ΕΠΛ425

Τεχνολογίες Διαδικτύου

(Internet Technologies)

The Basics of Cascading Style Sheets (CSS)

Διδάσκων

Δρ. Χριστόφορος Χριστοφόρου

christophoros@cs.ucy.ac.cy

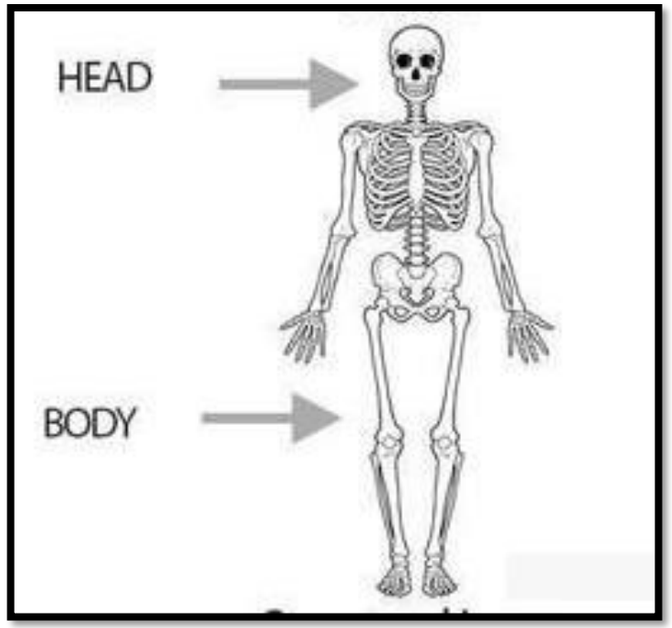
Goals

Introduction to Front-End Development:

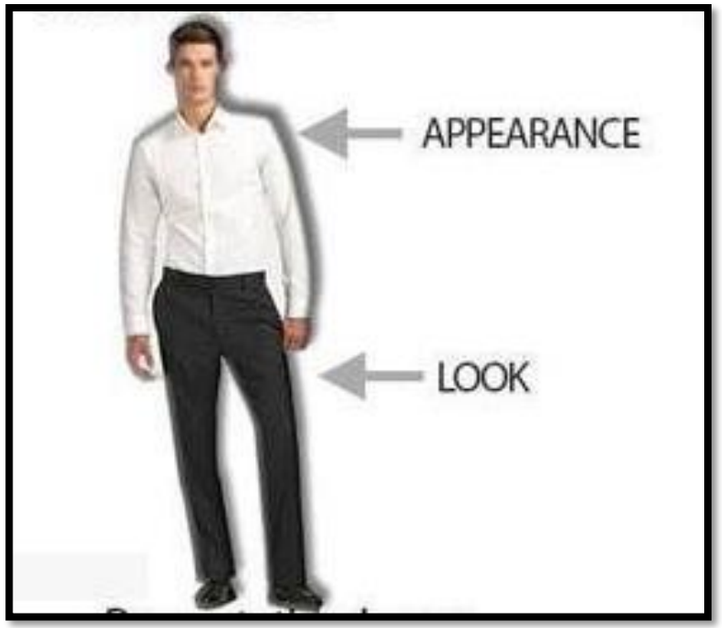
- HTML to create the document structure and content
- **CSS** to **control** its **visual/stylist aspect**
- Javascript for interactivity

Cascading Style Sheets (CSS)

- CSS (Cascading Style Sheets) is the **code** that **styles** **web content**.



HTML

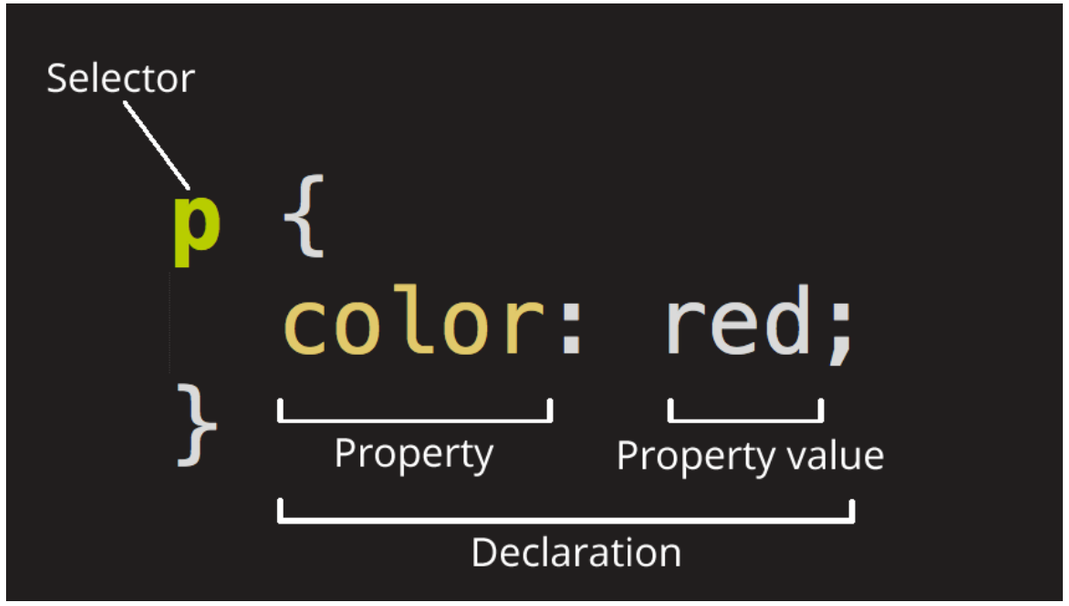


HTML + CSS

What is Cascading Style Sheets (CSS)?

- Like HTML, CSS is **NOT a programming language**. It's **NOT a markup language either**. **CSS** is a **style sheet language**.
- CSS is what you **use** to **selectively style HTML elements**.

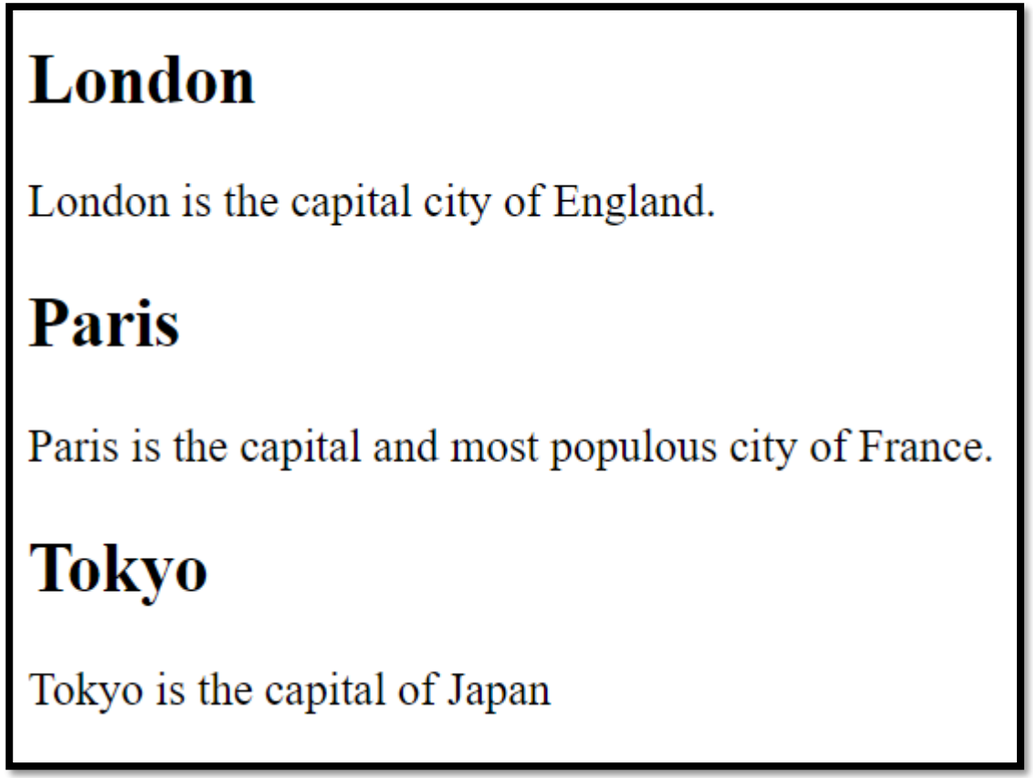
Anatomy of a CSS ruleset



The whole structure is called a **ruleset**.

HTML without CSS

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Web Page</title>
</head>

<body>
    <div class="cities">
        <h2>London</h2>
        <p>London is the capital city of England.</p>
    </div>
    <div class="cities">
        <h2>Paris</h2>
        <p>Paris is the capital and most populous city of France.</p>
    </div>
    <div class="cities">
        <h2>Tokyo</h2>
        <p>Tokyo is the capital of Japan </p>
    </div>
</body>

</html>
```



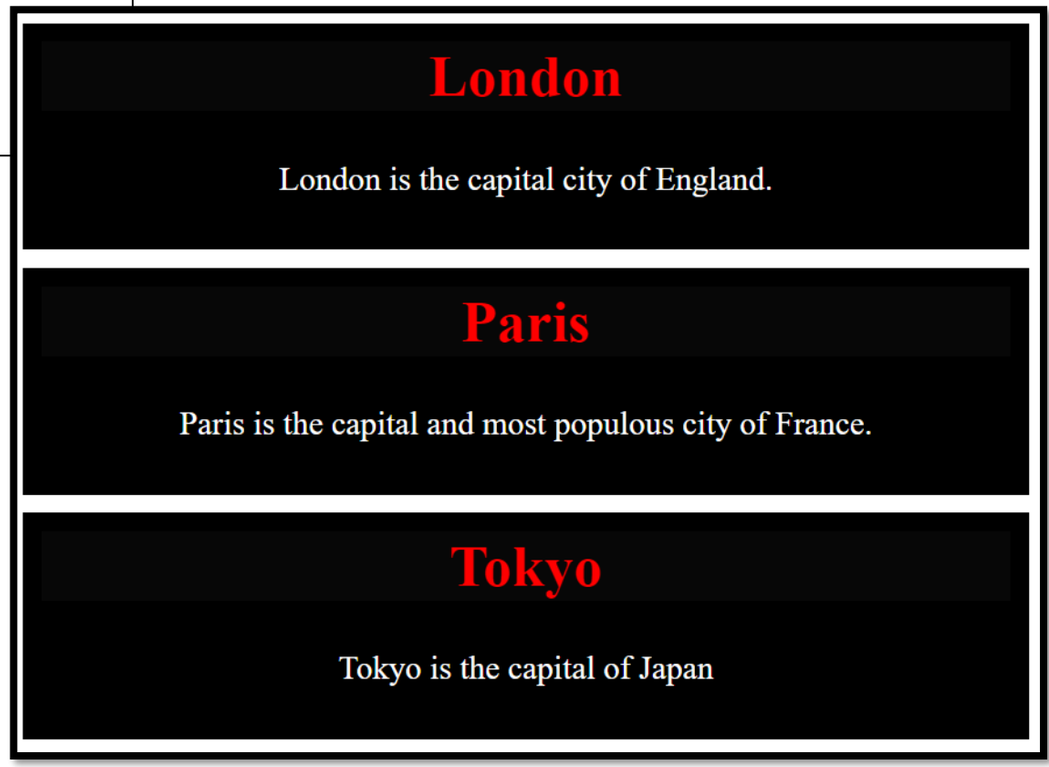
HTML with CSS

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Web Page With CSS</title>
    <link rel="stylesheet" href="CSS/style1.css">
</head>

<body>
    <div class="cities">
        <h2>London</h2>
        <p>London is the capital city of England.</p>
    </div>
    <div class="cities">
        <h2>Paris</h2>
        <p>Paris is the capital and most populous city of France.</p>
    </div>
    <div class="cities">
        <h2>Tokyo</h2>
        <p>Tokyo is the capital of Japan </p>
    </div>
</body>

</html>
```

```
div.cities {
    text-align: center;
    background-color: black;
    color: white;
    margin: 10px;
    padding: 10px;
}

h2 {
    text-align: center;
    margin-top: 0;
    font-size: xx-large;
    line-height: 1.2;
    color: red;
    background-color: #070707;
}

p {
    font-size: large;
    line-height: 1.2;
    color: white;
}
```

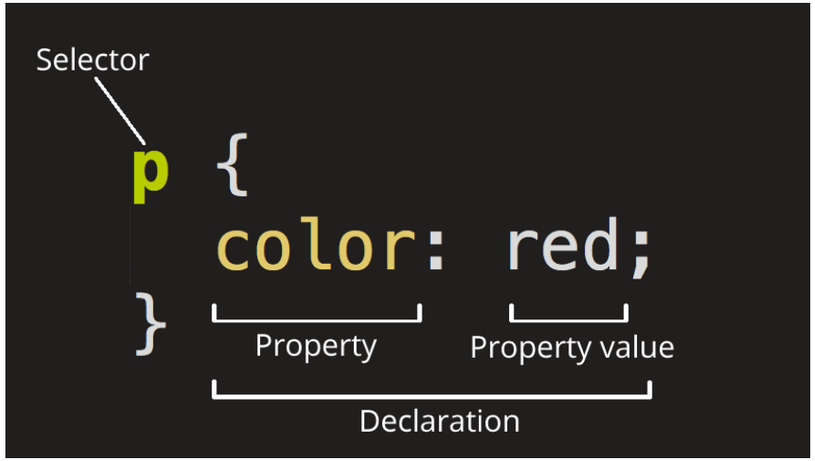
style1.css file in **CSS** folder

To **make the CSS code to work**, we still need to **link** this **style1.css** file to **our HTML document** using the **<link> element,** anywhere between the **<head>** and **</head>** tags.



