Internet Technologies

Introduction to HTML and CSS – Part 2

Block, inline-block, class, id, css pseudoclasses

University of Cyprus
Department of Computer Science

Types of HTML elements

Each HTML element is categorized by the HTML spec into one of the following 3 categories:

1. **block**: large blocks of content, **has height and width**

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

2. **inline**: small amount of content, **no height or width**

   `<a>, <em>, <strong>,<br>`

   a. **inline block**: inline content with height and width

      `<img>`

3. **metadata**: information about the page, usually not visible

   `<title>, <meta>`

Block elements

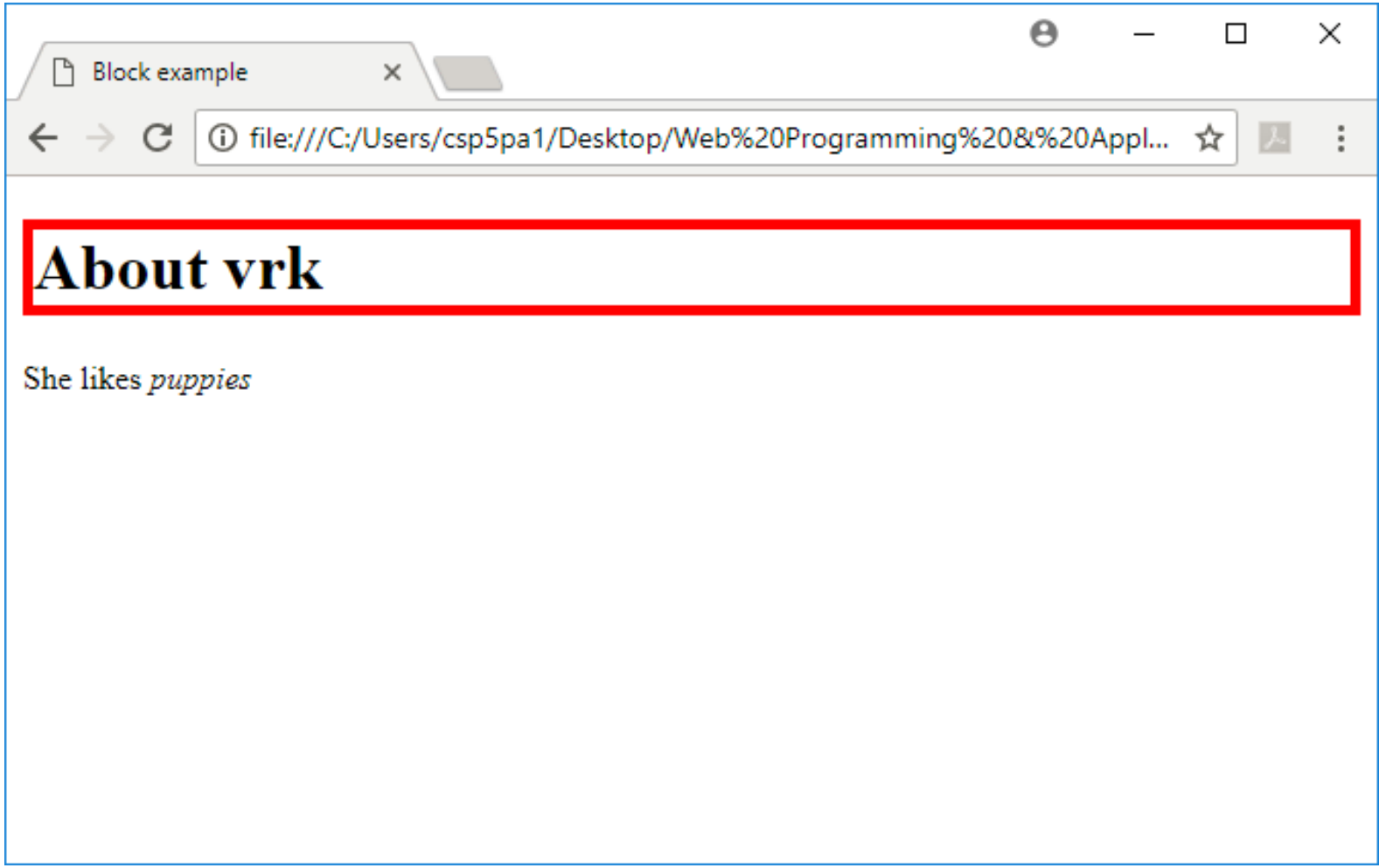
- Examples:
  `<p>, <h1>, <blockquote>, <ol>, <ul>, <table>`
- Take up the full width of the page (**flows top to bottom**)
- Have a `height` and `width`
- Can have block or inline elements as children

Example: Block

Q: What does this look like in the browser?

```
h1 {
    border: 5px solid red;
}
```

A:


Block example
file:///C:/Users/csp5pa1/Desktop/Web%20Programming%20&%20Appl...
About vrk
She likes puppies

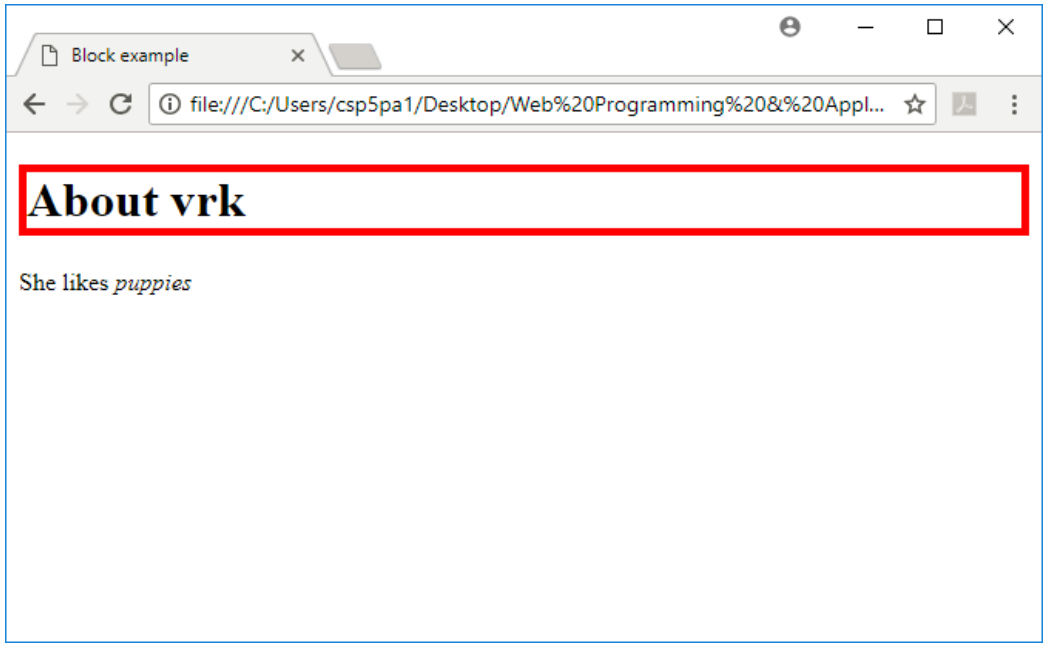
Block-level:

extends the full width of the page

```
h1 {
    border: 5px solid red;
}
```

```
<h1>About vrk</h1>
<p>
  She likes <em>puppies</em>
</p>
```

- <h1> is block-level, so it extends the full width of the page by default
- Note how block-level elements (h1, p) flow top to bottom

