

Yönetilebilir Arayüz Mimarisi

Web, Mobil ve Karmaşık Sistemler için HTML/CSS

Bilal Çınarlı

Front-end Architect

Senior UX Developer@Turkcell

@bcinarli

github.com/bcinarli

Siz?

Arayüz Atölyesi

Neler Yapacağız?

- Tasarımı Nasıl Ele Almayız?
- Tekrar Kullanılabilirlik
- Seçiciler
- İsimlendirme / Gruplama
- Kod tekrarlarını azaltma

Başlamadan

Chat Odası

<http://tlk.io/arayuz2015>

Çalışmalar İçin

<https://jsfiddle.net>

<http://codepen.io/>

Gün Boyu İhtiyacınız Olabilecekler

- Text editor
- Sass
- Photoshop (Size bağlı)
- Github

Arayüz Mimarisi

Neden?

- Sistemler ve takımlar büydü
- Arayüz geliştirme karmaşıklığı
- Sayfalardan uygulamalara geçildi

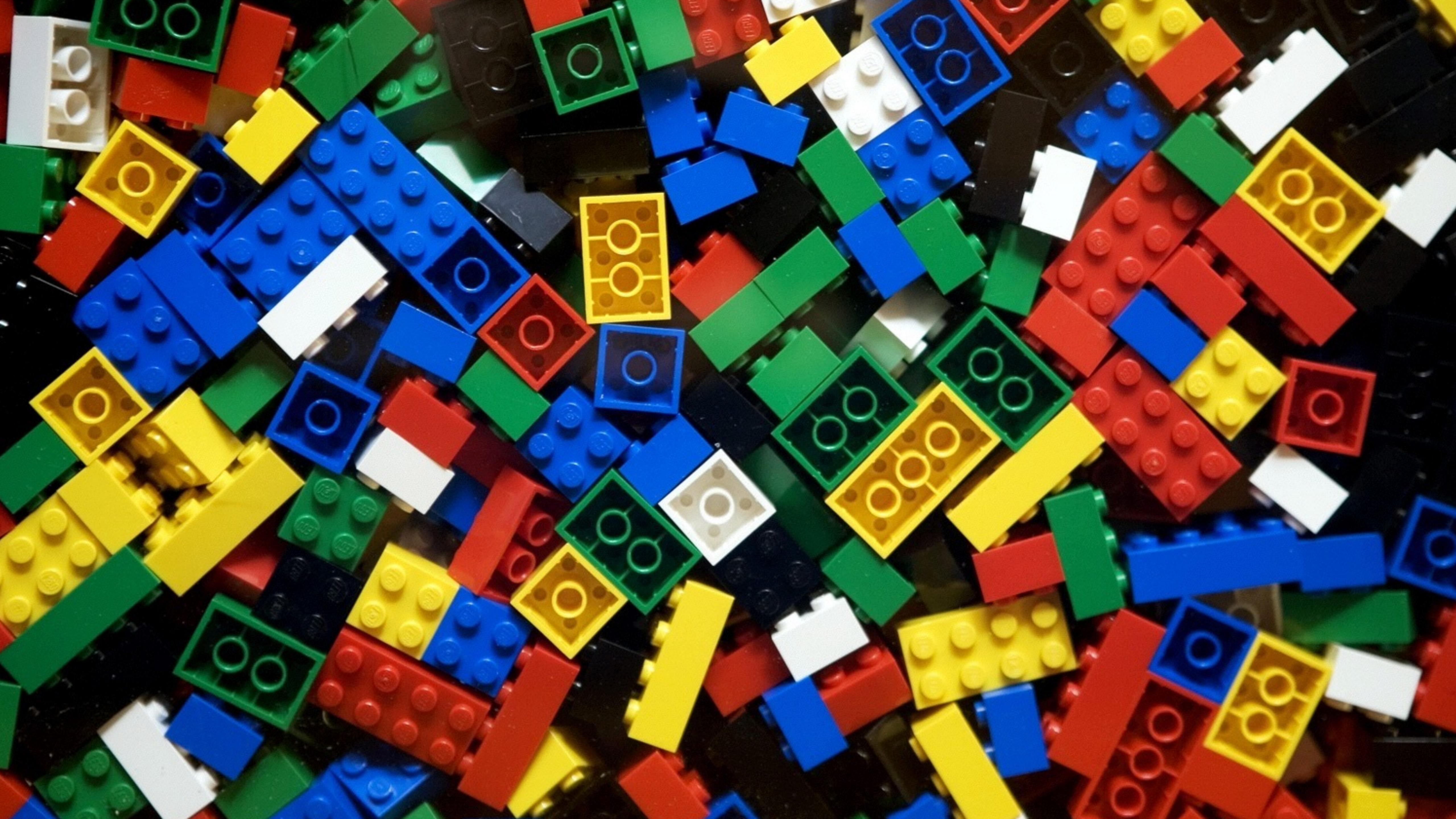
I've seen the
FUTURE
It's in my
BROWSER





Hedef

- Daha kolay ve hızlı çalışma
- Daha akıllıca çalışma
- Daha kısa öğrenme/adapte olma süresi
- Daha kolay yönetim ve geliştirme



SASS

- Fonksiyonlar
- Değişken
- Modüler Çalışabilme

Biraz Elimizi Kirletelim

GitHub

```
$ git clone https://github.com/bcinarli/scalable-frontend-architecture.git arayuz2015
```

docs/designs	New: Sample Design	7 hours ago
styles-sass	Update: Sample Style organization	22 minutes ago
styles	Update: Sample Style organization	22 minutes ago
.gitignore	Update: Sample Style organization	30 minutes ago
README.md	Update README.md	2 days ago
index.html	Update: Sample Style organization	30 minutes ago

README.md

Yönetilebilir Arayüz Mimarisi

Günlük hayatın bir parçası olan karmaşık sistemleri, web ve mobil arayüzleri daha kolay yönetmek için planlı bir arayüz kurgusu önemli bir yere sahiptir. Bu atölye çalışmasında, bu kurguyu nasıl planlamak gereklidir, geliştirirken nelere dikkat edilmeli konularının üzerinde duracağınız.