Anatomy of a CSS ruleset

- **Selector**: This is the **HTML element name** at the **start of the ruleset**. It **defines** the **element**(s) **to be styled…** In this example, is all **<p>** elements. To style **different element(s), change the selector.**



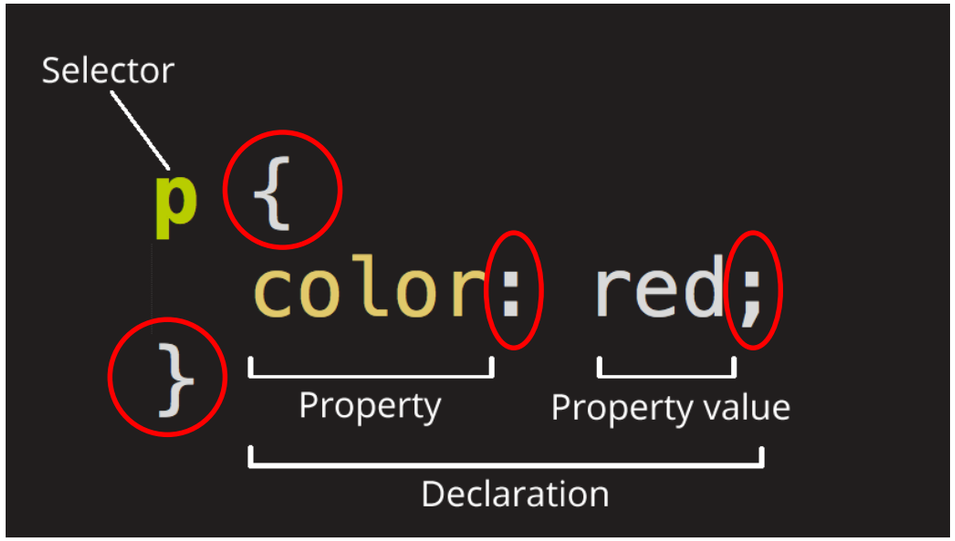
- **Declaration**: This is a **single rule** like **color: red;**

- It **specifies** which of the element's **property** you want to **affect** (in this case is the **color** of the **text** included in the **<p> element**) and the **property value** of how to **style** it (in this case the text of **<p>** elements will be **red**).

Anatomy of a CSS ruleset

Note the important parts of the syntax:

- Apart from the selector, **each ruleset must be wrapped** in **curly braces { }**
- **Within each declaration**, you must **use a colon :** to **separate** the **property** from its **value** or **values**.
- Each **declaration** should be followed by a **semi colon ;**



Types of Selectors

- There are **3 basic types** of CSS selectors:

Element selector (this is the one we've been using)	`p`	**All** `<p>` **elements**
ID selector	`#parag1`	**element** with `id="parag1"`
Class selector	`.parags`	**elements** with `class="parags"`

- **class** and **id** are **HTML attributes** that **can be used on any** HTML element:
  - **class**: Used on 1 or more elements; identifies a **collection of elements**
  - **id**: Used on **exactly 1 element** per page; identifies **one unique element**

Types of Selectors

- In case of **multiple declarations** in one ruleset, you must **use a semicolon (;)** to **separate each declaration** from the **next one**.

```
p {
    font-size: large;
    line-height: 1.2;
    color: white;
}
```

- You can also **select multiple HTML elements** and **apply a single CSS ruleset to all of them**. **Separate multiple selectors** by **commas ,**

```
p, h1, h2 {
    color: red;
    text-align: center;
}
```

Types of Selectors

- **To select** an **element with a specific id use the #.** Note that on a given HTML page**, each id value should be UNIQUE**.
- The example at the right side will **apply the CSS ruleset** to the element with attribute **id="id1"**

```
#id1 {
    color: red;
    text-align: center;
}
```

- **To select** all **elements with a specified class use the dot .**
- The example at the right side will **apply the CSS ruleset** to all the elements with attribute class="class1"

```
.class1 {
    color: red;
    text-align: center;
}
```

Types of Selectors

- **To select** all the ** elements with a specified class** use the **type of the element** plus the **dot . followed by the class name.**
- The example at the right side will **apply the CSS ruleset** to elements of type ** ** and with attribute class="class1".

```
div.class1 {
    color: red;
    text-align: center;
}
```

Types of Selectors

- To select an element but **only when it is in a specified state** use **two dots :**
- The example at the right side is referred as **pseudo-class selector**.
- It **applies the CSS ruleset** to an element of type **<a> but only** when the **cursor hovers over** the **link**.
- Pseudo-class selectors can also be applied on other HTML elements as well (e.g., <h1>, <p>, <img>, etc.)

```
a:hover  {
    color:red;
    text-decoration:underline
}
```

Selector	Example	Example description	CSS
.class	.intro	Selects all elements with class="intro"	1
#id	#firstname	Selects the element with id="firstname"	1
*	*	Selects all elements	2
element	p	Selects all <p> elements	1
element,element	div,p	Selects all elements and all <p> elements	1
element element	div p	Selects all <p> elements inside elements	1
element>element	div>p	Selects all <p> elements where the parent is a element	2
element+element	div+p	Selects all <p> elements that are placed immediately after elements	2
[*attribute*]	[target]	Selects all elements with a target attribute	2
[*attribute=value*]	[target=_blank]	Selects all elements with target="_blank"	2
[*attribute~=value*]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"	2
[*attribute|=language*]	[lang|=en]	Selects all elements with a lang attribute value starting with "en"	2
:link	a:link	Selects all unvisited links	1
:visited	a:visited	Selects all visited links	1
:active	a:active	Selects the active link	1
:hover	a:hover	Selects links on mouse over	1
:focus	input:focus	Selects the input element which has focus	2
:first-letter	p:first-letter	Selects the first letter of every <p> element	1
:first-line	p:first-line	Selects the first line of every <p> element	1
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent	2
:before	p:before	Insert content before every <p> element	2
:after	p:after	Insert content after every <p> element	2
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"	2
element1~element2	p~ul	Selects every ul element that are preceded by a p element	3
[*attribute^=value*]	a[src^="https"]	Selects every a element whose src attribute value begins with "https"	3
[*attribute$=value*]	a[src$=".pdf"]	Selects every a element whose src attribute value ends with ".pdf"	3
[*attribute**=*value*]	a[src*="w3schools"]	Selects every a element whose src attribute value contains the substring "w3schools"	3
:first-of-type	p:first-of-type	Selects every p element that is the first p element of its parent	3
:last-of-type	p:last-of-type	Selects every p element that is the last p element of its parent	3
:only-of-type	p:only-of-type	Selects every p element that is the only p element of its parent	3
:only-child	p:only-child	Selects every p element that is the only child of its parent	3
:nth-child(*n*)	p:nth-child(2)	Selects every p element that is the second child of its parent	3
:nth-last-child(*n*)	p:nth-last-child(2)	Selects every p element that is the second child of its parent, counting from the last child	3
:nth-of-type(*n*)	p:nth-of-type(2)	Selects every p element that is the second p element of its parent	3
:nth-last-of-type(*n*)	p:nth-last-of-type(2)	Selects every p element that is the second p element of its parent, counting from the last child	3
:last-child	p:last-child	Selects every p element that is the last child of its parent	3
:root	:root	Selects the document's root element	3
:empty	p:empty	Selects every p element that has no children (including text nodes)	3
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)	3
:enabled	input:enabled	Selects every enabled input element	3
:disabled	input:disabled	Selects every disabled input element	3
:checked	input:checked	Selects every checked input element	3
:not(*selector*)	:not(p)	Selects every element that is not a p element	3
::selection	::selection	Selects the portion of an element that is selected by a user	3

There are **many more selectors** to discover….

Types of Selectors: Some Common confusions

```
div.class1 {
    color: red;
    text-align: center;
}
```

Apply the CSS ruleset to all elements with **class = "class1"**

Vs

```
div .class1 {
    color: red;
    text-align: center;
}
```

Apply the CSS ruleset to all the elements with the **class = "class1"** that are **children** of ** elements**

Note that there is a comma here

Vs

```
div, .class1 {
    color: red;
    text-align: center;
}
```

Apply the CSS ruleset to **all** elements and all the elements with **class = "class1"**

Colliding styles

- When **styles collide**, the **most specific rule wins**.

```
div p {
   color: red;
}

p {
   color: blue;
}
```

Specificity precedence rules (details):

- **ids** are **more specific** than **classes**
- **classes** are **more specific** than **element names**
- **Style rules** that **directly target elements** are **more specific** than **style rules** that are **inherited.**

```
<div>
    <p>Hello EPL425 People!!</p>
</div>
```

Q: What will be the color of the paragraph's text???

Colliding styles

```
div p {
   color: red;
}

p {
   color: blue;
}
```

```
<div>
    <p>Hello EPL425 People!!</p>
</div>
```

A: Red!

Colliding styles

- However, if style rules have the **same specificity**, the **later rule** **wins**.

```
p {
   color: red;
}

p {
   color: blue;
}
```

```
<div>
    <p>Hello EPL425 People!!</p>
</div>
```

Q: What will be the color of the paragraph's text???

Colliding styles

```
p {
   color: red;
}

p {
   color: blue;
}
```

```
<div>
    <p>Hello EPL425 People!!</p>
</div>
```

A: Blue!

Combining selectors

- You can **combine** selectors:

```
#notes p.important strong {
    color: red;
}
```

Q: What does this select?

Hint: Read from right to left

Combining selectors

```
#notes p.important strong {
    color: red;
}
```

A: This will apply the CSS ruleset to all elements that are children of <p> tags with attribute class="important" that are children of the element with the attribute id="notes"

Some CSS properties

There are over 500 CSS properties that you can use! Below are a few examples:

Font face (mdn)	**font-family:** Helvetica;
Font color (mdn)	**color:** gray;
Background color (mdn)	**background-color:** red;
Border (mdn)	**border:** 3px solid green;
Text alignment (mdn)	**text-align:** center;

Note: Mozilla Developer Network (MDN) is one of the **best reference** for HTML elements and CSS properties. The actual **W3 spec** is very hard to read (meant for browser developers, not web developers)

Inheritance

```
p {
    font-family: Helvetica;
    text-align: center;
    color: white;
}

h1 {
    font-family: Helvetica;
    text-align: center;
    color: white;
}

h2 {
    font-family: Helvetica;
    text-align: center;
    color: white;
}
```

- CSS styles **can be inherited** from **parent** to **child**.
- For example if you want **some CSS styles** to be **used in all elements included** in the <body> of your **web page….**

…Instead of selecting all elements individually

```
body {
    font-family: Helvetica;
    text-align: center;
    color: white;
}
```

….you can style the parent and the children will inherit the styles…

Inheritance

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Web Page With CSS</title>
    <link rel="stylesheet" href="CSS/style1.css">
</head>

<body>
    <div class="cities">
        <h2>London</h2>
        <p>London is the capital city of England.</p>
    </div>
    <div class="cities">
        <h2>Paris</h2>
        <p>Paris is the capital and most populous city of France.</p>
    </div>
    <div class="cities">
        <h2>Tokyo</h2>
        <p>Tokyo is the capital of Japan </p>
    </div>
</body>

</html>
```

```
body {
    font-family: Helvetica;
    text-align: center;
    color: white;
}
```

With the above CSS ruleset, all elements included in body (e.g., p, h1, h2, etc.), will inherit the CSS styles declared in the body selector CSS ruleset.

Inheritance

```html
<!DOCTYPE html>
<html>

<head>
    <title>HTML Web Page With CSS</title>
    <link rel="stylesheet" href="CSS/style1.css">
</head>

<body>
    <div class="cities">
        <h2>London</h2>
        <p>London is the capital city of England.</p>
    </div>
    <div class="cities">
        <h2>Paris</h2>
        <p>Paris is the capital and most populous city of France.</p>
    </div>
    <div class="cities">
        <h2>Tokyo</h2>
        <p>Tokyo is the capital of Japan </p>
    </div>
</body>

</html>
```

```css
body {
    font-family: Helvetica;
    text-align: center;
    color: white;
}

h2 {
    font-size: 30px;
    color: red;
}
```

Then, you can override a style if you want. For example if you want to make the text color of h2 red and also declare a new font-size for this, you can declare a specific CSS rule for h2.

Inheritance

- Note that, while **many CSS styles** are **inherited** from parent to child, **sometimes NOT ALL CSS properties are inherited**.

- There's **no rule** for **what properties** are **inherited** or not; the **inheritance behavior** is defined in the CSS spec. **Generally:**
  - **text-related properties** (i.e., font-family, text-align, etc.) **are inherited**
  - **layout-related properties** (i.e., display, etc.) **are not**.

Inheritance

- Also **for some elements**, the **browser has its own default styles**.
- For example for anchor **<a>** elements **the browser "silently" loads its own default stylesheet** on every webpage.
- So to style **<a>** links, we have to **override** the **browser default link style by explicitly setting a color.**

Link-related CSS

- Since we're on the topic of links, lets see how do we style them using CSS.
- Also we will discuss about **pseudo-classes** and **how these are used** on **styling** elements (and not only anchor <a> elements).

Link-related CSS

- When you move the mouse over a link, two things will normally happen:
  - The **mouse arrow** will turn into a **little hand**
  - The **color** of the **link element** will **change**
- **By default**, a link will appear like this (in all browsers):
  - An **unvisited link** is underlined and **blue**
  - A **visited link** is underlined and **purple**
  - An **active link** is underlined and **red**

Pseudo-classes

- **Pseudo-classes** are **special keywords** you can **append** to selectors, **specifying a special state** of the **selected element(s).** For example, it can be used to:
  - **Style** an element when a **user move** the **mouse over it**
  - **Style visited** and **unvisited links differently**
  - **Etc.**

- The syntax of pseudo-classes:

```
selector:pseudo-class {
    property: value;
}
```

The **complete list** of pseudo-classes can be found here! Have a look!

Link related CSS pseudo-classes

Syntax	Explanation
a:link	Represents an element that has not yet been visited. Applies only for <a> and <area> elements that have an href attribute.
a:visited	Applies once the link has been visited by the user. Applies only for <a> and <area> elements that have an href attribute.
a:hover	Matches when a user designates an item with a pointing device, such as holding the mouse pointer over the item.
a:active	Matches when an item is being activated by the user. For example, when the item is clicked on.

Link related CSS pseudo-classes

- You can **change** the **default colors** a link will appear by using the following **CSS rulesets**.

