HTML + CSS

Instructor: Adnan Sami, SE at Talrock (Paris, France)
WhatsApp: +880 16 1744 2007

1-1 [HTML]: VSCode | HTML

Preference -> Settings -> Editor: font Size

Preference -> Settings -> Editor: Mouse Wheel Zoom

.pdf, .doc, .jpg, .mp4, .html, .css

These are file formats or file extensions. It tells my computer what type of file he is going to deal with. A computer treat different files in a different way. It could be a text file, a image file, a video file, a web file (which has a file type for instance .html, .css, .asp, .php etc.)

HTML -> Hyper Text Markup (Symbols) Language

Hyper Text (concept or an idea) -> not a plain text. usually refers to hyperlinks. it’s a feature or a nature of HTML that enables a user to navigate between different pages via hyperlinks. In short, to link one document with another.

Markup -> Symbols or Tags

Language -> Language of the browser

your-name.html

1. hey I am adnan sami. I am learning to build websites.
2. I am a student.
3. I am studying computer science.
4. My hobby is learning new stuffs. blah blah ..
5.
6.
7.
8. I am feeling sleepy.

How markup or tags looks?

Tags tell the web browser how to display text, images, links, and other html elements.

<> </>

<tagname>content</tagname>

<h1>this is a heading</h1>

** this is a paragraph. this is a sentence. this is another sentence. **

How markup or tags looks?

<> </>

<tagname>content</tagname>

<h1>this is a heading</h1>

** this is a paragraph. this is a sentence. this is another sentence. **

How markup or tags looks?

<> </>

Bold: content

Strong (Important): content

Italic: content

Emphasize (Priority): content

How markup or tags looks?

<> </>

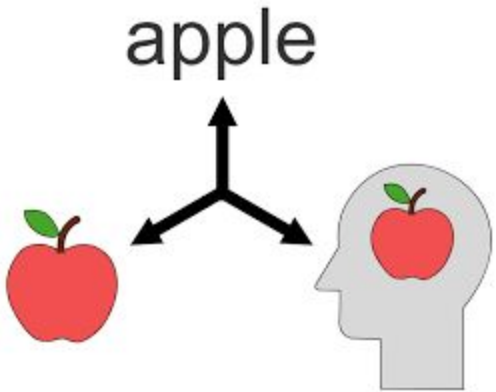
Small: **<small>content</small>**

Semantic Tags (Meaningful Tags. Meaningful to what?)

- to the developer
- to the user
- to the browser
- to accessibility tools
- to the search engines (specifically search engine crawlers)

Semantic HTML refers to the use of HTML tags that describes the meaning—or semantics—of the content contained within them.

1. Code Readability
2. Accessibility
3. SEO


apple

https://epaper.prothomalo.com/

Inline (in a line) Element: strong, em, i, small, span etc.

Block Element (creates a block or a new line): p, h1, div, etc.

How markup or tags looks?

```
<>  </>
<h1>this is a heading</h1>
<h2>this is a heading</h2>
<h3>this is a heading</h3>
<h4>this is a heading</h4>
<h5>this is a heading</h5>
<h6>this is a heading</h6>
```

list

- item 1
- item 2
- item 3
- . . .

List Item: **
item 1
**

unordered list by default.

**Unordered List: <ul> item1 **

```
<ul>
      <li>item 1</li>
      <li>item 2</li>
      <li>item 3</li>
</ul>
```

ul tag adds a margin (indent) to the unordered list. here ul is a container tag. another container tag is a div tag.

**Ordered List: <ol> item1 **

```
<ol>
    <li>item 1</li>
    <li>item 2</li>
    <li>item 3</li>
</ol>
```

ol tag adds a margin (indent) to the unordered list. here ol is a container tag. another container tag is a div tag.

Ordered List: `<ol type="1"> <li>item1</li> </ol>`

type="1" | type="a" | type="A" | type="i" | type="I"

Unordered List: `<ul type="disc"> <li>item1</li> </ul>`

type="disc" | type="circle" | type="square" | type="none"

Element Nesting

**Division: <h2>this is a heading</h2> **

```
<div>
        <h2>this is a heading</h2>
        <p>this is a paragraph. this is another text.</p>
        <ul>
                <li>fruits</li>
                <li>vegetables</li>
                <ul>
                        <li>pumpkin</li>
                        <li>tomatoes</li>
                </ul>
        </ul>
</div>
```

Empty Tag/ Void Tag/ Self Closing Tag:

**Break:
**

    <h2>this is a heading</h2>
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Animi eligendi dolorem unde consectetur
 minima sed reiciendis earum repudiandae quam, adipisci blanditiis impedit deserunt reprehenderit provident nostrum dolores praesentium recusandae qui!.

 is an empty tag.

