IS 242 Web Application Development 1

Lecture 7: Introduction to CSS (Part 3)

Outlines of today’s lecture

- CSS layout properties
- CSS combinators
- CSS pseudo-classes
- CS pseudo-elements
- Cascading order

CSS Positioning Properties

- The **CSS positioning** properties allow you to **position an element**.
- Elements can be positioned using the **top**, **bottom**, **left**, and **right** properties.
- These properties will not work unless the **position** property is set first. They also work differently depending on the **positioning method**.
- There are four different **positioning methods** specified using the position property:
  - Static
  - Fixed
  - Relative
  - Absolute

Static Positioning

- A static positioned element is always positioned according to the **normal flow** of the page.
- HTML elements are positioned static by **default**.
- Static positioned elements are **not affected** by the top, bottom, left, and right properties.

Static Positioning – Example

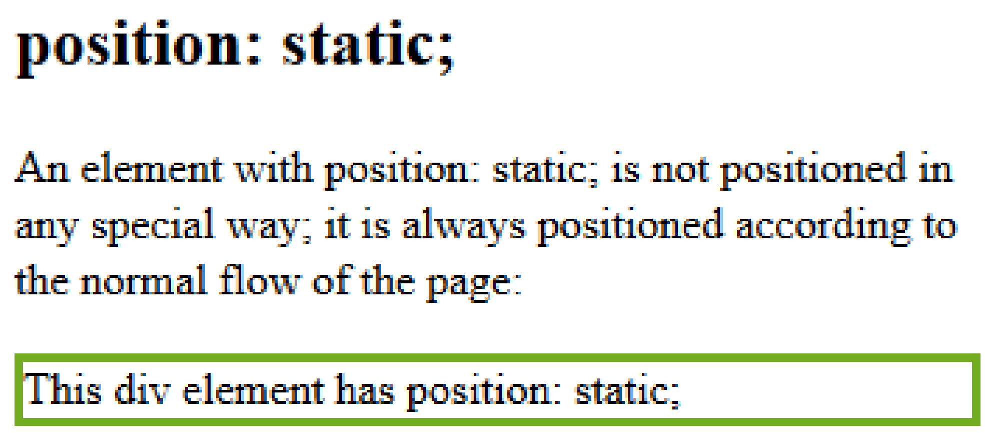
```html
<!DOCTYPE html>
<html>
<head>
<style>
div.fixed {
    position: fixed;
    bottom: 0;
    right: 0;
    width: 300px;
    border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: fixed;</h2>

<p>An element with position: fixed; is positioned relative to the viewport, which
means it always stays in the same place even if the page is scrolled:</p>

<div class="fixed">
This div element has position: fixed;
</div>

</body>
</html>
```



position: static;

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

This div element has position: static;

Fixed Positioning

- An element with a fixed position is positioned **relative to the browser window**, and will not move even if the window is scrolled.

- Fixed positioned elements are **removed from the normal flow (i.e. do not affect the position of surrounding elements)**. The document and other elements behave like the fixed positioned element does not exist.

- Fixed positioned elements can **overlap** other elements.

- You can indicate that an element should be fixed using the **position** property with a value of **fixed**.

- You then use the **offset properties** (top or bottom and left or right) to indicate where the element should appear in relation to the browser window.