```
a:link {color:green; background-color:transparent; text-decoration:none}
a:visited {color:pink; background-color:transparent; text-decoration:none}
a:hover  {color:red; background-color:transparent; text-decoration:underline}
a:active {color:yellow; background-color:transparent; text-decoration:underline}
```

Main ways to define CSS colors:

140 predefined color names (Check this list)

```
color: black;
```

Hex values

```
color: #00ff00;
color: #0f0;
color: #00ff0080;
```

rgb() and rgba()

```
color: rgb(34, 12, 64);
color: rgba(0, 0, 0, 0.5);
```

The "**a**" in **rgba ()** stands for **alpha channel** and is a **transparency** value!

Types of HTML elements: Block and Inline

Each **HTML element** is categorized by the HTML spec into one of following **three categories**:

- **block:** large blocks of content, has **height** and **width.** E.g., **<p>, <h1>, , <ol>, <ul>, <table>**
- **inline:** small amount of content, with **no height** or **width:** E.g., **<a>, ,
**
  - **Inline-block:** inline content **with height** and **width**: E.g., **<img>**
- **metadata**: information about the page, usually **not visible on the web page:** E.g., <title>, <meta>

Block elements

Examples:

<p>, <h1>, , <ol>, <ul>, <table>

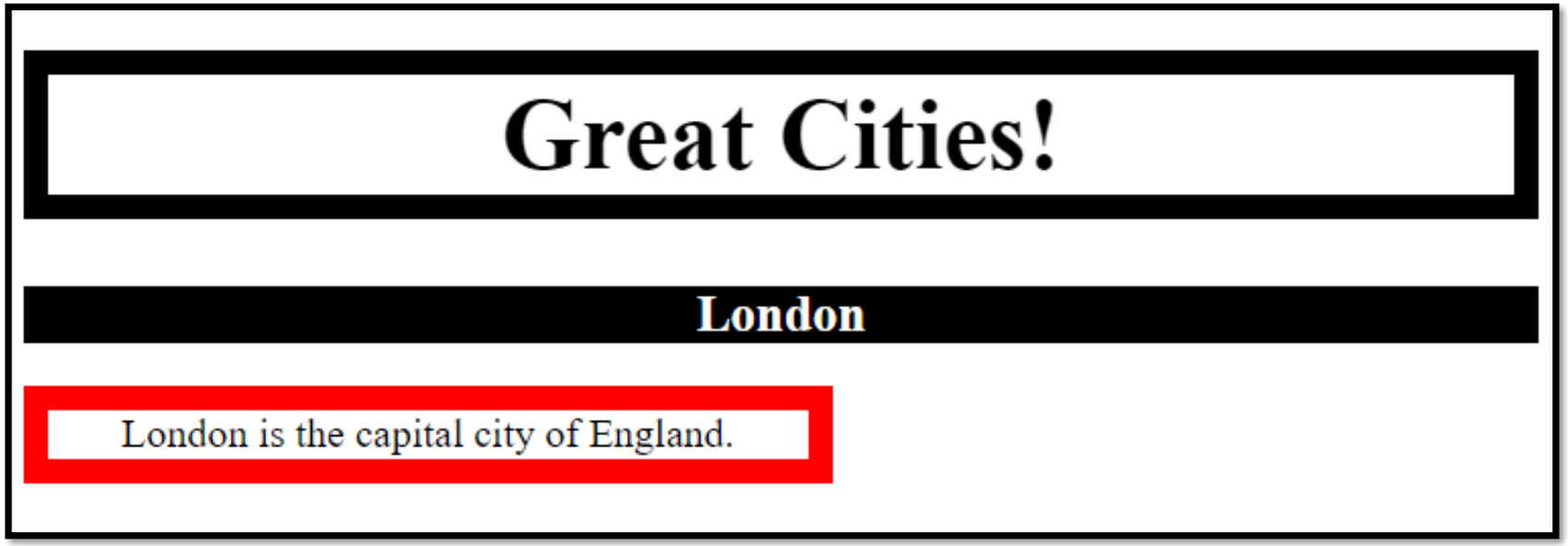
- Take up the **full width** of the page.
- **Flows from top to bottom**
- Have a **height** and **width**. Note that height and width **can be modified with CSS and JavaScript**.
- **Can have block** or **inline** elements as **children**.

Example: Block Elements

```html
<!DOCTYPE html>
<html>

<head>
    <title>Block Vs Inline Elements</title>
    <link rel="stylesheet" href="CSS/style2.css">
</head>

<body>
    <h1>Great Cities!</h1>
    <h2>London</h2>
    <p>London is the capital city of England.</p>
</body>

</html>
```

```css
h1 {
    text-align: center;
    font-size: 40px;
    border: 10px solid black;
    padding: 20px auto;
}

h2 {
    text-align: center;
    color: white;
    font-size: 20px;
    background-color: black;
}

p {
    text-align: center;
    border: 10px solid red;
    padding: 20px auto;
    width: 50%;
}
```

Example: This is how it will look on the browser

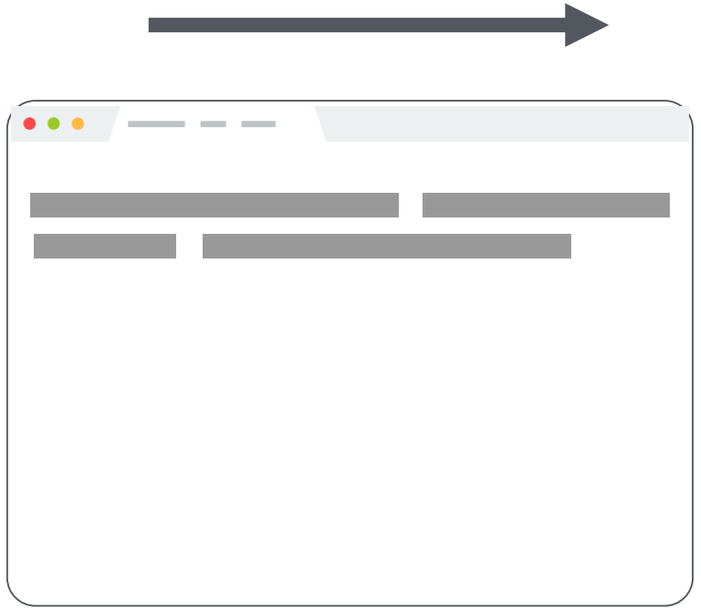


Block-level elements **by default**, unless stated otherwise (i.e., declare size of its width; **width: 50%;**), extends the **full width of the page**.

Also note how **block-level elements flow from top to bottom**.

Inline elements

Examples: <a>, , ,



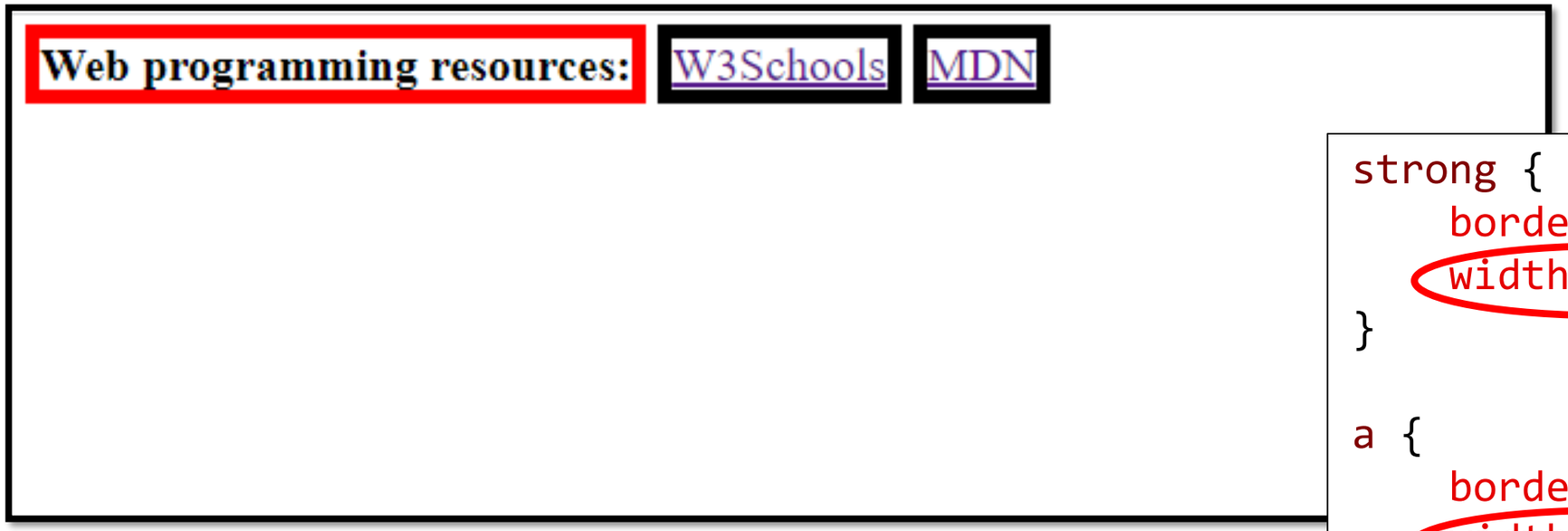
- **Take up only as much width as needed**
- **Flows from left to right.** If you want to put it on a **new line** then **use
**
- **CANNOT** have **height** and **width**
- **CANNOT** have a **block element childs**
- **CANNOT** be **positioned** (i.e., CSS properties like **float** and **position** **do not apply** to inline elements) → However you can **position its container block element** instead.

Example: Inline

```css
strong {
    border: 5px solid red;
    width: 1000px;
}

a {
    border: 5px solid black;
    width: 100px;
}
```

```html
<!DOCTYPE html>
<html>

<head>
    <title>Block Vs Inline Elements</title>
    <link rel="stylesheet" href="CSS/style2.css">
</head>

<body>
    <strong>Web programming resources:</strong>
    <a href="https://www.w3schools.com/" target="_blank">W3Schools</a>
    <a href="https://developer.mozilla.org/en-US/" target="_blank">MDN</a>
</body>

</html>
```

Example: This is how it will look on the browser

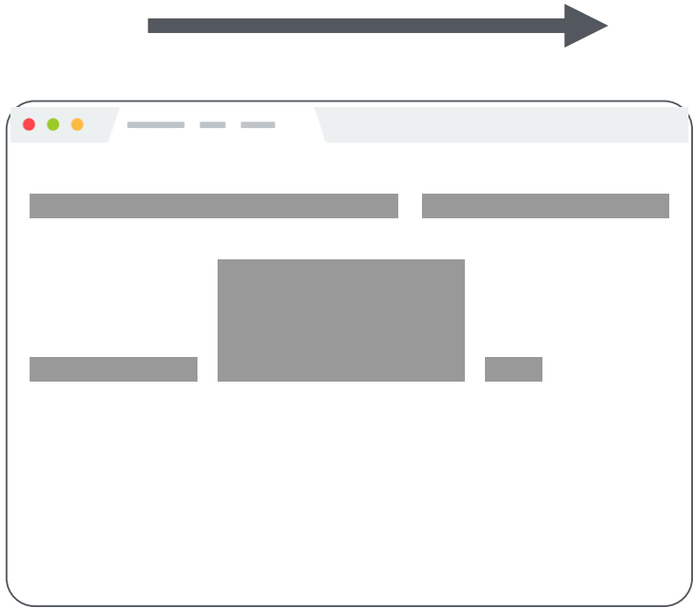



```
strong {
    border: 5px solid red;
    width: 1000px;
}

a {
    border: 5px solid black;
    width: 100px;
}
```

Remember that you cannot set **width** on an **inline element**, so in this case it is **ignored**!

Inline-block elements

- **Examples: <img>**, and any element with property **display: inline-block;**
- **Space allocated** on the browser is the **size of the content**, i.e., it **takes only as much space as needed!!!**
- **Flows from left to right**.
- **Can have height** and **width**
- **Can be positioned** (i.e., CSS properties like **float** and **position** apply)



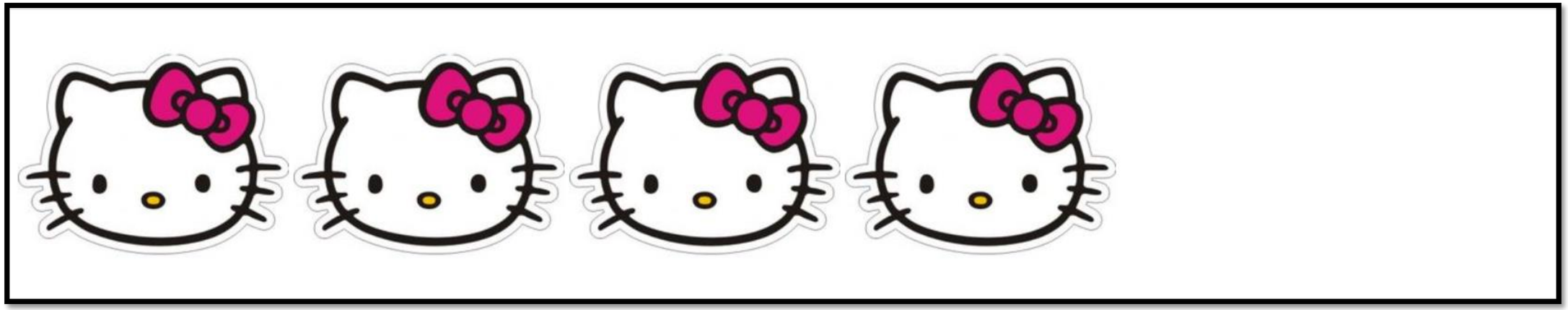
Example: Inline-block

```html
<!DOCTYPE html>
<html>

<head>
    <title>Block Vs Inline Elements</title>
    <link rel="stylesheet" href="CSS/style2.css">
</head>

