imarketinx.de

Webdesign // individuell • flexibel • robust

Rüdiger Alte ● Trögelsbyer Weg 38E ● 24943 Flensburg ● Tel.: 0461 / 67 999 667 ● Mobil: 0160 / 43 82 369 ● Mail: office@imarketinx.de ● Web: imarketinx.de

Modulare Website-Entwicklung

Flexible Frontend-Programmierung

Modulare Website-Entwicklung funktioniert ähnlich wie der Bau eines Fertighauses. Haben Sie schon einmal beobachtet, wie schnell und unkompliziert der Aufbau eines solchen Hauses abläuft?
Das geht auch im Kleinformat mit Lego.
Das Prinzip ist immer dasselbe: Vorgefertigte Module werden so kombiniert und zusammengesetzt, dass ein vorher festgelegtes Ganzes entsteht.

Wichtig ist dabei, dass die richtigen Module an die passenden Stellen kommen. Eine Flurtreppe, die an der Außenwand angebracht wird, kann zwar schön aussehen, verfehlt aber ihren eigentlichen Zweck.

Was sind Website-Module?

Website-Module sind bewährte Komponenten der Website-Gestaltung, die sich in (fast) jede Website problemlos integrieren lassen. Sie sind flexibel, leicht anzupassen und erleichtern die Pflege einer Website.

Modulare Website-Entwicklung ist ein Ansatz im Frontend Development oder UX Design. Dabei werden wiederverwendbare und von anderen Komponenten unabhängige Module mit HTML und CSS entwickelt.

Der HTML-Code eines Moduls bleibt unverändert – bis auf die Ergänzung um CSS-Klassen. Mittels dieser CSS-Klassen werden Varianten des Grund-Moduls erstellt.

Eine Webseite wird also aus Modulen – wie ein Fertighaus oder wie ein Haus aus Legosteinen – zusammengebaut. Dabei sind die Module unabhängig voneinander verwendbar und frei kombinierbar.

Wie funktioniert modulare Website-Entwicklung?

Modulare Website-Entwicklung zielt auf ein bestimmtes Feature, z. B. eine Fotogalerie, einen Absende-Button oder ein Menü. Dabei wird Komplexität in einfache und sinnvolle Einheiten aufgelöst.

Hier ein allgemeines Beispiel, wie eine Startseite oder Homepage modular zusammengesetzt werden kann:

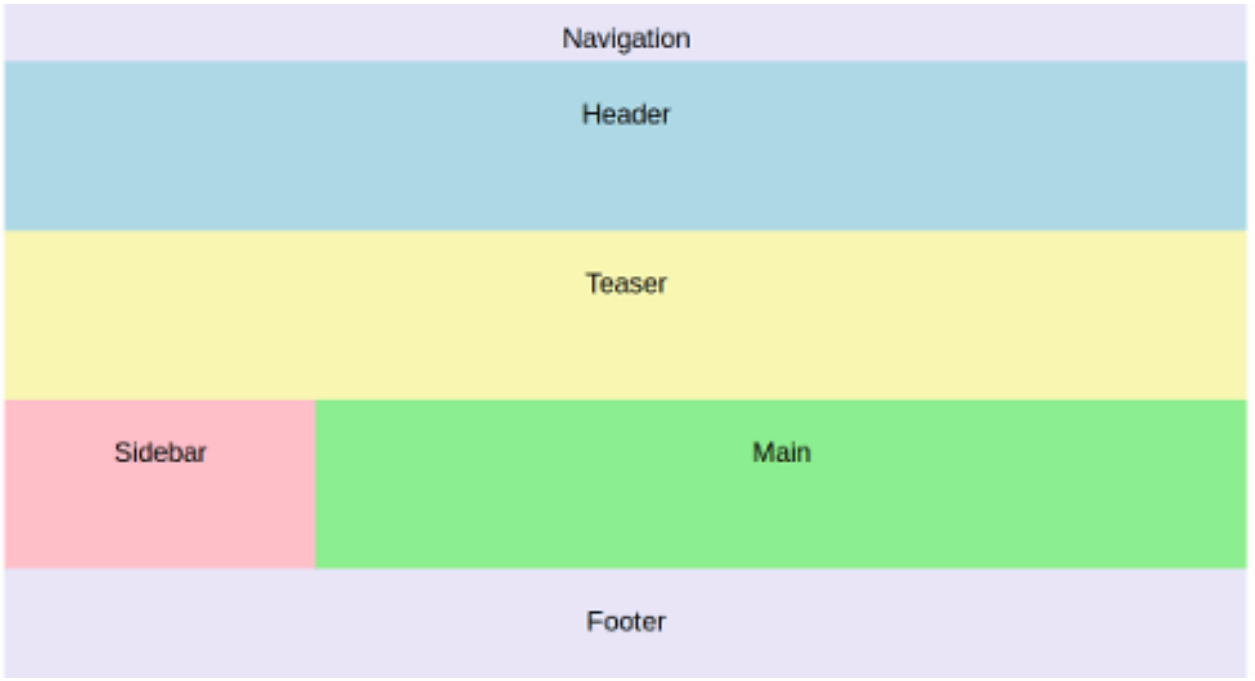
- Modul „Menu“ in der Navigation
- Modul „Banner“ (mit den Modulen „Logo“ und „Title“) im Header
- Modul „Cards“ im Teaser
- Modul „Box“ im Main Content

