{POWER.CODERS}

CSS Best Practice

Agenda

Today we will learn about

- Quiz CSS
- CSS Cascade
- Text formatting
- Backgrounds
- Online ressources

CSS Quiz

What is CSS?

What is CSS?

Cascading **S**tyle **S**heets say how your HTML looks like.

What is CSS?

Cascading **S**tyle **S**heets say how your HTML looks like.

- "This element…"
- "… or these elements…"
- "… should look like this"

What is CSS?

Cascading **S**tyle **S**heets say how your HTML looks like.

- "This element…"
- "… or these elements…"
- "… should look like this"

These are CSS **rules**.

Give me examples of an CSS rule

via Chat

Give me examples of an CSS rule

via Chat

```css
p {
  color: pink;
  font-weight: bold;
  margin-left: 2em;
}
```

What is a selector?

What is a selector?

The **selector** defines which element(s) the rule is applied to.
There are 3 types:

What is a selector?

The **selector** defines which element(s) the rule is applied to. There are 3 types:

- Selecting **all** elements of a particular **type**, e.g. `p {}` or `p strong {}`

What is a selector?

The **selector** defines which element(s) the rule is applied to. There are 3 types:

- Selecting **all** elements of a particular **type**, e.g. `p {}` or `p strong {}`
- Selecting **multiple** elements by using a **class**, e.g. `.warning {}` or `p.warning {}`

What is a selector?

The **selector** defines which element(s) the rule is applied to. There are 3 types:

- Selecting **all** elements of a particular **type**, e.g. `p {}` or `p strong {}`
- Selecting **multiple** elements by using a **class**, e.g. `.warning {}` or `p.warning {}`
- Selecting **one** element by using an unique **id**, e.g. `#about {}`

Best practice

Start with generic styles on elements and get more specific with classes and ids.

- Find first the **similarities** in your design (=generic) and afterwards the **differences** (=specific).

What is a declaration?

What is a declaration?

The selector is followed by a block of **one or more declarations**. The block starts and ends with the opening and closing curly brackets `{ }`.

What is a declaration?

The selector is followed by a block of **one or more declarations**. The block starts and ends with the opening and closing curly brackets `{ }`.

Inside the block each declaration consists of the name of a **property**, a colon, and the **value** for the property. And at the end there is a semi-colon.

RECAP: CSS RULE

```
selector {
  property_1: value_1;
  property_2: value_2;
  property_3: value_3;
  ...
}
```

The block `{ }` is called declaration.

Best practice

Always use the same structure / order inside a rule, e.g.

- go by alphabet: `color` before `width`
- go by type: group colors together, styles of font-styling, e.g. `font-size`, then block-styling, e.g. `height` and `width`
- merge both or find your own structure

BEST PRACTICE

```
* {
  color: black;
  font-family: Arial, sans-serif;
  font-size: 1em;
}

h1 {
  color: pink;
  font-family: Georgia, serif;
  font-size: 3em;
}
```

What is the box model?

What is the box model?

The box model is used to describe an element's dimensions and structure. It is made up of four boxes: content box, padding box, border box, and margin box

How can you build a basic layout in CSS?

How can you build a basic layout in CSS?

- Flex
- Grid
- Multi columns
- Float

Example Grid

```
.parent {
  display: grid;
  place-items: center;
}
```

Example Flex

```
.parent {
  display: flex;
}

.child {
  flex: 0 1 150px;
}
```

Tell me some properties and what they do?

How do you connect your CSS with your HTML?

How do you connect your CSS with your HTML?

In 3 ways

How do you connect your CSS with your HTML?

In 3 ways

- External

How do you connect your CSS with your HTML?

In 3 ways

- External
- Embedded (or internal)

How do you connect your CSS with your HTML?

In 3 ways

- External
- Embedded (or internal)
- Inline

Connecting CSS to HTML: External

```
<head>
  <link rel="stylesheet" href="style.css">
</head>
```

- Can be referenced from multiple pages.
- Reduced file size & bandwidth.
- Easier to maintain in larger projects.