Fixed Positioning-Example

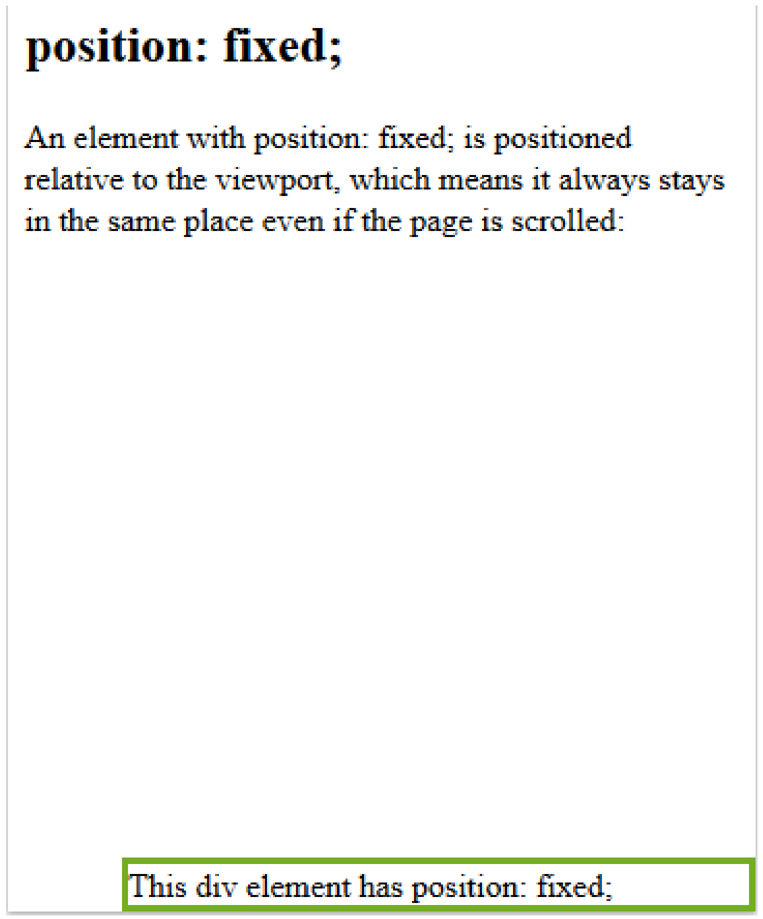
```
<!DOCTYPE html>
<html>
<head>
<style>
div.fixed {
    position: fixed;
    bottom: 0;
    right: 0;
    width: 300px;
    border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: fixed;</h2>

<p>An element with position: fixed; is positioned relative to the viewport, which
means it always stays in the same place even if the page is scrolled:</p>

<div class="fixed">
This div element has position: fixed;
</div>

</body>
</html>
```



position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled:

This div element has position: fixed;

Relative Positioning

- A relative positioned element is positioned **relative to its normal position**.

- The content of relatively positioned elements can be moved and overlap other elements, but the reserved space for the element is still **preserved** in the normal flow.

- You can indicate that an element should be relatively positioned using the **position** property with a value of **relative**.

- You then use the **offset properties** (top or bottom and left or right) to indicate how far to move the element from where it would have been in normal flow.

Relative Positioning-Example

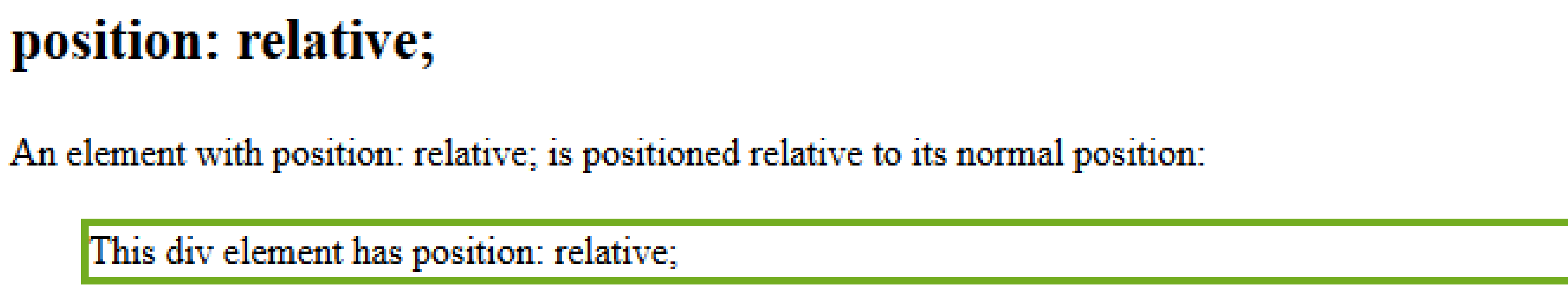
```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
    position: relative;
    left: 30px;
    border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: relative;</h2>

<p>An element with position: relative; is positioned relative to its normal
position:</p>

<div class="relative">
This div element has position: relative;
</div>

</body>
</html>
```



position: relative;

An element with position: relative; is positioned relative to its normal position:

This div element has position: relative;

Absolute Positioning

- An absolute position element is positioned **relative to the first parent element that has a position other than static.** If no such element is found, the containing block is <html>

- Absolutely positioned elements are **removed from the normal flow.** The document and other elements behave like the absolutely positioned element **does not exist.**

- Absolutely positioned elements can overlap other elements.

