

Web Developer

HTML, CSS e Strumenti di Digital Marketing
(SEO, SEM, SEA)

Docente: Shadi Lahham

Positioning & Display

Page flow

Shadi Lahham - Web development

Display

Property

Display

```
div {  
  display: inline;           /* Default of all elements, unless UA stylesheet overrides */  
  display: inline-block;     /* Characteristics of block, but sits on a line */  
  display: block;           /* block is default for elements like <div> and <section> */  
  display: none;            /* Hide */  
}
```

Inline:

- The default value for elements
- e.g. ``, ``, ``, etc.
- Doesn't break the flow of the text
- The element will accept margin and padding, but the element still sits inline
- Margin and padding will only push other elements horizontally, not vertically
- Important: an inline element will not accept height and width. No effect

Display

Inline Block

- Combines aspects of inline and block
- Very similar to inline in that it will sit inline with the natural flow of text
- Possible to set a width and height

Block

- Some elements are set to block by the browser UA (user agent) stylesheet
- Container elements, like `<div>`, `<section>`, ``, etc.
- Text block elements like `<p>`, `<h1>`, `<h2>`, etc.
- Do not sit inline
- By default take up as much horizontal space as they can

None

- `display:none` removes the element from the document flow
- The element does not take up any space

Hiding elements

There are Several methods to 'hide' elements

display:none

removes the element from the document flow

visibility:hidden

hides the element, but it still takes up space in the layout

opacity:0

hides the element, still takes up space in the layout, events work

Hiding elements

	Collapse	Events	Tab order
<code>display: none</code>	Yes	No	No
<code>visibility: hidden</code>	No	No	No
<code>opacity: 0</code>	No	Yes	Yes

[Hiding elements with display, opacity or visibility? \(Video\)](#)
[CSS Layout - The display Property](#)

Background

Property

Background

- Allows you to control the background of any element
- A shorthand property: allows to write multiple CSS properties in one
- All background definitions are option, but at least one must be stated
- Default values are given to background if some are not defined

```
body {  
  background: transparent image-url('image.png') left top no-repeat;  
}
```

Background

```
body {  
  background:  
    url('texture.jpg')      /* image */  
    top center / 200px 200px /* position / size */  
    no-repeat               /* repeat */  
    fixed                   /* attachment */  
    padding-box             /* origin */  
    content-box             /* clip */  
    red;                    /* color */  
}
```

It's possible to use any combination of properties in any order however the above order is **recommended** to avoid confusion

Anything not specified is automatically set to its default

Background

The background will be transparent, instead of red:

```
body {  
  background-color: red;  
  background: url(texture.jpg);  
}
```

Fixes:

```
body {  
  background: url(texture.jpg);  
  background-color: red;  
}
```

Or:

```
body {  
  background: url(texture.jpg) red;  
}
```

Background

background is made up of eight properties

background-image

background-position

background-size

background-repeat

background-attachment

background-origin

background-clip

background-color

Background

background-image

The path to the image. Examples: `url('../css/image.png')` or `url('../css/image.png')`

background-position

Position of the background relative to the HTML element

Can accept two unit values: X (left offset) and Y (top offset)

Can also accept Keywords: left,center,right and top,center,bottom. More details [here](#)

background-size

Specifies the dimensions of the background image for the element

background-repeat

Whether the background repeats if the width exceeds the background size

Possible values: no-repeat, repeat, repeat-x and repeat-y

Background

[background-attachment](#)

specifies whether the background image should scroll with the page or be fixed

[background-origin](#)

Sets whether the background image will start from the border, padding or content

[background-clip](#)

Decides how the image is clipped (cut)

[background-color](#)

Changes the background color of an element

[CSS Backgrounds](#)

[Background properties](#)

[background-origin - MDN](#)

[background-clip - MDN](#)

Static Positioning

Static Positioning

- All HTML elements are positioned static by default
- Static elements are positioned in the normal flow of the page
- Static elements ignore top, bottom, right, or left property specifications

Static Positioning: Block Elements

In normal flow block elements flow from top to bottom making a new line after every element

```
<p>Greetings</p>  
<p>Hello</p>  
<p>Hi there!</p>
```

Static Positioning: Inline Elements

In normal flow inline elements flow from left to right wrapping to next line when needed

```
  
  
  
  

```

Relative Positioning

Relative Positioning

- Keeps the element in the normal flow but allows it to be moved in relation to the top, bottom, right, or left
- Does not affect the elements surrounding it
- Makes an element a "positioning context" in which to position other elements relative to it
- The relative value will still put the element in the normal flow, but then offset it according to top, bottom, right and left properties

Absolute Positioning

Absolute Positioning

- Positions element outside of the normal flow
- Other elements act as if it's not there
- An absolutely positioned element is offset from its container block, set with the properties top, bottom, right and left
- Its container block is the first surrounding element that has any position other than static
- If no such element is found, the container block is <html>

IN BASE AL PADRE CHE CONTIENE
QUELL'ELEMENTO CHE NON ABBA POSIZIONE
STATIC!! (SE NON NE ESISTONO VA ALL'HTML

Absolute Positioning

```
.top {  
  position: absolute;  
  top: -40px;  
  right: 10px;  
  background-color: yellow;  
}
```

```
.bottom {  
  position: absolute;  
  bottom: -40px;  
  left: 60px;  
  background-color: green;  
}
```