Connecting CSS to HTML: Embedded

```
<head>
  <style>
    p {
      color: blue;
      font-size: 12px;
    }
  </style>
</head>
```

- Inside `<head>` element.
- Uses `<style>` tag.
- Only applies to that page.

Connecting CSS to HTML: Inline

```
<p style="color:red">Some text.</p>
```

- Uses the HTML `style` attribute.
- Only applies to one element at a time.
- Not recommended to use.

Best practice

Use external css files.

Best practice

Use external css files.

- order rules from **big** to **small**
- order rules from the **outside** to the **inside**
- add comments and whitespace

BEST PRACTICE

```
/* always declare html element first - it's the biggest */
html {
  background : #ffffff;
  box-sizing: border-box;
  font-size : 10px;
  /*font-size : 0.75vw;*/
  line-height : 1;
  height : 100%;
  scroll-behavior: smooth;
  overflow-x : hidden;
  width : 100%;
}

/* This selects ALL elements. Add it AFTER the html selector */
* {
  color : #333333;
```

CSS Cascade

What is Cascading?

What is Cascading?

The CSS Cascade is the way our browsers **resolve competing** CSS declarations.

What is Cascading?

The CSS Cascade is the way our browsers **resolve competing** CSS declarations.

Every time we add a CSS rule, it enters the CSS cascade.

What is Cascading?

The CSS Cascade is the way our browsers **resolve competing** CSS declarations.

Every time we add a CSS rule, it enters the CSS cascade.

The **further down** the cascade a declaration falls, the **less likely** it will end up as the final style.

IMPORTANCE

The first tier of the Cascade looks at the type of rule

IMPORTANCE

The first tier of the Cascade looks at the type of rule

1. **transition**: Rules that apply to an active transition take the utmost importance

Importance

The first tier of the Cascade looks at the type of rule

1. **transition**: Rules that apply to an active transition take the utmost importance
2. `!important`: When we add !important to the end of our declaration, it jumps to this level of the Cascade. Ideally, you reserve this level for Hail Marys, which are needed to override styles from third-party libraries.

Importance

The first tier of the Cascade looks at the type of rule

1. **transition**: Rules that apply to an active transition take the utmost importance
2. `!important`: When we add !important to the end of our declaration, it jumps to this level of the Cascade. Ideally, you reserve this level for Hail Marys, which are needed to override styles from third-party libraries.
3. **animation**: Rules that apply to an active animation jump up a level in the Cascade

Importance

The first tier of the Cascade looks at the type of rule

1. **transition**: Rules that apply to an active transition take the utmost importance
2. `!important`: When we add !important to the end of our declaration, it jumps to this level of the Cascade. Ideally, you reserve this level for Hail Marys, which are needed to override styles from third-party libraries.
3. **animation**: Rules that apply to an active animation jump up a level in the Cascade
4. **normal**: This level is where the bulk of rules live

Which rule would win?

```
p {
  color: orchid !important;
}

... [many rules in between] ...

p {
  color: sandybrown;
}
```

Which rule would win?

```
p {
  color: orchid !important;
}

... [many rules in between] ...

p {
  color: sandybrown;
}
```

The color of the p-tags would be orchid.
Remember that `!important` declarations fall on the second level, while normal declarations fall on the fourth level.

Origin

The second tier of the Cascade looks at where the rule was defined

Origin

The second tier of the Cascade looks at where the rule was defined

1. **website**: This is the only level that you have control over, as a web developer.

Origin

The second tier of the Cascade looks at where the rule was defined

1. **website**: This is the only level that you have control over, as a web developer.
2. **user**: The user can change styles via browser settings or Code Inspector.

Origin

The second tier of the Cascade looks at where the rule was defined

1. **website**: This is the only level that you have control over, as a web developer.
2. **user**: The user can change styles via browser settings or Code Inspector.
3. **browser**: Each browser has its own set of styles called `user agent stylesheet`.