Button: <button>this is a button</button>

Hyper Links

Hyper Text (concept or an idea) -> not a plain text. usually refers to hyperlinks. it’s a feature or a nature of HTML that enables a user to navigate between different pages via hyperlinks.

and how can we achieve this concept? by connecting pages together with hyperlinks both internally (same website) and externally. for example the wikipedia website. at old age, there were only simple html or web pages connected together.

Anchor: **this is a link**

this is a link

download image

here we need to specify where to go through href (hypertext reference) attribute. replace # with a reference (.html file path or external link).

attribute-name="value"

href="about.html" (relative path)

href="e:/pages/about.html" (absolute path)

href="https://www.google.com"

Image Tag: <img>

Image: <img src="#" alt="this is an image" width="100" height="100">

here we need to tell the image path through src (source) attribute. replace # with an image file path or external image url. alt, width and height attributes are optional.

<img> is an empty tag. it has no ending tag or closing tag.

HTML Form

to collect user data or information html form is used. for example you filled a form on admission at CSL Training.

Input: <input type="text">

type="password" (for password field)

type="checkbox"

type="date"

type="email"

type="file"

type="submit" | value="Submit"

Select: <select type="text"> <option value="text"></option> </select>

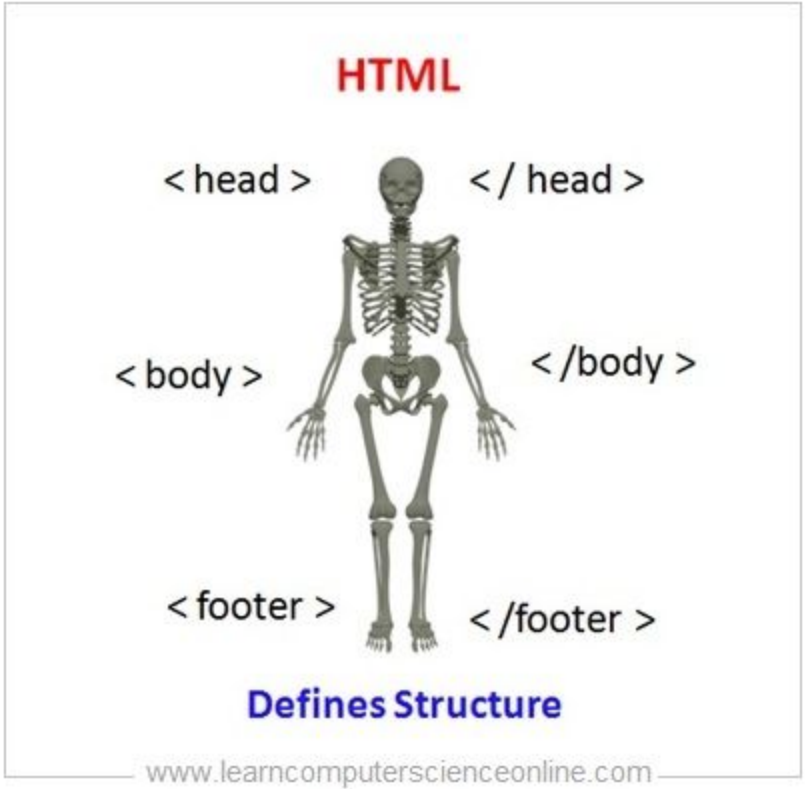
```
<select>
      <option value="option1">Option 1</option>
      <option value="option2">Option 2</option>
      <option value="option3">Option 3</option>
</select>
```

```
<form>
    <label for="name">Name</label>
    <input type="text" placeholder="your name" id="name">

    <label for="pass">Password</label>
    <input type="password" placeholder="your password" id="pass">


    <input type="submit" value="submit">
</form>
```

Task

https://www.w3schools.com/tags/tag_label.asp

What Is HTML ?