Pulse

Graphs

Settings

HTTPS clone URL

<https://github.com/I>



You can clone with [HTTPS](#), [SSH](#), or [Subversion](#).

Clone in Desktop

Download ZIP

Prepros/Codekit

- Klasörünüze Prepros ya da Codekit içine ekleyin
- “styles-sass/styles.scss” dosyasını “styles/styles.css” şeklinde çıktı vermeye ayarlayın

- ▶ docs
- ▼ styles
 - ◊ styles.css
- ▼ styles-sass
 - ▶ auxiliary
 - ▶ components
 - ▶ elements
 - ▶ global
 - ▶ layout
 - ▶ settings
 - ▶ tools
 - styles.scss
 - index.html
 - README.md

SASS

Open

Output style:

Expanded ▼

Debug info:

None ▼

Other options:

Create a source map

Use the libsass compiler

After compiling:

Run Bless on the CSS file

Run Autoprefixer on the CSS file

OUTPUT FILE ✓

/styles/styles.css ✖

Çalışma (5 dakika)

Örnek repoyu inceleyelim, istediğiniz kod örneğini ve denemeyi yapın

Markup

- Sayfa mimarisinin temeli
- Bütün sunum ve etkileşim katmanını etkiler
- Karmaşık kurguların kolay uygulanmasını sağlar ya da kabus haline gelmesine neden olur!

Tasarımı İnceleyelim



Web ve Mobil Arayüz Mimarisi

⌚ 28 Mart 2015 ⚽ İstanbul

Biletler tükendi

Siteler büyündükçe, ihtiyaçlar artıyor. Bu ihtiyaçları, planlı bir arayüz mimarisi ile yönetebilirsiniz!

Arayüz mimarisi atölyesinde, basit yaklaşımlar ile tekrar kullanılabilir, geliştirilebilir ve yönetilebilir bir altyapı oluşturmanın temelleri işlenmekte. Temel HTML ve Sass desteği ile örnek kurgular üzerinden modern yaklaşımlar incelenmeye.

