

# FWDP 1000 – Day 7

Course: Web Development 1

Instructor: Gabbie Bade

# Morning Review

- Download the files.
- Edit styles.css to follow the styling best practices from the Day 6 slides.
- You may work alone or with a group. Feel free to join any of the breakout rooms.

We will go over this in 15 minutes. Set up your day 7 branch as well.

# Morning Review

What questions do you have? Anything you want to review?

- Domains, hosting, FTP, servers...
- Flexbox...
- Media queries...
- GitHub...
- etc...

# Agenda

- CSS Floats
- CSS Pseudo-elements
- CSS Positioning
- Assignment #5

# CSS Floats

# CSS Floats

The CSS `float` property allows elements to be moved to the right or left side of a container. Other elements in the container will wrap around the floated element.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac molestie ante. Fusce semper consectetur fermentum. In pellentes

**Floated  
Left**

ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac molestie ante.

Fusce semper consectetur fermentum. In felis justo, et

Mauris vel lacus sapien. Cras neque tortortur at dapibus ut, elementum vel tellus.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac molestie ante. Fusce semper consectetur fermentum. In pellentes ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac molestie ante.

**Floated  
Right**

Fusce semper consectetur fermentum. In felis justo, et Mauris vel lacus sapien. Cras neque tortortur at dapibus ut, elementum vel tellus.

# Floats for Layouts

Before CSS Flexbox and CSS Grid, floats were used to do layouts on the web.

Unless you are supporting very old browsers, floats should not be used to create columns of content.

[https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout/Floats](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Floats)

# CSS Float Syntax

There are three main values for the float property:

```
float: left;
```

```
float: right;
```

```
float: none;
```

<https://developer.mozilla.org/en-US/docs/Web/CSS/float>



# Floats and Flow

Floated elements can expand outside of their container element.

If we float an image at the end of one section, it can expand into the next section, altering the layout in ways we don't intend.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac molestie ante. Fusce semper consectetur fermentum. In pellentes ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac molestie ante. Fusce semper consectetur fermentum. In felis justo, et



dolor sit amet, consectetur adipiscing elit.

## Heading

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac consectetur adipiscing elit. Nunc ac molestie ante. Fusce semper consectetur fermentum. In pellentes ipsum dolor sit amet, ante.

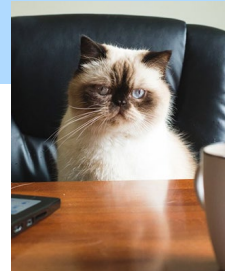
# Clearing Floats

To prevent floated elements from expanding outside of their containers, we can use the CSS `clear` property on the next element.

It still won't look great though.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac molestie ante. Fusce semper consectetur fermentum. In pellentes ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac molestie ante. Fusce semper consectetur fermentum. In felis justo, et

dolor sit amet, consectetur adipiscing elit.



## Heading

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ac dolor sit amet

# CSS Clear Syntax

There are four main values for the `clear` property:

```
clear: left;
```

```
clear: right;
```

```
clear: none;
```

```
clear: both;
```

<https://developer.mozilla.org/en-US/docs/Web/CSS/clear>

# Clearfix Hack

The most common solution to clearing floats is to add this code to the element that contains the floated element.

Class or element name

Pseudo-element

```
.container::after {  
  content: "";  
  display: block;  
  clear: both;  
}
```

Clear left and right floats

# Pseudo Elements

# CSS Pseudo Elements

A pseudo-element is a part of an existing HTML element that can be selected and styled.

For example: the first letter or the first line of some text.

Two common pseudo-elements are **::before** and **::after**. They act like a new element was added to the HTML.

<https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements>

# Pseudo Element Syntax

The pseudo-element selector must first select an element by its type, class, or ID. Then it can target the pseudo-element.

Double colon indicates  
a pseudo-element

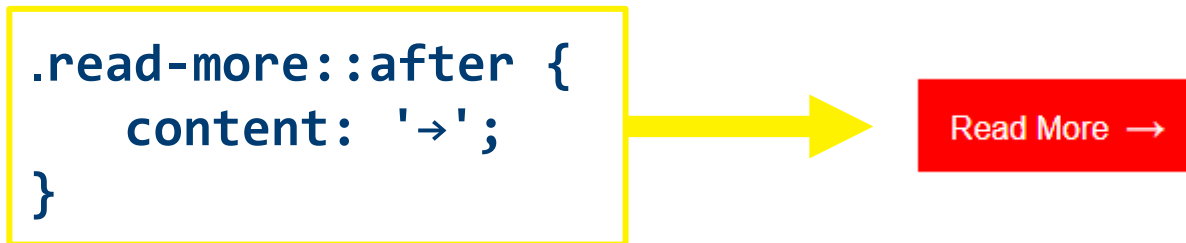


```
p::first-letter {  
  font-size: 2rem;  
  color: red;  
}
```

# Pseudo Elements for Design

Pseudo-elements should **not** be used for content and only to add to the design.