HTML
< head >
</ head >
< body >
</body >
< footer >
</footer >
Defines Structure
www.learncomputerscienceonline.com

we can separate the entire html document into three parts:

```
<html>

    <head></head> <!-- meta data: information about the website. for seo -->

    <body>
        <header>
            <nav></nav>
        </header>
        <main></main>
        <footer></footer>

    </body>

</html>
```

```
<html>

    <head>
        <title>this is my website</title>
        <!-- 16x16 pixels -->
        <link rel="icon" type="image/x-icon" href="favicon.ico">
    </head>

    <body>

        <footer></footer>

    </body>

</html>
```

<base>

```
<head>
	<base href="https://syntackle.live" />
</head>

<body>
	<a href="/contact/">Contact</a>
</body>
```

<datalist>

```
<label for="browser">Choose your browser from the list:</label>

<input list="browsers" name="browser" id="browser">

<datalist id="browsers">
      <option value="Edge">
      <option value="Firefox">
      <option value="Chrome">
      <option value="Opera">
      <option value="Safari">
</datalist>
```

<progress>

```
<progress value="0">0%</progress> <!-- default range: 0.0 to 1.0 -->

<progress value="0" max="100">0%</progress>

<progress>Indeterminate</progress>
```

<dialog>

```
<dialog open>
	<p>This is a one time message. Click the button to close it.</p>

	<form method="dialog">
		<button>Ok</button>
	</form>
</dialog>
```

[advance usage]
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/dialog#examples

<dialog> (advance usage)

```
<dialog>
    <p>This is a dialog box.</p>
    <button id="close">Close</button>
</dialog>
<button id="open">Open Dialog Box</button>

<script>
    document.querySelector("#open").addEventListener("click", () => {
        document.querySelector("dialog").showModal()
    })

    document.querySelector("#close").addEventListener("click", () => {
        document.querySelector("dialog").close()
    })
</script>
```

<optgroup>

```
<select>
	<optgroup label="1PM">
		<option value="titanic">Twister</option>
		<option value="nd">Napoleon Dynamite</option>
		<option value="wab">What About Bob?</option>
	</optgroup>
	<optgroup label="2PM">
		<option value="bkrw">Be Kind Rewind</option>
		<option value="stf">Stranger Than Fiction</option>
	</optgroup>
</select>
```

```
<form>
    <fieldset>
    <legend>Gender</legend>
        Male    <input name="gender" type="radio" >
        Female  <input name="gender" type="radio" >
    </fieldset>
</form>
```

The <legend> tag is used to define a caption for the <fieldset> element.

```
<details>
      <summary>Click to read more about me</summary>

      <p>I love sharing how to use the latest technology tools … </p>
</details>
```

Practice Task

- ❏ Create a basic HTML page structure with the doctype, html, head, meta, title and body tags
- ❏ The title will be "My Resume"
- ❏ Add comments to the code to explain each section. For example, "this section is about my educational qualification"

CSS

A typical webpage has 3 layers:

HTML

Content Layer

CSS

Presentation Layer

JavaScript

Behavior Layer

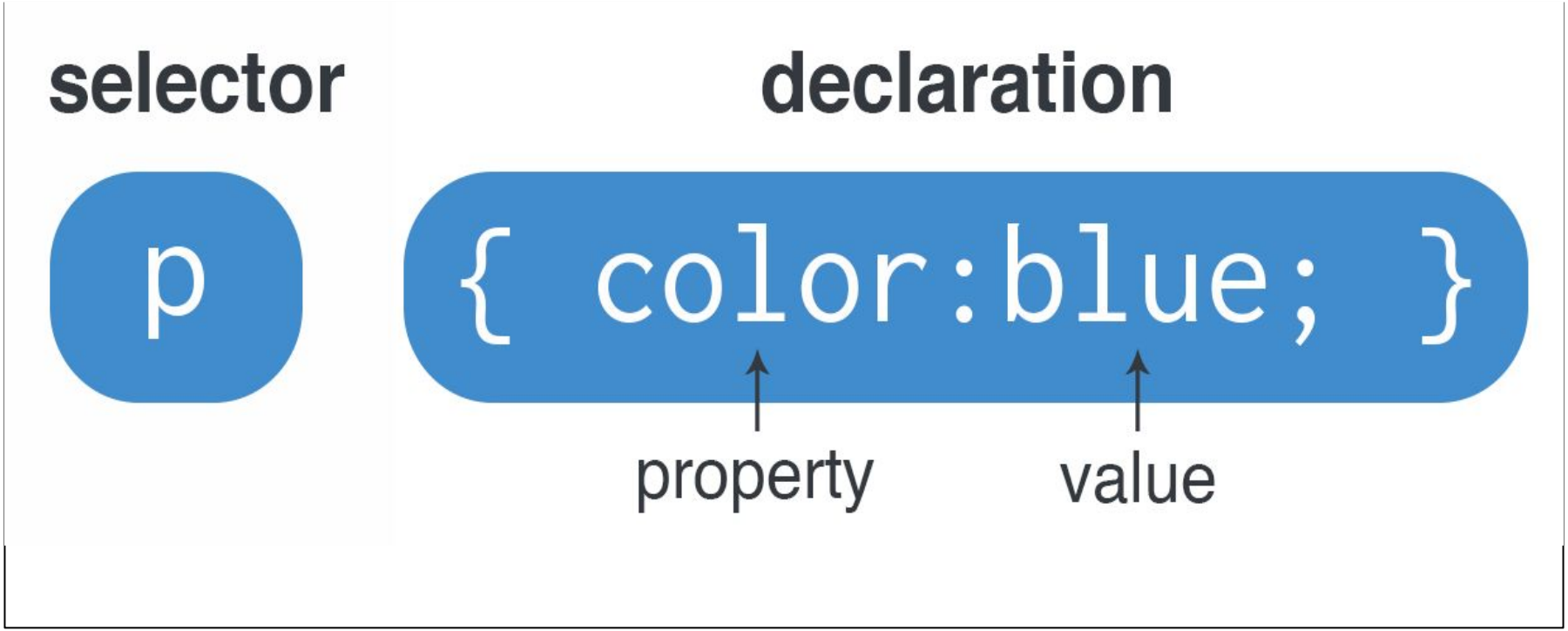
What can CSS do?

- Gives color to the webpage
- Changes position
- Can do a little bit of animation and many more …

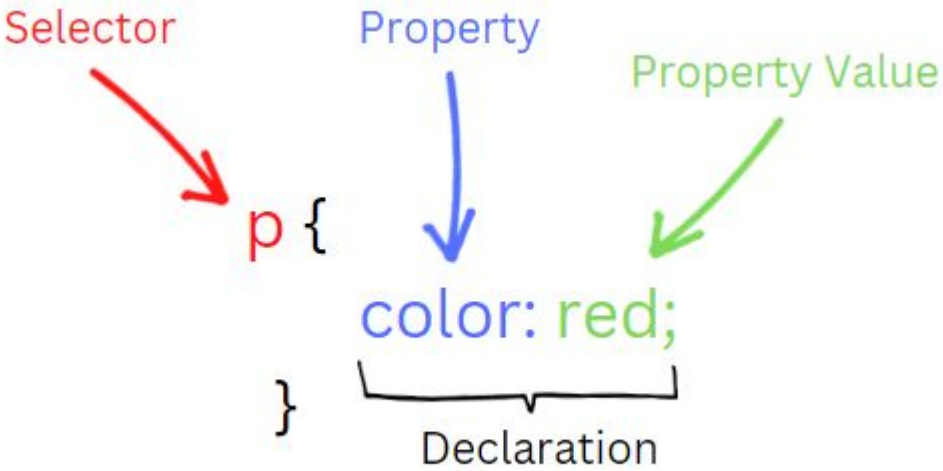
What actually is cascading?

```
h1 {
	color: red;
}

h1 {
	color: green;
}

h1 {
	color: black; // the last one applies
}
<h1>this is a heading</h1>
```