Specificity

The third tier of the Cascade looks at the Specificity of a rule.

SPECIFICITY

The third tier of the Cascade looks at the Specificity of a rule.

1. **inline**: Styles declared within a `style` HTML property are the most specific

SPECIFICITY

The third tier of the Cascade looks at the Specificity of a rule.

1. **inline**: Styles declared within a `style` HTML property are the most specific
2. **id**: Targeting elements based on their **id**, using the syntax `#id`

Specificity

The third tier of the Cascade looks at the Specificity of a rule.

1. **inline**: Styles declared within a `style` HTML property are the most specific
2. **id**: Targeting elements based on their **id**, using the syntax `#id`
3. **class | attribute | pseudo-class**: Targeting elements based on their **class**, using the syntax `.class`

   This level also includes **attribute selectors** that target HTML attributes, like `[checked]` and `[href="https://powercoders.org"]`, and **pseudo-selectors**, like `:hover` and `:first-of-type`

Specificity

The third tier of the Cascade looks at the Specificity of a rule.

1. **inline**: Styles declared within a `style` HTML property are the most specific
2. **id**: Targeting elements based on their **id**, using the syntax `#id`
3. **class | attribute | pseudo-class**: Targeting elements based on their **class**, using the syntax `.class`
   This level also includes **attribute selectors** that target HTML attributes, like `[checked]` and `[href="https://powercoders.org"]`, and **pseudo-selectors**, like `:hover` and `:first-of-type`
4. **type | pseudo-element**: Targeting elements based on their **tag type**, using the syntax `type`
   This level also includes **pseudo-elements**, like `::before` and `::selection`

Pseudo-class vs pseudo-element

Pseudo-class vs pseudo-element

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

Pseudo-class vs pseudo-element

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

A **pseudo-element** is something that acts like an element, but is **not** an element. Thus, it **cannot** be manipulated by JavaScript.

Fight time: Which rule wins?

A

```
<p style="color: sandybrown">...</p>
```

B

```
p {color: orchid;}
```

Fight time: Which rule wins?

A

```
<p style="color: sandybrown">...</p>
```

B

```
p {color: orchid;}
```

A wins! Remember that **inline** styles fall on the first level, while **type rules** fall on the fourth level.

Fight time: Which rule wins?

A

```
.paragraph {color: sandybrown;}
```

B

```
#paragraph {color: orchid;}
```

Fight time: Which rule wins?

A

```
.paragraph {color: sandybrown;}
```

B

```
#paragraph {color: orchid;}
```

B wins! Remember that rules with a **class** selector fall on the third level, while rules with an **id** selector fall on the second level.

THE NUMBER OF HITS MATTER

A

```
.paragraph:first-of-type {color: sandybrown;}
```

B

```
p.paragraph {color: orchid;}
```

THE NUMBER OF HITS MATTER

A

```
.paragraph:first-of-type {color: sandybrown;}
```

B

```
p.paragraph {color: orchid;}
```

Rule **A** has two "hits" on the **third level** (1 **class** and 1 **pseudo-class**), whereas Rule **B** has only one "hit" on the **third level** - its "hit" on a lower (fourth) level doesn't come into play.

Lower level counts if number of hits match

A

```
p#paragraph {color: sandybrown;}
```

B

```
#paragraph.paragraph {color: orchid;}
```

Lower level counts if number of hits match

A

```
p#paragraph {color: sandybrown;}
```

B

```
#paragraph.paragraph {color: orchid;}
```

Rules **A** and **B** both have 1 hit on the **second level** (1 id), but Rule **B** additionally has 1 hit on the **third level** (1 class), which beats Rule **A**'s hit on the **fourth level** (1 tag).

Position

The fourth and final tier looks at the order that the rules were defined in.

Rules lower in the file overwrite rules higher in the file

```
p {
  color: sandybrown;
}
p {
  color: orchid;
  color: black;
}
```

CSS Cascade: summary

There are four tiers in the cascade. The further down the cascade a declaration falls, the less likely it will end up as the final style.