[Etkinlik detayları »](#)

Geçmiş Etkinlikler

[CSS3 Animation ve Transition »](#)

CSS3 Animasyonlarına giriş ve temel animasyon özelliklerinin kullanımı.

Nesrin Kalender | @wtmistanbul

[Usable Front-end »](#)

Arayüz tasarımından entegrasyon aşamasına front-end development süreci.

Bilal Çınarlı | @uxkafe

[Mobil İçin Arayüz Geliştirme »](#)

Yaklaşım tercihi, mobil site mi, responsive tasarım mı, adaptive mı?

Bilal Çınarlı | @Mobilistanbul

HTML Magazin Bülten

Güncel haberler, etkinlik ve buluşma duyurularını yaptığız, her ayın 3. Salı günü gönderilen bültenimize hemen abone olun!

eposta adresi

Abone Ol

Kaynaklar

CSS Architecture

[Scalable and Modular Architecture for CSS](#)

Kitap | Websitesi

[Melange: CSS Framework](#)

Framework | Github



“ Ülkemizde bu tarz etkinlıklere olan yoğun ilgi gerçekten sevindirici ve sektör için umit verici bir durum. Emeği geçen herkese şimdiden teşekkürler”

John Doe | Front-end Developer

[Front-end Masters](#)

Komponentler

- Sayfaları unutun
- Sayfalar içindeki komponentleri düşünün
- Komponentleri birleştirerek sayfaları oluşturuyoruz

Komponentleri Planlama

- Bir tane komponenti ele alın
- Kullanım yapısını düşünün
- Aynı yapıyı başka elemanlarda arayın
- Farklı noktaları ayırin
- Benzer noktaları ortak tanımlayın

Çalışma (10 dakika)

Tasarımdaki ana menümüzü oluşturalım

[ANASAYFA](#) [ETKİNLİKLER HAKKINDA](#) [TAKVİM](#) [İLETİŞİM](#)

Çalışma (Beraber)

Menü için hazırladığımız yapıyı başka nerde kullanabiliriz?

[ANASAYFA](#) [ETKİNLİKLER HAKKINDA](#) [TAKVİM](#) [İLETİŞİM](#)

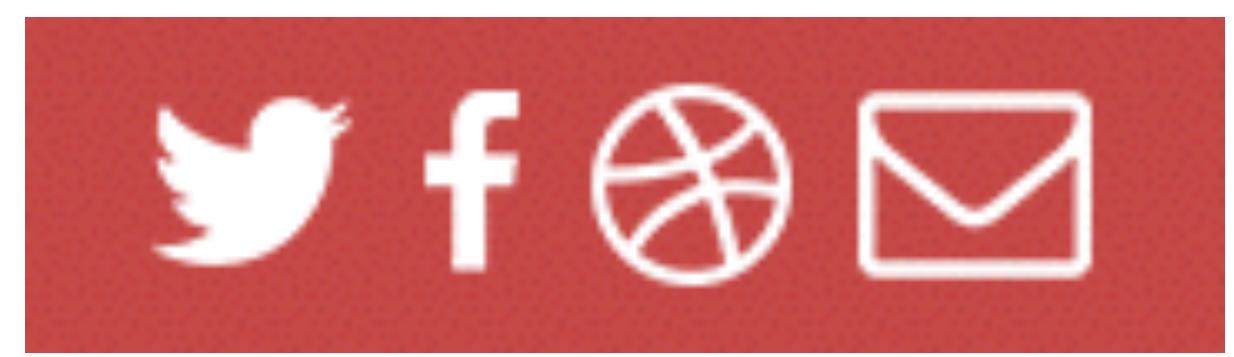
[Son Yazılar](#) [Blog Arşiv](#) [Kategoriler](#)

[HTML Mag »](#)

[CSS-Tricks »](#)

[Smashing Magazine »](#)