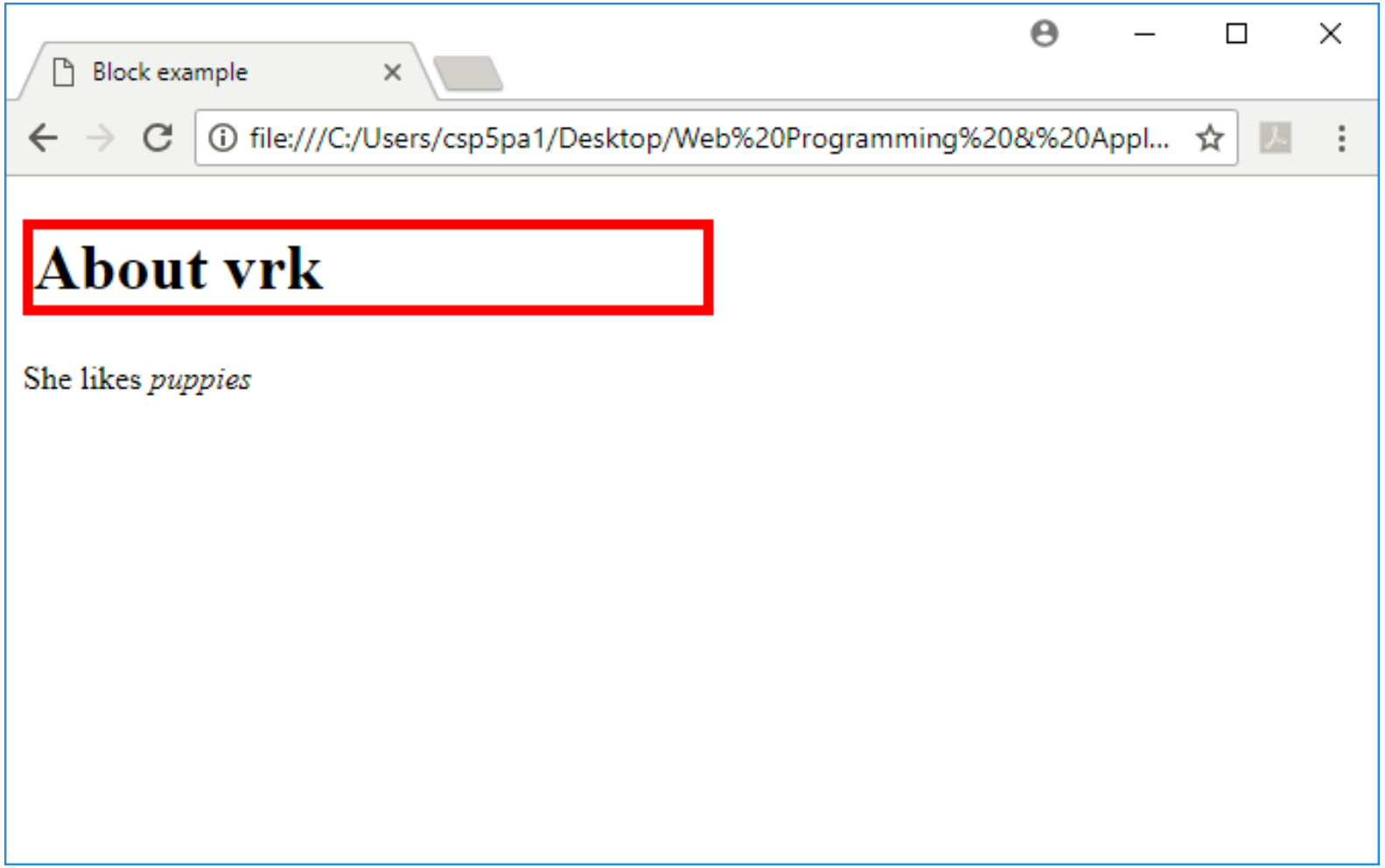


Q: What does this look like in the browser?

```
h1 {
    border: 5px solid red;
    width: 50%;
}
```


```
<h1>About vrk</h1>
<p>
  She likes <em>puppies</em>
</p>
```

A:


Block example
file:///C:/Users/csp5pa1/Desktop/Web%20Programming%20&%20Appl...
About vrk
She likes puppies

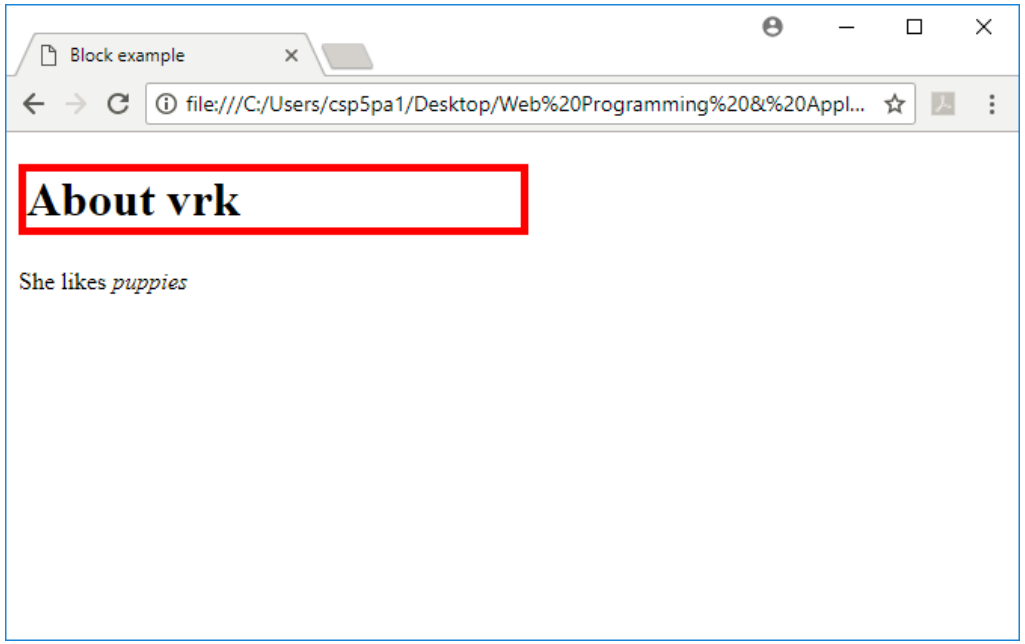
Block-level:

width can be modified

```
h1 {
    border: 5px solid red;
    width: 50%;
}
```

```
<h1>About vrk</h1>
<p>
  She likes <em>puppies</em>
</p>
```

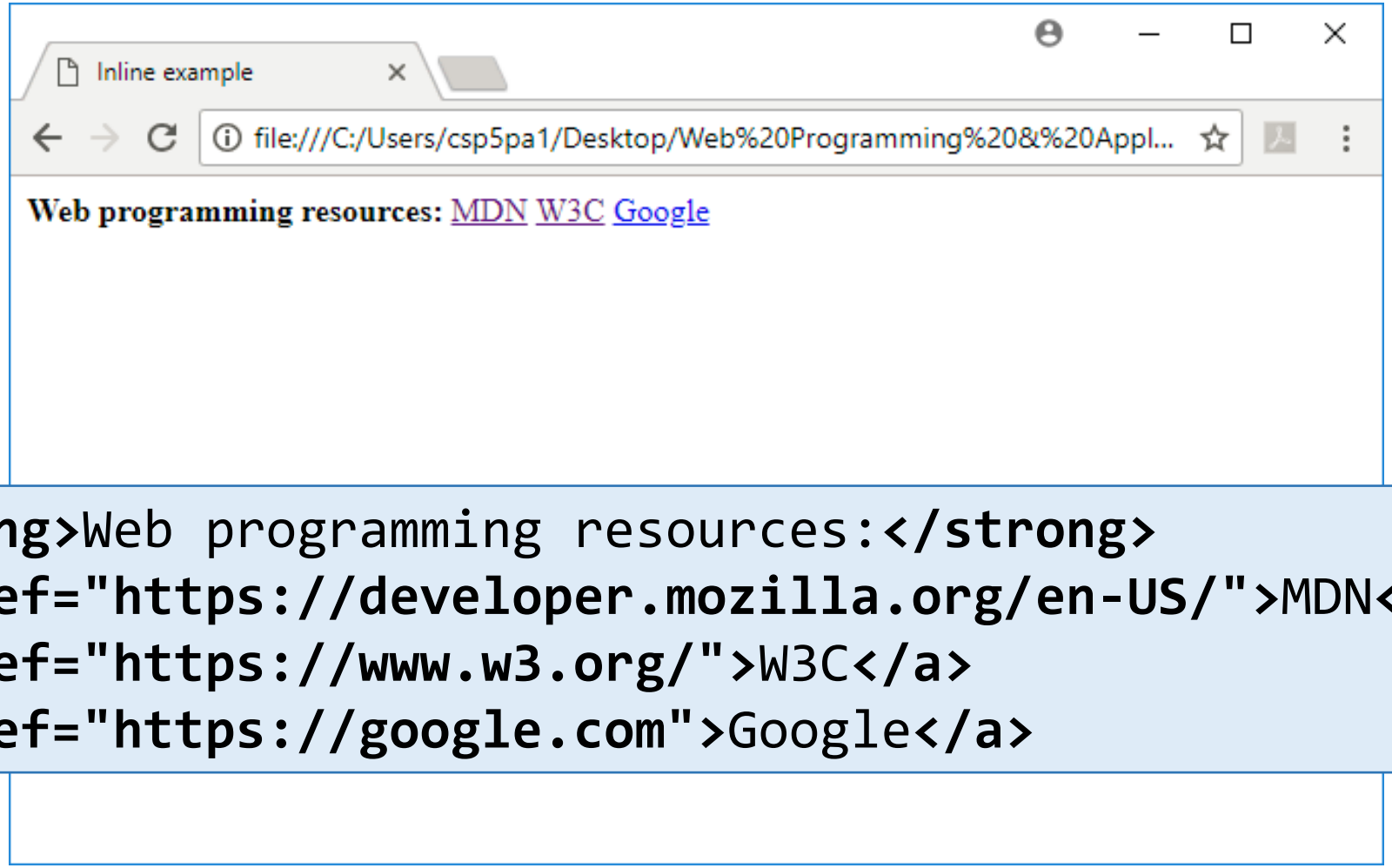
- <h1> is block-level, so its width can be modified
- Block-level elements still flow top to bottom



