CSS Positioning & Layout

Page structure

Shadi Lahham - Programmazione web - Frontend - HTML e CSS

Property - display

Property: display

```css
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. <span>, <em>, <b>, 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
- An inline element will not accept height and width. No effect

Property: 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
- Able 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>, <ul>, 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
display: none	Yes	No	No
visibility: hidden	No	No	No
opacity: 0	No	Yes	Yes

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

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

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

Property: background

```css
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

Property: 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;
}
```

Property: background

background is made up of eight properties

background-image
background-position
background-size
background-repeat
background-attachment
background-origin
background-clip
background-color

Property: background properties

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

Background-repeat
Whether the background repeats if the width exceeds the background size
Possible values: no-repeat, repeat, repeat-x and repeat-y

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

CSS Backgrounds
Background properties

Property: background properties

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-origin - MDN
background-clip - MDN

Floating elements

Property: Float

- Floating an element moves it as far to the left or right of its containing element as possible
- Other elements, e.g. paragraphs or lists, will wrap around the floated element
- Floated elements without a width will take up the whole width of the container and not appear floated
- It's possible to specify
  - Whether an element is floated or not
  - Which side it floats on

Float - example

```css
.float {
  float: left;
  width: 200px;
  background: #bada55;
  margin: 8px;
  padding: 5px;
}
```

I am a floater

Lorem ipsum dolor sit amet consectetuer accumsan Sed vitae mauris risus. Magna eget a Phasellus a nibh nisl risus netus mauris nec.

```html
<div class="float">I am a floater</div>
<p>Lorem ipsum dolor sit amet consectetuer accumsan Sed vitae mauris risus...</p>
```

Note: floated elements will fill the space from left to right and go to a new line based on remaining space

16

Float - multiple float example

```css
.float {
 float: left;
 width: 200px;
 background: #bada55;
}

.float-right {
 float: right;
 width: 150px;
 background: darkorange;
}
```



```html
<div class="float">I am a floater</div>
<div class="float">I am a floater</div>
<div class="float">I am a floater</div>
<div class="float-right">I am a floater</div>
<div>Lorem ipsum dolor sit amet consectetuer accumsan Sed vitae mauris risus...</div>
```

Float - Placing elements side by side

```css
.float-child-left {
 float: left;
 width: 200px;
 background: #4fc3f7;
}

.float-child-right {
 float: right;
 width: 150px;
 background: #ffa726;
}
```


To place two block level elements to be side by side float both elements; One left and one right

Clear

- Clearing tells the element on which side (right, left, both) other elements cannot appear
- If an image is floated left, to avoid that a paragraph is wrapped next to it, clear:left; should be added to the style of the paragraph
- Clearing both ensures the element doesn't wrap next to floats on either side

```
Examples:
clear: right;
clear: left;
clear: both;
```

Clear - example

```css
.float {
 float: left;
 width: 200px;
 background: #bada55;
 margin: 8px;
 padding: 5px;
}

.clear-both {
 clear: both;
}
```

```html
<div class="float">I am a floater</div>
<div class="clear-both">cleared: Lorem ipsum dolor sit amet consectetuer </div>
```

I am a floater

cleared: Lorem ipsum dolor sit amet consectetuer

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

```
<img src="img/otter.jpg" />
<img src="img/otter.jpg" />
<img src="img/otter.jpg" />
<img src="img/otter.jpg" />
<img src="img/otter.jpg" />
```

Relative Positioning

Relative Positioning

- Takes the element out of the normal flow, allowing 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

Relative positioning example

```css
.post-it {
 background-color: orange;
 padding: 5px;
 width: 100px;
 height: 100px;
 box-sizing: border-box;
 position: static;
 box-shadow: 0 0 7px 0 darkgray;
 left: 200px; /* no impact */
}

#two {
 background-color: yellow;
 position: relative;
 left: 20px;
 top: 2px;
}
```

```html
<div id="one" class="post-it">one</div>
<div id="two" class="post-it">two</div>
<div id="three" class="post-it">three</div>
```

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>

Absolute Positioning