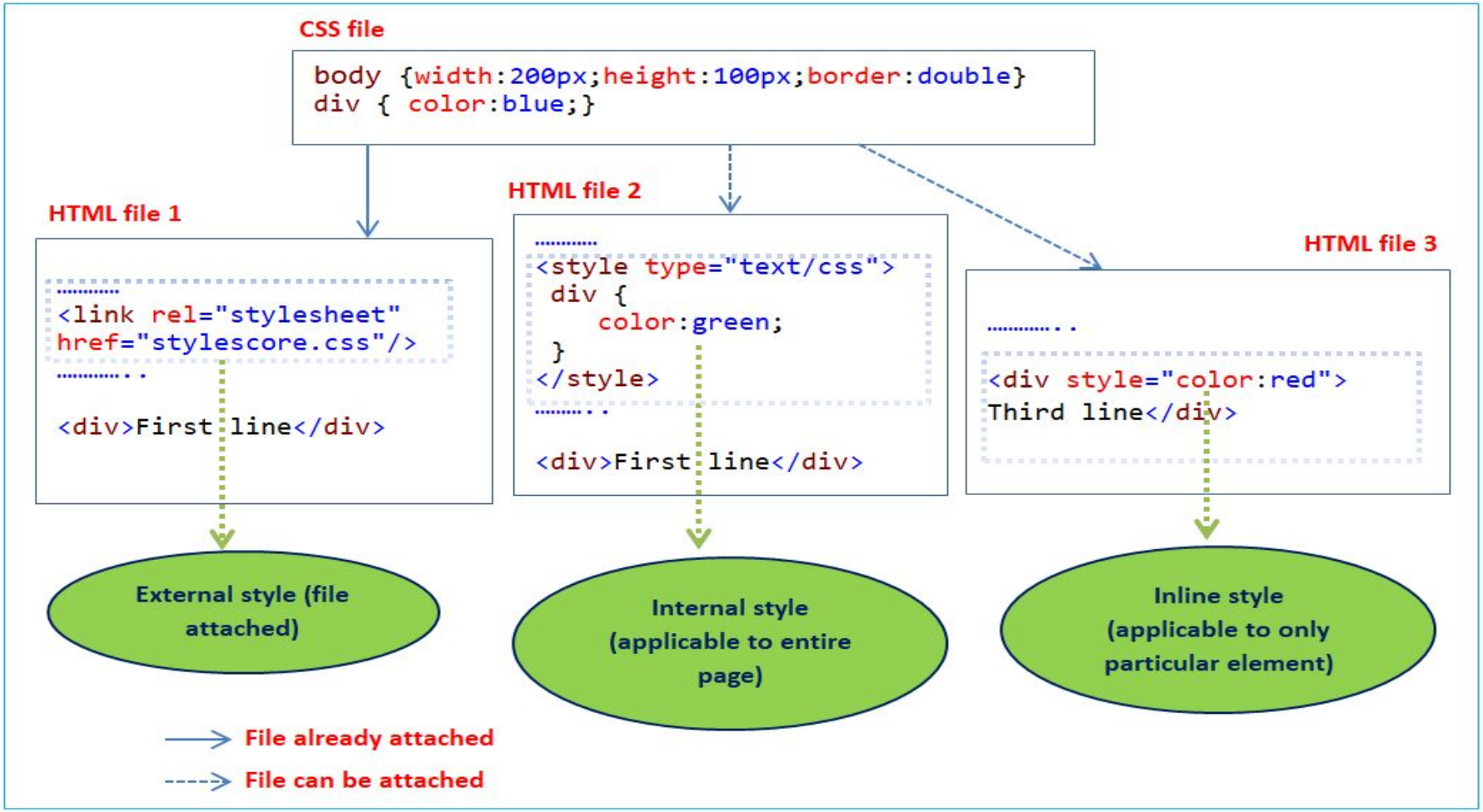
CSS Syntax



CSS Syntax



Types of CSS


CSS file
body {width:200px;height:100px;border:double}
div { color:blue;}
HTML file 1
<link rel="stylesheet"
href="stylescore.css"/>
First line
HTML file 2
<style type="text/css">
div {
color:green;
}
</style>
First line
HTML file 3

Third line
External style (file attached)
Internal style (applicable to entire page)
Inline style (applicable to only particular element)
File already attached
File can be attached

HTML attributes

HTML attributes are special words used inside the opening tag to control the element's behaviour. HTML attributes are a modifier of a HTML element type. An attribute either modifies the default functionality of an element type or provides functionality to certain element types unable to function correctly without them.