Inline elements

- Examples:
  `<a>, <em>, <strong>, <br>`
- Take up only as much width as needed (flows left to right)
- Cannot have `height` and `width`
- Cannot have a block element child
- Cannot be positioned (i.e. CSS properties like `float` and `position` do not apply to inline elements)
  - Must position its containing block element instead
- Only left & right `padding`/`margin` have effect on inline element (we will speak later about that)

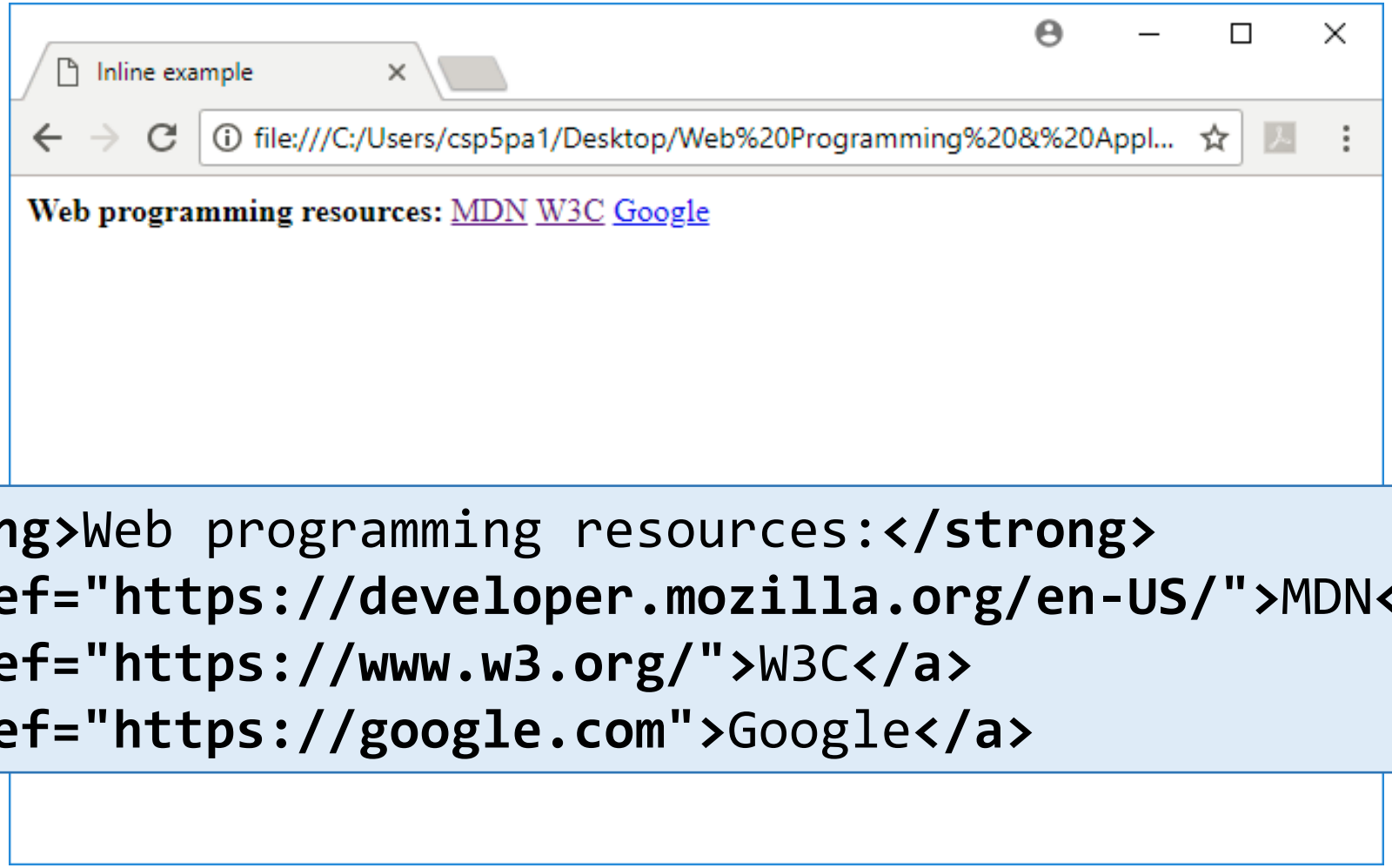
Example: Inline




```
<strong>Web programming resources:</strong>
<a href="https://developer.mozilla.org/en-US/">MDN</a>
<a href="https://www.w3.org/">W3C</a>
<a href="https://google.com">Google</a>
```