1. Importance
2. Origin
3. Specificity
4. Position

CSS RESET / NORMALIZE

In order to deal with different browser styles people came up with stylesheets which would **reset** these styles or **normalize** them.

- https://bitsofco.de/a-look-at-css-resets-in-2018/
- https://medium.com/@tinydinosaur/a-wordy-history-of-default-browser-styles-and-css-resets-befdd614d93b

CSS RESET / NORMALIZE

In order to deal with different browser styles people came up with stylesheets which would **reset** these styles or **normalize** them.

- https://bitsofco.de/a-look-at-css-resets-in-2018/
- https://medium.com/@tinydinosaur/a-wordy-history-of-default-browser-styles-and-css-resets-befdd614d93b

Even with reset or normalizing stylesheets in place, browser testing is mandatory.

CSS COLORS

CSS Color Values

Your browser can accept colors in many different ways:

Color name:	`red`
Hexadecimal value:	`#FF0000`
RGB value:	`rgb(255, 0, 0)`
RGBA value:	`rgba(255, 0, 0, 0)`
HSL value:	`hsl(0, 100%, 100%)`

W3Schools Color Picker

CSS Color Names

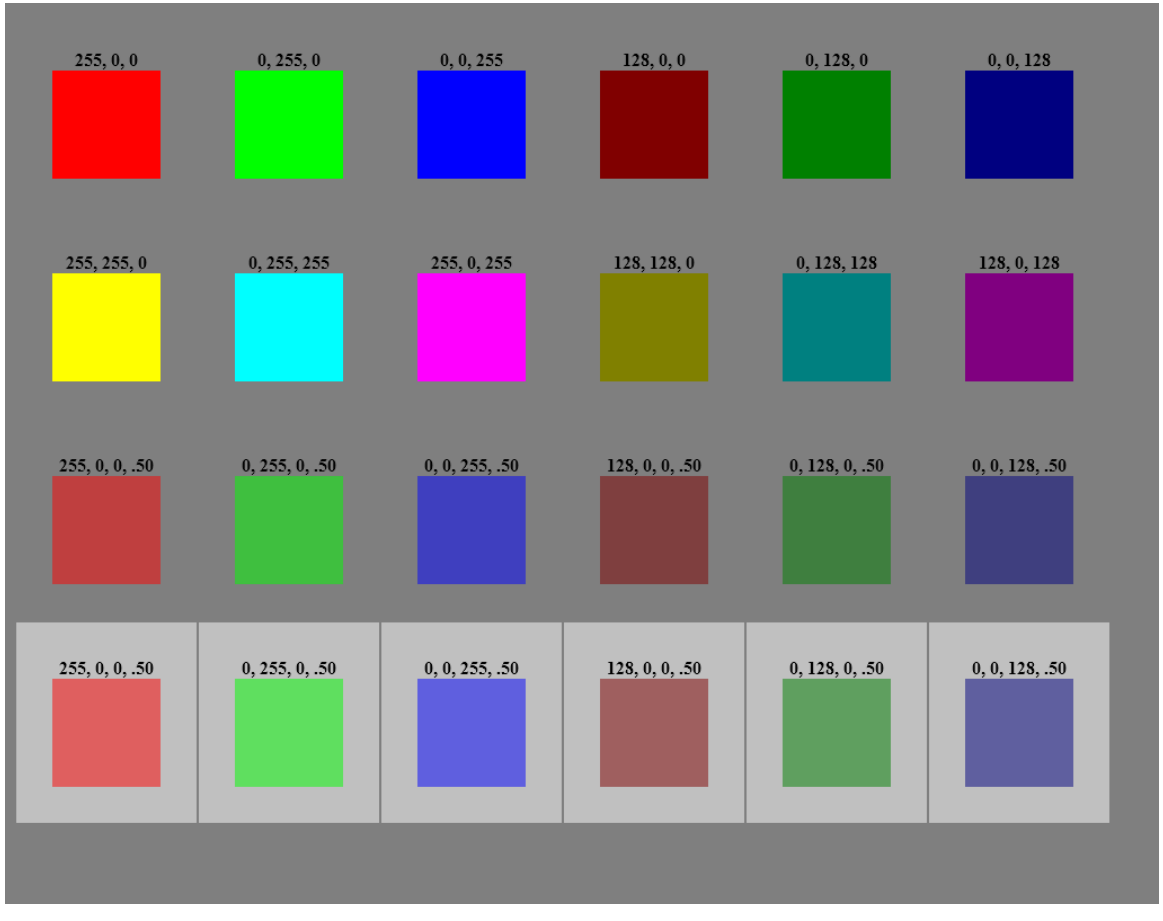
Best practice

- hex codes
- rgba

Quick color theory

- RGBA = Red, Green, Blue, Alpha
- R, G, B in range [0 - 255]
  - 0 = none of that colour
  - 255 = maximum amount of that color
- Alpha in range [0.0 - 1.0]
  - 0.0 = completely transparent, 0% of the color
  - 1.0 = completely opaque, 100% of the color
  - 0.5 = 50% of the color
  - 0.25 = 25% of the color, etc

Color examples



Quick color theory

- hexadecimal codes are also based on RGB
- RGB in range [00 - FF]
    - 00 = none of that colour
    - FF = maximum amount of that color
- There is no transparency possible with hex codes

Quick color theory

- hexadecimal codes are also based on RGB
- RGB in range [00 - FF]
    - 00 = none of that colour
    - FF = maximum amount of that color
- There is no transparency possible with hex codes

Examples:

QUICK COLOR THEORY

- hexadecimal codes are also based on RGB
- RGB in range [00 - FF]
  - 00 = none of that colour