The content should make sense without the pseudo-element.





# Selectors vs Elements

**Pseudo-elements** are used to style part of an element.

Example: the first letter of a paragraph.

**Pseudo-classes** are used to style an element based on its state.

Example: the hover state of a button.

# Recap

**True or False:** The CSS float property should only be used to wrap text around an HTML element.

**True or False:** The content inside the **::before** or **::after** pseudo element able to be read by screen readers?

# CSS Positioning

# CSS Position Property

The CSS `position` property sets how an element is positioned in a document.

The `top`, `right`, `bottom`, `left` properties determine the final location of positioned elements on the x and y axis.

The `z-index` property determines the final location of the positioned elements on the z axis.

<https://developer.mozilla.org/en-US/docs/Web/CSS/position>

# Use Position Sparingly

Do **not** overuse the CSS `position` property.

Flexbox and Grid should be your default ways to handle layout.

Positioning is used for specific cases when standard methods of layout do not work.

# CSS Position Values

The position property has five possible values:

- `static`
- `relative`
- `absolute`
- `fixed`
- `sticky`

# Related Properties

In addition to setting position, you may also use these CSS properties but they are not required:

- top
- bottom
- left
- right
- z-index

# Position – Static

This is the default value of the position property for every HTML element and means the element will appear where it is in the HTML document.

The top, bottom, left, right, z-index properties are ignored if set on an element with `position: static`.

If you ever use this, it will only be to override another position value.



# Position – Relative

Relative is similar to Static except you can now use top, bottom, left, right, z-index.

The element is positioned normally then offset **relative to itself** based on the values of top, right, bottom, and left.

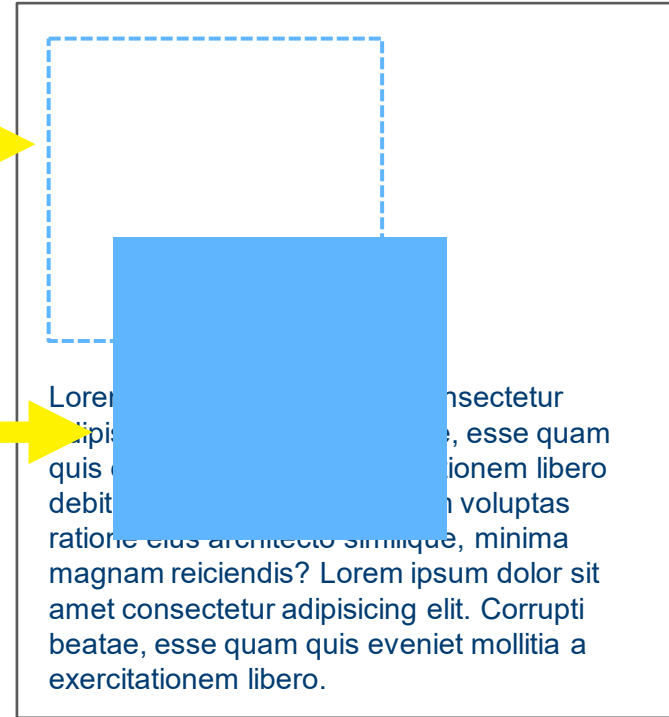
Moving the element does not impact elements around them and the original space for the element remains.

# Position – Relative

Original location  
of the element



```
.box {  
  position: relative;  
  top: 20rem;  
  left: 2rem;  
}
```



# Position – Relative

You generally use `position: relative` so you can use the remaining positions **inside** of a container.

In particular, `position: absolute` works well when the parent is relatively positioned.

# Position – Absolute

Absolute is similar to relative except it removes the element from the normal flow so the original space is removed.

If no parent element has a position set other than static, then the element will position based on the browser window when using top, bottom, left, right, z-index.

# Position – Absolute

Original location of the element.  
The text has moved up to fill that  
empty space.

