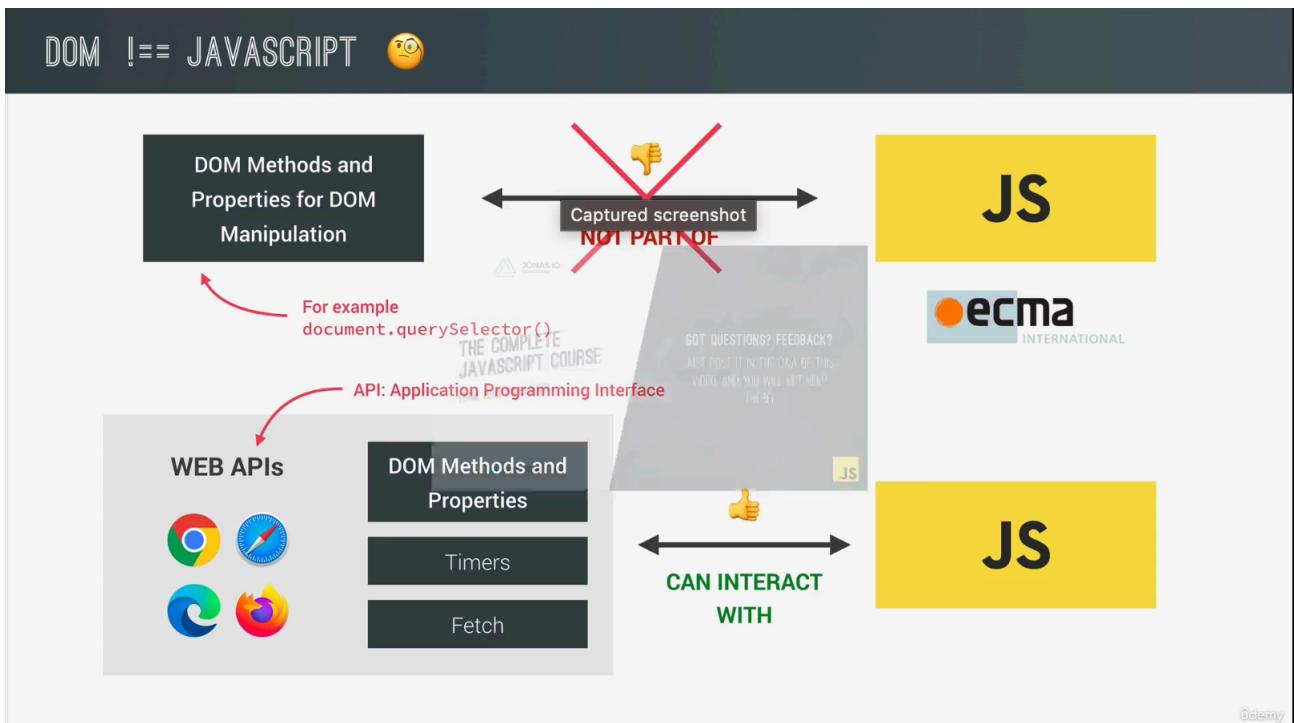
The right window is a web browser with the title 'JavaScript Fundamentals – Part 2'. The browser's developer tools are open, specifically the 'Console' tab, which displays the output of the console.log statements:

```

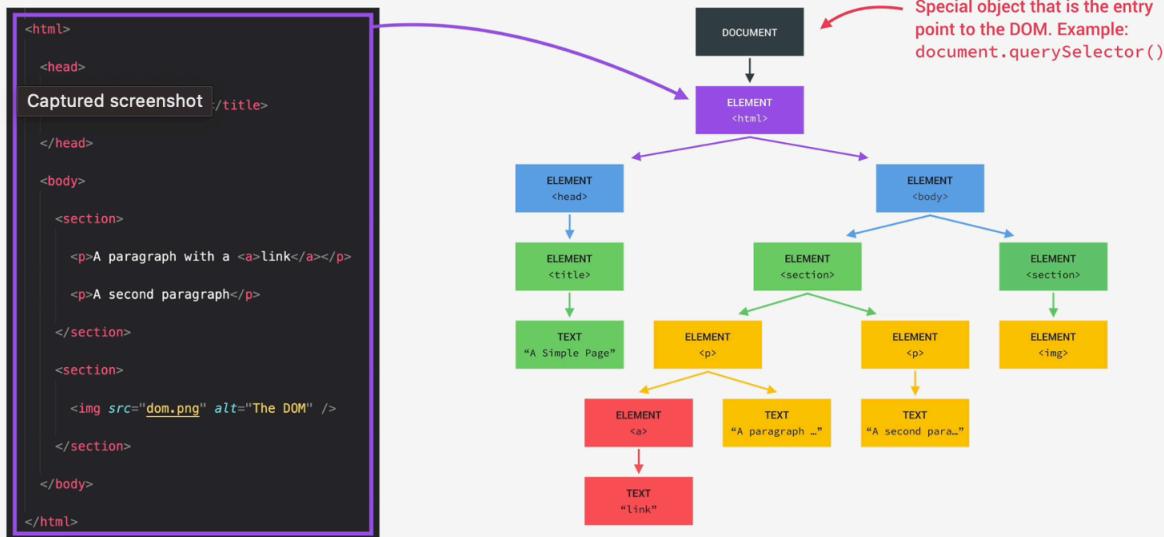
script.js:300 {firstName: "Jonas", lastName: "Schmedtmann", age: 46, job: "teacher", friends: Array(3)}
script.js:302 Schmedtmann
script.js:303 Schmedtmann
script.js:306 Jonas
script.js:307 Schmedtmann