- Modul „Video“ in der „Sidebar“
- Modul „List“ im Footer

Module sind keine Templates und Templates sind keine Inhaltselemente

Templates sind Vorlagen für Websites, die die Struktur für das Layout einer Seite vorgeben. Im folgenden Beispiel sind das Navigation, Header, Teaser, Main Content, Sidebar und Footer.

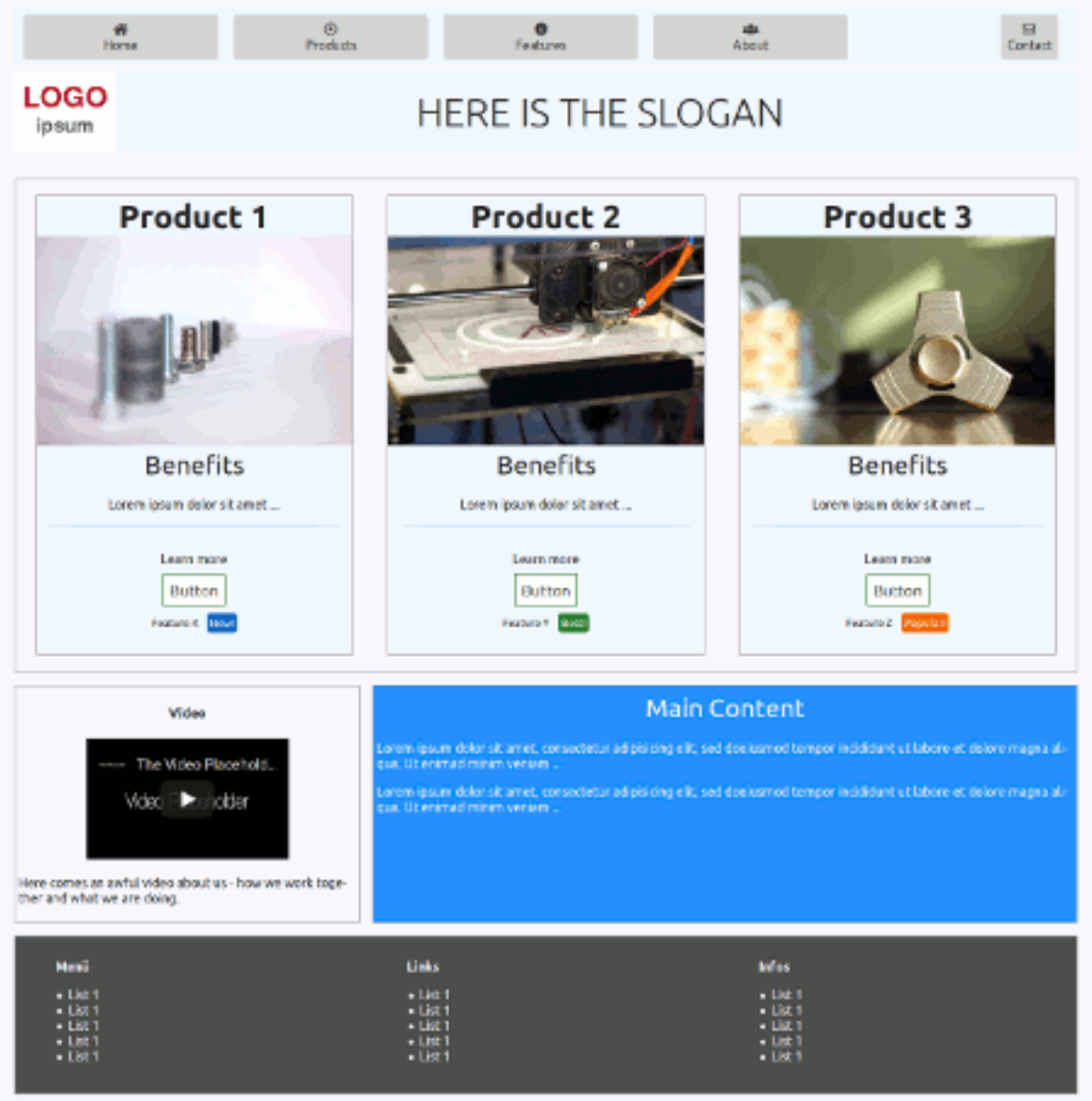
Beispiel für ein Website-Template



Ein Website-Template - schematische Darstellung

Module sind flexible Inhaltselemente oder Inhalts-Komponenten wie Menü, Boxen, Buttons, Listen, Formulare, Tabellen, Slider, Tabs oder Accordions. Also all das, was eine Website zum Leben erweckt.

Beispiel für einen modularen Website-Aufbau



Ein Website gestaltet aus Modulen - schematische Darstellung

Module bestehen aus einer HTML-Grundstruktur, die durch CSS-Klassen unterschiedlich gestaltet werden.

Vorteile von Website-Modulen

- wiederverwendbar und sofort einsetzbar – out of the box
- vielseitig einsetzbar und anpassungsfähig
- frei kombinierbar und flexibel
- einfach zu pflegen und zu ändern
- schnelle Entwicklung und Implementierung

- einfaches gezieltes Redesign

Beispiel Content-Box

Module erleichtern das Coden, da komplexe Codes vermieden und in einfache Einheiten aufgeteilt werden. Nehmen wir als Beispiel eine Content-Box, die in ein HTML-Element `<div>...</div>` gepackt wird und für unterschiedlichste Inhalte verwendet werden kann: Texte mit und ohne Bilder, Hinweise und Tipps, Zitate und andere Elemente, die sich vom normalen Fließtext abheben sollen. Ich werde vier unterschiedliche Boxen gestalten.

Das Grundgerüst ist dabei stets dasselbe Markup:

```
<div class="container">
     <article class="box box-bordered">
          <h4 class="text-center">Simple Bordered Box</h4>
          <p>Hier steht ein Beispieltext, der nichts aussagt.</p>
     </article>