Inline CSS

```
<p style="color: red;">this is a paragraph</p>
```

cons:

- not resulable
- limited functionality

Internal CSS

```
<head>
        <style>
                p {
                        color: red;
                }
        </style>
</head>
```

- COLOR NAMES
- RGB
- HEX
- HSL
- RGBA
- HSLA

CSS Units

- em
- rem
- vw
- vh
- %

```
<!-- fixed unit: px -->
<p style="width: 50px">this is a paragraph</p>

<!-- relative unit: % -->
<p style="width: 50%">this is a paragraph</p>
```

CSS Selectors: Tag

Tag Selector:

```
<head>
	<style>
		h1 {
			color: red;
			background-color: green;
		}
	</style>
</head>
```

CSS Selectors: Id (Specific One)

Id Selector:

```
<head>
	<style>
		#foo {
			color: red;
			background-color: green;
		}
	</style>
</head>
```

CSS Selectors: Class

Id Selector:

```
<head>
	<style>
		.foo {
			color: red;
			background-color: green;
		}
	</style>
</head>
```

CSS Selectors: Grouping

```
<head>
	<style>
		h1, p {
			color: red;
			background-color: green;
		}
	</style>
</head>
```

CSS Selectors: Attribute

```
<head>
	<style>
		input {
			font-size: 32px;
		}
		input[type="password"] {
			color: green;
		}
	</style>
</head>

<body>
	<form>
		<input type="text">
		<input type="password">
	</form>
</body>
```

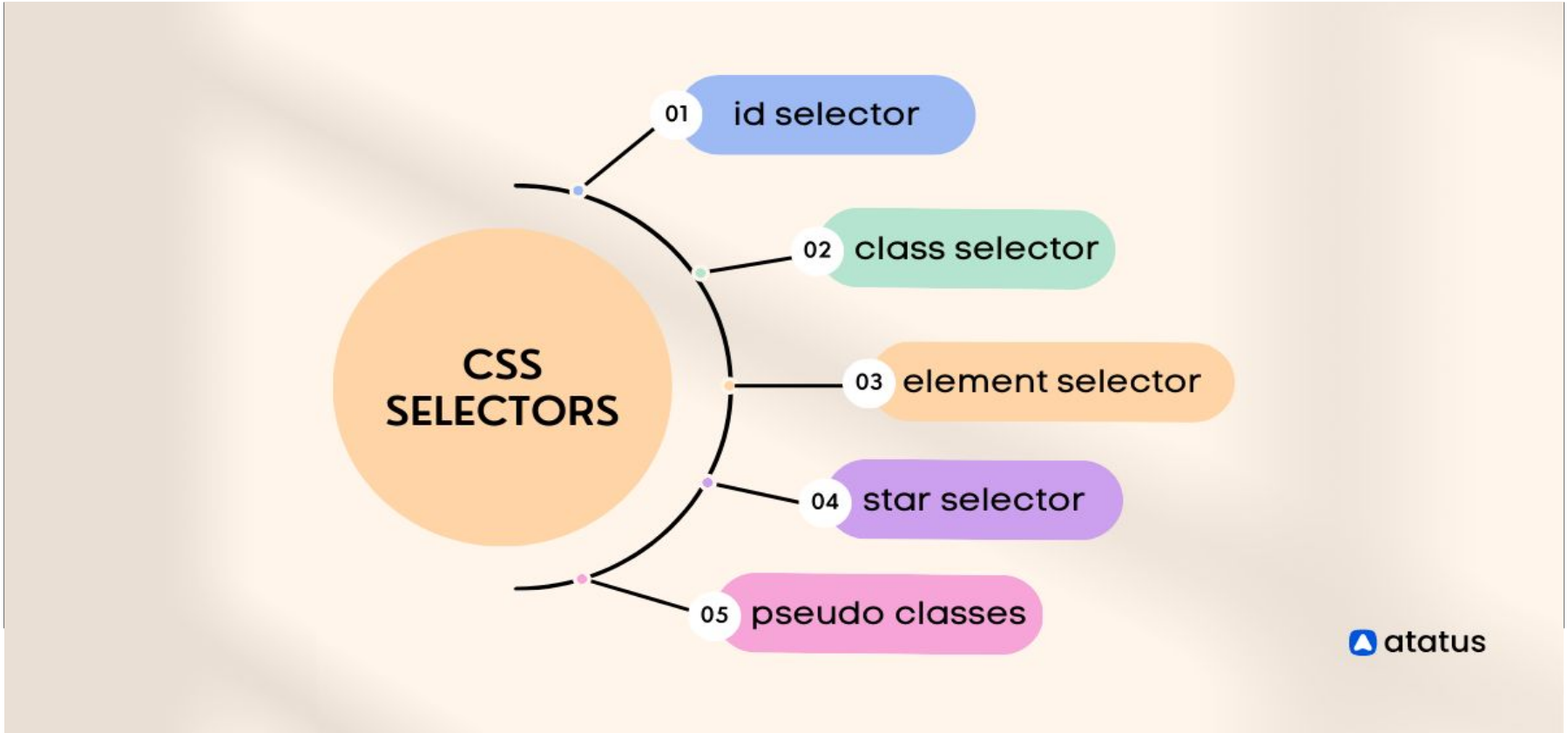
CSS Selectors: Universal

```
<head>
	<style>
		* {
			margin: 0; /* to remove default margin */
			padding: 0;
		}
	</style>
</head>
```