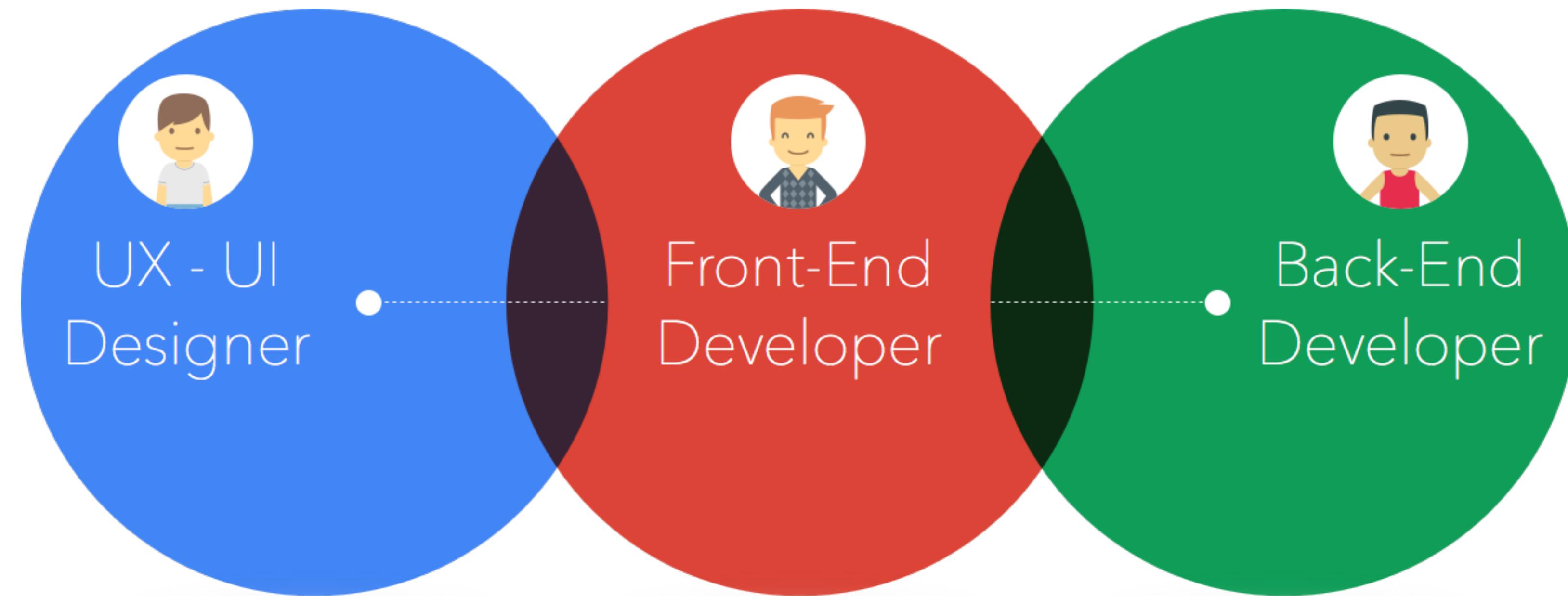
[CSS Zen Garden »](#)



Mimari'nin Temelleri

Front-end Kimdir?

- OOCSS - Arayüz geliştirmeye mühendis bakış açısı
- Kurgu - Sadece renkleri ekleyen değil, akışı planlayan
- Adaptive - Sürekli değişen/güncellene teknoloji ve sistemlere hızlı çözüm üretebilen



Tasarımı Yönetmek

Öğeleri Nasıl Sadeleştirebiliriz?

Benzer görüntüye sahip elemanları en sade şekilde tanımlama

Biletler tükendi

Hemen Başla

Abone Ol

Çalışma (Beraber)

Ögeler için genel grupları en sade haliyle planlayalım

Reusable CSS

Single Responsibility Principle

Object Oriented Programlamada, bütün ortamlar (class, function, variable vs), sadece bir işlemden sorumlu olmalı ve bu işlem o ortamın içinde başlayıp sonlanmalı.

CSS'de Karşılığı

Tanımlanan her stil, temel anlamda her zaman bir iş için tanımlanmalı ve her zaman o iş için uygun kullanılmalı



Bunu istemiyoruz!

HTML Magazin Bülten

Güncel haberler, etkinlik ve buluşma duyurularını
yaptığımız, her ayın 3. Salı günü gönderilen
bültenimize hemen abone olun!

eposta adresi

Abone Ol

Kaynaklar

CSS Architecture

Scalable and Modular Architecture
for CSS

Kitap | Websitesi

Melange: CSS Framework

Framework | Github

Front-end Masters

Video | Websitesi

KENDİNİZ DENEYEREK ÖĞRENİN

Videoları izleyerek, örnek projeler ile
kendinizi geliştirin.

