Responsives UI Design

Wirkungsvolles Mittel gegen zunehmende Gerätefragmentierung?

Nicolas Leyking
Ergosign GmbH
Europaallee 12
66113 Saarbrücken
leyking@ergosign.de

Jan Groenefeld
Ergosign GmbH
Europaallee 12
66113 Saarbrücken
groenefeld@ergosign.de

Markus Kühner
Ergosign GmbH
Europaallee 12
66113 Saarbrücken
kuehner@ergosign.de

Abstract
Softwarehersteller sehen sich heute mit dem Problem einer fast unbeherrschbaren Fragmentierung von Geräten und Betriebssystemen konfrontiert. Der Ansatz, jeder Applikation ein natives Pendant zur Verfügung zu stellen, erscheint schlicht unwirtschaftlich. Im Gegensatz zu einem statischen Design, welches nur sehr beschränkt auf Veränderungen der eigentlichen Zielgröße reagieren kann, wird bei responsivem Design auf ein flexibles, programmtechnisches und konzeptuelles Regelwerk zurückgegriffen. Dabei ist es das Ziel, mit Hilfe dynamischer Platzausnutzung eines beliebigen Anzeigegerätes zu jedem Zeitpunkt eine optimale Präsentation der Inhalte zu gewährleisten. Somit ermöglicht responsives Design einen „Cross Platform"-Ansatz, mit welchem die Gerätefragmentierung beherrscht werden kann.

Keywords:
/// Responsives Design
/// Cross-Platform
/// Layouts
/// Flexibilität
/// Mobile

1. Motivation

Das starke Interesse am Thema „responsives Webdesign" ist in erster Linie der anhaltend starken Verbreitung von mobilen Geräten in unterschiedlichsten Ausprägungen geschuldet. Die seit ca. 2007 auf dem Markt existierenden mobilen Geräte, wie zum Beispiel das iPhone von Apple Inc., erlauben aufgrund ihres ausreichend großen Touch-Screens erstmals eine umfassende Internetnutzung mit einem nahezu vollwertigen Internetbrowser. Die seit Jahren und laut Prognosen auch in Zukunft noch stark wachsende Verbreitung von Smartphones und Tablets [5] erlaubt nicht nur Unternehmen im Bereich des „Casual Computing", also im klassischen Consumer Segment, neue Einsatzfelder für sich zu erschließen, sondern lässt die mobilen Geräte auch für „Business Solutions" immer interessanter werden. Unabhängig von ihrem konkreten Kontext wurden die Inhalte bis vor kurzem meistens nur in einem statischen Design angeboten. Das heißt, Inhalte wie z.B. Websites wurden in der Regel für eine Geräteklasse und teilweise auch Bildschirmauflösung optimiert und programmiert. Die Ursache hierfür liegt augenscheinlich häufig im damit verbundenen Programmieraufwand. Mittels Zoom-Mechanismen können Inhalte dieser Art prinzipiell zwar ohne Probleme auch in mobilen Browsern angezeigt werden, ihre Bedienbarkeit leidet jedoch erheblich. Von einer guten User Experience kann erst recht keine Rede sein. Der daraus hervorgegangene gestalterische und technische Ansatz des responsiven Webdesigns erlaubt es, den grafischen Aufbau und die verwendeten Medien einer Website an ein (fast) beliebiges Ausgabemedium flexibel anzupassen und so eine größere Gerätekompabilität erreichen.

Für viele Unternehmen, die neue Geschäftsfelder erschließen wollen oder an eigenen Prozessoptimierungen interessiert sind, bietet die Mobiliät der Smartphones und Tablets neue Möglichkeiten, innovative Lösungen zu entwickeln. Dabei liegt es auf der Hand, dass eine 1:1 Portierung einer Applikation nicht die bestmögliche Lösung darstellt und mehr Probleme schafft als löst. Probleme entstehen durch Platzmangel und Interaktionskonzepte, die ursprünglich für Maus und Tastatur ausgelegt waren. Die Vorstellung, eine komplexe Layout-Struktur, wie sie zum Beispiel bei klassischen CRM-Systemen anzutreffen ist, 1:1 auf mobile Geräte zu übertragen, verdeutlicht dies. Eine Neu-Konzipierung für den mobilen Kontext bedeutet in der Regel eine komplexe Aufgabe für den Designer und erfordert viel Erfahrung im User Experience Design an sich und im Speziellen in der Domäne mobiler Endgeräte, funktional wie technologisch.