- You can indicate that an element should have an absolute positioning using the **position** property with a value of **absolute.**

- The box **offset properties (top or bottom and left or right)** specify where the element should appear in relation to its containing element

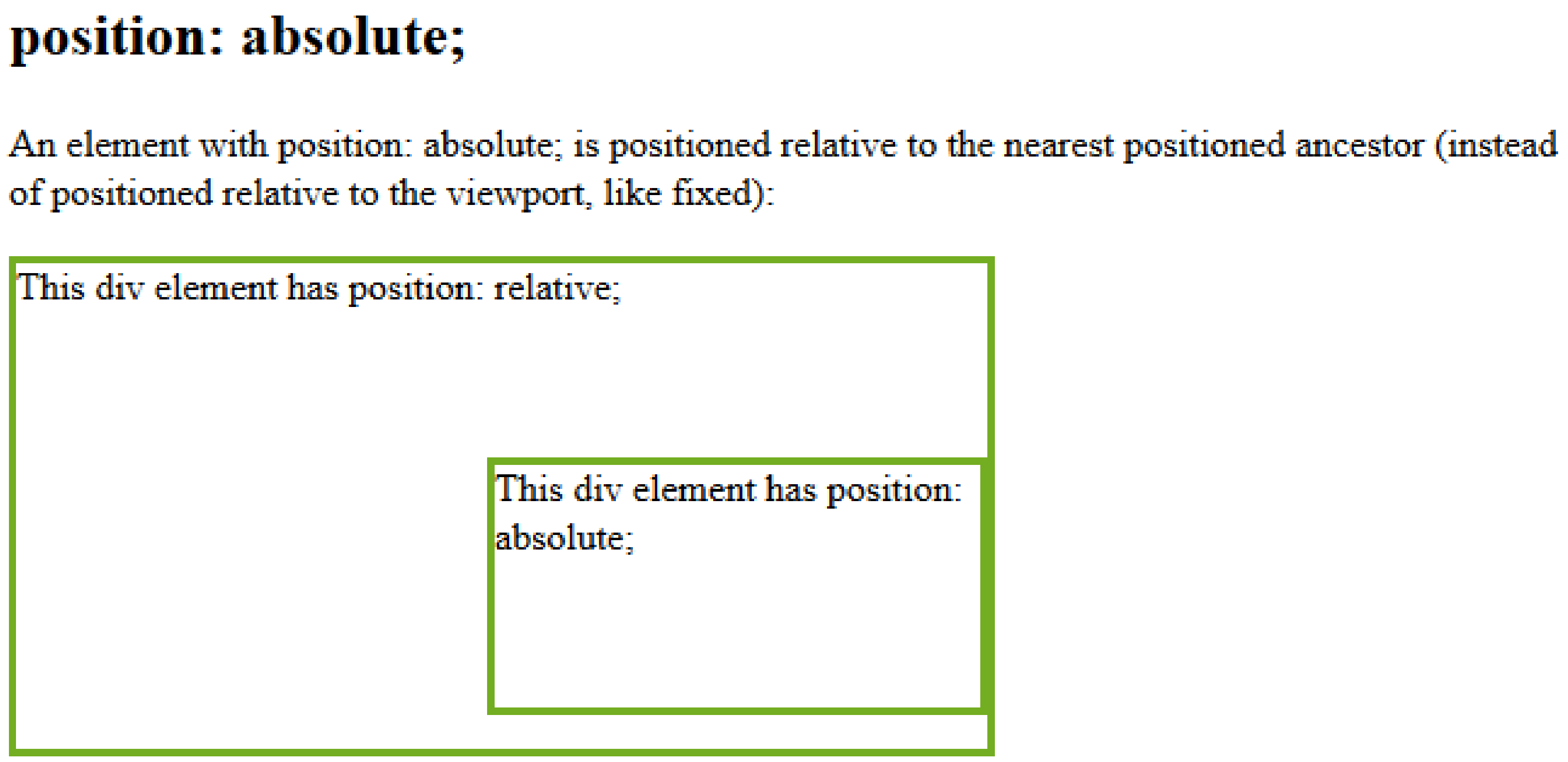
Overlapped Elements

- When elements are positioned outside the normal flow, they can overlap other elements.
- The **z-index** property specifies the stack order of an element (which element should be placed in front of, or behind, the others).
- An element can have a positive or negative stack order.
- Example: http://www.w3schools.com/css/tryit.asp?filename=trycss_zindex
- An element with **greater stack order** is always in front of an element with a lower stack order.
- **Note:** If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.

Absolute Positioning-Example

```
div.relative {
    position: relative;
    width: 400px;
    height: 200px;
    border: 3px solid #73AD21;
}

div.absolute {
    position: absolute;
    top: 80px;
    right: 0;
    width: 200px;
    height: 100px;
    border: 3px solid #73AD21;
}
```



position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):

This div element has position: relative;

This div element has position: absolute;

CSS Float

- With CSS **float**, an element can be pushed to the left or right, allowing other elements to wrap around it.