Hemen Başla

Widget - Banner

Başlıksız, kırmızı arkaplan, beyaz metin

KENDİNİZ DENEYEREK ÖĞRENİN

Videoları izleyerek, örnek projeler ile kendinizi geliştirin.

Hemen Başla

```
<div class="sidebar-banner">
```

...

```
</div>
```

Widget - Banner

- Widget
- Beyaz Metin
- Kırmızı Arkaplan

KENDİNİZ DENEYEREK ÖĞRENİN

Videoları izleyerek, örnek projeler ile kendinizi geliştirin.

Hemen Başla

```
<div class="widget  
    widget-banner">
```

...

```
</div>
```

Single Responsibility Principle

- Daha sade kodlama
- Tekrar kullanılabilen kod bloklarına sahip olma
- Farklı tanımları bir arada kullanabilme avantajı

Çalışma (Beraber)

Tasarım'dan SRP kullanımlarını inceleyelim

Biletler tükendi

Hemen Başla

Abone Ol

Usable Front-end »

Arayüz tasarımından entegrasyon aşamasına
front-end development süreci.

Bilal Çınarlı | @uxkafe

Seçim Sorunu

- İyi kurgulanmamış tanımlarda sayfaya göre küçük değişiklikleri olan komponentleri düzenleme zorlaşmaktadır
- İstenilen görünüm vermek için çok spesifik seçici tanımı gerekmektedir
- Kod tekrarı artar, dolayısıyla kontrol zorlaşır

CSS Seçiciler - Atomik Yaklaşım

Uyulması Gerekenler

- Tahmin edilebilir - isimleri, tanımları ile yaptıkları uyumlu olmalı
- Tekrar kullanılabilir - farklı öğeler için de kullanılabilirmeli
- Stabil olmalı - eklendiği ögeyi etkilerken başka bir ögeyi bozmamalı
- Düşük Spesifiklikte olmalı - bir ögeye “nokta atışı” yapmamalı

- * (global selector)
- a, div (tag selector)
- [class*="like"], [class^="like"], [class\$="like"], [class~="like"]
- .class, .media
- #unique

Spesificity

Belki de bütün kötü kodun temeli!

```
header .menu ul { }
```

.main-navigation { }

`li:first-child h2 .title`

0

Inline styles

0

IDs

2

Classes, attributes
and pseudo-classes

2

Elements and
pseudo-elements

+ Duplicate

`#nav .selected > a:hover`

0

Inline styles

1

IDs

2

Classes, attributes
and pseudo-classes

1

Elements and
pseudo-elements

+ Duplicate

Anlamı/Aynışılık Tanımları

- HTML - tanımları itibariyle anlamlı ve makine/tarayıcı için bir ifadeye sahiptir
- Class - cihaz için anlamlı değildir, insanın okuyup, anlamlandırması için tanımlanır, subjektiftir

<div class="headline"></div>

<div class="baslik"></div>

<div class="publizierte"></div>

```
<article class="widget widget-secondary resources">
  <h1 class="resources-title">Kaynaklar</h1>
  <p class="resources-text">Lorem ipsum dolor...</p>
</article>
```

```
<article class="widget widget-secondary resources">
  <h1 class="resources-title">Kaynaklar</h1>
  <p class="resources-text">Lorem ipsum dolor...</p>
</article>
```

```
<article class="widget widget-secondary resources">
  <h1 class="resources-title">Kaynaklar</h1>
  <p class="resources-text">Lorem ipsum dolor...</p>
</article>
```

Anlamlı/Anlaşılır Tanımlar

- Class ismi verirken görünümden bahsetmek gereksizdir
- Tanımın yaptığı iş/eyleme göre isimlendirmek her zaman daha doğrudur

```
<div class="border mt10 blue pill">
```

```
...
```

```
</div>
```

```
<div class="widget widget-round">
```

```
...
```

```
</div>
```

```
<button class="blue-button">...</button>
```

```
<button class="primary-action">...</button>
```

