

# CSS for Software Engineers for CSS Developers



# Harry Roberts

<http://csswizardry.com>



inuitcss

1959

2015



Flow Matic



CSS



1996



5

# Principles of Software Engineering





1

DRY

Don't repeat yourself



“Every piece of knowledge must have a **single, unambiguous, authoritative** representation within a system”

[wikipedia.org/wiki/Don't\\_repeat\\_yourself](https://wikipedia.org/wiki/Don't_repeat_yourself)



“**DRY** is not about no repeating **anything** but is about no  
repeating yourself”

H. Roberts

“If you manually type a declaration 50 times in a project, **you are repeating yourself**: this is not DRY. If you can generate that declaration 50 times without having to manually repeat it, this is DRY: **you are generating repetition without actually repeating yourself**. This is quite a subtle but important distinction to be aware of”

[csswz.it/1ytQkxp](https://csswz.it/1ytQkxp)





```
.u-margin-top      { margin-top:      12px; }  
.u-margin-right    { margin-right:    12px; }  
.u-margin-bottom   { margin-bottom:   12px; }  
.u-margin-left     { margin-left:     12px; }
```

`$unit : 12px`

```
.u-margin-top      { margin-top:    $unit; }  
.u-margin-right    { margin-right:   $unit; }  
.u-margin-bottom   { margin-bottom:  $unit; }  
.u-margin-left     { margin-left:    $unit; }
```



```
.page-title {  
    font-family: "Custom Font", sans-serif;  
    font-weight: 700;  
}
```

```
.btn {  
    font-family: "Custom Font", sans-serif;  
    font-weight: 700;  
}
```

```
.pagination {  
    font-family: "Custom Font", sans-serif;  
    font-weight: 700;  
}
```

```
@mixin custom-font() {  
  font-family: "Custom Font", sans-serif;  
  font-weight: 700;  
}
```

```
.page-title {  
  @include custom-font();  
}
```

```
.btn {  
  @include custom-font();  
}
```

```
.pagination {  
  @include custom-font();  
}
```

Gives the exact same  
output, but at least we  
haven't duplicated  
anything manually



```
.btn {  
  color : white;  
  font-weight : bold;  
}
```

```
.calendar__title {  
  font-size : 14px;  
  font-weight : bold;  
}
```