```css
.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

```css
.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;
}
```

```html
<figure class="geo-image">
    <img src="img/fires.jpg"
         alt="fires in russia">
    <figcaption>Forest fires</figcaption>
</figure>
```


Forest fires in Russia

Fixed and Sticky Positioning

Fixed Positioning

```css
.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

```css
.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

Property: Z-index - example

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

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

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

Your turn

1.Columns columns

- Create a page with
  - Header, footer, sidebar, and two main columns
- Header and footer
  - Should stay in place when the page scrolls
- The sidebar and two columns
  - Should fill the rest of the page, vertically and horizontally
  - Should expand proportionally when window is resized
- One of the columns should contain
  - An image and a caption placed on top of the image
- Use semantic HTML5 elements
- Fill the page with meaningful content
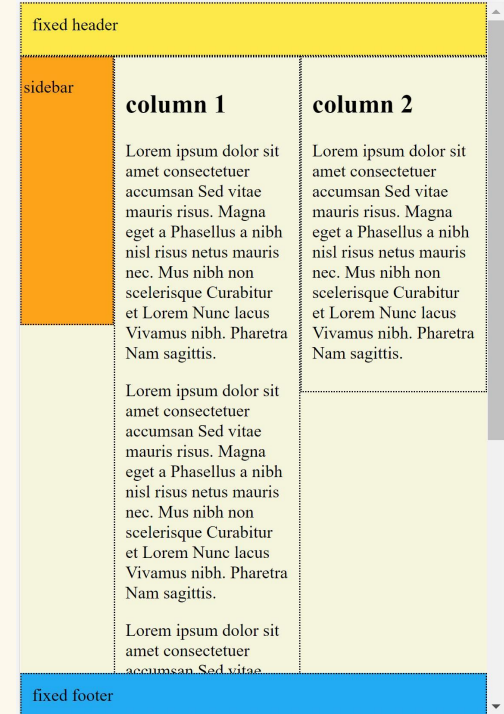
2.Alien frogs

- 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
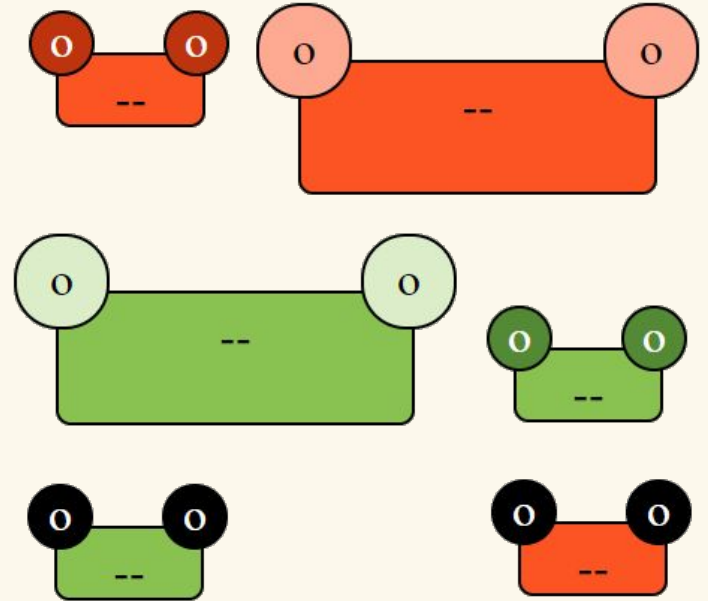
2.Alien frogs

- Create a page full of all possible combinations of aline 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:

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

Bonus

3.United nations

- Make a <ul> list of countries
  - Each list item should contain: the flag, the name and the population number
  - The name should be left aligned and the population should be right aligned
  - Countries with large populations should be highlighted in some way
  - The flags area should have a background color
  - If the flag is not rectangular (e.g. Switzerland) the flag should be centered in the flag area
  - Include at least one country with an irregularly sized flag (e.g Switzerland)
  - The list should contain at least 50 countries and should be scrollable
- Include a header above the list
  - It should contain the following titles: 'Flag', 'Country', 'Population'
  - It should be spaced and aligned properly
  - It should remain in place when the list is scrolled
  - Not to be confused with the <header> tag

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