- Float is often used with images, but it is also useful when working with layouts.

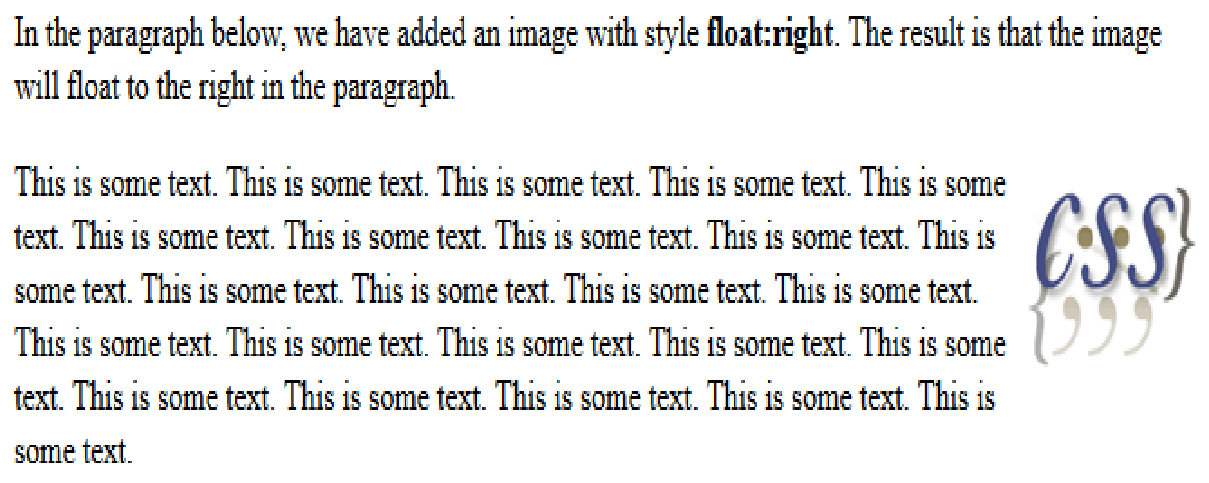
- Elements are floated **horizontally**, this means that an element can only be floated left or right, not up or down.

- The elements after the floating element will flow around it.

- The elements before the floating element will not be affected.

CSS Float-Example

```html
<!DOCTYPE html>
<html>
<head>
<style>
img {
    float: right;
}
</style>
</head>
<body>
<p>In the paragraph below, we have added an image with style <b>float:right</b>.
The result is that the image will float to the right in the paragraph.</p>
<p>
<img src="logocss.gif" width="95" height="84" />
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>
</body>
</html>
```



In the paragraph below, we have added an image with style **float:right**. The result is that the image will float to the right in the paragraph.

This is some text. This is some text.

CSS Float (Cont.)

- If you place several floating elements after each other, they will float next to each other if there is room.

- Here we have made an image gallery using the float property: http://www.w3schools.com/css/tryit.asp?filename=trycss_float_elements

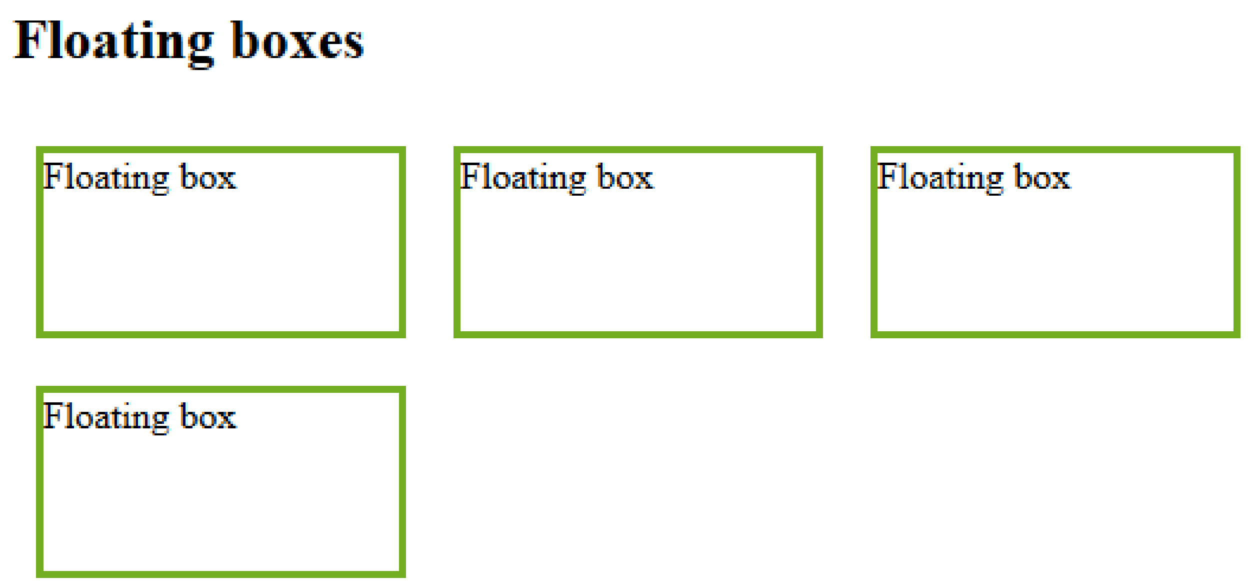
- Elements after the floating element will flow around it. To avoid this, use the **clear** property.