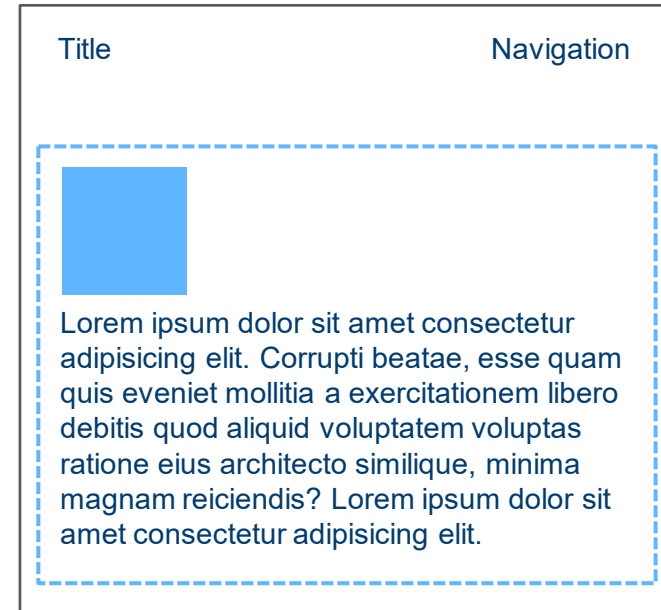
```
.box {  
  position: absolute;  
  top: 20rem;  
  left: 2rem;  
}
```

Lorem ipsum dolor sit amet consectetur  
adipiscing elit. Corrupti beatae, esse quam  
quis eveniet mollitia a exercitationem libero  
debitis quod aliquid voluptatem voluptas  
ratione eius architecto similique, minima  
magnam reiciendis? Lorem ipsum dolor sit  
amet elit. Corrupti  
beatae et mollitia a  
exercitationem

# Position – Absolute

An example of a page with no positioning set.

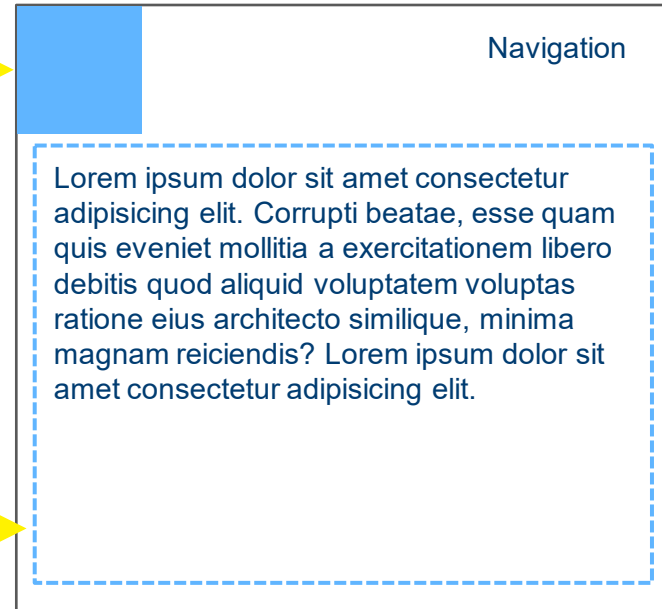
This is simply the default layout.



# Position – Absolute

```
.box {  
  position: absolute;  
  top: 0;  
  left: 0;  
}
```

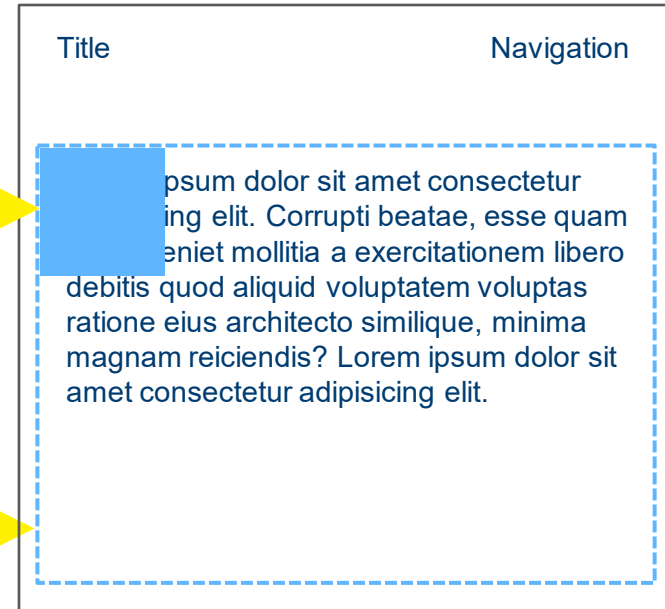
Parent element has no position  
set so defaults to static.