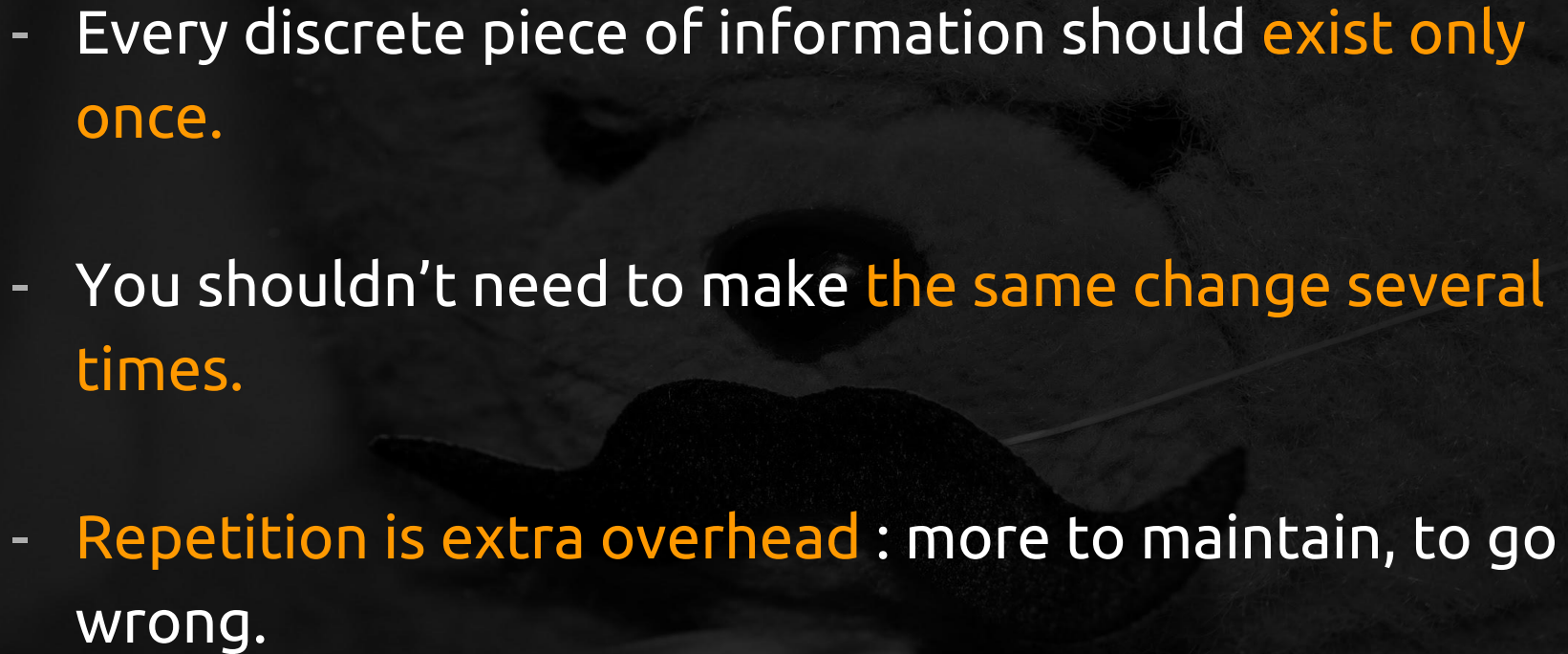
```
.message {  
  font-weight : bold;  
}
```

*This is purely  
coincidental.  
Don't try to  
DRY it out.*



“Repetition is better than wrong abstraction”

H. Roberts

- 
- Every discrete piece of information should **exist only once**.
  - You shouldn't need to make **the same change several times**.
  - **Repetition is extra overhead** : more to maintain, to go wrong.