- The clear property specifies which sides of an element other floating elements are not allowed.

- Add a text line into the image gallery, using the clear property: http://www.w3schools.com/css/tryit.asp?filename=trycss_float_clear

CSS Float - Several floating elements

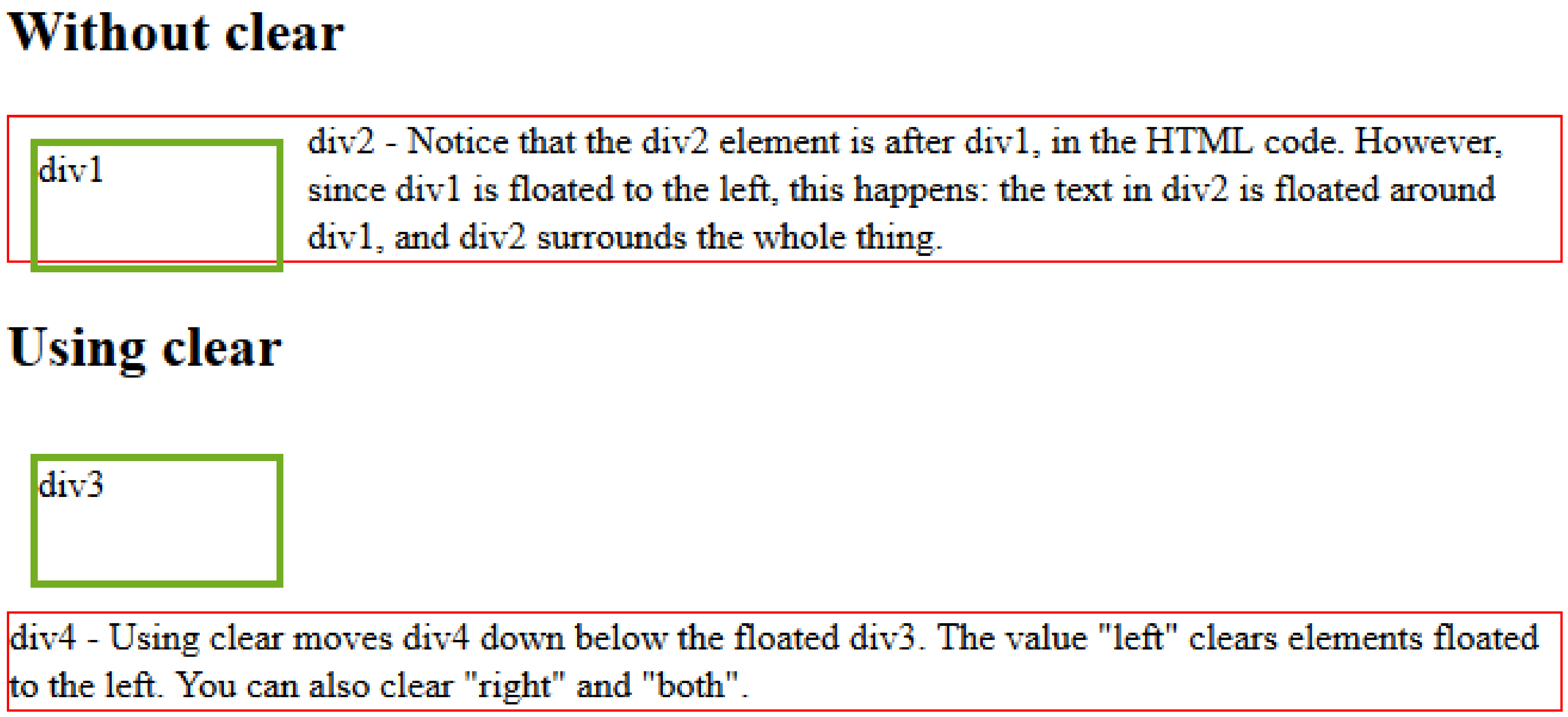
```
<!DOCTYPE html>
<html>
<head>
<style>
.floating-box {
    float: left;
    width: 150px;
    height: 75px;
    margin: 10px;
    border: 3px solid #73AD21;
}
</style>
</head>
<body>
<h2>Floating boxes</h2>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
<div class="floating-box">Floating box</div>
</body>
</html>
```