# Position – Absolute

```
.box {  
  position: absolute;  
  top: 0;  
  left: 0;  
}
```

```
.parent {  
  position: relative;  
}
```





# Position – Fixed

Fixed is similar to absolute except it is **always** relative to the viewport.

Using the top, bottom, left, right, z-index properties is based on the viewport, not the webpage.

For example, setting bottom: 0; will place the element at the bottom of the viewable browser window, not the bottom of the webpage.

# Position – Fixed

```
.box {  
  position: fixed;  
  top: 0;  
  left: 0;  
}
```



et consectetur  
eatae, esse quam  
quis eveniet mollitia a exercitationem libero  
debitis quod aliquid voluptatem voluptas  
ratione eius architecto similique, minima  
magnam reiciendis? Lorem ipsum dolor sit  
amet consectetur adipisicing elit.

# Position – Fixed

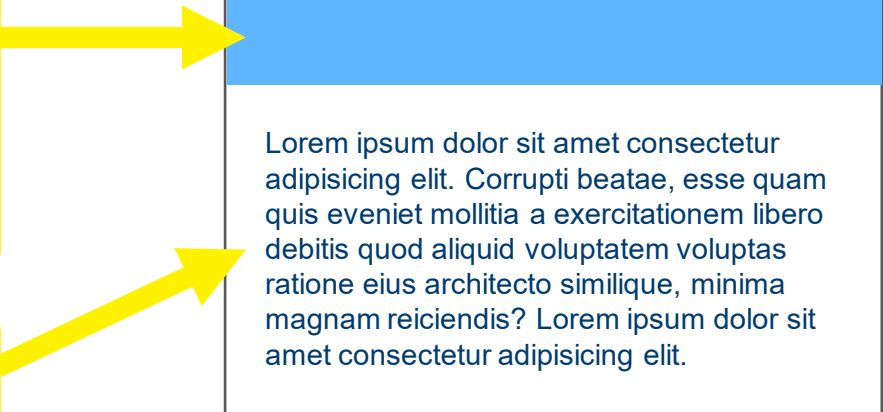
When you set an element to `position: fixed`, make sure you consider that some elements may have flown behind it.

You will often want to offset other elements by the height or width of the fixed element.

# Position – Fixed

```
.box {  
  position: fixed;  
  top: 0;  
  left: 0;  
  right: 0;  
}
```

```
.element {  
  margin-top: 5rem;  
}
```



Lorem ipsum dolor sit amet consectetur adipisicing elit. Corrupti beatae, esse quam quis eveniet mollitia a exercitationem libero debitis quod aliquid voluptatem voluptas ratione eius architecto similique, minima magnam reiciendis? Lorem ipsum dolor sit amet consectetur adipisicing elit.

# Position – Sticky

Sticky is a hybrid of relative and fixed. It must be used with at least one of top, bottom, left, right otherwise it will be the same as relative.

It functions as relative to start until the viewport reaches the position defined by one of those four properties, then the element functions similar to fixed but stays within its parent container.

# Position – Sticky

Check the example provided by MDN to see it in action...

[https://developer.mozilla.org/en-US/docs/Web/CSS/position#Result\\_3](https://developer.mozilla.org/en-US/docs/Web/CSS/position#Result_3)

# Z-Index

The z-index property can be used on any element with a position set other than static to determine how items stack on one another visually.

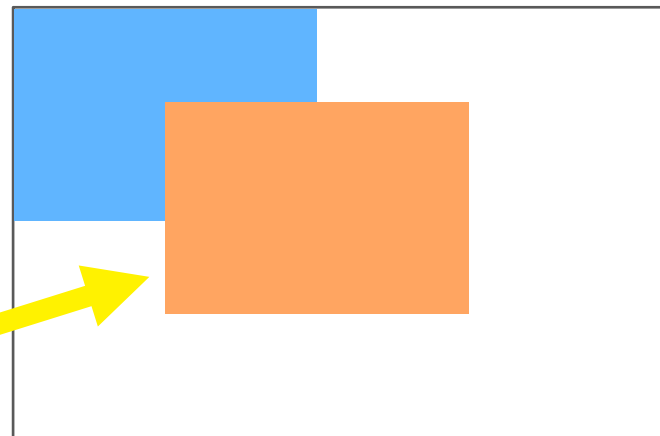
It accepts positive or negative number values with higher values stacking over smaller values.