  - FF = maximum amount of that color
- There is no transparency possible with hex codes

Examples:

- #FFFFFF - white
- #0000FF - **blue**
- #FF0000 - **red**
- #00FF00 - **green**
- #000000 - **black**

CSS COLOR PROPERTIES

- `color`
- `background-color`
- `border-color`
- `text-decoration-color`
- `...`

Best practice

- Use only a **few colors**. Usually 1 text color, 1 link color (plus 1 hover), 1 highlighting color.

Best practice

- Use only a **few colors**. Usually 1 text color, 1 link color (plus 1 hover), 1 highlighting color.
- Use colors in a **logical way**, e.g. not the same color for link (with action) and highlight (without action).

Best practice

- Use only a **few colors**. Usually 1 text color, 1 link color (plus 1 hover), 1 highlighting color.
- Use colors in a **logical way**, e.g. not the same color for link (with action) and highlight (without action).
- Let them be readable for anyone and choose a **good contrast** between text and background.

CSS FONTS

- CSS gives a lot of control over the appearance of text
- Collectively, these are the `font-*` properties.

font-family

```
<!DOCTYPE html>
<html>
<head>
  <style>
      h1.serif { font-family: serif; }
      h1.sans { font-family: sans-serif; }
      h1.mono { font-family: monospace; }
    </style>
  </head>
  <body>
    <h1 class="serif">This is serif</h1>
    <h1 class="sans">This is sans-serif</h1>
    <h1 class="mono">This is monospace</h1>
  </body>
</html>
```

Serif fonts

Serif fonts are called that because they have the little sticking out bits at the ends of bits of the letters. For example, at the left and right ends of the top of the T.

Sans-serif fonts

Sans-serif literally means "without serifs" – as you can see in the second heading these letters have no sticking out bits.

Monospace fonts

And monospace fonts might be serif or sans-serif. Their main characteristic is that all the letters are the same width.

Default fonts

The browser defines the default fonts.