<body>
    <img src="http://i.imgur.com/WJToVGv.jpg" alt="Kitty"/>
    <img src="http://i.imgur.com/WJToVGv.jpg" alt="Kitty"/>
    <img src="http://i.imgur.com/WJToVGv.jpg" alt="Kitty"/>
    <img src="http://i.imgur.com/WJToVGv.jpg" alt="Kitty"/>
</body>

</html>
```

```css
img {
    width: 40px;
    height: auto;
}
```

http://i.imgur.com/WJToVGv.jpg =

Example: This is how it will look on the browser



You can set width and **height** on **inline-block** elements, so image **width** is set to **40px** and **height** to **auto** to **lock ratio**.

inline-block elements flows **from left to right**, so images are right next to each other.

The `display` CSS property

- You can **change** an element's default **rendering type** **by changing the display property**. Examples:

```
p {
    display: inline;
}
```

```
img {
    display: block;
}
```

- The most used values for display are:
  - block
  - inline
  - inline-block
  - For more details check this link!

To Sum up

- **block:** flows **top-to-bottom**; **has height** and **width.** Takes the **whole line**. **Examples:** <p>, <h1>, <blockquote>, <ol>, <ul>, <table>
- **inline:** flows **left-to-right**; **does not have height** and **width. Take up only as much width as needed. Examples:** <a>, ,
- **inline-block:** flows **left-to-right**; **has height** and **width equal to size of the content. Takes only as much space as needed. Examples:** <img>

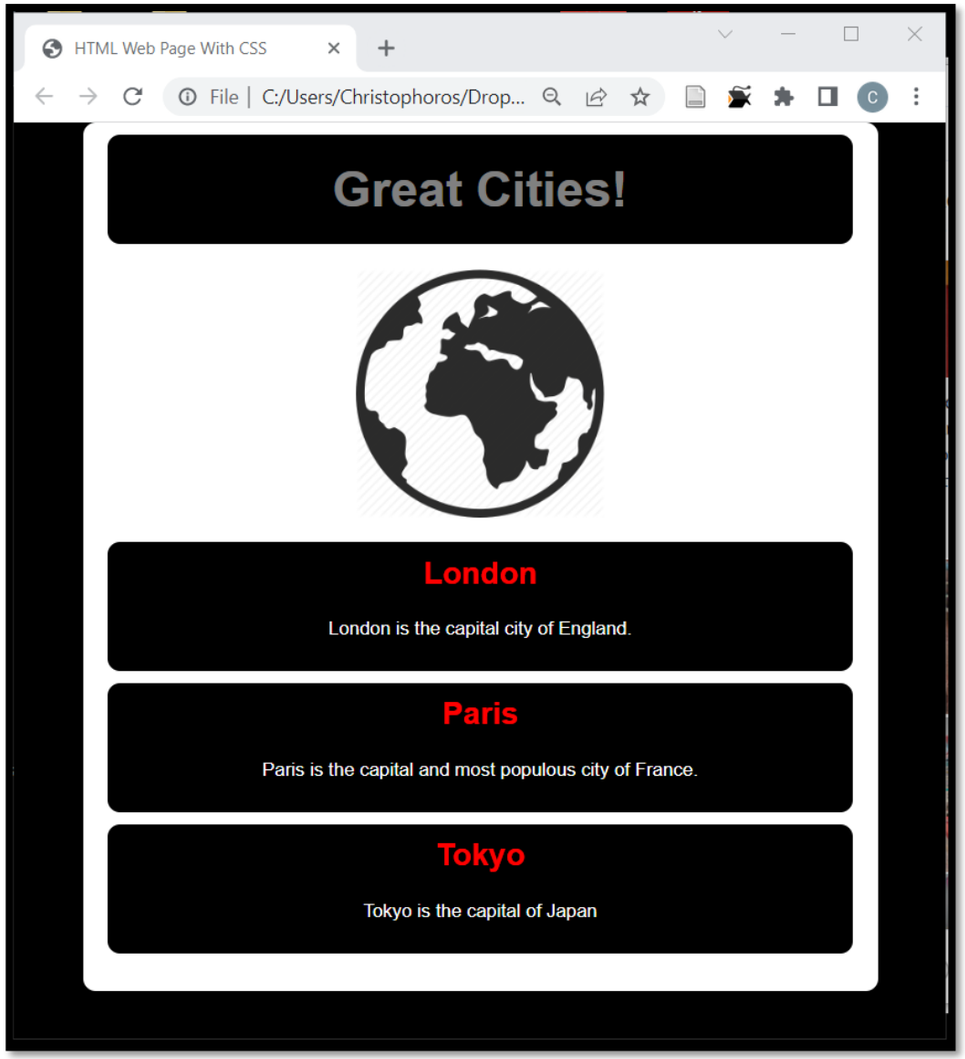
Applying CSS in an HTML document – An example

Now that we've explored some CSS fundamentals, let's **start an example** and see **how we can improve the appearance** of the following **HTML document...**

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Web Page With CSS</title>
 </head>

<body>
    <h1>Great Cities!</h1>
    <img src="images/Earth_globe.png" alt="The Globe" width="200" height="auto"/>

    <div class="cities">
        <h2>London</h2>
        <p>London is the capital city of England.</p>
    </div>
    <div class="cities">
        <h2>Paris</h2>
        <p>Paris is the capital and most populous city of France.</p>
    </div>
    <div class="cities">
        <h2>Tokyo</h2>
        <p>Tokyo is the capital of Japan </p>
    </div>
</body>

</html>
```

Applying CSS in an HTML document – An example

…into THIS!!!



Applying CSS in an HTML document – An example

- **First** we have to **link** the HTML file to the **style1.css** file **in which the CSS rules will be coded.**
- This **style1.css** is **located** in the **CSS folder** or our web site…
- Add the <link> **element anywhere between** the <head> and </head> **tags**.

```
<!DOCTYPE html>
<html>

<head>
    <title>HTML Web Page With CSS</title>
    <link rel="stylesheet" href="CSS/style1.css">
 </head>

<body>
    <h1>Great Cities!</h1>
    <img src="images/Earth_globe.png" alt="The Globe" width="200" height="auto">

    <div class="cities">
        <h2>London</h2>
        <p>London is the capital city of England.</p>
    </div>
    <div class="cities">
        <h2>Paris</h2>
        <p>Paris is the capital and most populous city of France.</p>
    </div>
    <div class="cities">
        <h2>Tokyo</h2>
        <p>Tokyo is the capital of Japan </p>
    </div>
</body>

</html>
```

Applying CSS in an HTML document – An example

- Then start **adding some rules and information** to the **style1.css file** based on **how we want** our **web page to look like**.
- Lets start from **changing** the **html page color**.
- This **first CSS ruleset**, which is applied on the whole html document, sets a background-color for the entire page to **black**.

style1.css

```
html {
    background-color: black;
}
```

Applying CSS in an HTML document – An example

- Lets continue by styling the body:
  - **width: 600px;** This forces the body to always be 600 pixels wide.
  - **background-color: white;** This sets the <body> background color to white
  - **text-align: center; font-family: Arial, Helvetica, sans-serif; line-height: 1.2;** These will **style the text** of the **body**.

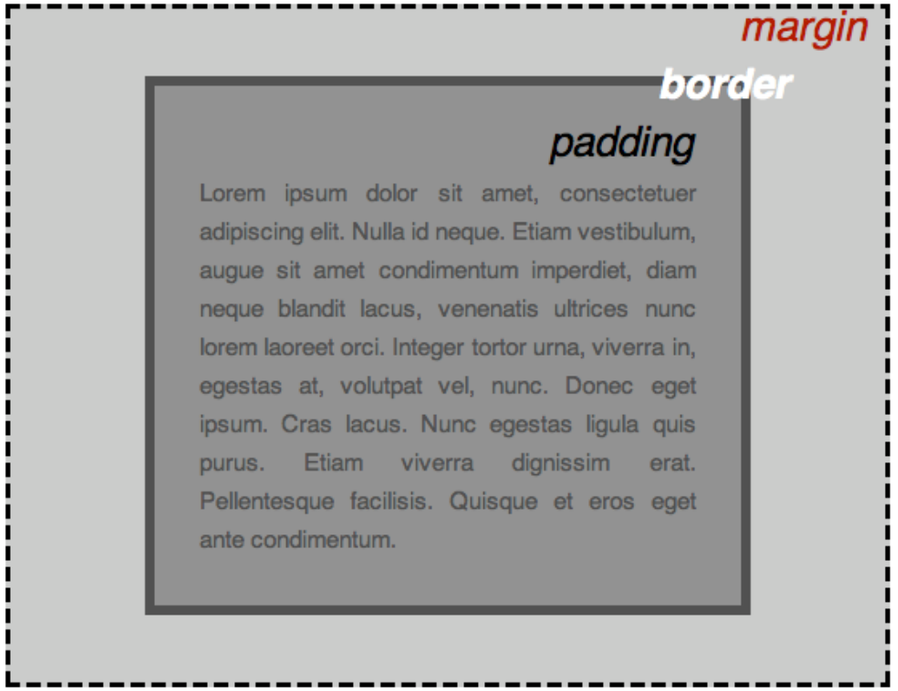
style1.css

```
body {
    width: 600px;
    background-color: white;
    text-align: center;
    font-family: Arial, Helvetica, sans-serif;
    line-height: 1.2;
    margin: 0 auto;
    padding: 10px 20px 20px 20px;
    border-radius: 10px;
}
```

- **All the text-related attributes** will be also **inherited** to the **child elements** of <body> element, like , <h1>, <h2>, and <p> elements.

The Box Model (margin, padding and border)

- **CSS layout** is mostly based on the ***box model.*** Each element taking up space on your page has properties like:
  - **margin**, the **space outside of the element's border**.
  - **padding**, the **space around the element's content from** the **element's border**. In the example provided, it is the **space around** the **paragraph text**.
  - **border**, the solid line that defines the **space allocated** by the **element**.



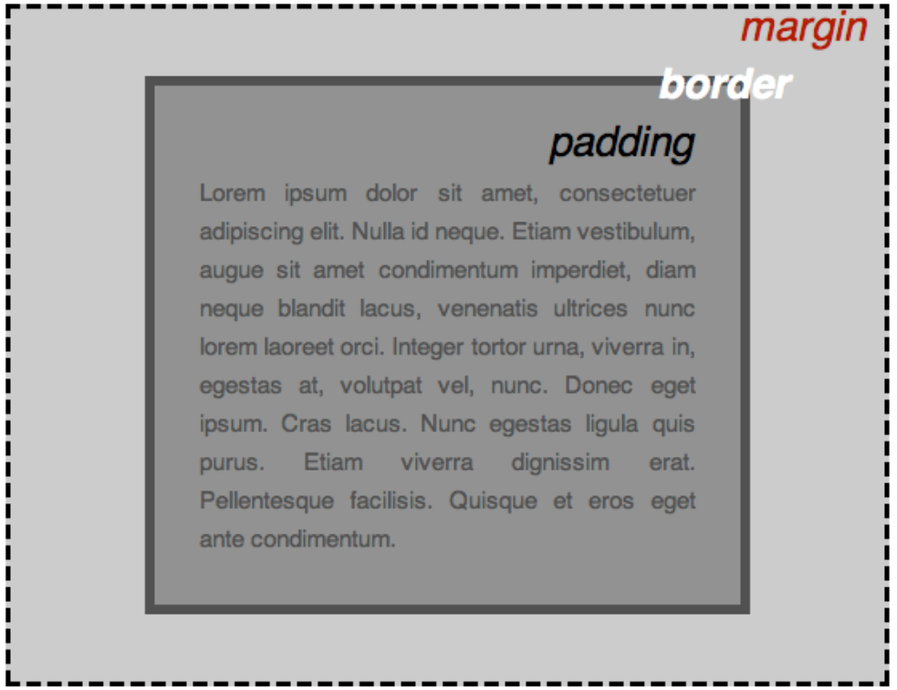
margin

margin is the **space between** the **element's border** and **other elements**.

- Can specify margin-top, margin-bottom, margin-left, margin-right, individually.

There's also a shorthand:

- margin: **2px 4px 3px 1px**;
  top|right|bottom|left
- margin: **10px 2px**;
  top+bottom|left+right

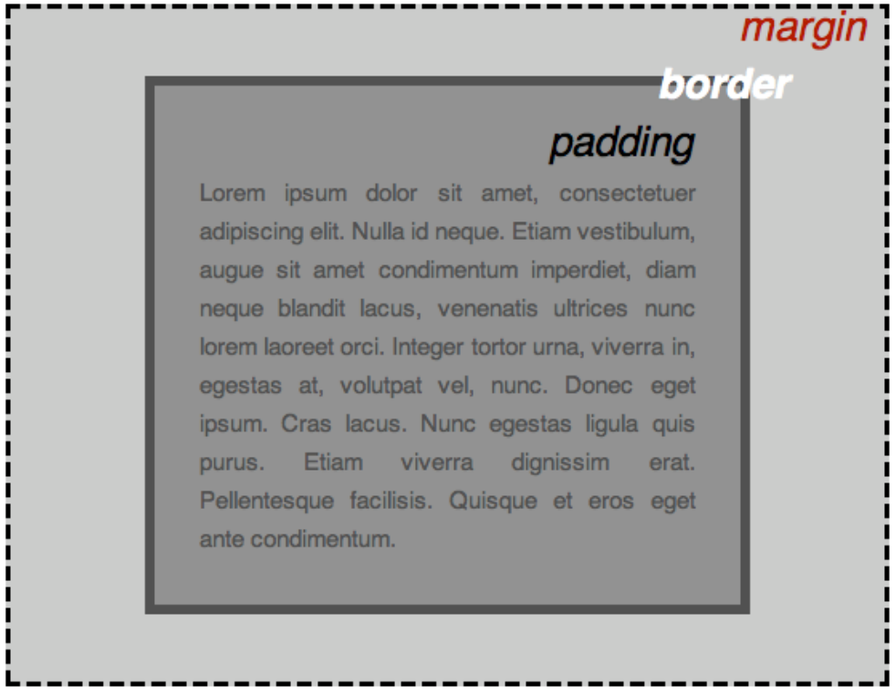


Note: There is also the case of a negative margin!!! We try this later to see what happens!!!

padding

padding is the **space between** the **element's border** and **the element's content**.