Floating boxes

Floating box
Floating box
Floating box
Floating box

CSS float - clear



Without clear

div1

div2 - Notice that the div2 element is after div1, in the HTML code. However, since div1 is floated to the left, this happens: the text in div2 is floated around div1, and div2 surrounds the whole thing.

Using clear

div3

div4 - Using clear moves div4 down below the floated div3. The value "left" clears elements floated to the left. You can also clear "right" and "both".

CSS Combinators

CSS Combinators contain more than one simple selector. There are four different combinators in CSS3:

- Descendant combinator/selector
- Child combinator/selector
- Adjacent sibling combinator/selector
- General sibling combinator/selector

Descendant Selector

- The descendant selector **matches all elements that are descendants** of a specified element.
- A descendant selector is made up of **two or more** selectors separated by **white space.**

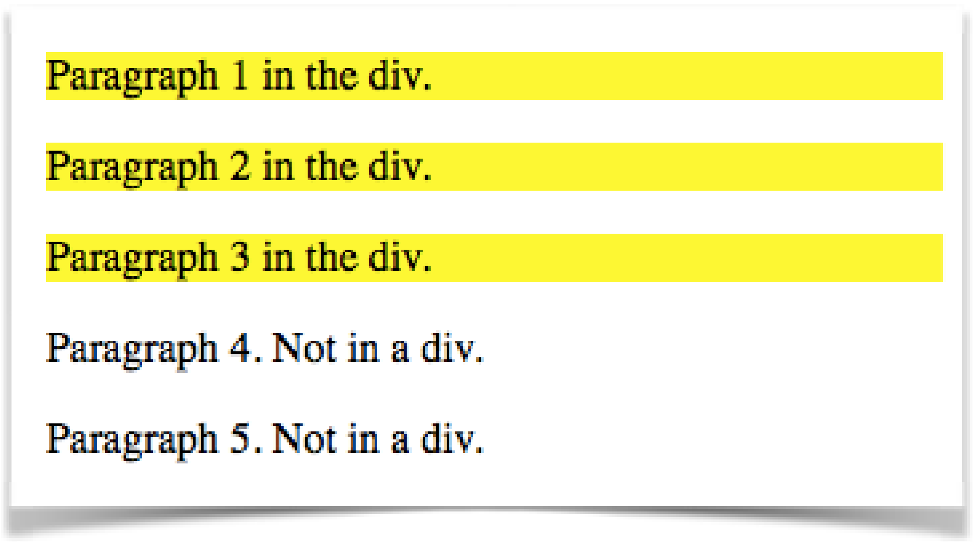
```
p em { color: #FF0066; }
```

Example

```
div p {
    background-color: yellow;
}
```

```
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <span><p>Paragraph 3 in the div.</p></span>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>
```



Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4. Not in a div.

Paragraph 5. Not in a div.

Child Selector

- The child selector selects all elements that are **the immediate children** of a specified element.