     <article class="box box-shadow">
          <h4 class="text-center">Box with Boxs-Shadow</h4>
          <p>Hier steht ein Beispieltext, der nichts aussagt.</p>
     </article>
     <article class="box box-material">
          <h4 class="text-center">Material Box</h4>
          <p>Hier steht ein Beispieltext, der nichts aussagt.</p>
     </article>
</div>
```

Gestaltet werden die Boxen mit CSS. Das Basisdesign („default“ engl. Standard, Standardwert, Voreinstellung) wird über die Klasse `box` bestimmt:

```
.box {
     width: 100%;
     margin-bottom: 1rem;
     padding: .5rem;
     background-color: #fcfcfc;
     border-radius: .125rem;
}
```

Erste Erweiterung

Nun kann ich durch weitere Klassen die Box nach Wunsch verändern. Eine Box mit Rand sieht dann so aus:

HTML-Code:

```
<article class="box box-bordered">
    …
</article>
```

CSS-Code:

```
.box-bordered {
    border: 1px solid #e5e5e5;
    transition: opacity .25s;
}
.box-bordered:hover {
    opacity: .7;
}
```

Einfache Änderungen durch CSS-Klassen

Hier der Code für eine Box ohne Rand, dafür mit einem leichten Schatten, der den Rand elegant ersetzt und die Box stärker hervorhebt:

HTML-Code:

```
<article class="box box-shadow">
…
</article>
```

CSS-Code:

```
.box-shadow {
    box-shadow: 0 0 2px rgba(0, 0, 0, .5);
    transition: box-shadow 0.25s ease-in-out;
}
.box-shadow:hover {
```

```
    box-shadow: 0 0 2px 2px rgba(0, 0, 0, .3);
}
```

Noch stärker werden die Box und der Box-Inhalt durch folgenden Code betont:

HTML-Code:

```
<article class="box box-material">
    …
</article>
```

CSS-Code:

```
.box-material {
    box-shadow: 0 1px 3px rgba(0, 0, 0, .16), 0 2px 5px rgba(0, 0, 0, .26);
    transition: box-shadow .25s ease-in-out;
}
.box-material:hover {
    box-shadow: 0 7px 21px rgba(0, 0, 0, .26), 0 2px 5px rgba(0, 0, 0, .22);
}
```