CSS Classes

```
<head>
	<style>
		.class1 {
			color: red;
		}
		.class2 {
			font-weight: bold;
		}
	</style>
</head>

<body>
	<h1 class="class1 class2">this is a heading</h1>
</body>
```

CSS Selectors



CSS Borders: Example 1

```
.blog {
	border-style: solid;
	/* border-top-style: solid; */
	border-width: 2px;
	border-color: purple;
	/* border: 2px solid purple; border-style is required */
}
```

CSS Borders: Example 2

```
.blog {
	border: 2px solid purple; /* <!-- border-style is required --> */
	border-bottom: none;
}
```

CSS Margin

```
.blog {
	margin: 10px 50px; /* TB LR */
	margin: 10px 50px 100px; /* T LR B */
	margin: 10px 30px 60px 120px; /* T R B L */
}
```

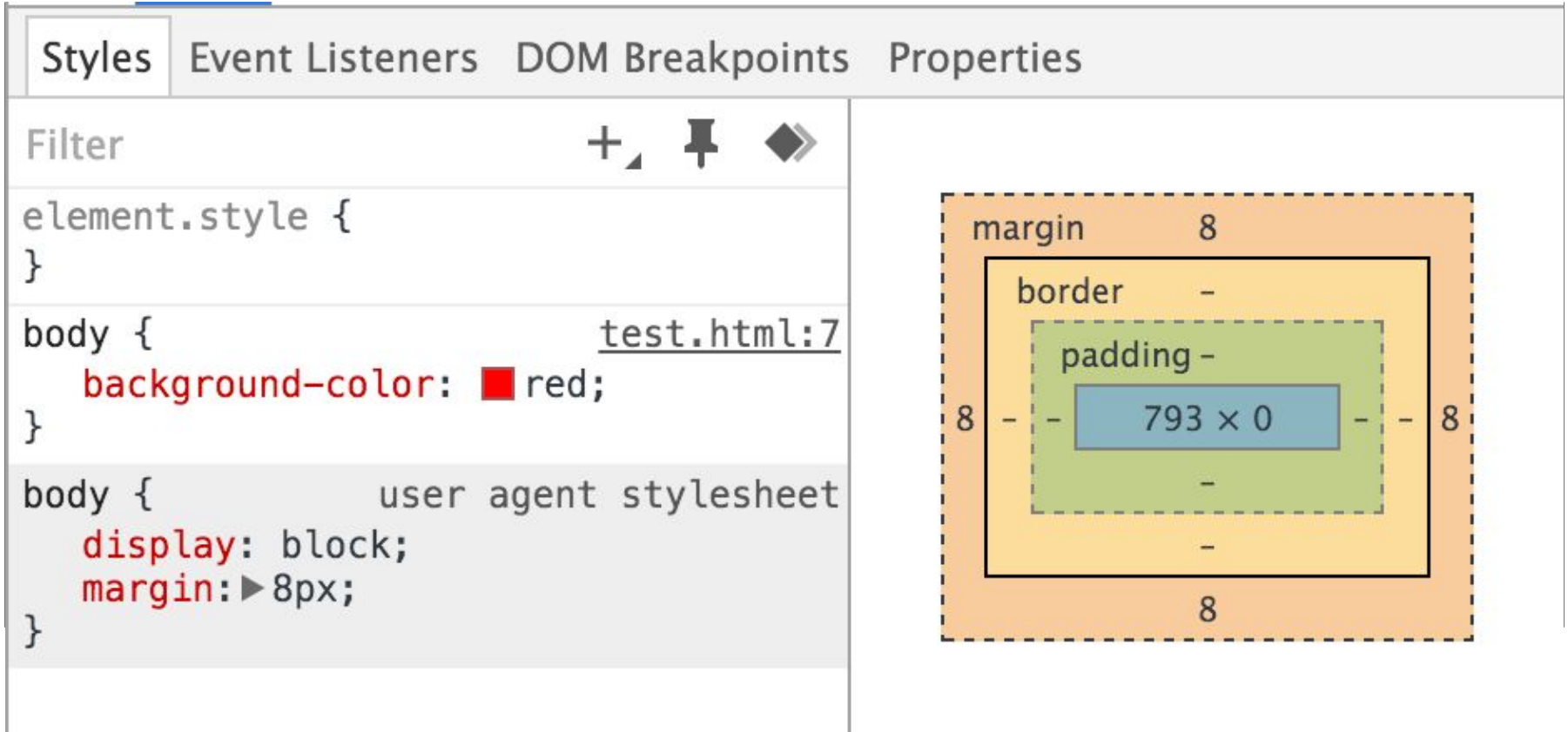
CSS Margin

```
<head>
	.highlight {
		background-color: yellow;
		margin: 20px; /* applies to left and right only (because span is an inline element)*/
	}
</head>

<body>
	<p>this is a <span class="highlight">paragraph</span>. this is another sentence.</p>
</body>
```