## The single Responsibility Principle



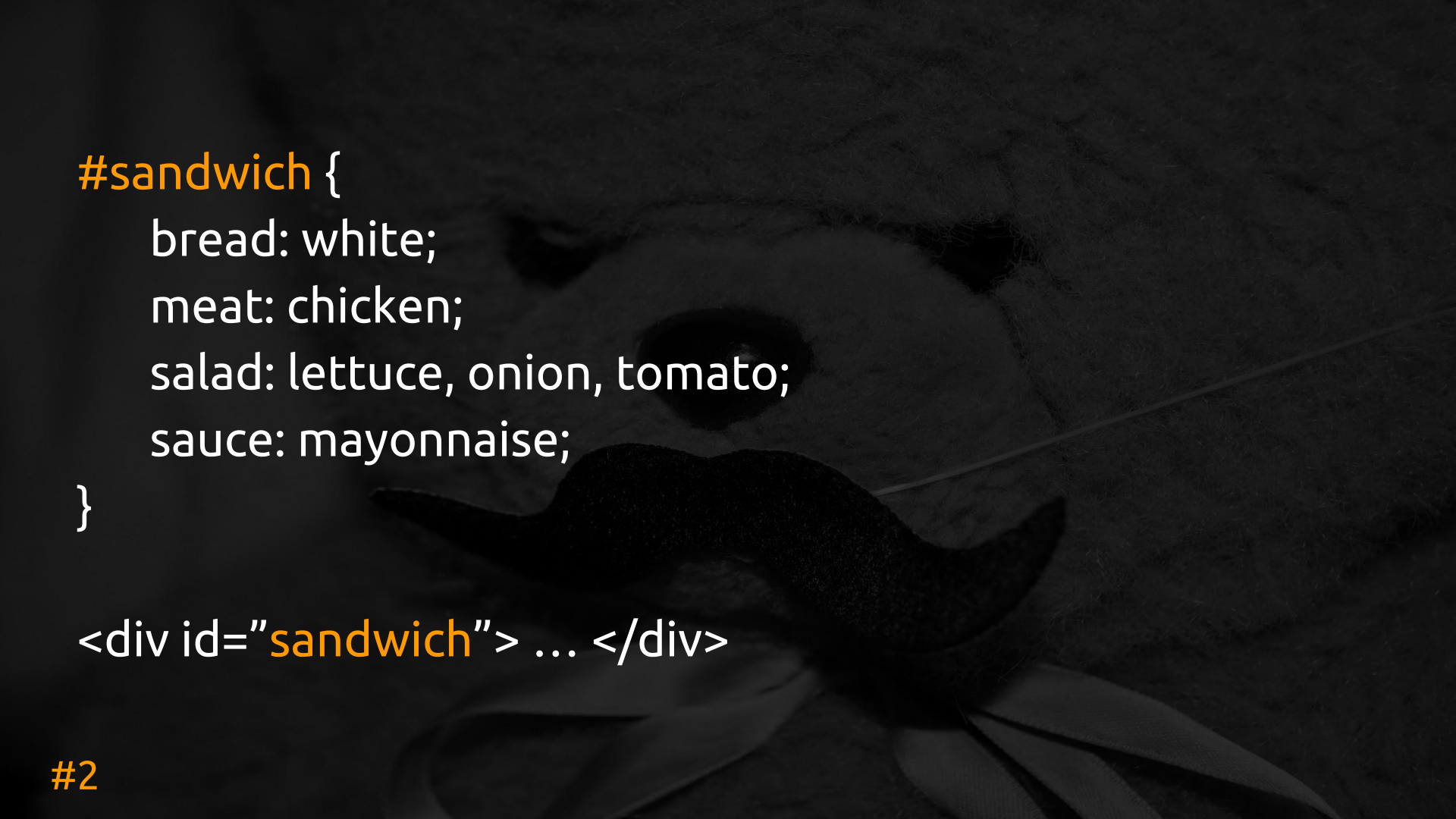
“[...] the single responsibility principle states that **every class should have responsibility over a single part of the functionality** provided by the software, and that responsibility should be **entirely encapsulated by the class.**”

[wikipedia.org/wiki/Single\\_responsibility\\_principle](https://wikipedia.org/wiki/Single_responsibility_principle)



Everything should do **one job**, one job only and  
should do that job very very well.