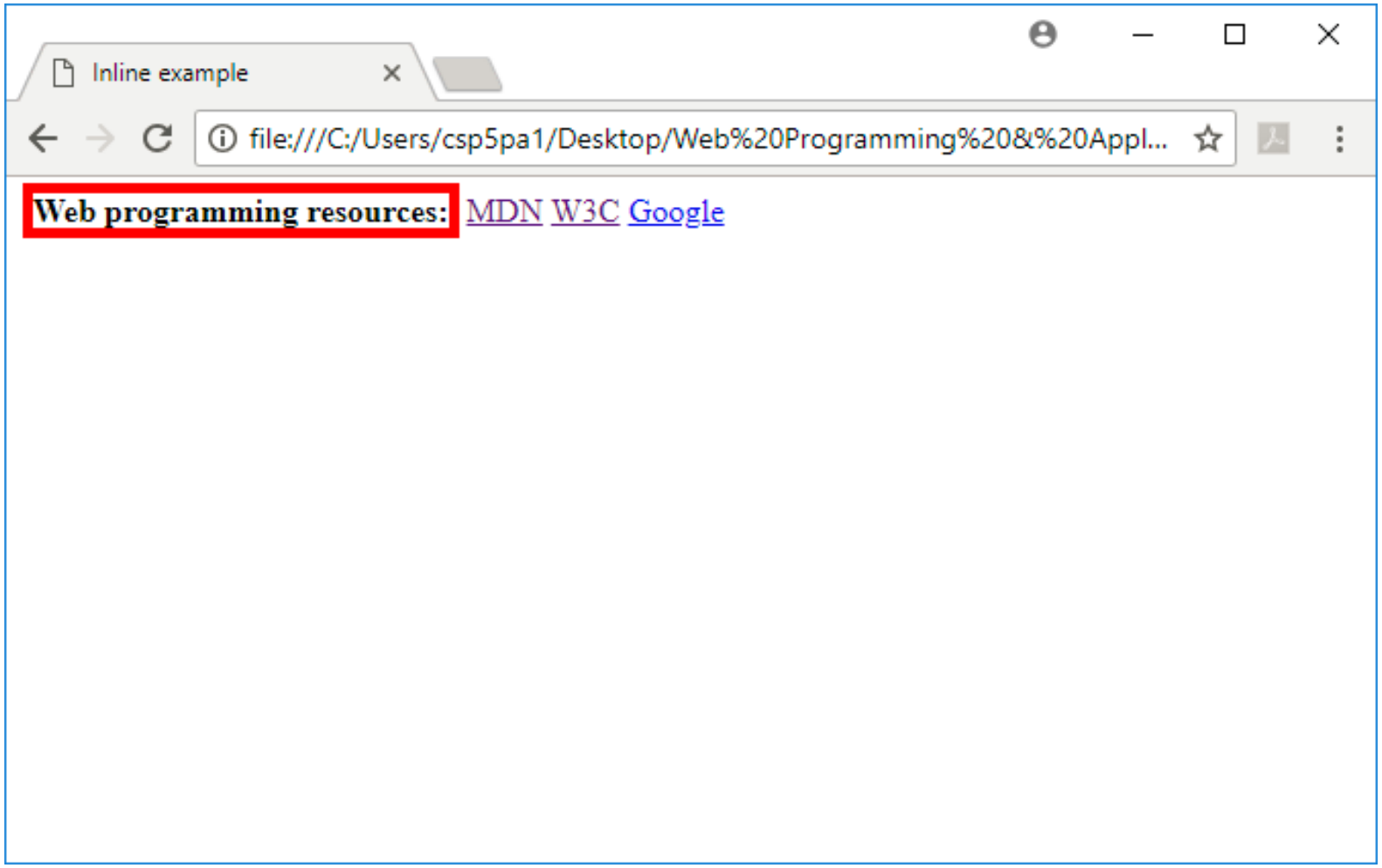
Q: What does this look like in the browser?

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




```
<strong>Web programming resources:</strong>
<a href="https://developer.mozilla.org/en-US/">MDN</a>
<a href="https://www.w3.org/">W3C</a>
<a href="https://google.com">Google</a>
```

A:

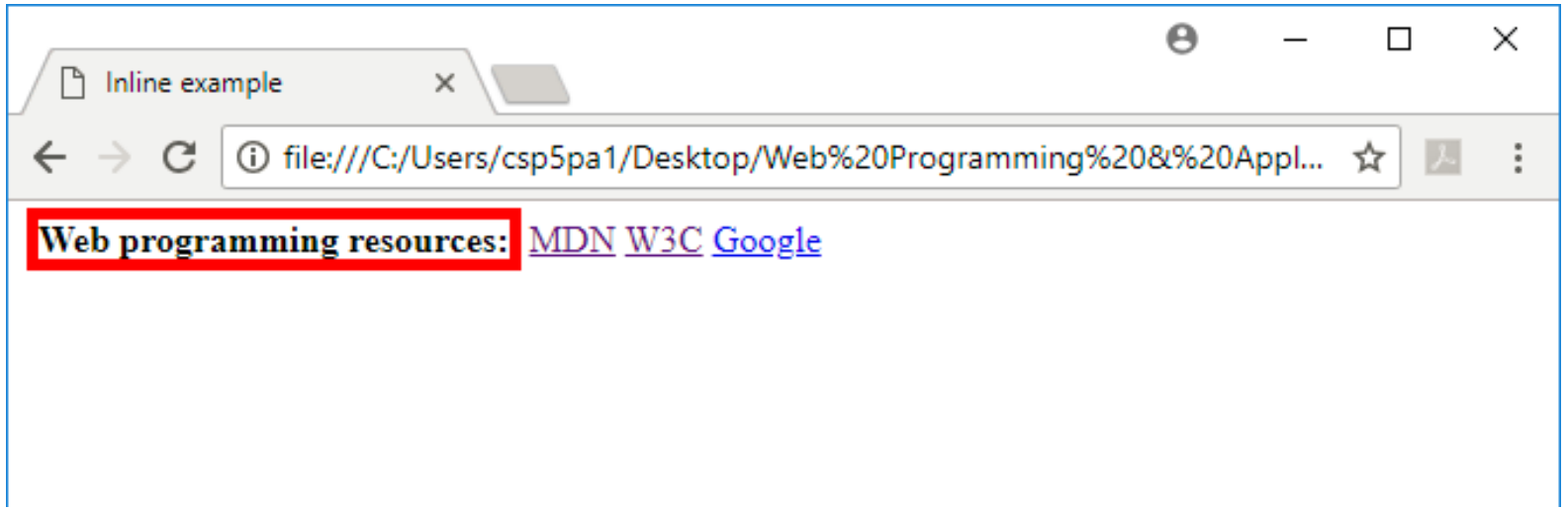

Inline example
file:///C:/Users/csp5pa1/Desktop/Web%20Programming%20&%20Appl...
Web programming resources:
MDN
W3C
Google

Inline elements ignore `width`

width cannot be modified

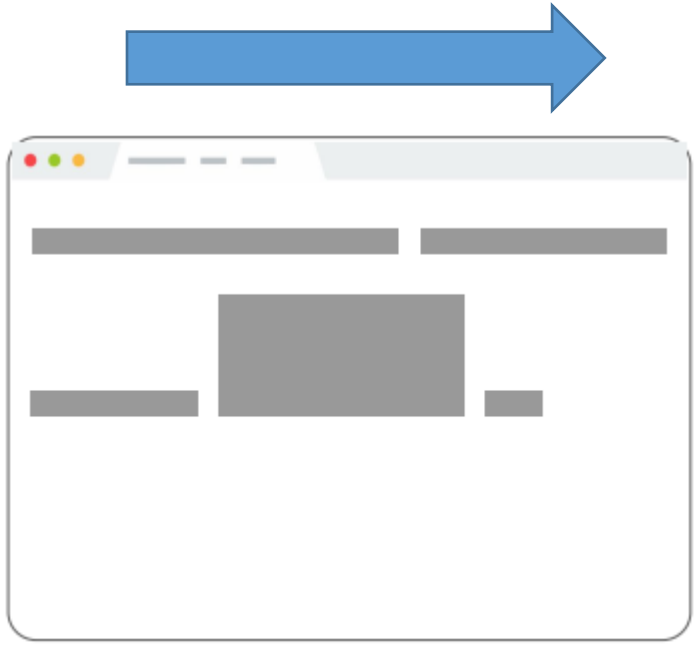
```
strong {
    border: 5px solid red;
    width: 1000px;
    /* Will not work;
       strong is inline! */
}
```

```
<strong>Web programming resources:</st
<a href="https://developer.mozilla.org
<a href="https://www.w3.org/">W3C</a>
<a href="https://google.com">Google</a
```



inline-block

- Examples:
  `<img>`, & any element with `display: inline-block;`
- Width is the size of the content,
  i.e. it takes only as much space as needed (flows left to right)
- Can have `height` and `width`
- Can have a block element as a child
- Can be positioned (i.e. CSS properties like `float` and `position` apply)
- `Padding/margin` have effect on all sides of `inline-block` element



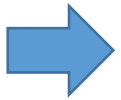
Example: Inline-block

Q: What does this look like in the browser?