- chrome://settings/fonts

A FONT STACK

```
body {
  font-family: 'DejaVu Serif', 'Times New Roman', Times, serif;
}

h1, h2, h3, h4, h5, h6 {
  font-family: 'DejaVu Sans', Arial, Helvetica, sans-serif;
}

pre, code, tt {
  font-family: 'DejaVu Sans Mono', monospace;
}
```

What is a font stack?

You can list multiple fonts for a `font-family` entry. This is called a **font stack**.

When the browser sees a font stack, it checks to see if the first font in the list is installed on the user's computer. If it is, it uses it.

If it's not, it moves on to the next font in the list. So you should list the fonts in the stack from most specific to least specific.

"Web safe" fonts

Names / Font stack	Type
Helvetica, Arial	sans-serif
Courier New, Courier	monospace
Georgia	serif
Times New Roman, Times	serif
Verdana	sans-serif

MDN Web Safe fonts

Downloadable web fonts

```
@font-face {
  font-family: 'Open Sans';
  src: local('OpenSans Regular'),
       url('/fonts/OpenSans-Regular-webfont.woff2') format('woff2'),
       url('/fonts/OpenSans-Regular-webfont.woff') format('woff');
}
```

Google Fonts

https://fonts.google.com

One of several online repositories of free fonts that make it very easy to include them in web pages

Google Fonts

https://fonts.google.com

One of several online repositories of free fonts that make it very easy to include them in web pages

Try to find the Lato font

Google Fonts

https://fonts.google.com

One of several online repositories of free fonts that make it very easy to include them in web pages

Try to find the Lato font

```
<link href="https://fonts.googleapis.com/css?family=Lato" rel="stylesheet">
```

Fonts best practice

- Always use font stacks
- Only use 1-2 different fonts per website
- Use monospace for `code`

Icon fonts

Icon fonts are text files that can be modified using CSS.

Pro:

Icon fonts

Icon fonts are text files that can be modified using CSS.
Pro:

- Scale better than images

Icon fonts

Icon fonts are text files that can be modified using CSS.
Pro:

- Scale better than images
- Change color, size etc. via CSS

Icon fonts

Icon fonts are text files that can be modified using CSS.

Pro:

- Scale better than images
- Change color, size etc. via CSS

Con:

Icon fonts

Icon fonts are text files that can be modified using CSS.

Pro:

- Scale better than images
- Change color, size etc. via CSS

Con:

- Additional request to server

Icon fonts

Icon fonts are text files that can be modified using CSS.

Pro:

- Scale better than images
- Change color, size etc. via CSS

Con:

- Additional request to server
- Flash of invisible icons while font is loading

Icon fonts

Icon fonts are text files that can be modified using CSS.

Pro:

- Scale better than images
- Change color, size etc. via CSS

Con:

- Additional request to server
- Flash of invisible icons while font is loading
- Some browser might not be able to interpret the font

SVG

... stands for Scalable Vector Graphics.

SVG

... stands for Scalable Vector Graphics.

- Best responsive image choice, as it can **infinitely scale**.

SVG

... stands for Scalable Vector Graphics.

- Best responsive image choice, as it can **infinitely scale**.
- Possible to use CSS and Javascript (JS) to interact with SVG.

SVG

... stands for Scalable Vector Graphics.

- Best responsive image choice, as it can **infinitely scale**.
- Possible to use CSS and Javascript (JS) to interact with SVG.
- Mainly used for icons, logos and infographics.

SVG

... stands for Scalable Vector Graphics.

- Best responsive image choice, as it can **infinitely scale**.
- Possible to use CSS and Javascript (JS) to interact with SVG.
- Mainly used for icons, logos and infographics.

It is both an image format and a document format. There's so much possible with it.

The future of icons

SVGs are quickly becoming the new standard of web icons and animations.

They offer

The future of icons

SVGs are quickly becoming the new standard of web icons and animations.

They offer

- superior scaling

The future of icons

SVGs are quickly becoming the new standard of web icons and animations.

They offer

- superior scaling
- faster rendering than icon fonts

The future of icons

SVGs are quickly becoming the new standard of web icons and animations.