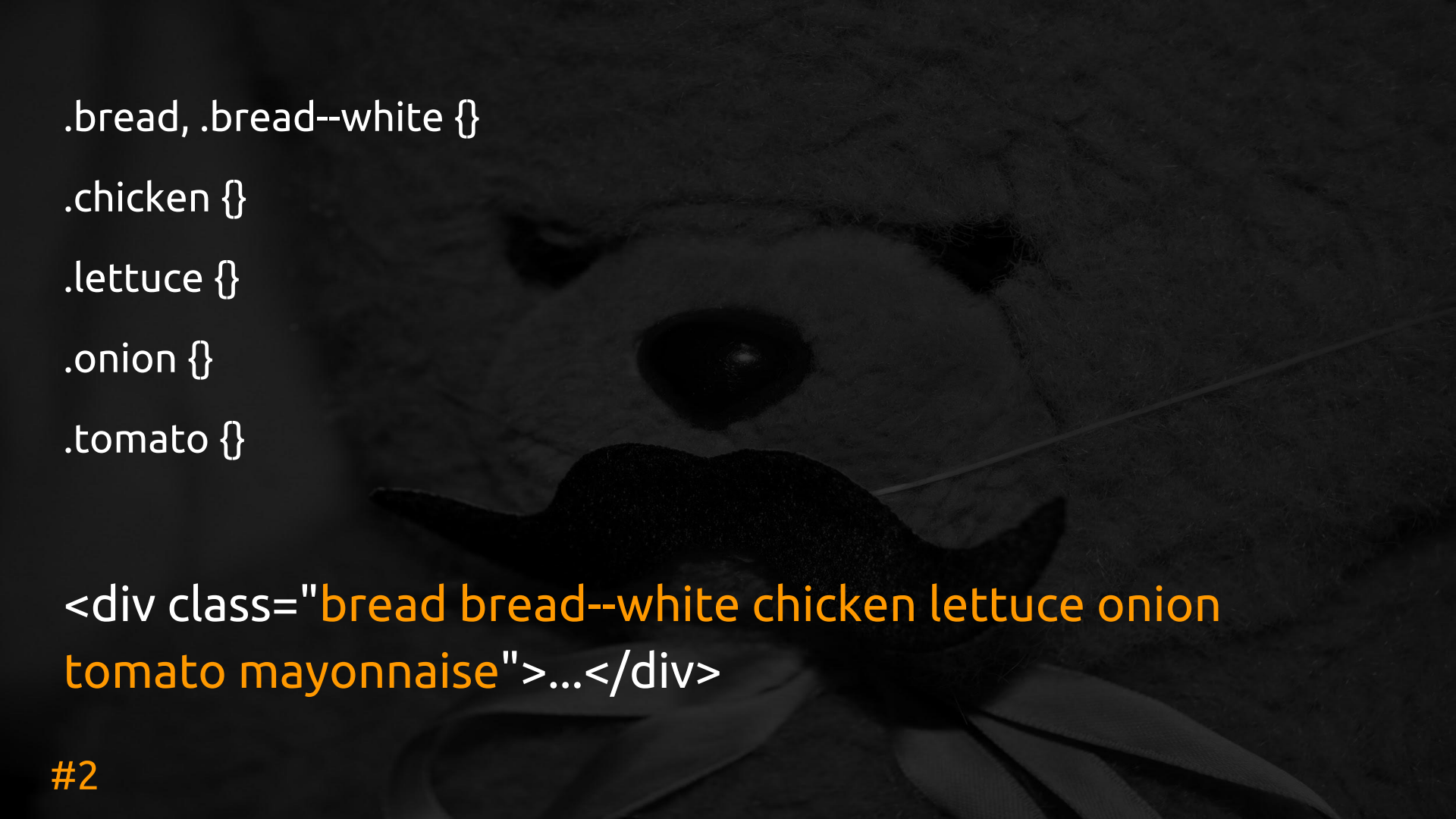




```
#sandwich {  
  bread: white;  
  meat: chicken;  
  salad: lettuce, onion, tomato;  
  sauce: mayonnaise;  
}
```

```
<div id="sandwich"> ... </div>
```





`.bread, .bread--white {}`

`.chicken {}`

`.lettuce {}`

`.onion {}`

`.tomato {}`

`<div class="bread bread--white chicken lettuce onion  
tomato mayonnaise">...</div>`

```
.btn-login {
```

```
  display: inline-block;
```

```
  padding: 2em;
```

```
  background-color: green;
```

```
  color: white;
```

```
}
```

Structural  
Responsability

Cosmetic  
Responsability

```
<button class="btn-login"> ... </button>
```

```
.btn {  
  display: inline-block;  
}
```

```
.btn--large {  
  padding: 2em;  
}
```

```
.btn--positive {  
  background-color: green;  
  color: white;  
}
```

```
<button class="btn btn--large  
btn-positive"> ... </button>
```



Provide developers with the ingredients.  
Let them make the meals.





## The separation of Concerns

“[...] It is, that one is willing to study in depth an aspect of one’s subject matter in isolation **for the sake of its own consistency** [...] But nothing is gained—on the contrary! —by **tackling these various aspects simultaneously**. It is what I sometimes have called ‘the separation of concerns’ [...] **it does not mean ignoring the other aspects, it is just doing justice** to the fact that from this aspect’s point of view, the other is irrelevant. It is being **one- and multiple-track minded simultaneously**.