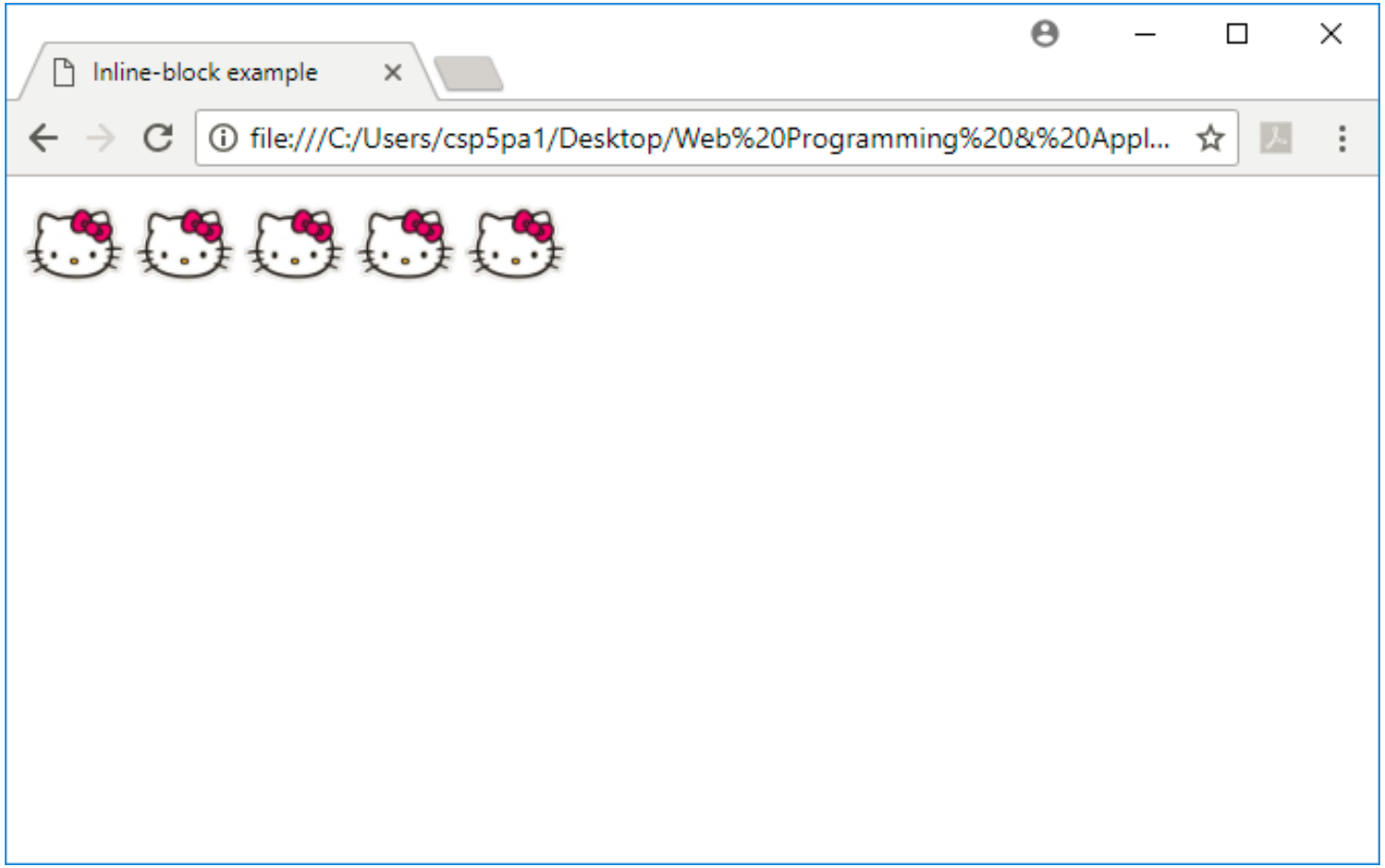
```
img {
    width: 50px;
}
```

```
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
```

```
<img src="http://i.imgur.com/WJToVGv.jpg" />
```



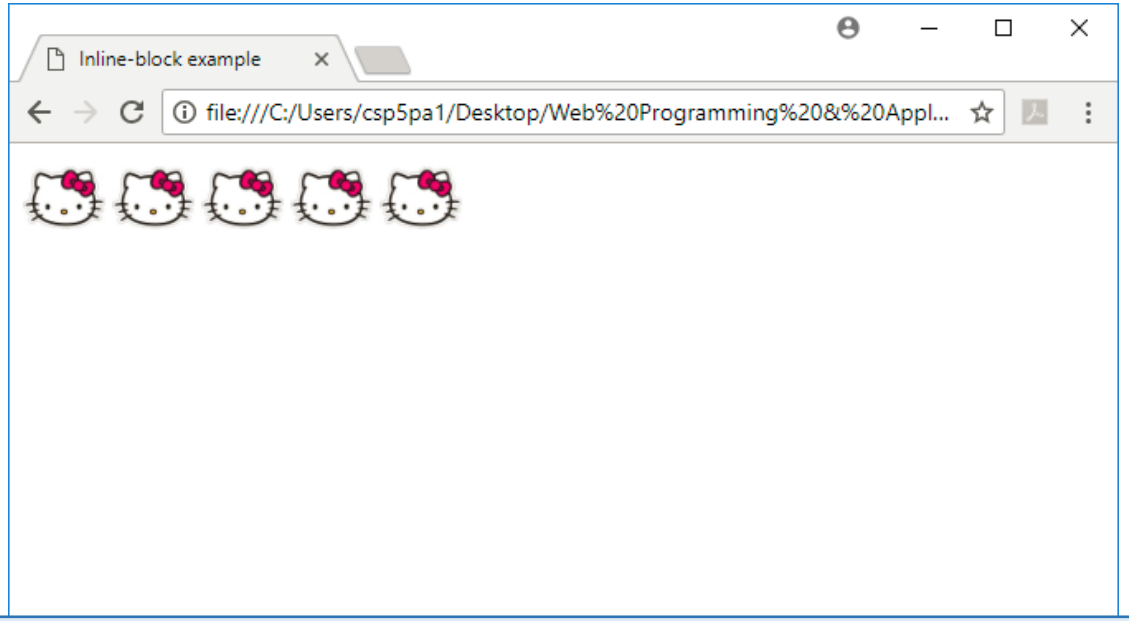
A:


Inline-block example
file:///C:/Users/csp5pa1/Desktop/Web%20Programming%20&%20Appl...

Inline-block

Has width and height; flows left to right

- Can set `width` on inline-block element, so image width is set to 50px.
- `inline-block` flows left to right, so images are right next to each other.




```
img {
    width: 50px;
}
```

```
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
<img src="http://i.imgur.com/WJToVGv.jpg" />
```

The `display` CSS property

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

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

```
a {
   display: block;
}
```

- Possible values for display:
  - `block`
  - `inline`
  - `inline-block`
  - some others: see here

Review

1. **block**: flows **top-to-bottom**; **has** `height` and `width`

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

2. **inline**: flows **left-to-right**, **does not have** `height` or `width`

   `<a>, <em>, <strong>,<br>`

   a. **inline block**: flows **left-to-right**, **has** `height` and `width` equal to size of the content

      `<img>`

Questions?

Moral of the story:

If your CSS isn't working, see if you're trying to apply block-level properties to inline elements

Example: h1 vs strong

<h1> heading is on a line of its own, and is not. -- Why?

Because h1 is a block-level element, and strong is an inline-level element

```
<h1>CS 425: Internet Technologies</h1>
<strong>Announcements</strong>
4/3: Homework 0 is out!<br/>
4/3: Office hours are now posted.<br/>
```

CS 425: Internet Technologies

Announcements 4/3: Homework 0 is out!
4/3: Office hours are now posted.

```
<h1>CS 425: Internet Technologies</h1>
<strong>Announcements</strong><br/>
4/3: Homework 0 is out!<br/>
4/3: Office hours are now posted.<br/>
```

CS 425: Internet Technologies

Announcements
4/3: Homework 0 is out!
4/3: Office hours are now posted.

Example: `text-align` mystery

- Set `text-align: center;` on the `<h1>` and works!
- Set `text-align: center;` on the `<a>` tag but does not work -- Why?

```
h1 { /* works! */
   text-align: center;
}

a { /* fails :( */
   text-align: center;
}
```

CS 425: Internet Technologies

Announcements
4/3: Homework 0 is out!
4/3: Office hours are now posted.

View Course Contract

Let's try looking at the MDN description of text-align ...

Example: `text-align` mystery

- The `text-align` CSS property describes how content like text is aligned within its parent **block** element. `text-align` does not control the alignment of block elements, only their inline content.

Initial value	`start`, or a nameless value that acts as `left` if `direction` is `ltr`, `right` if `direction` is `rtl` if `start` is not supported by the browser.
Applies to	block containers

Example: `text-align` demystified!