- The special character used in child selector is **the > (greater than) sign.**

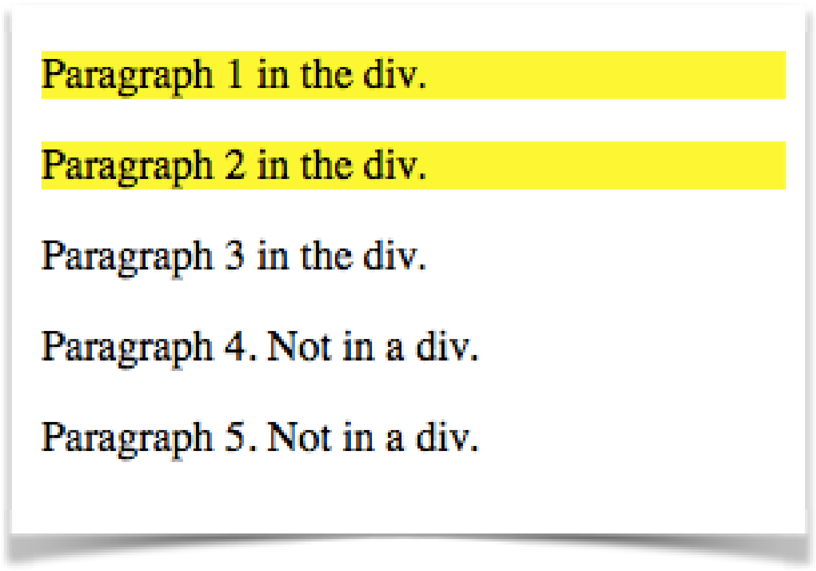
```
p > em { color: #FF0066; }
```

Example

```
div>p {
    background-color: yellow;
}
```

```
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <span><p>Paragraph 3 in the div.</p></span>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>
```



Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4. Not in a div.

Paragraph 5. Not in a div.

Adjacent Sibling Selector

- The adjacent sibling selector selects all elements that are the **adjacent siblings** of a specified element.
- Sibling elements must have the **same parent element**, and **"adjacent" means "immediately following”**.
- The special character used in Adjacent Sibling selector is **the + (plus) character**.

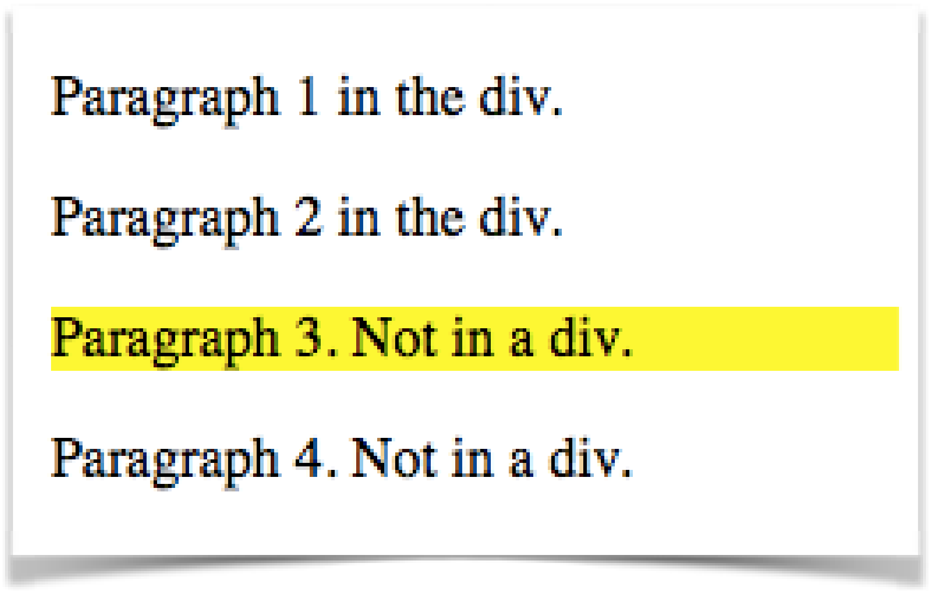
```
div+p {color: #FF0066;}
```

Example

```
div+p {
    background-color: yellow;
}
```

```
<body>
  <div>
    <p>Paragraph 1 in the div.</p>
    <p>Paragraph 2 in the div.</p>
  </div>

  <p>Paragraph 3. Not in a div.</p>
  <p>Paragraph 4. Not in a div.</p>
</body>
```



Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3. Not in a div.

Paragraph 4. Not in a div.

General Sibling Selector

- The general sibling selector selects **all elements that are siblings of a specified element and which are following it.**

- The special character is the **~ (tilde) character.**

```
div ~ p {color: #FF0066;}
```