[wikipedia.org/wiki/Separation\\_of\\_concerns](https://wikipedia.org/wiki/Separation_of_concerns)



Each thing responsible for itself and  
nothing more



Using HTML to provide cosmetics.

```
<font color="red">
```

```
<div style="color : red; ">...</div>
```

```
<nav role="navigation">
  <ul>
    <li> <a>...</a></li>
    <li><a>...</a> </li>
  </ul>
</nav>
```

```
[role="navigation"] { ... }
[role="navigation"] > ul { ... }
[role="navigation"] > ul > li { ... }
```



```
<nav class="site-nav js-site-nav" role="navigation">  
  <ul class="site-nav__list">  
    <li class="site-nav__item">  
      <a class="site-nav__link">...</a>  
    </li>  
    ....  
  </ul>  
</nav>
```

```
<nav class="site-nav js-site-nav" role="navigation">  
  <ul class="site-nav__list">  
    <li class="site-nav__item">  
      <a class="site-nav__link">...</a>  
    </li>  
    ....  
  </ul>  
</nav>
```

Semantic  
Concern

```
<nav class="site-nav js-site-nav" role="navigation">  
  <ul class="site-nav__list">  
    <li class="site-nav__item">  
      <a class="site-nav__link">...</a>  
    </li>  
    ....  
  </ul>  
</nav>
```

Accessibility  
Concern

```
<nav class="site-nav js-site-nav" role="navigation">  
  <ul class="site-nav__list">  
    <li class="site-nav__item">  
      <a class="site-nav__link">...</a>  
    </li>  
    ....  
  </ul>  
</nav>
```

Behaviors  
Concern



```
<nav class="site-nav js-site-nav" role="navigation">  
  <ul class="site-nav__list">  
    <li class="site-nav__item">  
      <a class="site-nav__link">...</a>  
    </li>
```

```
.site-nav { ...  
  .site-nav__list { ...  
    .site-nav__link { ... }  
  }  
}
```

Stylistic  
Concern





Only bind CSS onto CSS-based classes only.

Don't write DOM-like selectors.

Don't bind CSS onto data-\* attributes.

Don't bind JS onto CSS classes.