- Can specify padding-top, padding-bottom, padding-left, padding-right, **individually**



There's also a shorthand:

- padding: **2px 4px 3px 1px**;
  top|right|bottom|left
- padding: **10px 2px**; **<-**
  top+bottom|left+right

border (setting width style color)

Shorthand: border: *width style color*;
Example: border: 3px dotted black;

You can also **set each property individually** (See all styles)

```
border-style: dotted;
border-width: 5px;
border-color: black;
```

Can also **specify each border individually**:

```
border-top
border-bottom
border-left
border-right
```

Can also specify the **border-radius** to make **rounded corners**:

```
border-radius: 5px;
```

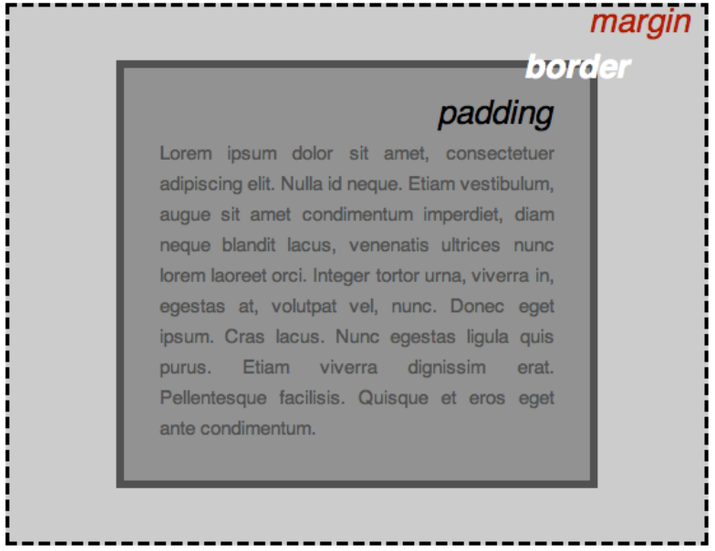
Applying CSS in an HTML document – An example

- **margin: 0 auto;** The **first value** affects the **element's top** and **bottom side** (setting it to 0 pixels in this case); the **second value** affects the **left** and **right side**.

- Here, **auto** is a **special value** that **divides the available horizontal space evenly between left and right**, thus **putting the body in the center of the HTML page.**

style1.css

```
body {
    width: 600px;
    background-color: white;
    text-align: center;
    font-family: Arial, Helvetica, sans-serif;
    line-height: 1.2;
    margin: 0 auto;
    padding: 10px 20px 20px 20px;
    border-radius: 10px;
}
```

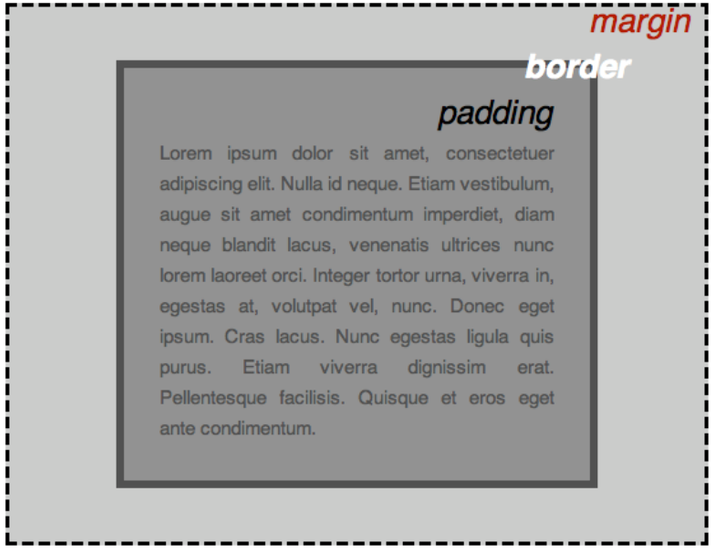


Applying CSS in an HTML document – An example

- **padding: 10px 20px 20px 20px;**
  This sets four values for padding.
- The goal is to **put some space around the content from the borders of its container (i.e., from the <body> border)**.
- In this example, there is 10 pixels from the top of the body, and 20 pixels from the right, bottom and left.

style1.css

```
body {
    width: 600px;
    background-color: white;
    text-align: center;
    font-family: Arial, Helvetica, sans-serif;
    line-height: 1.2;
    margin: 0 auto;
    padding: 10px 20px 20px 20px;
    border-radius: 10px;
}
```



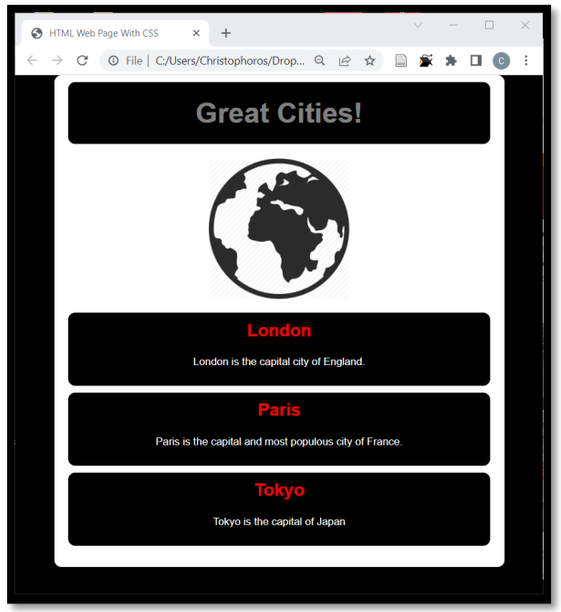
Applying CSS in an HTML document – An example

- **border-radius: 10px;** This property **defines** the **radius** of the **element's corners**.
- It allows you to **add rounded corners to elements**!
- This property can have **from one to four values**.
- **One value border-radius: 10px; means that** the **value applies to all four corners**, which are **rounded equally**.

style1.css

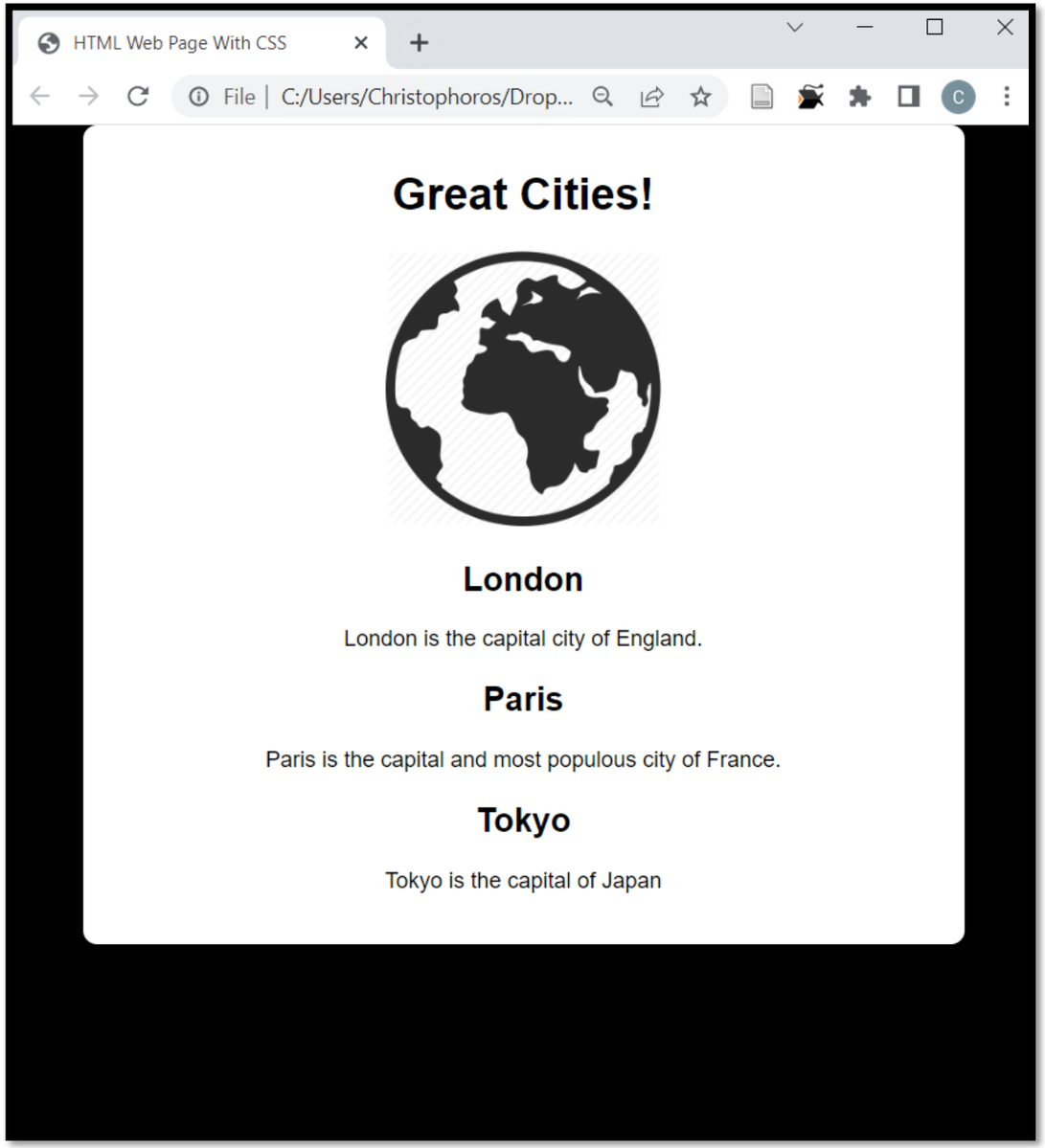
```
body {
    width: 600px;
    background-color: white;
    text-align: center;
    font-family: Arial, Helvetica, sans-serif;
    line-height: 1.2;
    margin: 0 auto;
    padding: 10px 20px 20px 20px;
    border-radius: 10px;
}
```

This is the desired

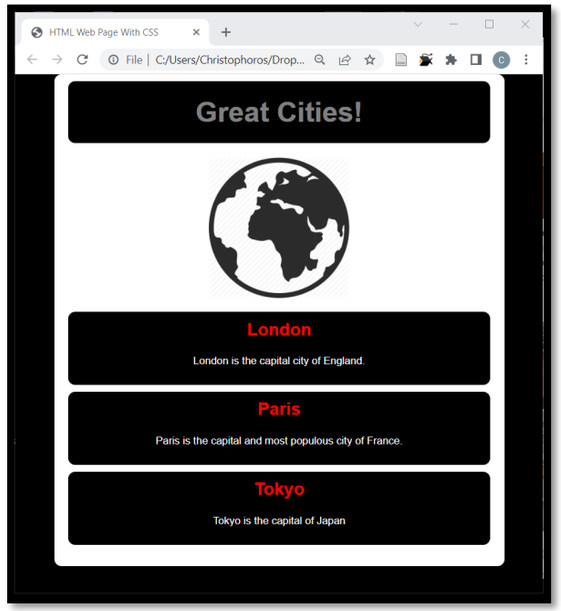


Applying CSS in an HTML document – An example

- Our work in progress, with the previous CSS rules applied will look like this….



This is the desired

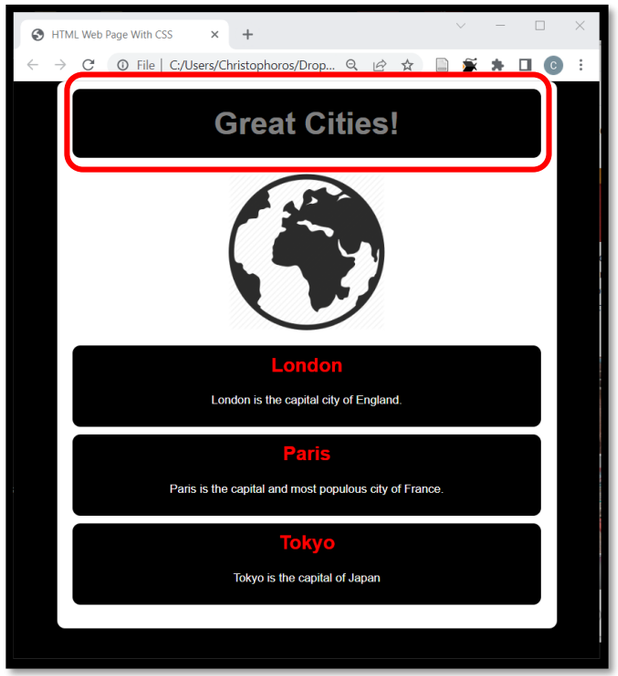


Applying CSS in an HTML document – An example

- Lets continue from styling the <h1> element, which **includes** the title that will be displayed on the top web page….

style1.css

```
h1 {
    font-size: 40px;
    background-color: black;
    color: rgba(255, 255, 255, 0.5);
    padding: 20px 20px;
    margin: 0;
    border-radius: 10px;
}
```



Applying CSS in an HTML document – An example

style1.css

```css
h1 {
    font-size: 40px;
    background-color: black;
    color: rgba(255, 255, 255, 0.5);
    padding: 20px 20px;
    margin: 0;
    border-radius: 10px;
}
```