Weitere Gestaltungsmöglichkeiten

Die Schrift- und Hintergrundfarbe kann ich durch folgende beispielhafte Klassen verändern:

CSS-Code:

```
.box-bg-maroon {
    background-color: maroon;
}
.box-color-snow {
    color: snow;
}
```

Der entsprechende HTML-Code für die Box mit Rand:

HTML-Code:

```
<article class="box box-bordered box-bg-maroon box-color-snow">
    …
</article>
```

Modul-Präfixe

Das Präfix `box-` dient dazu, das Modul „Content-Box“ unabhängig von anderen Elementen spezifisch als Modul gestalten zu können. Natürlich ist es auch möglich – und in konkreten Projekten auch üblich – soweit wie möglich auf generische Klassen zurückzugreifen. So könnten die Klassen für die Farben auch so lauten:

> `bg-maroon` und `snow`

Rand und Schatten bei den Klassen `box-bordered`, `box-shadow` und `box-material` könnten ebenfalls verallgemeinert werden, wenn die Stilangaben für weitere Elemente wie Buttons, Bilder oder Cards ebenfalls gelten sollen.

Oft jedoch benötigen solche Elemente eigene Stile und daher eigene Klassen wie z. B. `button-bordered`, `button-bg-blue`, `button-color-white`. Das heißt, dass auch diese Elemente modular gestaltet werden.

Schlankes CSS – aufgeblähtes HTML?

Modulare Website-Entwicklung birgt die Gefahr der „Classitis“, d. h. die übermäßige Verwendung von CSS-Klassen. Damit kann der HTML-Quellcode rasch unübersichtlich und aufgebläht werden.

Vorbeugen ist auch bei „Classitis“ billiger als behandeln.

Weder CSS-Dateien noch HTML-Code beeinträchtigen die Performance einer Website, wenn man folgende Hinweise beachtet:

- HTML-Code sinnvoll strukturieren
- semantischen HTML-Code schreiben
- CSS-Code organisieren
- CSS-Code zusammenfassen und nicht wiederholen (DRY-Prinzip: Don‘t repeat yourself)
- unnötige CSS-Styles löschen – weniger Code schreiben
- CSS-Dateien zu einer Datei zusammenführen
- Code richtig schreiben (testen, testen, testen)
- Code minimieren und komprimieren

Methoden der Naming Conventions wie SMACSS, BEM, OOCSS, atomic.css oder suit.css helfen beim modularen Gestalten von Websites.

Zwar geht bei modularer Webentwicklung ein wenig Übersichtlichkeit im HTML-Code verloren, dafür wird der CSS-Code umso übersichtlicher, nicht-redundant und performanter,

wartungsfreundlich und pflegeleicht.

Für modulare Website-Entwicklung sind CSS-Custom Properties („CSS-Variable“) äußerst hilfreich. Mit dem Konzept der CSS-Custom-Properties sollte sich jede Person, die mit Web-Development beruflich zu tun, vertraut machen.

CSS Preprocessors wie SASS, LESS oder Stylus sind von Haus aus modular angelegt und daher sehr gut geeignet für modulare Webentwicklung.

Links

Modulare Website-Entwicklung

https://www.toptal.com/designers/ux/getting-started-with-modular-front-end-development

https://www.knucklepuckmedia.com/blog/why-you-should-adopt-website-modules/

https://medium.com/@wereheavyweight/how-were-using-component-based-design-5f9e3176babb

HTML-Struktur und -Semantik

https://alistapart.com/article/meaningful-css-style-like-you-mean-it/

https://developer.mozilla.org/de/docs/Learn/HTML/Einf%C3%BChrung_in_HTML/Document_and_website_structure

https://www.mediaevent.de/xhtml/html-section-header.html

https://webmasterparadies.de/webdesign/strukturierung-einer-website-in-html-5

https://blog.kulturbanause.de/2008/01/html-elemente-und-semantik/

CSS-Code organisieren

https://speckyboy.com/good-bad-css-practices/

https://www.mediaevent.de/css-organisieren/

https://www.developerdrive.com/2017/11/6-ways-to-organize-your-css/

Hier – auch in den Kommentaren unter dem verlinkten Artikel – gibt es interessante Ansätze, CSS-Klassen im HTML-Code zusammenzufassen:

https://css-tricks.com/could-grouping-html-classes-make-them-more-readable/

Test-Tools

Es gibt jede Menge Werkzeuge, um online Code zu testen, daher hier nur die beiden Tools, die das World Wide Web Consortium (W3C) anbietet:

CSS testen: http://jigsaw.w3.org/css-validator/

HTML testen: https://validator.w3.org/nu/