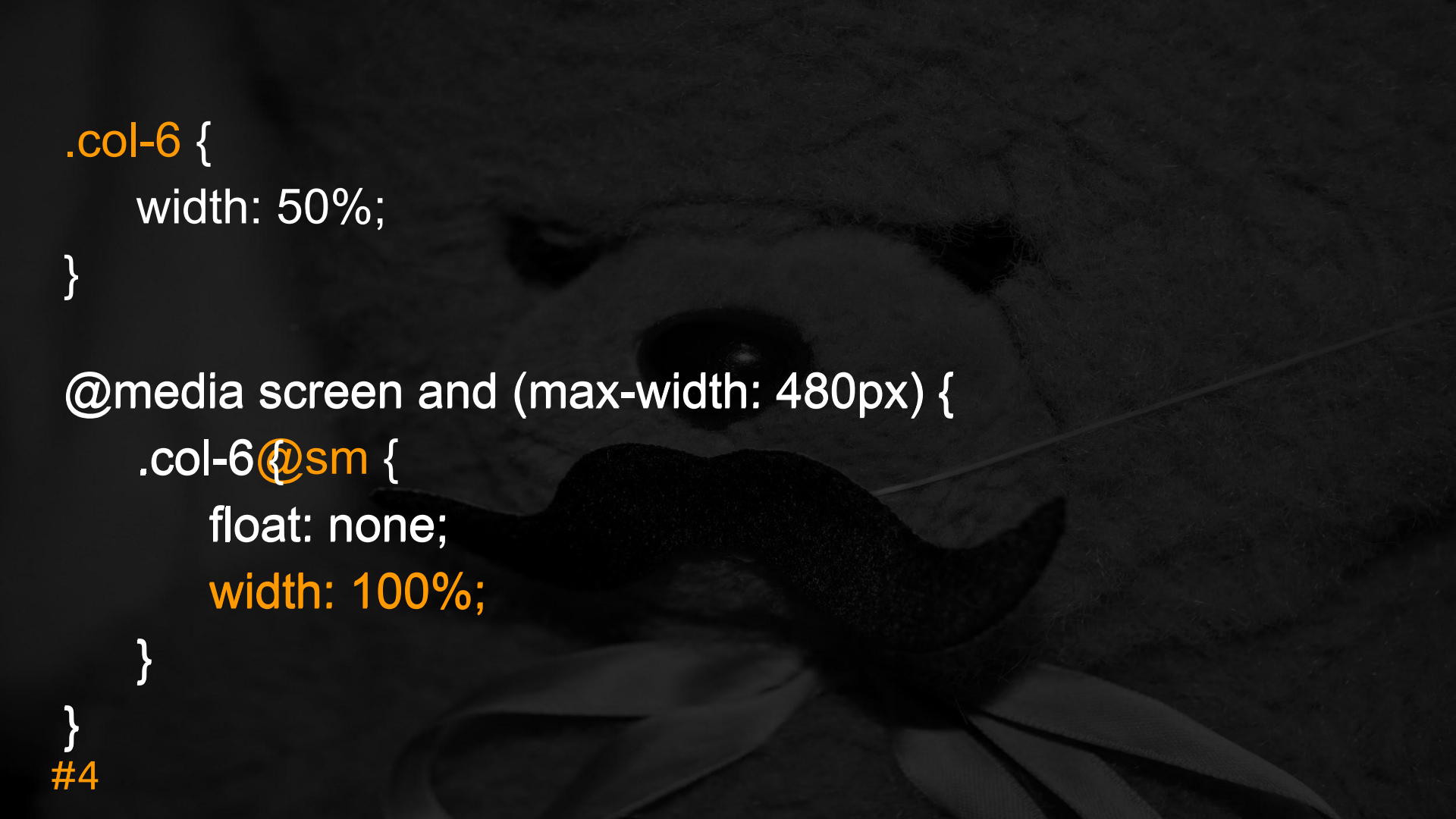


# 4 Immutability



“...an immutable object is an object whose state cannot be modified after it is created.

[wikipedia.org/wiki/Immutable\\_object](https://wikipedia.org/wiki/Immutable_object)



```
.col-6 {  
  width: 50%;  
}
```

```
@media screen and (max-width: 480px) {  
  .col-6@sm {  
    float: none;  
    width: 100%;  
  }  
}
```

#4



```
.btn {  
    font-size: 1em;  
}
```

```
.promo .btn {  
    font-size: 1.2em;  
}
```





```
.btn {  
    font-size: 1em;  
}
```

```
.btn--large {  
    font-size: 1.2em;  
}
```



Don't have several states of the same thing.