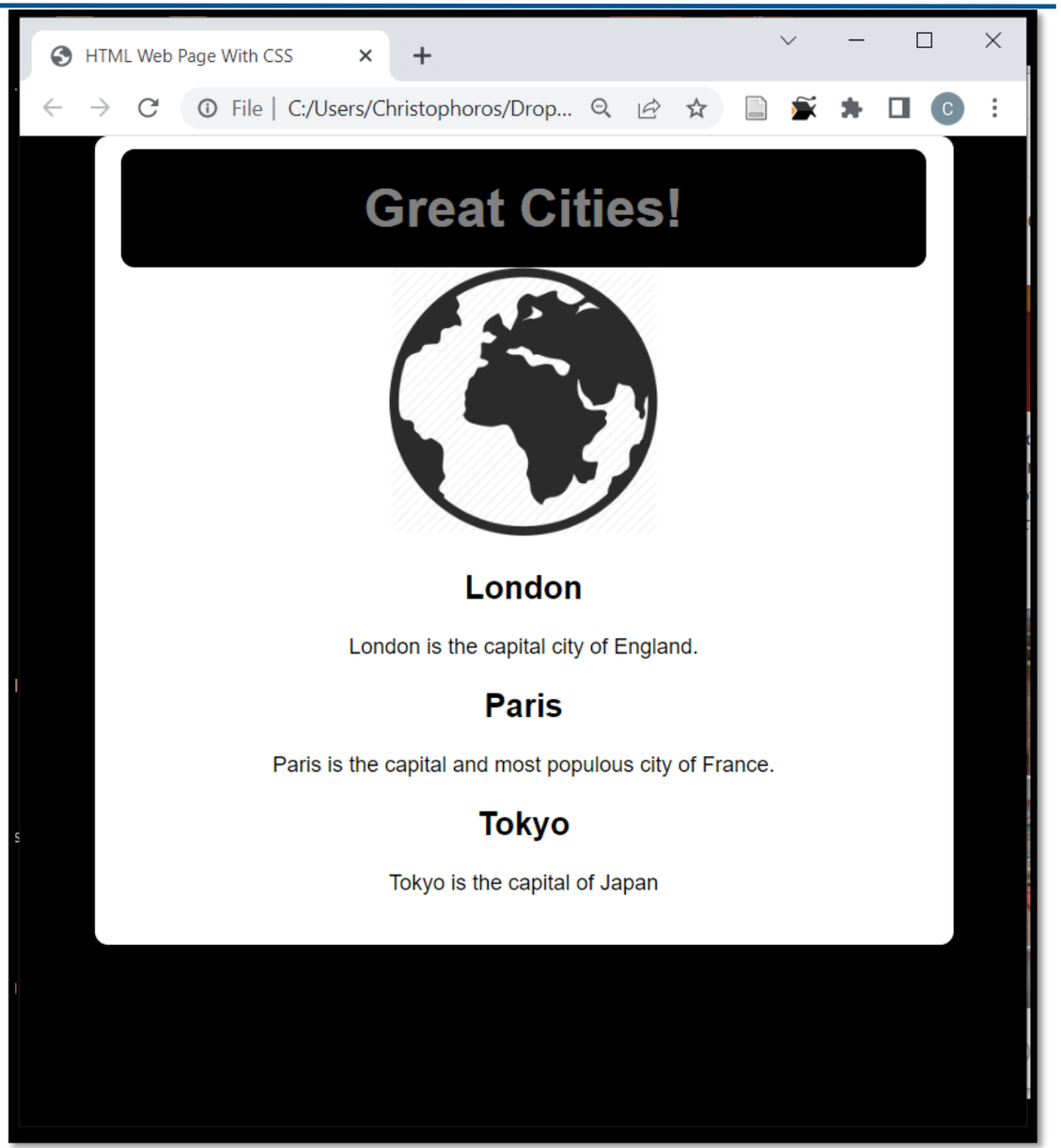
color: rgba(255, 255, 255, 0.5);

The text **color** of <h1> elements will be **white with transparency 50%**!

The "a" in rgba (), which has the value 0.5 in the declaration provided in the CSS ruleset, stands for **alpha channel** and is a **transparency.**

Applying CSS in an HTML document – An example

- Our work in progress, with the h1 rule applied will now look like this….



Applying CSS in an HTML document – An example

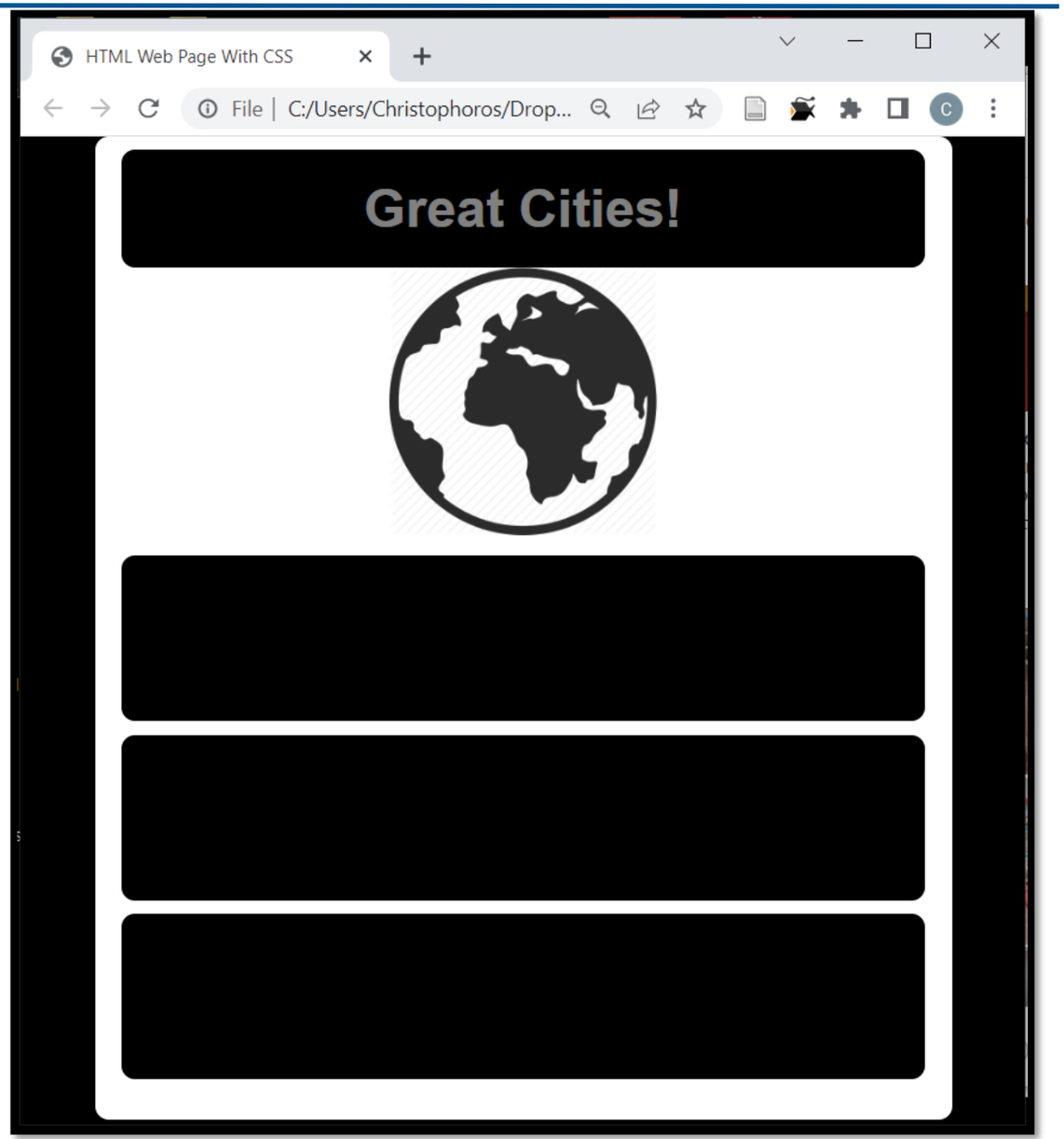
- Lets continue from styling the ** ** elements with **class** name **cities**….

style1.css

```
div.cities {
    background-color: black;
    margin: 10px 0;
    padding: 10px;
    border-radius: 10px;
}
```

Applying CSS in an HTML document – An example

- Our work in progress, with the **div.cities CSS ruleset** applied will now look like this….



Applying CSS in an HTML document – An example

```
div.cities {
    background-color: black;
    margin: 10px 0;
    padding: 10px;
    border-radius: 10px;
}
```

style1.css

```
div.cities {
    background-color: black;
    padding: 10px;
    border-radius: 10px;
}
```

Note: With **margin** attribute we **add space** between the elements, **so as not to sit flush against each other**.

In the example **at the right**, we **did not add margins** between the **div.cities** elements….

Applying CSS in an HTML document – An example

style1.css

```
img {
    margin-top: -30px;
}
```

Note: Also **margins** can be **negative**.

See how the image flows on top of <h1> element

Applying CSS in an HTML document – An example

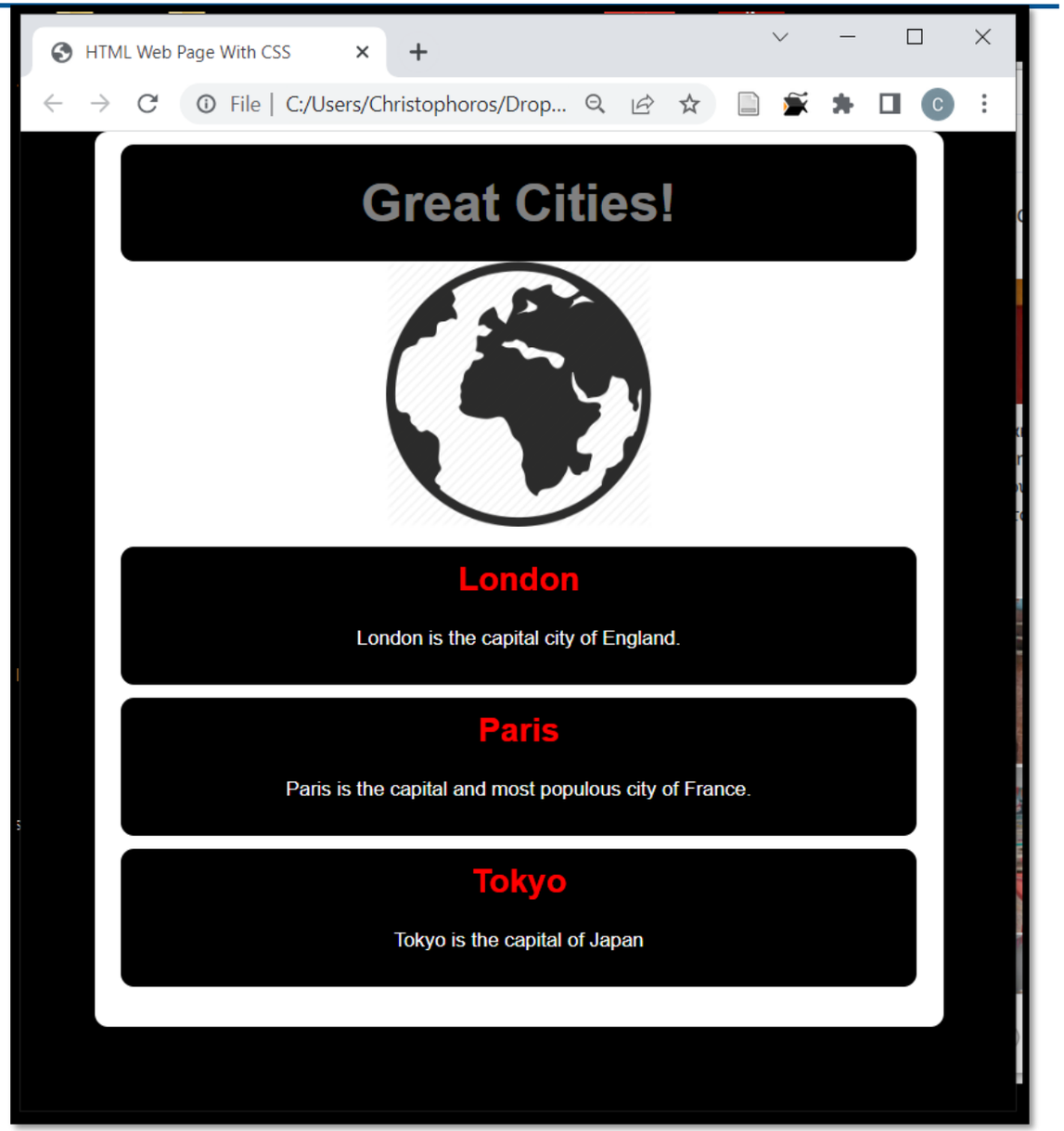
- Lets continue from styling the **<h2>** and **<p>** elements….

style1.css

```
h2 {
    margin-top: 0;
    font-size: 25px;
    color: red;
}

p {
    font-size: 15px;
    color: white;
}
```

Applying CSS in an HTML document – An example

- Our work in progress, with the h2 and p **CSS rulesets** applied will now look like this….



Applying CSS in an HTML document – An example

- And finally lets **put some margins around the image** to make it look better….