Example

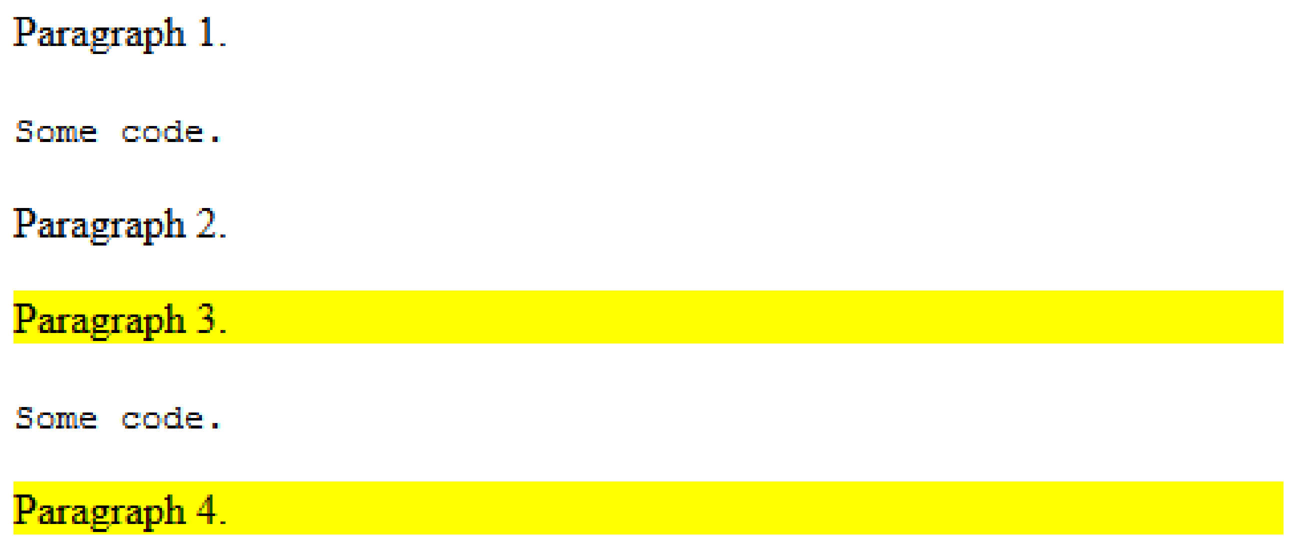
```
<!DOCTYPE html>
<html>
<head>
<style>
div ~ p {
    background-color: yellow;
}
</style>
</head>
<body>

<p>Paragraph 1.</p>

<div>
  <code>Some code.</code>
  <p>Paragraph 2.</p>
</div>

<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>

</body>
</html>
```



```
Paragraph 1.

Some code.

Paragraph 2.

Paragraph 3.

Some code.

Paragraph 4.
```

Pseudo-Classes

- Pseudo classes allow you to **control how the element should appear under different conditions**.

- A pseudo-class is used to define **a special state of an element**.

- For example, it can be used to:
  - Style an element when a user mouses over it
  - Style visited and unvisited links differently

- The syntax of pseudo-classes:

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

Example: Styling Links

- Links can be styled differently depending on what state they are in.

The **four links states** are:

- link - a normal, unvisited link
- visited - a link the user has visited
- hover - a link when the user mouses over it
- active - a link the moment it is clicked

Styling Links (Cont.)

```
/* unvisited link */
a:link {
    color: #FF0000;
}

/* visited link */
a:visited {
    color: #00FF00;
}

/* mouse over link */
a:hover {
    color: #FF00FF;
}

/* selected link */
a:active {
    color: #0000FF;
}
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

Pseudo-Eelements

A CSS pseudo-element is used to style **specified parts of an element**.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

The syntax of pseudo-elements:

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

Examples

- The **::first-line** pseudo-element is used to add a special style to the first line of a text.
- The ::first-line pseudo-element can only be applied to **block elements**.

```
p::first-line {
    color: #ff0000;
    font-variant: small-caps;
}
```

- The **::first-letter** pseudo-element is used to add a special style to the first letter of a text.
- The ::first-letter pseudo-element can only be applied to **block elements**.

```
p::first-letter {
    color: #ff0000;
    font-size: xx-large;
}
```

Examples (Cont.)

- The **::selection** pseudo-element matches the portion of an element that is selected by a user.

- The following example makes the selected text red on a yellow background:

```
::selection {
     color: red;
     background: yellow;
}
```

Read more on pseudo-classes and elements

Cascading Order

- What style will be used when there is more than one style specified for an HTML element?

- Generally speaking we can say that all the styles will "cascade" into **a new "virtual" style sheet** by the following rules, where number four has the highest priority:
    1 Browser default
    2 External style sheet
    3 Internal style sheet (in the head section)
    4 Inline style (inside an HTML element)

- So, an inline style (inside an HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or in a browser (a default value).

- Note: If the link to the external style sheet is placed after the internal style sheet in HTML <head>, the external style sheet will override the internal style sheet!

References

- www.w3schools.com
- Duckett, J. (2011). *HTML and CSS: Design and Build Websites*. John Wiley & Sons.
- Deitel & Deitel (2011). *Internet and World Wide Web How to Program, 5th Edition, Harvey & Paul Deitel & Associates.*