Use Modifiers or Responsive Suffixes



# The Open/Close Principle

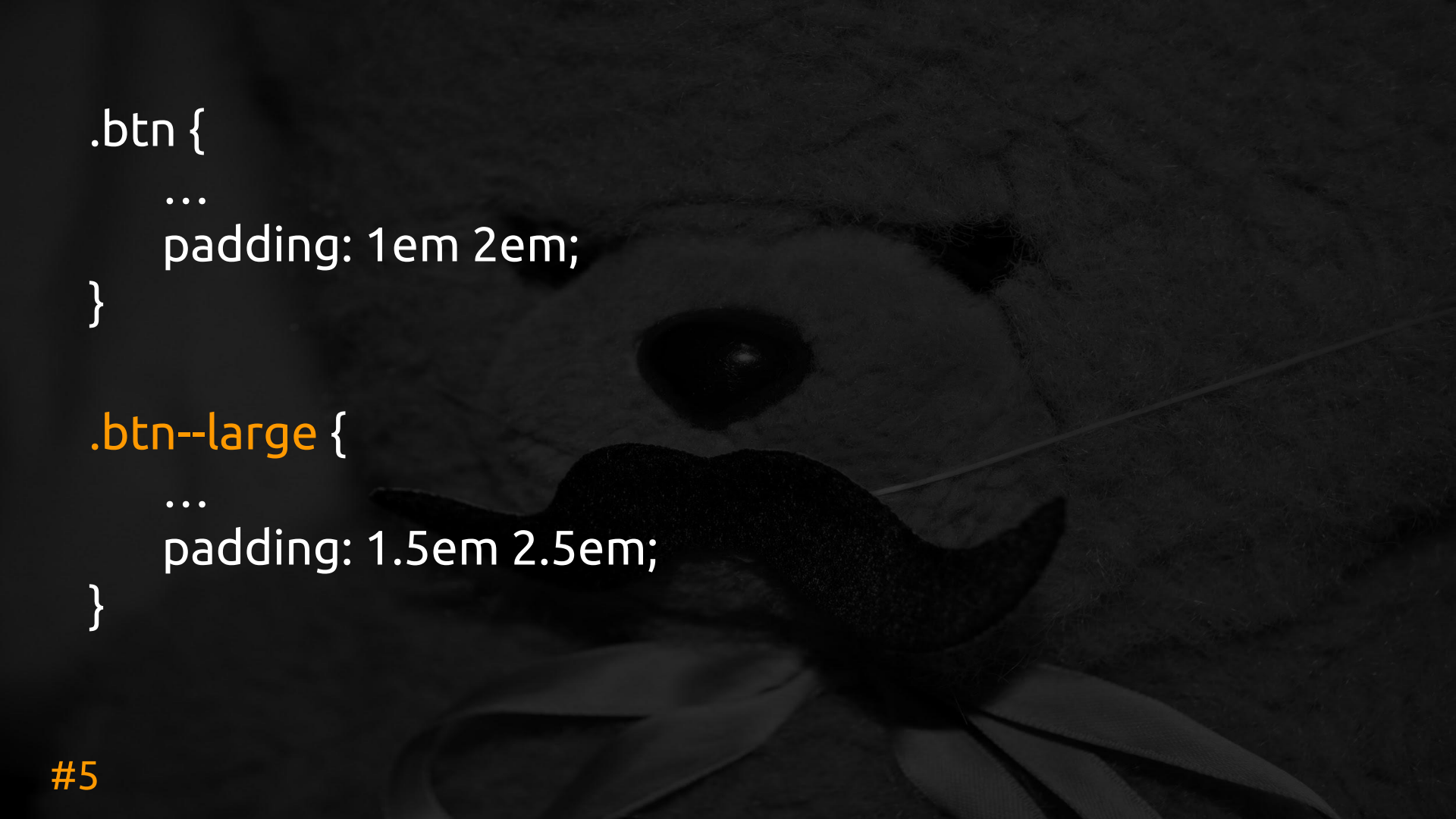


“Software entities (classes, modules, functions, etc.) should be **open for extension, but closed for modification.**”

[wikipedia.org/wiki/Open/closed\\_principle](https://wikipedia.org/wiki/Open/closed_principle)

“[...] once completed, the implementation of a class **could only be modified to correct errors; new or changed features would require that a different class be created.** That class could reuse coding from the original class through inheritance.