- Why? From the spec, can't apply `text-align` to an inline element; must *either* apply `text-align` to its block container, *or* set `a { display : block; }`

Solution 1

CSS

```
p {
  text-align: center;
}
```

HTML

```
<p>
   <a href="https://cs.ucy.ac
      View Course Contract
   </a>
</p>
```

Solution 2

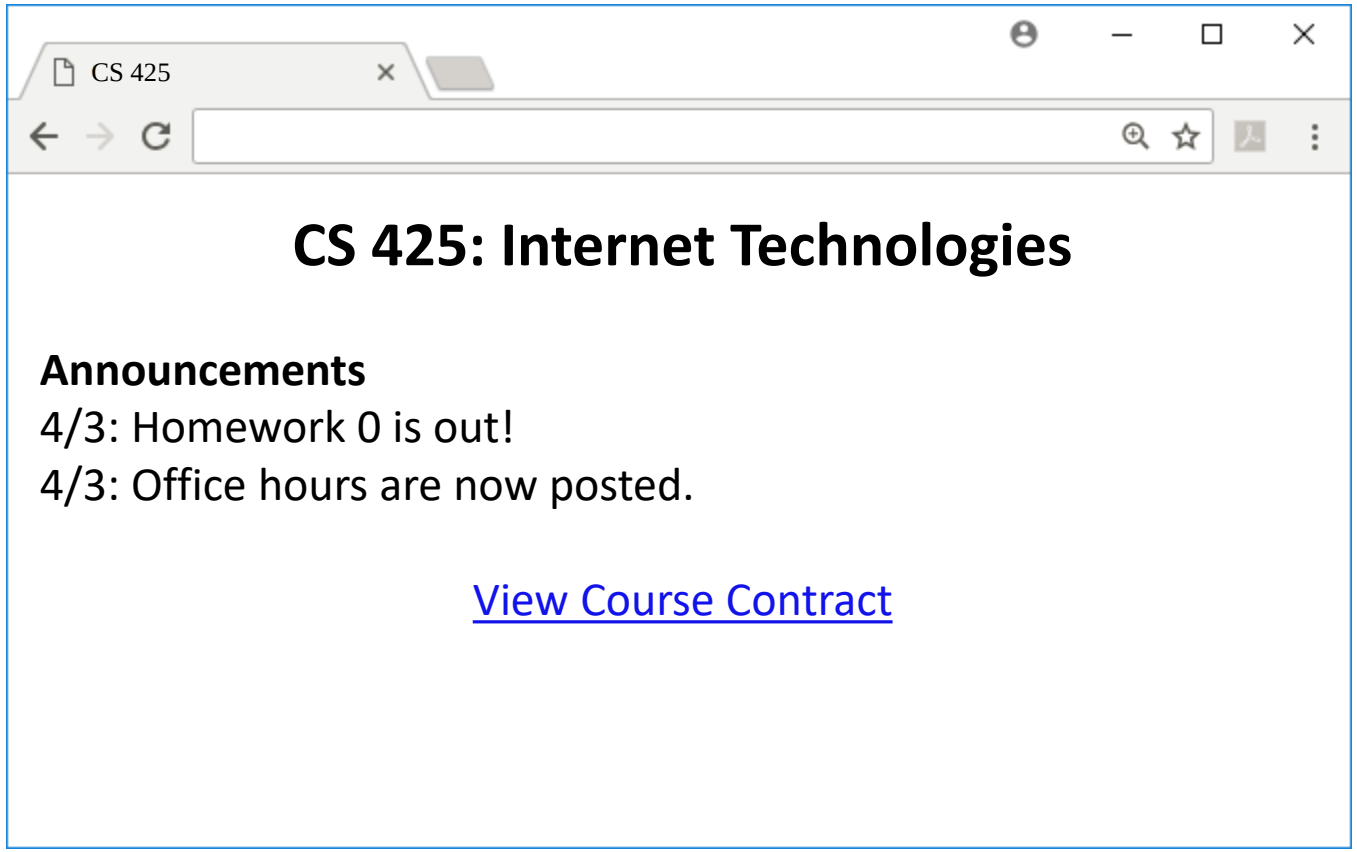
CSS

```
a {
   display: block;
   text-align: center;
}
```

HTML

```
<a href="https://cs.ucy.ac.cy
   View Course Contract
</a>
```

Example: `text-align` demystified!



`<div>` and `<span>`

- Two generic tags with no intended purpose or style:

1. `<div>` : a generic **block** element
2. `<span>` : a generic **inline** element

in action

- We can use as a generic inline HTML container:

CSS

```
span {
   background-color: yellow;
}
```

HTML

```
<strong>Announcements</strong><br/>
4/3: Homework 0 is out! <span>Due Friday</span>.<br/>
4/3: Office hours are now posted.
```

CS 425: Internet Technologies

Announcements
4/3: Homework 0 is out! Due Friday.
4/3: Office hours are now posted.

View Course Contract

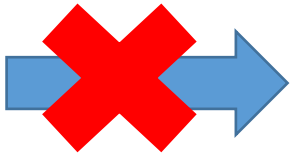
Multiple generic containers?

- But won't we often want multiple generic containers?
- How do we distinguish two generic containers?
- In other words, how do we select a subset of elements instead of all elements on the page?

CS 425: Internet Technologies

Announcements
4/3: Homework 0 is out! Due Friday.
4/3: Office hours are now posted.

View Course Contract



```
span {
    background-color: yellow;
}
span {
    background-color: cyan;
}
```

Colliding styles …

Classes and ids

- There are 2 basic types of CSS selectors:

Element selector		Selected elements
ID selector	**#abc**	element with **`id="abc"`**
Class selector	**.abc**	elements with **`class="abc"`**

```
<h1 id="title">Homework</h1>
<em class="hw">HW0</em> is due Friday.<br/>
<em class="hw">HW1</em> goes out Monday.<br/>
<em>All homework due at 11:59pm.</em>
```

Classes and ids

HTML

```
<h1 id="title">Homework</h1>
<em class="hw">HW0</em> is due Friday.<br/>
<em class="hw">HW1</em> goes out Monday.<br/>
<em>All homework due at 11:59pm.</em>
```

CSS

```
#title {
    color: purple;
}

.hw {
    color: hotpink;
}
```

Homework

HW0 is due Friday.
HW1 goes out Monday.
All homework due at 11:59pm.

More on `class` and `id`

- Special HTML attributes that can be used on any HTML element
  - **`class`**: Applied on 1 or more elements; identifies a **collection** of elements
  - **`id`**: Applied on exactly 1 element per page; identifies **one unique** element
- There are no browser defaults for any `id` or `class`
  - Adding a class name or id to an element does nothing to that element by default
  - They require CSS to target them and apply styling
- Can apply multiple classes by space-separating them:
  `<span class="hw new">HW1</span>`
  - Highest priority: class declared last in the stylesheet file (.css)
- Often used with span and div to create generic elements: e.g.
  `<span class="highlight">` is like creating a "highlight" element

More on `class` and `id`

- `classes` have no special abilities in the browser
- `ids` have special browser functionality: "hash (#) value" in the URL
  - Example: http://yourdomain.com#comments
  - Browser attempts to locate the element with an `id` of `"comments"` and will automatically scroll the page to show that element
- This is an important reason why having `ids` be absolutely unique is important. So your browser knows where to scroll!