They offer

- superior scaling
- faster rendering than icon fonts
- smaller file size than jpg or png

The future of icons

SVGs are quickly becoming the new standard of web icons and animations.

They offer

- superior scaling
- faster rendering than icon fonts
- smaller file size than jpg or png
- better accessibility

The future of icons

SVGs are quickly becoming the new standard of web icons and animations.

They offer

- superior scaling
- faster rendering than icon fonts
- smaller file size than jpg or png
- better accessibility
- extensive customization

`font-size`

`font-size` sets the height of the font.

default font-size in the browser is set to `16px`

`font-size`

`font-size` sets the height of the font.

default font-size in the browser is set to `16px`

16px = 1em

Measurement units

- `pt`: **Points** are a fixed unit, from the printing era. 1 pt = 1/72 of an inch. Only used for letter-spacing or word-spacing.
- `px`: **Pixels** are a fixed unit. 1 Pixel is the smallest unit of a digital image that can be displayed.
- `%`: **Percentage** is a scalable unit and calculates the size based on the default 16px (= 100%).
- `em`: **Scalable unit** and calculates the size based on the default 16 px (= 1em). The em takes the width of the **M** as base.
- `rem`: **Scalable unit** similar to `em`. The r in rem stands for **root**. The rem unit is **relative to the root** (html) element.
- `vw / vh`: **Scalable** unit based on **viewport width / height**. More used for block-level elements than text.
- ... and more

`em` versus `rem`

Try it yourself

- Add a new section to a HTML file
- Add several paragraphs within the section
- Emphasise some text as `strong` or `em` into each paragraph
- Set the section font-size to 1em
- Set each paragraph to a different font-size in em
- What happens?

`em` versus `rem`

Try it yourself

- Add a new section to a HTML file
- Add several paragraphs within the section
- Emphasise some text as `strong` or `em` into each paragraph
- Set the section font-size to 1em
- Set each paragraph to a different font-size in em
- What happens?

Now change `em` to `rem`. What changes?

font-weight

says how bold the text should be

```
strong {
  font-weight: bold; /* Default */
  font-weight: 700; /* Default */
}
```

```
strong {
  font-weight: normal; /* Disabled bold */
  font-weight: 400; /* Default */
}
```

```
.bolder {
  font-weight: bolder; /* bolder than bold */
  font-weight: 800; /* Default */
}
```

```
.lighter {
  font-weight: lighter; /* less bold */
```

```
  font-weight: 200; /* Default */
}
```

font-style

used for italic variations, rare

```
em {
  font-style: italic; /* Default */
}
```

```
em {
  font-style: normal; /* Disabled italic */
}
```

```
em {
  font-style: oblique; /* simulates italic */
}
```

line-height

- defines the extra space around each line
- Gives the text some "breathing room"
- Value is a multiplier of the font-size

```
body {
  font-height: 1.45;
}
```

`font` shorthand

```
p {
  font: italic normal bold normal 3em/1.5 Helvetica, Arial, sans-serif;
}
```

```
p {
  font-style: italic;
  font-variant: normal;
  font-weight: bold;
  font-stretch: normal;
  font-size: 3em;
  line-height: 1.5;
  font-family: Helvetica, Arial, sans-serif;
}
```

CSS BACKGROUND

Background properties

```
background-color: lightblue;
```

Background properties

```
background-color: lightblue;
background-image: url("img_tree.gif");
```

Background properties

```
background-color: lightblue;
background-image: url("img_tree.gif");
background-position: 0 0;
```

Background properties

```
background-color: lightblue;
background-image: url("img_tree.gif");
background-position: 0 0;
background-size: cover;
```

Background properties

```
background-color: lightblue;
background-image: url("img_tree.gif");
background-position: 0 0;
background-size: cover;
background-repeat: no-repeat;
```

Background properties

```
background-color: lightblue;
background-image: url("img_tree.gif");
background-position: 0 0;
background-size: cover;
background-repeat: no-repeat;
background-origin: content-box;
```