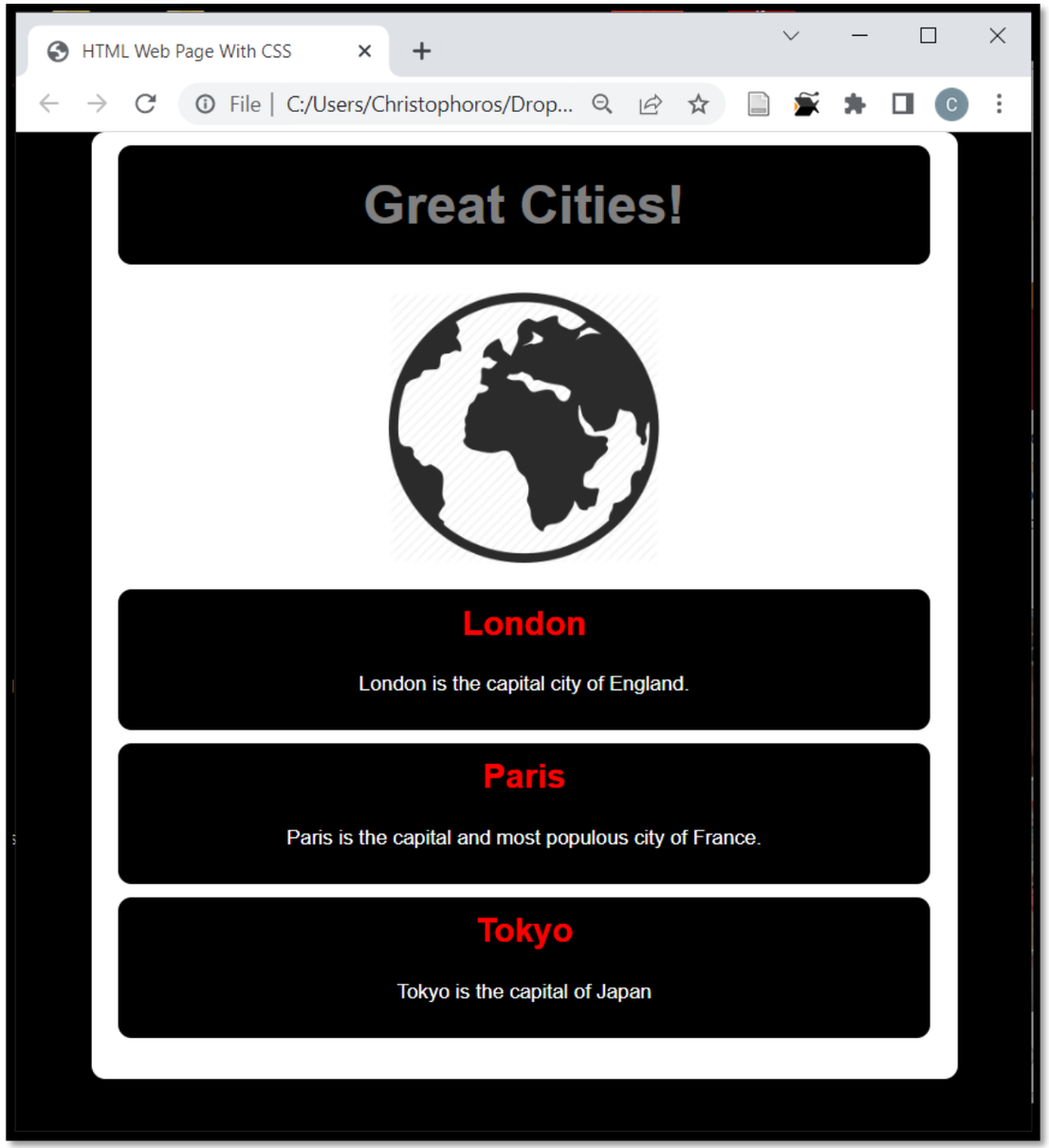
style1.css

```
img {
    margin: 20px auto;
    display: block;
}
```

- **Note:** In case your image is not centered **(with margin: 20px auto;)**, remember that the <body> is a **block element**, meaning it takes up the **full width** of the page. The **margin** applied to a **block element** will be **respected by other block elements** on the page.
- In **contrast**, **images** are mainly **inline elements.** For the **auto** margin trick to work on this image, we must **give it block-level behavior** using **display: block;**

Applying CSS in an HTML document – An example

- With this, **our work is done** and will **now look like this**….



CSS Custom Properties (CSS Variables)

- **Complex websites** have **very large amounts** of CSS, often with **a lot** of **repeated values**.
- For example, the **same color** might be **used in hundreds of different places**, **requiring global search** and **replace if that color needs to change**.
- **CSS custom properties** or **CSS variables** **allow a value** to be **stored in one place**, then **referenced** (using the **var() function**) in **multiple other places**.

CSS Custom Properties (CSS Variables)

- **Declaring** a **custom CSS property** is done using a **custom** property name that begins with a double hyphen (--) and a property value that can be **any valid CSS value**. Like any other property, this is **written inside a ruleset.**
- Note that the **selector given** to the **ruleset** defines the **scope that the custom property can be used within**.

```
:root {
    --main-color: brown;
}
```

- A common **best practice** is to define **CSS custom properties** on the **:root pseudo class**, so that it can be **applied globally across your HTML document.**

CSS Custom Properties (CSS Variables)

- You can **use** the **custom property value by specifying** your **custom property name inside** the **var()** function.

```
:root {
    --main-color: brown;
}


div.cities {
    background-color: var(--main-color);
    margin: 10px 0;
    padding: 10px;
    border-radius: 10px;
}
```

The **:root** CSS pseudo-class matches the root element of a tree representing the **document**. In HTML, :root represents the <html> element and is identical to the selector **html**, except that **its specificity is higher**.

Mobile vs Desktop browsers

- **Unless directed otherwise** via **HTML** or **CSS** indications, **mobile browsers render** web pages **at a desktop screen width** (~1000px), then **"zooms out" until the entire page fits on screen.**
- That's why you sometimes **get web pages** with **teeny-tiny font** on your **phone**: these webpages **have not added support for mobile**.

Mobile vs Desktop browsers

- To **prevent phone browsers** from **rendering the page at desktop width** and **zooming out**, **use the meta viewport tag**:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- This belongs in the **<head>** section of your HTML, where the <title>, <link>, and **other metadata elements** are included.

Mobile vs Desktop browsers: Meta viewport tag

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- name="viewport" → Browser, I am going to tell you how I want the viewport to look."
- content="width=device-width, initial-scale=1":
  - width=device-width → The viewport's width should always start at the device's width.
  - initial-scale=1 → Start at zoom level of 100%.

You should pretty much always include this tag in your HTML!!

Mobile vs Desktop browsers: Meta viewport tag

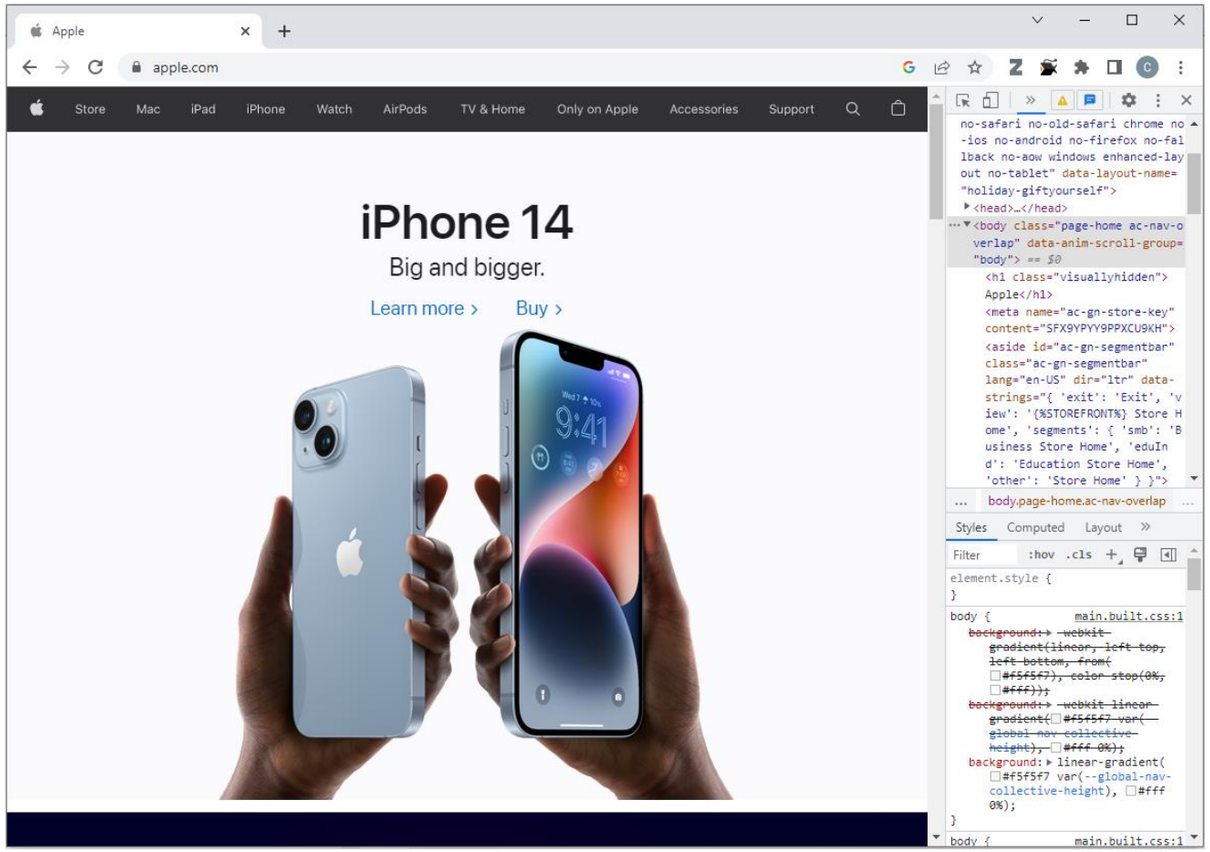
- The meta **viewport** **tag gets us almost all the way there**, but a **few more adjustments need to be made** to **make web pages** that **render well** on **all screen sizes** and **resolutions** **while ensuring good usability**.
- For more details on Responsive Web Design* see this link.

* **Responsive web design** is about creating web sites which **automatically adjust themselves** to **look good on all devices**, from small phones to large desktops.

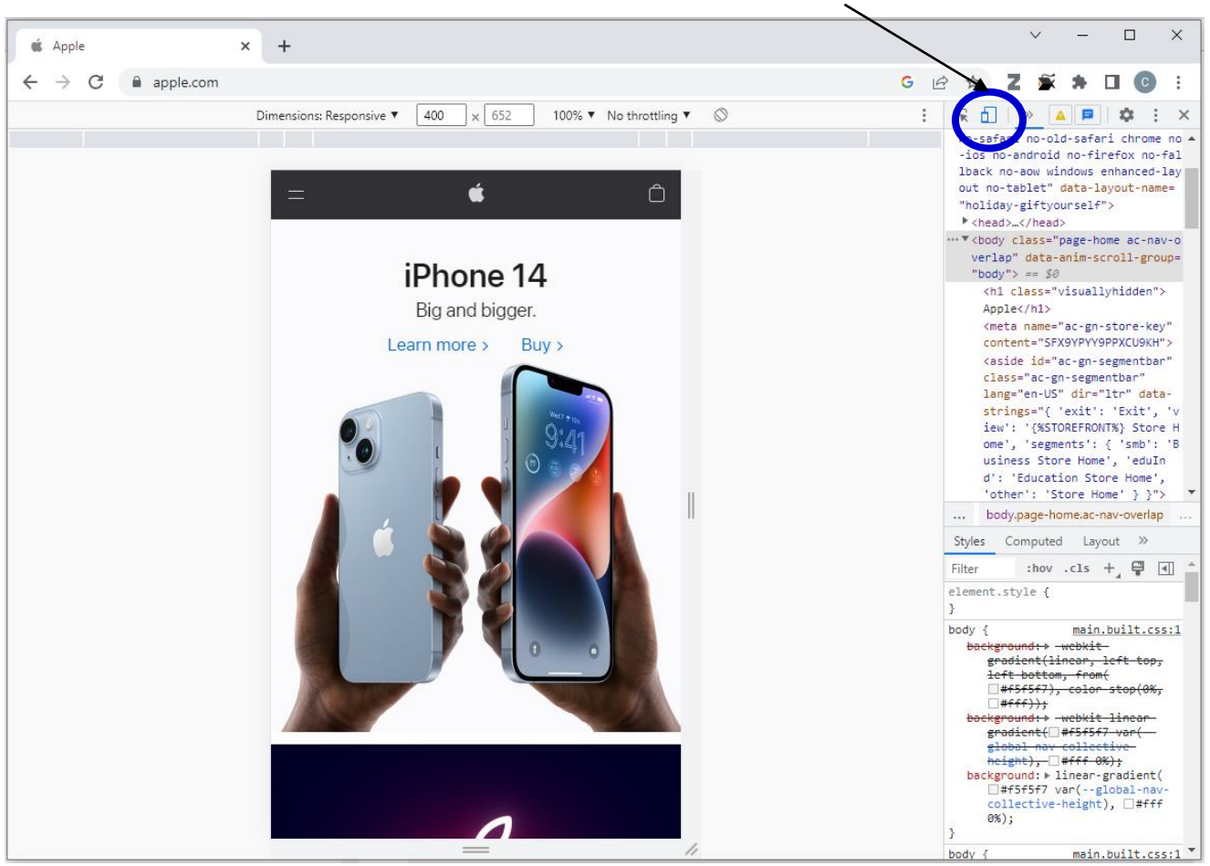
Mobile vs Desktop browsers

- To check **how your web page is rendered** on **mobile devices**, you can **simulate** a **web page** in a **mobile layout** via **Chrome device mode**.

Right Click → Inspect ….



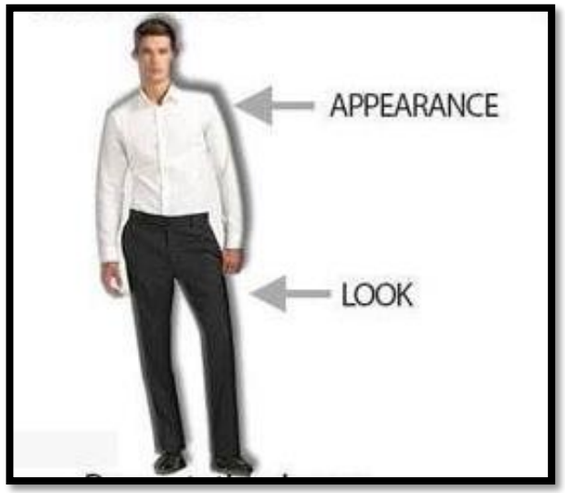
Select



Bootstrap

- **Bootstrap** is the most popular **open-source front-end framework** for **faster** and **easier** **web development**. **Bootstrap 5** is the **newest version**.
- Bootstrap includes **HTML** and **CSS** **based** **design templates** for **text/typography, forms, buttons, tables, navigation, modals, image carousels** and **many other**, as well as **optional** **JavaScript plugins**.
- Bootstrap also gives you the **ability** to easily **create responsive designs.**

Note: Bootstrap 3 and **Bootstrap 4** is **still supported** by the team for critical bugfixes and documentation changes, and it is perfectly safe to continue to use them. However, **new features will NOT be added** to them.