<https://developer.mozilla.org/en-US/docs/Web/CSS/z-index>

# Z-Index

```
.box-1 {  
  position: absolute;  
  top: 0;  
  left: 0;  
  z-index: 1;  
}
```

```
.box-2 {  
  position: relative;  
  z-index: 2;  
}
```

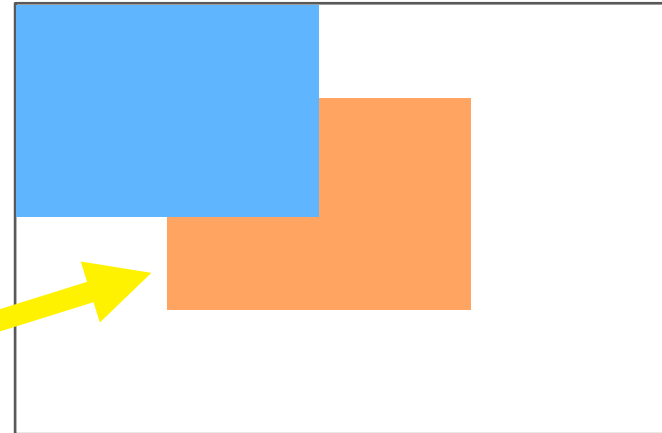




# Z-Index

```
.box-1 {  
  position: absolute;  
  top: 0;  
  left: 0;  
  z-index: 2;  
}
```

```
.box-2 {  
  position: relative;  
  z-index: 1;  
}
```



# Recap

Which **two** CSS position properties remove the element from the normal flow of the document?

Which CSS position property ignores the values of left, right, top, bottom, and z-index?

Which CSS position property can be used to make an element scroll with the page normally but stop scrolling when it reaches the top of the page?

static      relative      absolute      fixed      sticky

# Scroll-padding

The **scroll-padding** property can be used to prevent content appearing behind position: fixed or position: sticky elements when clicking in-page links.

We can set **scroll-padding-top** on the HTML element of our FAQ page to address this issue.

<https://developer.mozilla.org/en-US/docs/Web/CSS/scroll-padding>

# Scroll-behavior

The **scroll-behavior** property can be used to make the browser either jump to an element or smooth scroll to it when clicking an in-page link.

This property has two possible values: **auto** or **smooth**.

<https://developer.mozilla.org/en-US/docs/Web/CSS/scroll-behavior>

# Position + Pseudo-elements

Let's combine CSS Positioning with CSS pseudo-elements.

We'll use CSS to create a symbol at the end of each section element...

Sed vel hendrerit sem. Aenean auctor lacus nec nisl blandit ultrices. Proin tincidunt vehicula vestibulum. Suspendisse blandit felis ornare ante cursus, et blandit enim eleifend. Curabitur nec tortor eu leo facilisis mattis. Proin tempus vitae nisl laoreet pretium. Vivamus tristique eget leo eu elementum. Donec rhoncus posuere eros vitae rhoncus. Nam a lobortis nisl. Praesent eu orci at lectus vestibulum tincidunt a eu dolor. Curabitur tempus dignissim erat, ac consectetur mi congue vitae. Integer quis enim diam.



[https://www.w3schools.com/cssref/css\\_entities.asp](https://www.w3schools.com/cssref/css_entities.asp)

# Assignment #5

# Assignment #5

- Please refer to Assignment #5 in the Learning Hub.
- To submit the assignment, you can do **one** of these:
  - Have me check your assignment in class before 4pm.
  - Zip today's **folder** and upload it to the Learning Hub before next class.
- If you have questions or need guidance, just ask!

# Country Website



# Project #2 – Country Website

Check the Learning Hub for the requirements and links to examples of previous projects.

Today's files also include an example structure of the site. Yours does not have to be identical, it is simply an example.

# Resources

## CSS Floats (MDN)

[https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout/Floats](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Floats)

## CSS Position (MDN)

<https://developer.mozilla.org/en-US/docs/Web/CSS/position>

## CSS Position (CSS-Tricks)

<https://css-tricks.com/almanac/properties/p/position/>

---

# QUESTIONS & ANSWERS