The absolute value takes the element out of the normal flow

It positions it in relation to the window, or the closest non-static ancestor

Example: Absolute Positioning

```
.geo-image {  
  max-width: 450px;  
  margin: 0 auto;  
  width: 100%;  
  position: relative;  
}  
.geo-image img {  
  width: 100%;  
  display: block;  
}  
.geo-image figcaption {  
  background-color: orange;  
  position: absolute;  
  left: -4px;  
  bottom: 8px;  
}
```

```
<figure class="geo-image">  
    
  <figcaption>Forest fires</figcaption>  
</figure>
```



Fixed and Sticky Positioning

Fixed Positioning

```
.footer {  
  position: fixed;  
  bottom: 0;  
  left: 10px;  
  width: calc(100% - 20px);  
  background-color: #0099ff;  
  color: white;  
  font-weight: bold;  
  padding: 4px 8px;  
  box-sizing: border-box;  
}
```

The fixed value takes an element out of the normal flow
It positions it relative to the viewport
Parent positioning will no longer affect fixed elements

Sticky Positioning

```
.header {  
  background: #b8c1c8;  
  border-bottom: 1px solid #989ea4;  
  border-top: 1px solid #717d85;  
  color: #fff;  
  margin: 0;  
  padding: 2px 0 0 12px;  
  position: -webkit-sticky;  
  position: sticky;  
  top: -1px;  
}
```

Sticky positioning is a hybrid of relative and fixed positioning

The element is treated as relative positioned until it crosses a specified threshold, at which point it is treated as fixed positioned

Note: sticky is not supported by IE and still needs to be prefixed for webkit based browsers

Z-index

Property

Z-index

- When elements overlap, the order of overlapping can be changed with z-index
- The element with highest z-index goes on top
- Without z-index, elements stack in the order that they appear in the DOM
- Elements with non-static positioning will always appear on top of elements with default static positioning
- Nesting is important
 - If element B is on top of element A, a child of element A can never be higher than element B

Z-index

```
.bottom {  
  position: absolute;  
  top: 30px;  
  left: 60px;  
  background-color: #ff5722;  
}
```

```
.top {  
  position: absolute;  
  top: 40px;  
  left: 70px;  
  background-color: #cddc39;  
  z-index: 2;  
}
```

```
<div class="top">hello</div>  
<div class="bottom">there</div>
```

Z-index

```
.bottom {  
  position: absolute;  
  top: 30px;  
  left: 60px;  
  background-color: #ff5722;  
}
```

```
.top {  
  position: absolute;  
  top: 40px;  
  left: 70px;  
  background-color: #cddc39;  
  z-index: 200000; /* can never win */  
}
```

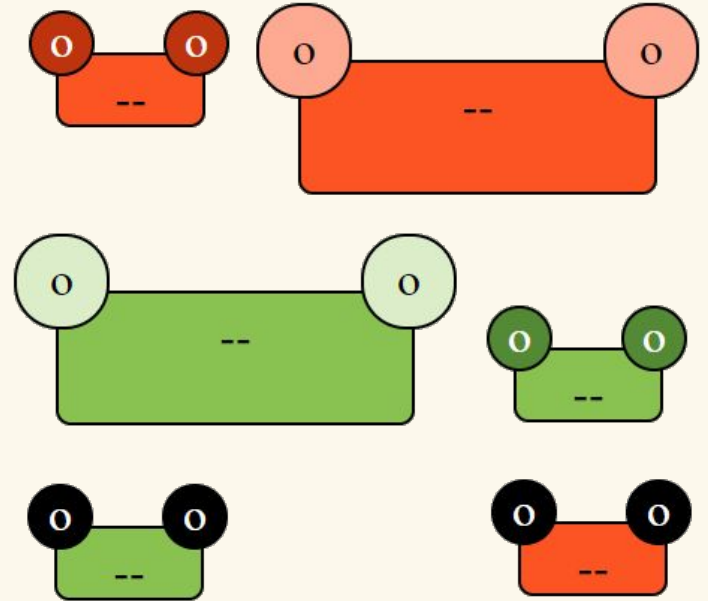
```
.container {  
  position: relative;  
  z-index: 1;  
}
```

```
<div class="container">  
  <div class="top">top thing</div>  
</div>  
<div class="container">  
  <div class="bottom">bottom thing</div>  
</div>
```


Your turn

1. Alien frogs

- On an alien planet you find alien frogs
 - Two types: green and orange
 - And two sizes: big and small
 - Small frogs have dark eyes
 - Big frogs have bright eyes
 - When frogs are touched their eyes become black
- See next page for technical details
- See frogs.gif animation for behaviors



1. Alien frogs

- Create a page full of all possible combinations of alien frogs
- Frogs should fill the page from left to right
- Frogs should wrap if there isn't enough space
- All frogs should change their eye color to black when touched
- The HTML should be valid and complete
- In the readme.md explain your CSS code
- Do all the work in CSS, the HTML should only include elements of this type:

```
<div class="frog type1 small">--</div>  
<div class="frog type2 small">--</div>  
<!-- etc... -->
```

References

[Display - CSS: Cascading Style Sheets](#)

[CSS Layout - The display Property](#)

[Visibility - CSS: Cascading Style Sheets](#)

[Position - CSS: Cascading Style Sheets](#)

[Using z-index - CSS: Cascading Style Sheets](#)