HTML + CSS

HTML + CSS + Bootstrap

Where to Get Bootstrap 5?

There are **two ways** to start **using Bootstrap 5** on **your own web site**.

- Download and host Bootstrap 5 yourself. Go to https://getbootstrap.com/, and follow the instructions there.
- If you **don't want** **to** **download** **and** **host** Bootstrap 5 yourself, you can **include it from a CDN** (Content Delivery Network).

```
<!-- Latest compiled and minified CSS -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">

<!-- Latest compiled JavaScript -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
```

Where to Get Bootstrap 5?

Advantages of using the Bootstrap 5 from a CDN:

- Many users **already have downloaded Bootstrap 5** from **jsDelivr when visiting another site**.
- As a result, it will be **loaded from cache** when **they visit your site**, which leads to **faster loading time**.
- Also, most CDN's will make sure that once a user requests a file from it, it will be **served from the server closest to them**, which also leads to **faster loading time**.

Create Your First Web Page With Bootstrap 5

1. Add the HTML5 doctype

- Bootstrap 5 uses HTML elements and CSS properties that require the HTML5 doctype.
- Always include the HTML5 doctype at the beginning of the page, along with the lang attribute and the correct title and character set:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Bootstrap 5 Example</title>
    <meta charset="utf-8">
  </head>
</html>
```

Create Your First Web Page With Bootstrap 5

2. Bootstrap 5 is mobile-first*

- Bootstrap 5 is designed to be **responsive to mobile devices**. Mobile-first styles are part of the core framework.
- To ensure proper rendering and touch zooming, add the following

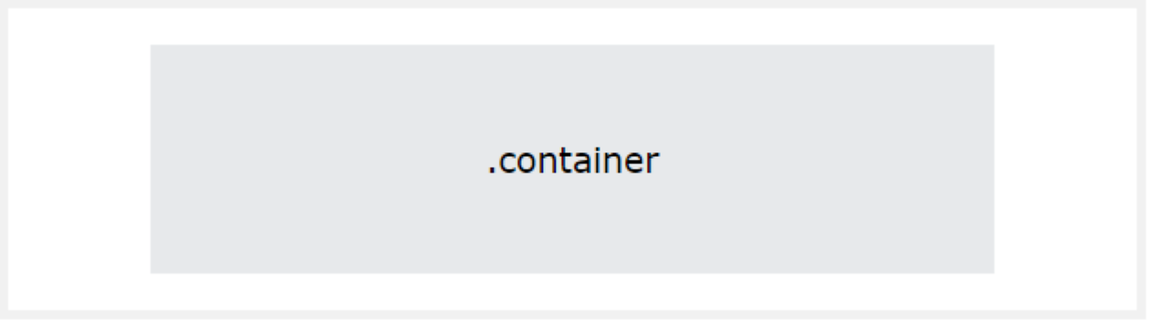
```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

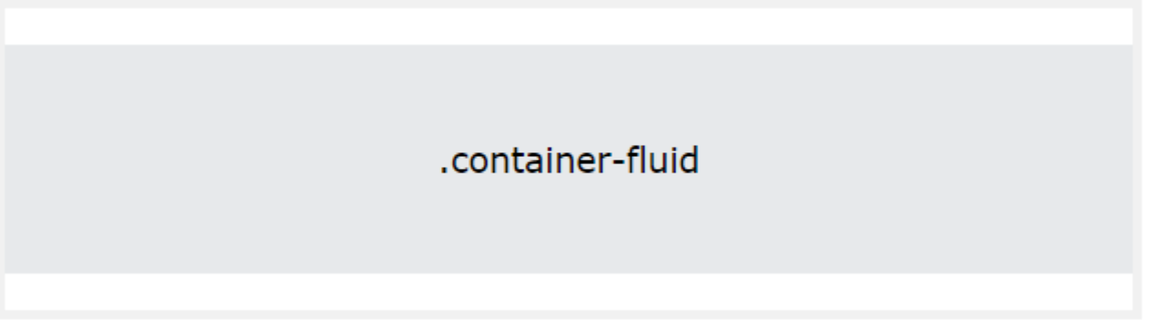
* A **"mobile-first"** approach involves **designing a desktop site starting with the mobile version, which is then adapted to larger screens** (contrary to the traditional approach of starting with a desktop site and then adapting it to smaller screens).

Create Your First Web Page With Bootstrap 5

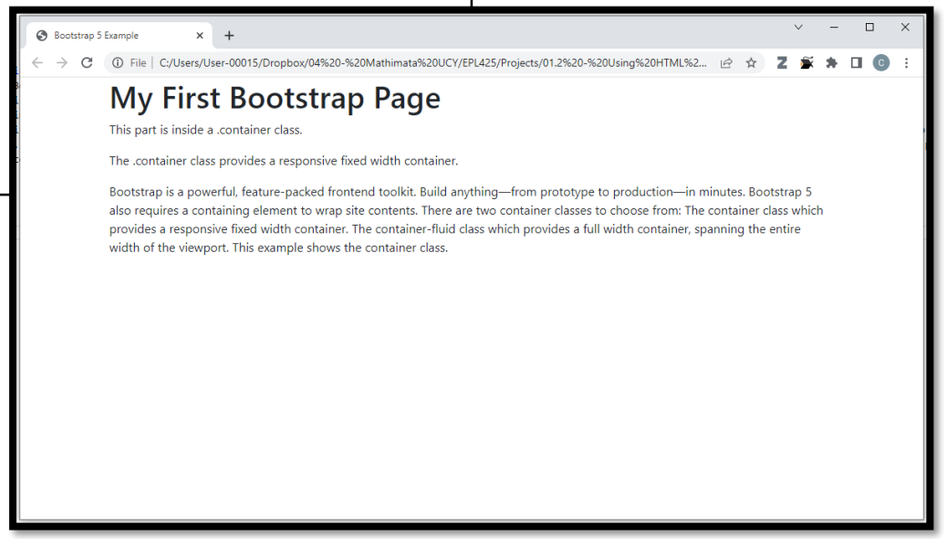
3. Containers

- Bootstrap 5 also requires a **containing element** to wrap site contents. There are **two container classes** to choose from:
  - The **container** class provides a responsive **fixed width container**
  - The **container-fluid** class provides a **full width container**, spanning the entire width of the viewport.

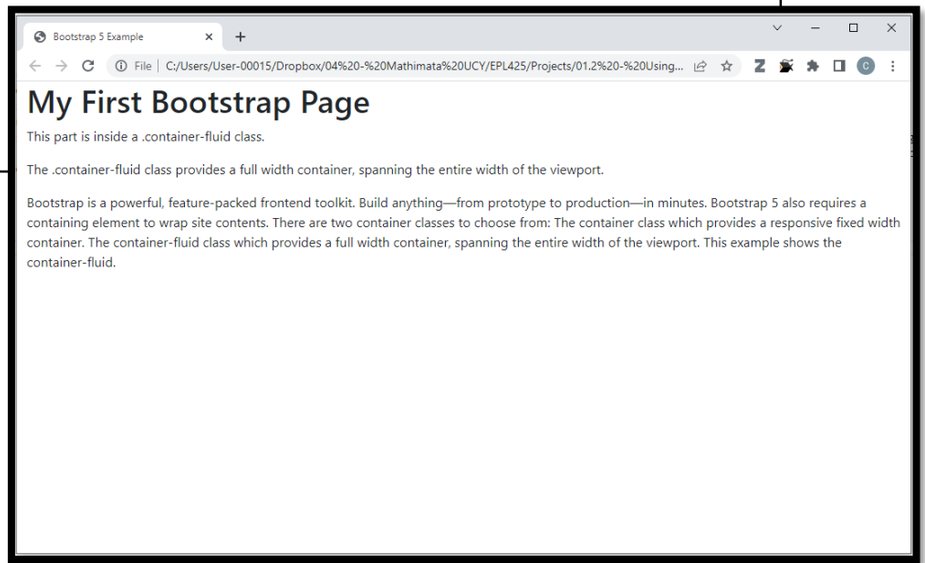




```html
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Bootstrap 5 Example</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
</head>

<body>
    <div class="container">
        <h1>My First Bootstrap Page</h1>
        <p>This part is inside a .container class.</p>
        <p>The .container class provides a responsive fixed width container.</p>
        <p>Bootstrap is a powerful, feature-packed frontend toolkit. Build anything—from prototype to
production—in minutes. Bootstrap 5 also requires a containing element to wrap site contents. There are two
container classes to choose from: The container class which provides a responsive fixed width container.
The container-fluid class which provides a full width container, spanning the entire width of the
viewport. This example shows the container class. </p>
    </div>
</body>

</html>
```



```html
<!DOCTYPE html>
<html lang="en">

<head>
    <title>Bootstrap 5 Example</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
</head>

<body>
    <div class="container-fluid">
        <h1>My First Bootstrap Page</h1>
        <p>This part is inside a .container-fluid class.</p>
        <p>The .container-fluid class provides a full width container, spanning the entire width of the viewport.</p>
        <p>Bootstrap is a powerful, feature-packed frontend toolkit. Build anything—from prototype to production—in minutes.
Bootstrap 5 also requires a containing element to wrap site contents. There are two container classes to choose from: The
container class which provides a responsive fixed width container. The container-fluid class which provides a full width
container, spanning the entire width of the viewport. This example shows the container-fluid class. </p>
    </div>
</body>

</html>
```



Bootstrap

- For more details on learning bootstrap check the following links:
  - https://www.w3schools.com/bootstrap5/index.php
  - https://getbootstrap.com/docs/5.3/getting-started/introduction/

You will also learn how to use Bootstrap during the Labs!!!

Flex Layout

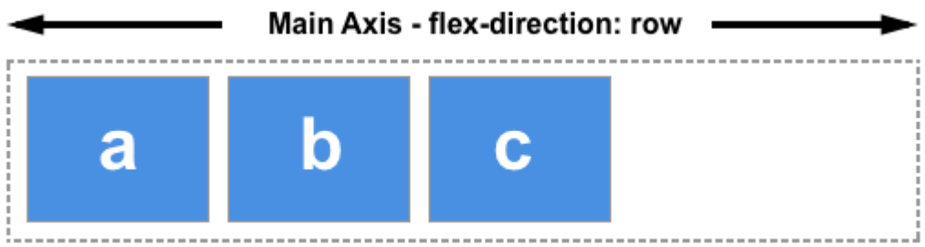
- CSS Layouts we saw so far…

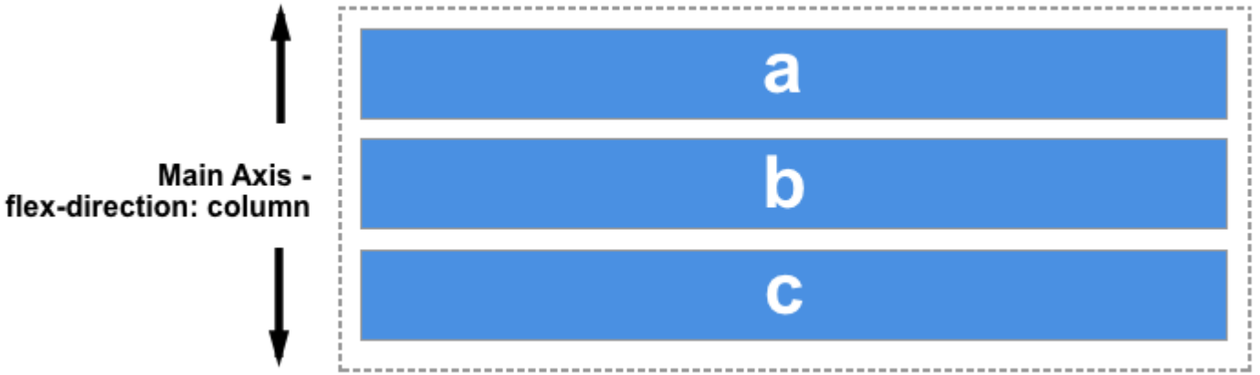
Block layout

Inline layout

Flex Layout

- To achieve **more complicated layouts**, we can **enable** a **different kind** of **CSS layout rendering mode**: **Flex layout.**
- **Flex layout** defines a **special set of rules** for **laying out** items in **rows** or **columns**.

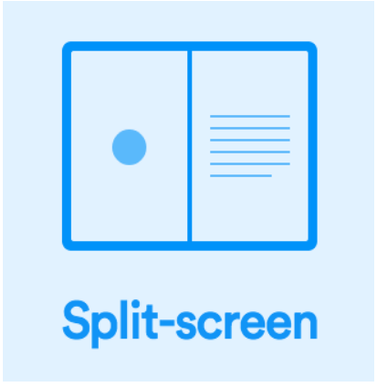


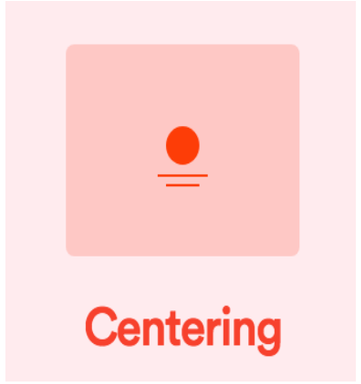


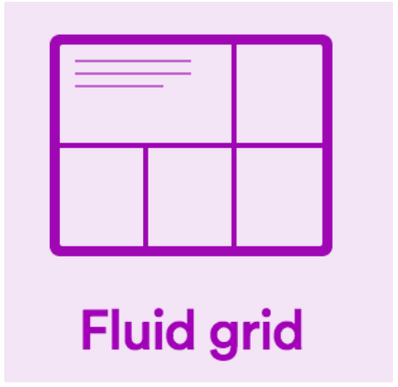
Flex layout

Flex layout solves all sorts of problems.

- Here are some examples of layouts that are easy to create with flex layout (and really difficult otherwise).







For more details on flex layout see this link!

Ερωτήσεις?