More on `class` and `id`

- A single element can have BOTH `id` and `class`
  - `class` is applied to element for styling purposes
  - `id` value is useful for direct linking to that point of a webpage

```
<h1 id="homework" class="hw">Homework</h1>
```

More on `class` and `id`

- Regarding CSS, there is nothing you can do with an `id` that you can't do with a `class` and vise versa. So **CSS doesn't care whether you use id or class!**
- **But Javascript** (as we will see later) **does cares!**
  - JavaScript depends on there being only one page element with any particular id, or else the commonly used `getElementById` function wouldn't be dependable.
  - More on Javascript later…

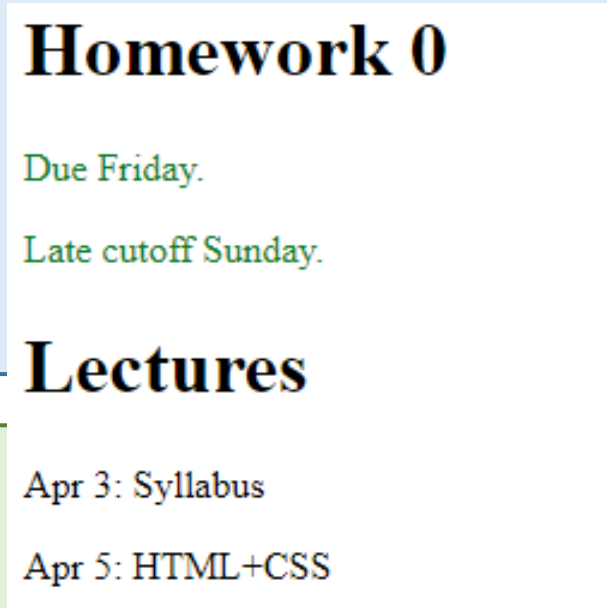
APPENDIX: Other selectors

element.className

Syntax	Example	Example described
element.className	p.hwc	<p> elements with hwc class

HTML

```
<h1 class="hwc">Homework 0</h1>
<p class="hwc">Due Friday.</p>
<p class="hwc">Late cutoff Sunday.</p>
<h1>Lectures</h1>
<p>Apr 3: Syllabus</p>
<p>Apr 5: HTML+CSS</p>
```



CSS

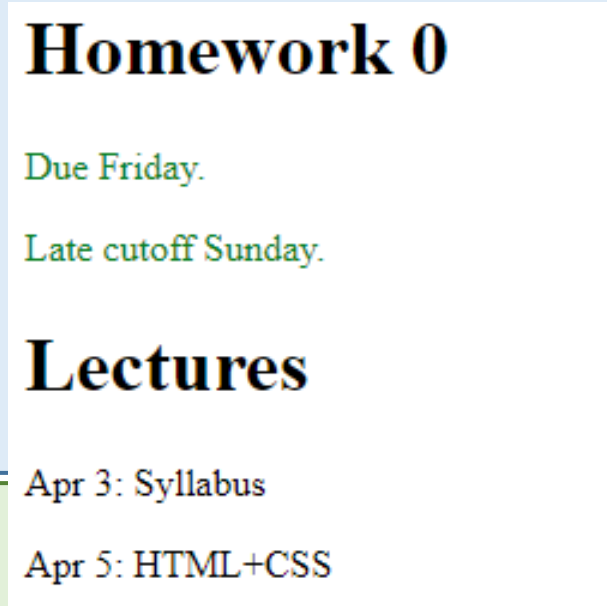
```
p.hwc {
    color: green;
}
```

Descendent selector

Syntax	Example	Example described
selector selector class element	.hwc p	<p> elements that are descendants of hwc class

```html
<div class="hwc">
   <h1>Homework 0</h1>
   <p>Due Friday.</p>
   <p>Late cutoff Sunday.</p>
</div>
<h1>Lectures</h1>
<p>Apr 3: Syllabus</p>
<p>Apr 5: HTML+CSS</p>
```

HTML




```css
.hwc p {
   color: green;
}
```

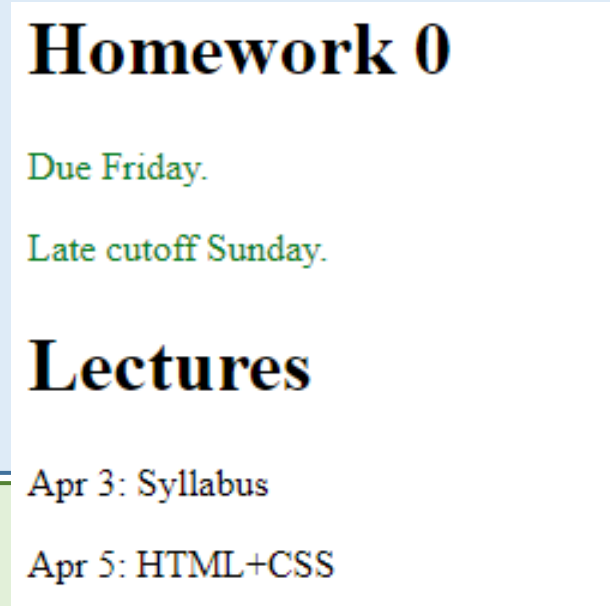
CSS

Descendent selector

Syntax	Example	Example described
selector selector class element	.hwc p	<p> elements that are descendants of hwc class

HTML

```
<div class="hwc">
  <div>
   <h1>Homework 0</h1>
   <p>Due Friday.</p>
   <p>Late cutoff Sunday.</p>
  </div>
</div>
<h1>Lectures</h1>
```



CSS

```
.hwc p {
   color: green;
}
```

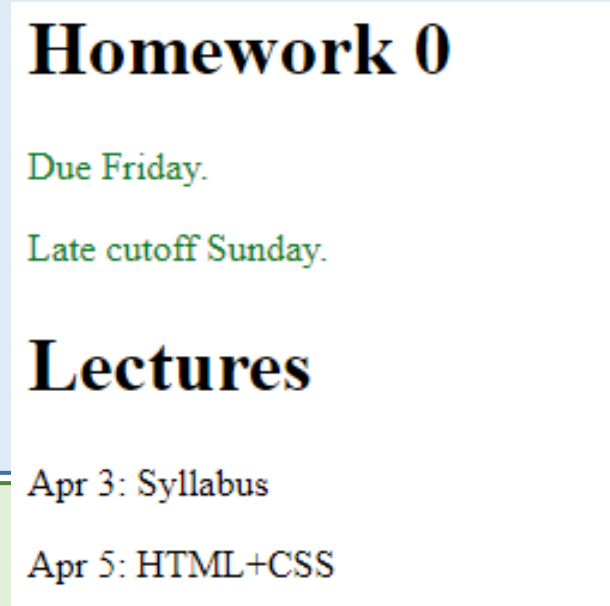
Note: The element does not have to be a direct child. The descendent may be nested many layers in.

Descendent selector

Syntax	Example	Example described
selector selector element element	`div p`	`<p>` elements that are descendants of a `<div>`

HTML

```
<div>
   <h1>Homework 0</h1>
   <p>Due Friday.</p>
   <p>Late cutoff Sunday.</p>
</div>
<h1>Lectures</h1>
<p>Apr 3: Syllabus</p>
<p>Apr 5: HTML+CSS</p>
```



CSS

```
div p {
   color: green;
}
```

Descendent selector

Syntax	Example	Example described
selector selector class element	`div p`	`<p>` elements that are descendants of a `<div>`

Discouraged:

```html
<h1>Homework 0</h1>
<p class="hwc">Due Friday.</p>
<p class="hwc">Late cutoff Sunday.</p>
```

HTML

Preferred:

```html
<div class="hwc">
   <h1>Homework 0</h1>
   <p>Due Friday.</p>
   <p>Late cutoff Sunday.</p>
</div>
```

HTML

Instead of applying a class to several adjacent elements, wrap the group in a `<div>` container and style the contents via descendent selectors.

selector, selector (comma separated)

Syntax	Example	Example described
selector, selector	h1, h2	<h1> and <h2> elements

HTML

```
<h1>Course Info</h1>
<h2>Lectures</h2>
<p>Mon-Thu 13.30-15.30</p>
<h2>Honor Code</h2>
<p>Do the right thing</p>
```

Course Info

Lectures

Mon-Thu 13.30-15.30

Honor Code

Do the right thing

```
h1, h2 {
   background-color: yellow;
}
```

CSS

Selector summary

Example	Description
`p`	All `<p>` elements
`.abc`	All elements with the `abc class`, i.e. `class="abc"`
`#abc`	Element with the `abc id`, i.e. `id="abc"`
`p.abc`	`<p>` elements with `abc class`
`p#abc`	`<p>` element with `abc id` (p is redundant)
`div p`	`<p>` elements that are descendants of a `<div>`
`h1, h2`	`<h1>` and `<h2>` elements

Grouping selectors

2 Common bugs:

- `p.abc` vs `p .abc`
- `p .abc` vs `p, .abc`

- A **<p>** element with the **abc** class vs
  An element with the **abc** class that descends from **<p>**
- An element with the **abc** class that descends from **<p>** vs
  All **<p>** elements and all elements with the **abc** class

Grouping selectors

- You can combine selectors:

```css
#main li.important strong {
    color: red;
}
```

CSS

- Q: What does this select?