Die folgenden Abschnitte sollen Führung und Hilfestellungen für Designer, Entwickler und Projektmanager geben, um eine bestmögliche User Experience für responsive Applikationen zu erzielen.

2. Begrifflichkeiten und ihre Ansätze

Die folgenden Begriffsdifferenzierungen entstanden auf Basis persönlicher, praktischer Erfahrungen im Themengebiet und sollen durch die eigene Expertise die verwendeten Begriffsdefinitionen insbesondere im Rahmen dieses Papers schärfen.

2.1. Responsives und adaptives Design

Grundsätzlich wird zwischen responsivem und adaptivem Design unterschieden. Die

Schlüsselaspekte sind Flexibilität in der Darstellung und Wirtschaftlichkeit in der Umsetzung, die unter anderem durch sogenannte „Media Queries“ und eine vorausschauende Projektierung erreicht werden.

Bei responsivem Design, das sich insbesondere durch sein dynamisches Layout auszeichnet, werden textuelle und mediale Inhalte mittels „Media Queries“ in Größe, Platzierung und automatischen Umbrüchen variabel an den zur Verfügung gestellten Platz angepasst [1]. Wichtig ist, dass die Neusortierung nicht willkürlich erfolgt, sondern zu jedem Zeitpunkt eine für den Nutzer sinnvolle Anordnung darstellt. Eine wichtige technische Grundlage ist hierbei die Fähigkeit der „Media Queries“, bei allen gängigen Browsern die Breite und Höhe des Browserfensters und Bildschirms sowie die Ausrichtung (Porträt oder Landscape Mode) direkt im CSS Code abzufragen [2] und das Design entsprechend zur Laufzeit zu variieren.

Adaptives Design verfolgt grundsätzlich ein sehr ähnliches Ziel wie responsives Design, baut jedoch ausschließlich auf ein fixes Layout auf, das abhängig von der aktuellen Gerätebreite programmtechnisch auf vordefinierte Darstellungen umstellt.

Der Unterschied zwischen adaptivem und responsivem Design liegt vor allem in der unterschiedlichen Methodik und dem Grad der Anpassungsfähigkeit der Layout-Struktur.

2.1. Webapplikationen und native Applikationen

Responsives Design ist grundsätzlich keiner Technologie unterworfen. Dennoch spricht im Sinne der wirtschaftlichen Projektierbarkeit über viele Geräte und Betriebssysteme hinweg einiges für den Ansatz einer Webapplikation. Denn durch den Ausbau und die Professionalisierung von JavaScript und CSS Frameworks wurden Webapplikationen in den letzten Jahren verstärkt zu einer echten Alternative gegenüber nativ laufenden Applikationen. Webtechnologien wie HTML, CSS und JavaScript ermöglichen es, auf nur einer Codebasis eine dynamisch funktionierende Cross-Platform-Strategie zu verfolgen.

Auch wenn Web-Technologien stark aufgeholt haben, bieten native Applikationen weiterhin die beste Performance und damit das beste Bedienerlebnis [8]. Durch ihre individuelle Entwicklung mit nativem Code erreichen sie auf dem jeweiligen Betriebssystem eine flüssige und stabile UI-Darstellung, die bei Webapplikationen kaum zu erreichen ist. Der Implementierung einer Applikation für mehrere unterschiedliche Plattformen wie zum Beispiel iOS, Android und Windows 8 steht hingegen ein sehr hoher Aufwand in der Entwicklung und Wartung gegenüber.

3. Responsive UI Patterns

Im Folgenden werden einige Problemstellungen und Lösungen aus den Bereichen Layouting und Navigation behandelt und diese mit praktischen Beispielen näher erläutert.

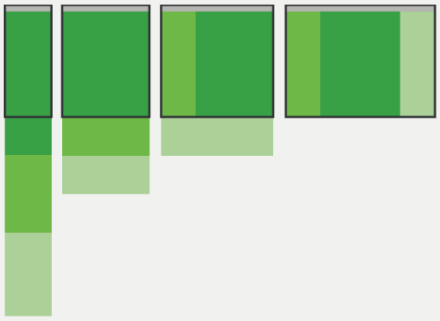

Abb. 1.
Dreispaltiges responsives Layoutkonstrukt [6]

3.1. Layouting – Patterns

Um den Rahmen dieses Papers nicht zu sprengen, wird nun lediglich ein responsives Seitenlayout näher vorgestellt. Weitere Layout Patterns finden sich im Internet, zum Beispiel auf der Website von Luke Wroblewski [6].