Background properties

```
background-color: lightblue;
background-image: url("img_tree.gif");
background-position: 0 0;
background-size: cover;
background-repeat: no-repeat;
background-origin: content-box;
background-clip: content-box;
```

Background properties

```
background-color: lightblue;
background-image: url("img_tree.gif");
background-position: 0 0;
background-size: cover;
background-repeat: no-repeat;
background-origin: content-box;
background-clip: content-box;
background-attachment: fixed;
```

Background properties

```
background-color: lightblue;
background-image: url("img_tree.gif");
background-position: 0 0;
background-size: cover;
background-repeat: no-repeat;
background-origin: content-box;
background-clip: content-box;
background-attachment: fixed;
```

```
background: lightblue url("img_tree.gif") no-repeat
fixed center / cover;
```

Multiple backgrounds

You can add multiple backgrounds as layers on top of each other by adding them in one line, seperated by comma.

```
background-color: white;
background-image:
 linear-gradient(45deg, black 25%, transparent 25%, transparent 75%, black 75%),
 linear-gradient(45deg, black 25%, transparent 25%, transparent 75%, black 75%);
background-size:100px 100px;
background-position: 0 0, 50px 50px;
```

ANIMATED BACKGROUNDS

```
body {
 width: 100%;
 height: 100vh;
 background: linear-gradient(-45deg, #ee7752, #e73c7e, #23a6d5, #23d5ab);
 background-size: 400% 400%;
 animation: gradientBG 15s ease infinite;
}

@keyframes gradientBG {
 0% {
  background-position: 0% 50%;
 }
 50% {
  background-position: 100% 50%;
 }
 100% {
```

Online ressources

- https://www.cssbasics.com/
- https://tiny.cc/w53zcz
- https://alistapart.com/article/journey/
- https://typespiration.com/
- https://contrastchecker.com/
- https://developer.mozilla.org/en-US/docs/Learn/CSS/Styling_text

Online Ressources: Images and Patterns

- UNSPLASH - Free images
- Subtle background patterns
- More background patterns
- Gradient Pattern Gallery by Lea Verou
- Ultimate CSS3 Gradient Generator

Online ressources: Web fonts

- Free icon font: Font awesome
- Icon font generator: Icon Moon
- Icon font generator: Glypther

EXERCISES

EXERCISES: CASCADE

1. Look at the source code and try to find out how the paragraph in each exercise will look like.
2. Explain in the group why? What rules apply, how does the cascade work in the specific case.
3. Copy the code to your editor (VSC or replit) and see if you are right.

Cascade 1

What color is the paragraph?

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body p { color: rgb(0, 128, 0); } /*green*/
      html p { color: rgb(128, 0, 128); } /*purple*/
    </style>
  </head>
  <body>
    <p>Here is a paragraph</p>
  </body>
</html>
```

CASCADE 2

How does the paragraph look like?

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p {
        color: rgb(255, 255, 255);
        background-color: rgb(0, 0, 0);
      }
      p.info {
        background-color: rgba(0, 0, 255, 0.5);
      }
    </style>
  </head>
  <body>
    <p class="info">Here is a paragraph</p>
  </body>
```

Cascade 3

What will the paragraph look like?

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p {
        color: rgb(255, 255, 255);
        background-color: rgb(0, 0, 0);
        text-decoration: none;
        border: 1px solid rgb(0, 0, 0);
      }
      p.info {
        background-color: rgba(0, 0, 255, 0.5);
        text-decoration: underline;
        padding: 1em;
      }
      p#tip {
```

Google Fonts

Choose three fonts from Google Fonts

- One sans-serif, for headings
- One serif, for almost everything else
- One monospace

Add them to your replit assignment and use them

Play a game

CSS Diner

Work on your project

Can you already add the first styles to your website project?

If you need some inspiration, you can check out sites like themeforest.net for a layout you like. BUT you have to write your own css, don't copy :)