İsimlendirme

- Her zaman kafa patlatıcı bir iş olmuştur
- Programlamadaki en zor işlerden biri olarak düşünülür
- Düzgün yapıldığında her zaman değerli olur

İsimlendirme Nasıl Yapılmalı

- Görünümden çok işlevselliğe odaklanmalı
- İş güdüşel olmalı
- Farklı ögelere uygulanabilir olmalı

```
<nav class="navigation-list">  
  <a href="#" class="navigation-list-item">...</a>  
  <a href="#" class="navigation-list-item">...</a>  
  <a href="#" class="navigation-list-item">...</a>  
  <a href="#" class="navigation-list-item">...</a>  
</nav>
```

Çalışma (Beraber)

Tasarımızdaki anlamlı öğeleri bulalım

Componentler

- Tasarımdaki öğelerin özelleşmiş durumlarıdır
- Objelere göre daha tanımlayıcı isimleri olabilir

```
<nav class="navigation-list main-navigation">  
  <a href="#" class="navigation-list-item main-navigation-item">...</a>  
  <a href="#" class="navigation-list-item main-navigation-item">...</a>  
  <a href="#" class="navigation-list-item main-navigation-item">...</a>  
  <a href="#" class="navigation-list-item main-navigation-item">...</a>  
</nav>
```

[ANASAYFA](#) [ETKİNLİKLER HAKKINDA](#) [TAKVİM](#) [İLETİŞİM](#)

.main-navigation { }

Biletler tükendi

.button-primary { }

İsimlendirme Yöntemleri

- Elemanın ne işe yaradığını anlatmalı
- Elemanın ek özelliğe sahip olup olmadığını anlatmalı
- Elemanın hangi durumda olduğunu anlatmalı

BEM Metodolojisi

- Yandex!
- Block / Element / Modifiers

Block

```
<div class="block block- -modifier">  
  <p class="block__element">...</p>  
</div>
```

Element

```
<div class="block block- -modifier">  
  <p class="block__element">...</p>  
</div>
```

Modifier

```
<div class="block block- -modifier">  
  <p class="block__element">...</p>  
</div>
```

BEM Metodolojisi

- Elemanın ne olduğunu anlatıyor
- Markup ve CSS içinde elemanların ilişkisini gösteriyor
- Tanımları bir namespace içine ekliyor

BEM Metodolojisi - Alternatif

- Tanıtım ayraçlarından hoşlanmıyorsanız
- modifier için kullanılan “- -” tanımı
- element için kullanılan “__” tanımı

Block

```
<div class="block mofied-block">  
  <p class="block-element">...</p>  
</div>
```

Element

```
<div class="block modified-block">  
  <p class="block-element">...</p>  
</div>
```

Modifier

```
<div class="block modified-block">  
  <p class="block-element">...</p>  
</div>
```

