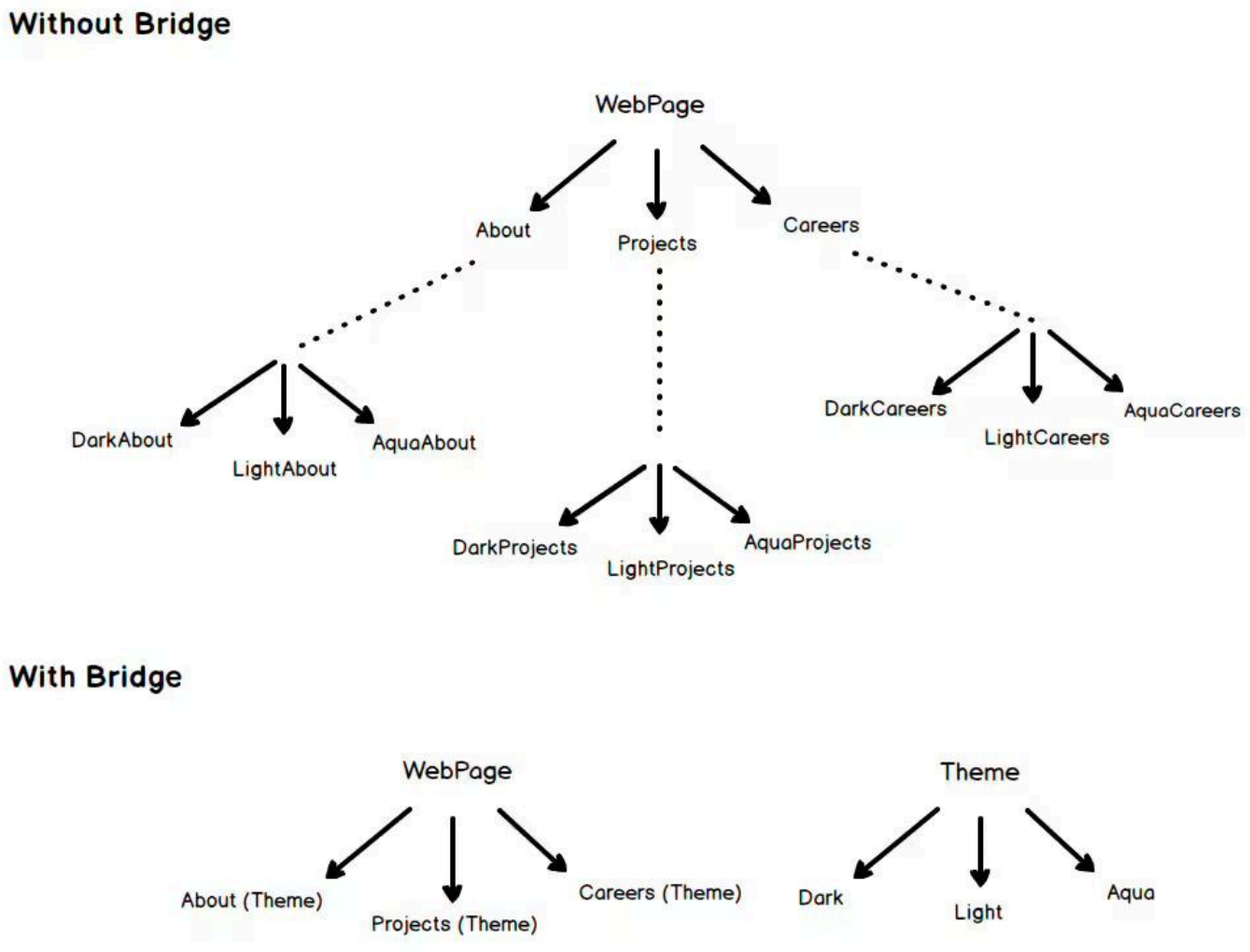
Мост — это структурный паттерн, который применяется для разделения абстракции и реализации таким образом, чтобы они могли меняться независимо друг от друга.

В паттерне **мост** происходит передача деталей реализации из одной иерархии в другой объект с отдельной иерархией.

Если говорить простыми словами, то при использовании шаблона **Мост** обеспечивается **предпочтение композиции над наследованием**.

Давайте представим, что у нас есть веб-сайт с разными страницами, а нам надо разрешить пользователям менять тему этих страниц. Что делать? Вряд ли будет разумно создавать множественные копии для каждой веб-страницы и для каждой темы. Вопрос можно решить благодаря шаблону Мост.

Схема:



Реализация:

```
interface WebPage
{
   public function __construct(Theme $theme);
   public function getContent();
}

class About implements WebPage
{
   protected $theme;
   public function __construct(Theme $theme)
   {
       $this->theme = $theme;
   }
   public function getContent(): string
   {
       return "About page in " . $this->theme->getColor();
   }
}

class Careers implements WebPage
{
   protected $theme;
   public function __construct(Theme $theme)
   {
       $this->theme = $theme;
   }
   public function getContent(): string
   {
       return "Careers page in " . $this->theme->getColor();
   }
}

interface Theme
{
   public function getColor();
}

class DarkTheme implements Theme
{
   public function getColor(): string
   {
       return 'Dark Black';
   }
}
```

```php
class LightTheme implements Theme
{
    public function getColor(): string
    {
        return 'Off white';
    }
}

class AquaTheme implements Theme
{
    public function getColor()
    {
        return 'Light blue';
    }
}

$darkTheme = new DarkTheme();

$about = new About($darkTheme);
$careers = new Careers($darkTheme);

echo $about->getContent();
echo $careers->getContent();
```