```

DOM Elements :



THE DOM TREE STRUCTURE



Odemy

```
'use strict';

console.log(document.querySelector('.message').textContent);

document.querySelector('.message').textContent = 'Correct Number!';

document.querySelector('.number').textContent = 13;
document.querySelector('.score').textContent = 20;
```

12/05/2025

Different types of functions :

A screenshot of a browser developer tools console. The title bar says "How JavaScript Works Behind the Scenes" and the address bar shows "127.0.0.1:64116". The console tab is active, showing the output of the following code:

```
script.js:46
Live reload enabled.
(index):55
```

The status bar at the bottom indicates "Captured screenshot" and "that this is how the arrow function works."

Short-circuiting using OR logic:

A screenshot of a browser developer tools console. The title bar says "Data Structures and Modern Operators" and the address bar shows "127.0.0.1:57034". The console tab is active, showing the output of the following code:

```
script.js:52
Captured screenshot
script.js:53
script.js:54
script.js:55
script.js:56
script.js:57
script.js:61
script.js:64
(index):55
```

The status bar at the bottom indicates "Captured screenshot" and "than having to deal with this turnary operator".

OR operator:

```
1 const rest1 = {
2   name: 'Capri',
3   numGuests: 20,
4 };
5
6 const rest2 = {
7   name: 'La Piazza',
8   owner: 'Giovanni Rossi',
9 };
10
11 rest1.numGuests = rest1.numGuests || 10;
12 rest2.numGuests = rest2.numGuests || 10;
13
14 rest1.numGuests ||= 10; // rest1.numGuests ||= 10;
15
16
17 console.log(rest1);
18 console.log(rest2);
```

Nullish operator:

```
const rest1 = {
  name: 'Capri',
  // numGuests: 20,
  numGuests: 0,
};

const rest2 = {
  name: 'La Piazza',
  owner: 'Giovanni Rossi',
};

// OR assignment operator
// rest1.numGuests = rest1.numGuests || 10;
// rest2.numGuests = rest2.numGuests || 10;
// rest1.numGuests ||= 10;
// rest2.numGuests ||= 10;

// nullish assignment operator (null or undefined)
rest1.numGuests ??= 10;
rest2.numGuests ??= 10;

console.log(rest1);
console.log(rest2);
```

```
// AND assignment operator  
// rest1.owner = rest1.owner && '<ANONYMOUS>';  
// rest2.owner = rest2.owner && '<ANONYMOUS>';  
rest1.owner &&= '<ANONYMOUS>';  
rest2.owner &&= '<ANONYMOUS>';  
  
console.log(rest1);  
console.log(rest2);
```

Working with strings :

```
Captured screenshot = 'TAP Air Portugal';  
const plane = 'A320';  
  
console.log(plane[0]);  
console.log(plane[1]);  
console.log(plane[2]);  
console.log('B737'[0]);  
  
console.log(airline.length);  
console.log('B737'.length);  
  
console.log(airline.indexOf('r'));  
console.log(airline.lastIndexOf('r'));  
console.log(airline.indexOf('portugal'));  
  
console.log(airline.slice(4));  
console.log(airline.slice(4, 7));
```

13/05/2025 & 14/05/2025 Functions and API calls

Call functions :

```
// call methods
const airInida = {
  airline: 'AirIndia',
  iataCode: 'AI',
  bookings: [],
  book(flightNum, name) {
    console.log(` ${name} booked a seat on ${this.airline} flight ${this.iataCode}${flightNum}`);
  };
  this.bookings.push({ flight: `${this.iataCode}${flightNum}`, name });
},
};

airInida.book(238, 'Jascar Benish');
airInida.book(240, 'Shiny Fedora');

const indiGo = {
  airline: 'IndiGo',
  iataCode: 'IG',
  bookings: [],
};

const book = airInida.book;
//book(23, 'Jascar Benish');
book.call(airInida, 23, 'Jascar Benish')
console.log(airInida);

book.call(indiGo, '240', 'Shiny ');
console.log(indiGo);

const spiceJet = {
  airline: 'SpiceJet',
  iataCode: 'SJ',
  bookings: [],
};

book.call(spiceJet, 503, 'Ponsekar');
console.log(spiceJet);

//apply method
const flightData = [583, 'Rohan Joel'];
book.apply(airInida, ...flightData);
console.log(airInida);

book.call(spiceJet, ...flightData);

//Bind method
```