Örnek

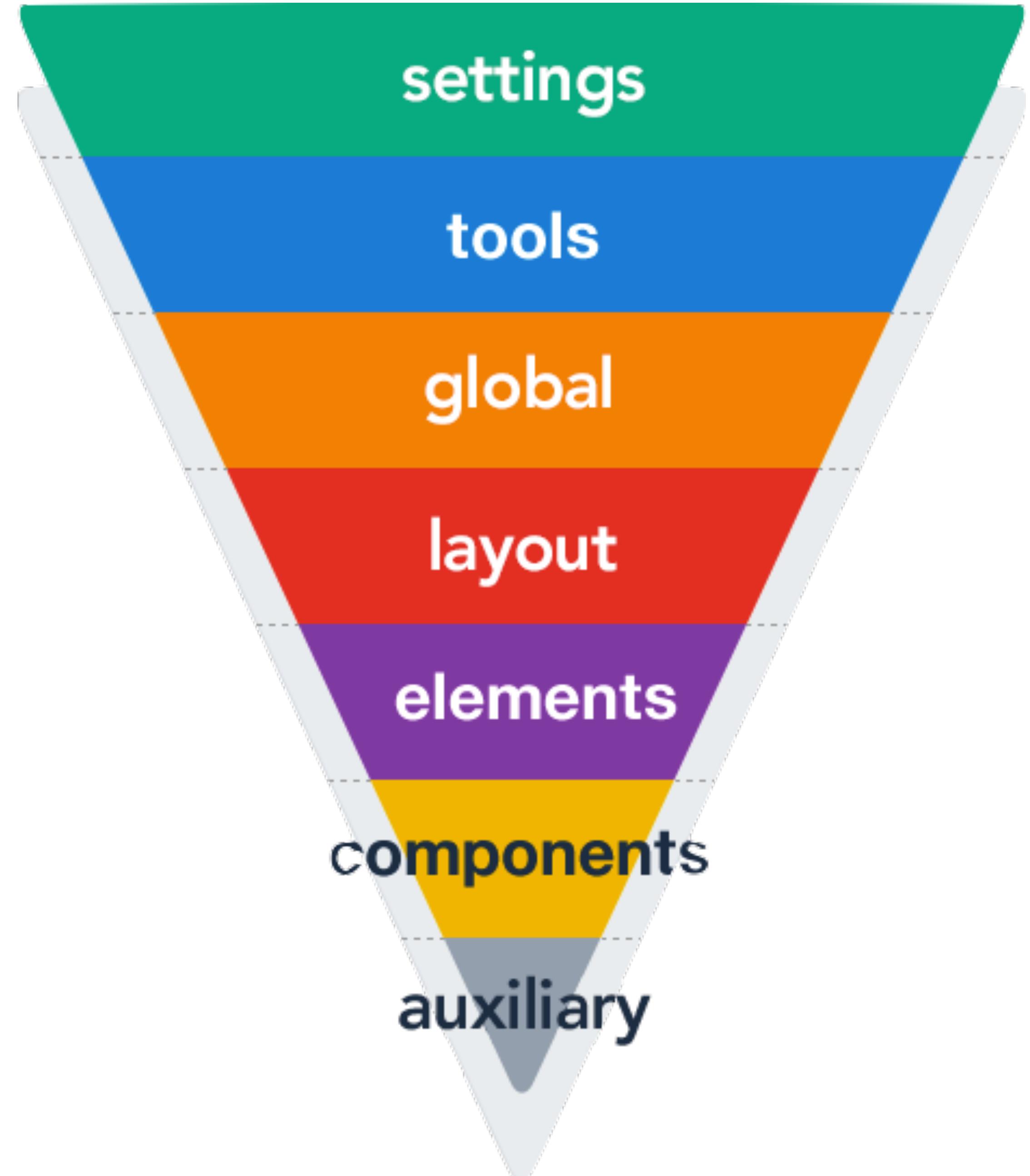
```
<nav class="navigation main-navigation">  
  <a href="#" class="navigation-item">...</a>  
  <a href="#" class="navigation-item">...</a>  
  <a href="#" class="navigation-item">...</a>  
</nav>
```

Örnek

```
<div class="widget promo-widget">  
  <h1 class="widget-title">...</h1>  
  <div class="widget-content">...</div>  
</div>
```