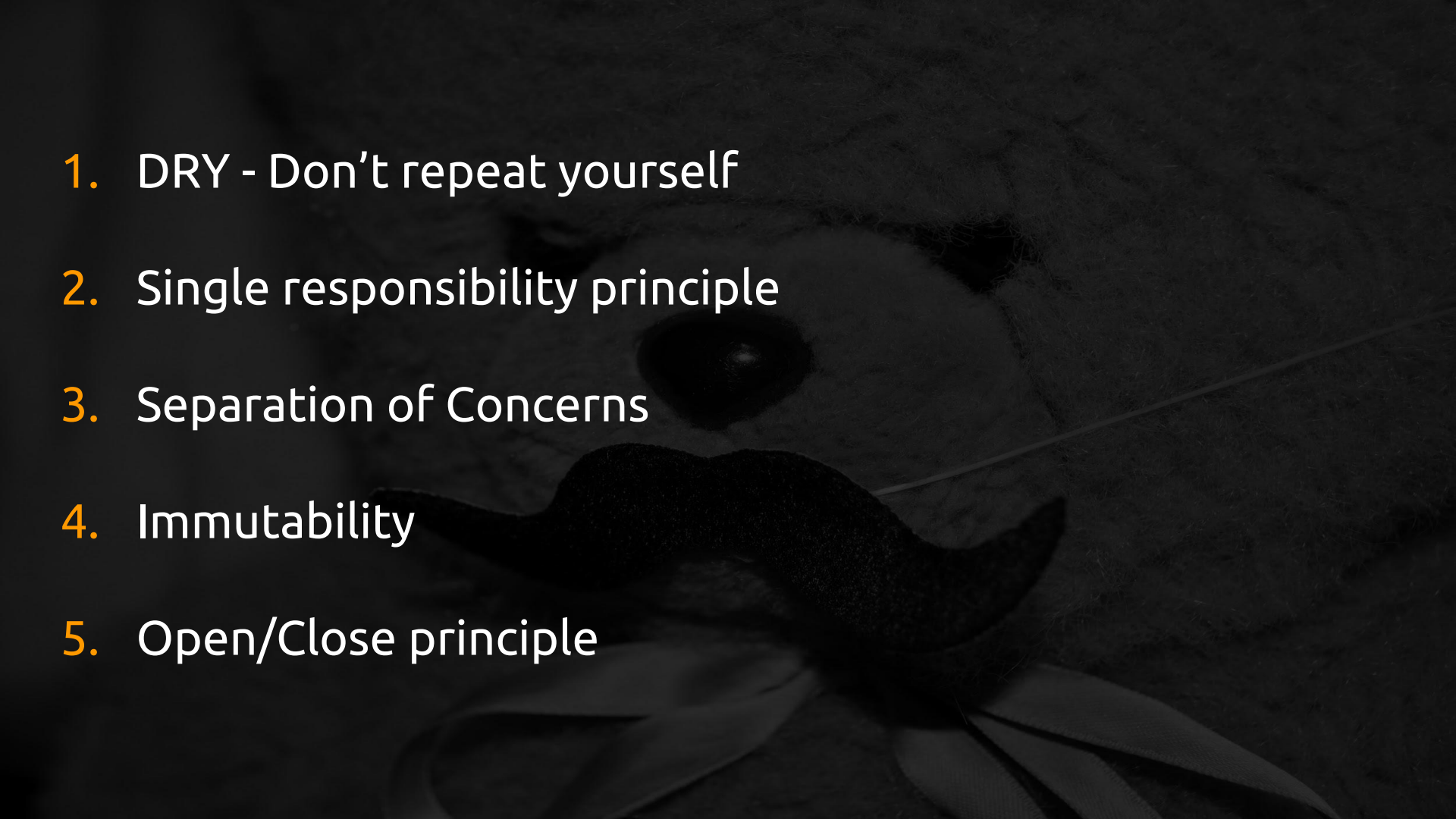


```
.btn {  
  ...  
  padding: 1em 2em;  
}  
  
.btn--large {  
  ...  
  padding: 1.5em 2.5em;  
}
```



Safe way to **make changes.**

Safe way of **working with legacy.**

- 
1. DRY - Don't repeat yourself
  2. Single responsibility principle
  3. Separation of Concerns
  4. Immutability
  5. Open/Close principle



# The Moustache Principle





“Just because you can, it doesn't mean that you should”

H. Roberts





# Thank you!!

@aldopizagalli

do-not-reply@frontedersTicino.com

Talk inspired by Henry Roberts

slides: <https://speakerdeck.com/csswizardry/css-for-software-engineers-for-css-developers>

video: <https://vimeo.com/140641366>