Grouping selectors

- You can combine selectors:

```css
#main li.important strong {
    color: red;
}
```

CSS

- Q: What does this select?
- A: Read from right to left:
  - **`<strong>`** tags that are children of **`<li>`** tags that have an **`"important"`** class that are children of the element with the **`"main"`** id.

Colliding styles

- When styles collide, the most specific rule wins (specificity)

CSS

```
div strong { color: red; }
strong { color: blue; }
```

HTML

```
<div>
   <strong>What color am I?</strong>
</div>
```

Colliding styles

- Specificity precedence rules (details):
  - Inline CSS rules e.g. `<h1 style="font-size:22px;">Hello</h1>`
  - `ids` are more specific than `classes`
  - `classes` are more specific than element names
  - Element selector e.g. `h1 { color: yellow; }`
    - Style rules that directly target elements are more specific than style rules that are inherited
      - CSS styles are inherited from parent to child

```
body {
    color: black;
}
```
CSS

Text color in every element inside <body> will be black unless otherwise.

Colliding styles

- If elements have the same specificity, the later rule wins.

CSS

```
strong { color: red; }
strong { color: blue; }
```

HTML

```
<div>
   <strong>What color am I?</strong>
</div>
```

Aside: The process of figuring out what rule applies to a given element is called the cascade. This is where the "C" in *Cascading* Style Sheets comes from.

Colliding styles

- The `!important` rule overrides any other declarations.

CSS

```
strong { color: red !important; }
strong { color: blue; }
```

HTML

```
<div>
   <strong>What color am I?</strong>
</div>
```

Aside: The process of figuring out what rule applies to a given element is called the cascade. This is where the "C" in *Cascading* Style Sheets comes from.

Inheritance

- We saw earlier that CSS styles are inherited from parent to child.

Instead of selecting all elements individually:

```css
a, h1, p, strong {
    font-family: Helvetica;
}
```

CSS

You can style the parent and the children will inherit the styles.

You can override this style via specificity:

```css
body {
    font-family: Helvetica;
}

h1, h2 {
    font-family: Consolas;
}
```

CSS

Inheritance

- While many CSS styles are inherited from parent to child, **not all CSS properties are inherited**.

`<em>` inherits the `font-family` property, but not `display`:

```
a {
    display: block;
    font-family: Arial;
}
em {
    border:2px solid #000;
}
```

CSS

Back to *Home*

HTML

```
<a href="/home">
 Back to <em>Home</em>
</a>
```

Inheritance

- While many CSS styles are inherited from parent to child, **not all CSS properties are inherited**.
- There's no rule for what properties are inherited or not; the inheritance behavior defined in the CSS spec.
- You can look it up via MDN, e.g.

font-family:	Inherited	yes
display:	Inherited	no

- Generally, text-related properties are inherited, and layout-related properties are not.
- (You can also change this via the `inherit` CSS property, which is somewhat esoteric and not often use)

<a> colors?

- MDN says color is inherited… but if I set the body color to deeppink, links don't change color:

```css
body {
    color: deeppink;
    font-family: Helvetica;
}
```

CSS

```html
<body>
  <h1>Chocolate</h1>
  <p>
    <a
href="https://ghirardelli.com">Ghirardelli</a>
is not overrated.
  <p>
</body>
```

HTML

<a> inherits font-family…

Why doesn't inherit color?

Chocolate

Ghirardelli is not overrated.

User agent styles

- This is because the browser has its own default styles:
  - Browser loads its own default stylesheet on every webpage
  - Not governed by spec, but there are recommendations

```
<!DOCTYPE html>
<html>
  <head>
    <title>       </title>
    <!--
      NOT TOTALLY ACCURATE: This isn't actually injected
      in the HTML, but it is loaded silently!
    -->
    <link rel="stylesheet" href="user-agent-style.css" />
  </head>
```

<a> colors?

- So to style <a> links, we have to override the browser default link style by explicitly setting a color:

```
body {
    color: deeppink;
    font-family: Helvetica;
}
a {
    color: deeppink;
}
```

CSS

```
<h1>Chocolate</h1>
<p>
   <a
href="https://ghirardelli.com">Ghirardelli</a>
is not overrated.
<p>
```

HTML

Chocolate

Ghirardelli is not overrated.

Link-related CSS

- Since we're on the topic of links:
  - How do we style visited links differently from unvisited?

CSS pseudo-classes

- pseudo-classes: special keywords you can append to selectors, specifying a *state* or *property* of the selector

Syntax	Explanation
`a`	All anchor tags (links) in all states
`a:visited`	A visited link
`a:link`	An unvisited link
`a:hover`	The style when you hover over a link
`a:active`	The style when you have "activated" a link (downclick)

There are more pseudo-classes than this; have a look!

CSS pseudo-classes

Syntax	Explanation
`:first-child`	Matches the first element among a group of sibling elements

CSS

```
p:first-child {
   color: blue;
}
```

HTML

```
<p>This is some text.</p>
<p>This is some text.</p>
<p>This is some text.</p>
```

This is some text.

This is some text.

This is some text.

CSS pseudo-classes

Syntax	Explanation
`:not(selector)`	Selects every element that is NOT in the selector

CSS

```
p {
    color: #000000;
}
:not(p) {
    color: #ff0000;
}
```

```
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<div>This is some text in a div element.</div>
<a href="https://www.w3schools.com"target="_blank">W3Schools</a>
```

This is a heading

This is a paragraph.

This is another paragraph.

This is some text in a div element.
W3Schools

CSS pseudo-classes

Syntax	Explanation
`:nth-child`	Selects every element that is that is the n child of its parent (the index of the first child is 1)

CSS

```
p:nth-child(2) {
   background: red;
}
```

HTML

```
<p>The first paragraph.</p>
<p>The second paragraph.</p>
<p>The third paragraph.</p>
<p>The fourth paragraph.</p>
```

The first paragraph.

The second paragraph.

The third paragraph.

The fourth paragraph.

CSS pseudo-classes

Syntax	Explanation
`:nth-child`	Odd and even are keywords that can be used to match child elements whose index is odd or even

CSS

```
p:nth-child(odd) {
   background: red;
}
p:nth-child(even) {
   background: blue;
}
```

HTML

```
<p>The first paragraph.</p>
<p>The second paragraph.</p>
<p>The third paragraph.</p>
<p>The fourth paragraph.</p>
```

The first paragraph.

The second paragraph.

The third paragraph.

The fourth paragraph.

CSS pseudo-classes

Syntax	Explanation
`:nth-child`	Using formula an+b where a represents a cycle size, n is a counter (starts at 0) and b is an offset value

CSS

```
p:nth-child(3n+0) {
   background: red;
}
```

HTML

```
<p>The first paragraph.</p>
<p>The second paragraph.</p>
<p>The third paragraph.</p>
<p>The fourth paragraph.</p>
<p>The fifth paragraph.</p>
<p>The sixth paragraph.</p>
<p>The seventh paragraph.</p>
<p>The eight paragraph.</p>
<p>The ninth paragraph.</p>
```

The first paragraph.

The second paragraph.

The third paragraph.

The fourth paragraph.

The fifth paragraph.

The sixth paragraph.

The seventh paragraph.

The eight paragraph.

The ninth paragraph.

CSS pseudo-classes

Syntax	Explanation
`::after`	Creates a pseudo-element that is the last child of the selected element. It is often used to add cosmetic content to an element with the content property. It is inline by default.

CSS

```
p::after {
    content: ' !!!';
    color: red;
}
```

HTML

```
<p>I like web programming</p>
<p>My favorite course</p>
```

I like web programming !!!

My favorite course !!!

Debugging Cascades in Chrome Dev Tools