CSS Padding

```
.blog {
	padding: 10px 50px; /* TB LR */
	padding: 10px 50px 100px; /* T LR B */
	padding: 10px 30px 60px 120px; /* T R B L */
}
```

CSS Box Model



CSS Display

```
.foo {
	display: none; /* visibility: hidden; */
}
```

CSS Display

```
<head>
	.highlight {
		background-color: yellow;
		display: block; /* (inline-block) to make height and width work */
		height: 200px;
	}
</head>

<body>
	<p>this is a <span class="highlight">paragraph</span>. this is another sentence.</p>
</body>
```

CSS Box Shadow

```
.foo {
	box-shadow: 10px 10px 20px purple;
}
```

CSS Font Family

```
.foo {
	font-family: 'Courier New', Courier, monospace; /* fallbacks */
}
```

CSS Background Image

```
.foo {
	height: 200px;
	background-image: url('images/cat.jpeg'); /* '. . /images/cat.jpeg' */
	background-repeat: no-repeat; / * repeat-x | repeat-y */
	background-size: cover; / * 200px */
}
```

CSS Background Image

Custom Cursor

```
body {
	/* cursor: pointer; */
	cursor: url("images/smile.png"), auto; /* auto is fallback value */
}
```

CSS Flex: justify-content

```
.parent {
        display: flex;
        justify-content: space-between;
        /* start | end | center | space-around etc. */
}

.child {
        padding: 16px;
        background-color: bisque;
        width: 100px;
}

.red { background-color: red; }
.green { background-color: green; }
.blue { background-color: blue; }
```

```
<div class="parent">
        <div class="child red">1</div>
        <div class="child green">2</div>
        <div class="child blue">3</div>
</div>
```

CSS Flex: flex-direction

```
.parent {
        display: flex;
        justify-content: space-between;
        flex-direction: row-reverse;
}

.child {
        padding: 16px;
        background-color: bisque;
        width: 100px;
}

.red { background-color: red; }
.green { background-color: green; }
.blue { background-color: blue; }
```

```
<div class="parent">
        <div class="child red">1</div>
        <div class="child green">2</div>
        <div class="child blue">3</div>
</div>
```

CSS Flex: align-items

```
.parent {
	display: flex;
	height: 400px;
	align-items: center;
}

.child {
	padding: 16px;
	background-color: bisque;
	width: 100px;
}

.red { background-color: red; }
.green { background-color: green; }
.blue { background-color: blue; }
```

```
<div class="parent">
	<div class="child red">1</div>
	<div class="child green">2</div>
	<div class="child blue">3</div>
</div>
```