Tanım Sıralaması

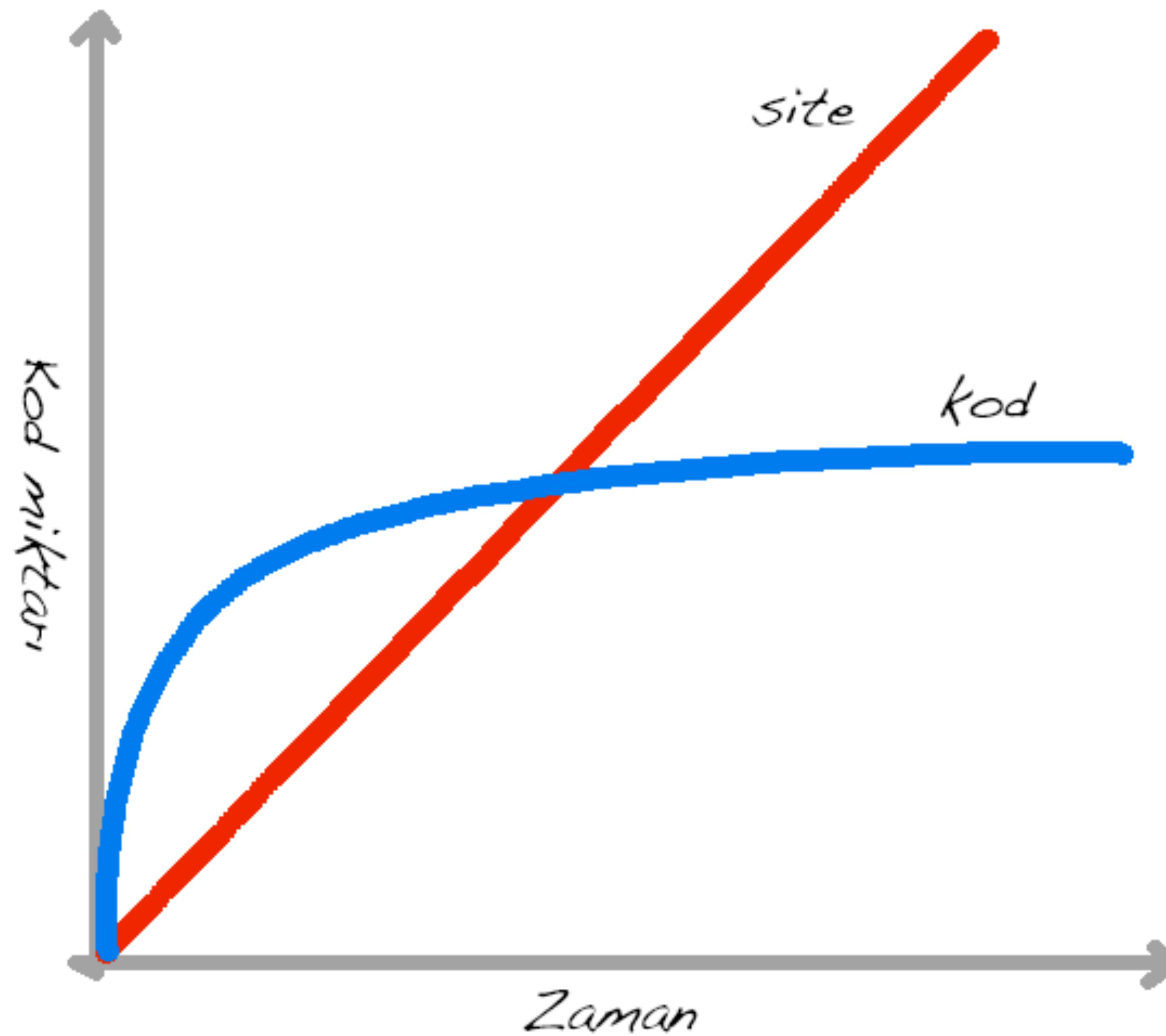
- Spesifikliğin yanında, tanım kurgusu da overwriteleri azaltır.
- Piramidin durumuna göre yazılan kodlar, SRP kullanımını arttırmır.
- Tanım kurgusu içinde ilk en üstteki kodlar genel kullanım içindir
- En altta kalan tanımlar ise, atomik yapıda, ögeye ya da duruma özel kodlardır



Kod/Yeni Geliştirme Süreci

Kod Kurgusu

- Bir sürü dosya
- Farklı klasörler



Layout

- Bir bütün olarak düşünülmeli
- Elemanların kendi genişlikleri yerine bulundukları yerler planlanmalı
- Grid sistem ile karıştırmamak önemli

Layout Kurgusu Yoksa



Layout Kurgusu Olduğunda



Toparlarsak



- Her zaman html ögesini olabilecek en sade yapı ile kurgulayın
- CSS seçicileri kısa, düşük spesifiklikte ve tekrar kullanılabilir olacak şekilde tanımlayın
- Küçük parçaları yönetmek daha kolaydır, kodunuzu küçük bloklara bölgerek yönetin
- Uygulamayı bir bütün olarak düşünüp, her zaman tekrar kullanım ve extend etmeyi planlayarak çalışın

Teşekkürler

@bcinarli