Der sogenannte „Column Drop“ ist ein gängiges, responsives Layouting-Konstrukt, bei dem der Seitenaufbau von einem dreispaltigen Desktop-Layout über ein zweispaltiges Tablet-Layout zu einem einspaltigen Smartphone-Layout umgestaltet wird (8). **[Abb. 1], [Abb. 2]**

Ein solches dynamisch gestaltetes Applikationslayout basiert im Wesentlichen auf Regeln des automatischen Umbruchs von Inhalten und begegnet den unterschiedlichen Bildschirmformaten effektiv und automatisiert. Das Regelwerk für eine solche Logik wird häufig auf nur einer

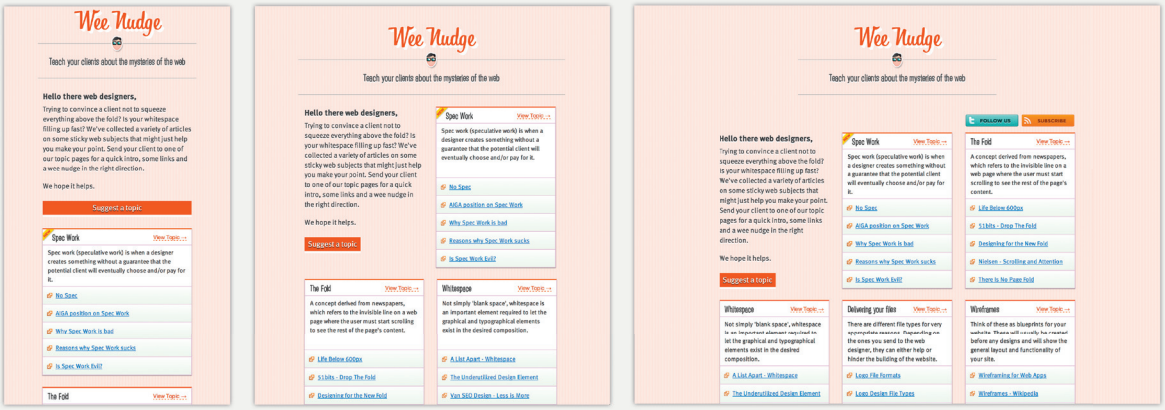

Abb. 2.
Praktisches Beispiel für ein dreispaltiges Layoutkonstrukt [7]

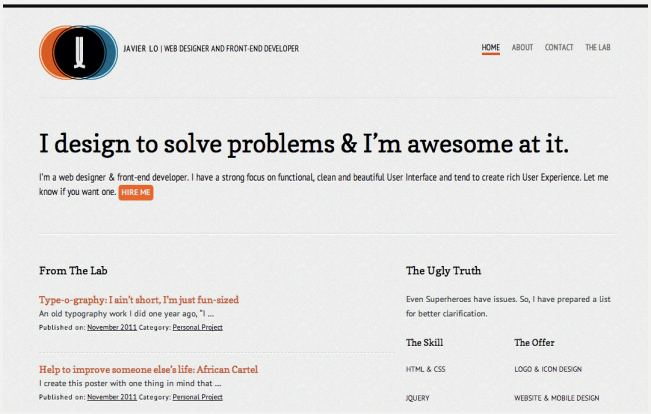

Abb. 3.
Listenartige Darstellung des Menüs in der Desktopvariante [9]

Abb. 4.
Fast identische Menüstruktur im Vergleich zur Desktop-Version [9]

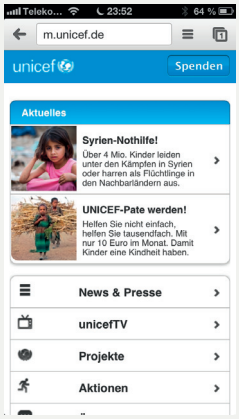

Abb. 5.
Footer-Navigation auf der mobilen Website von Unicef für Smartphones [10]

Abb. 6.
Horizontale Listendarstellung des Menüs für Tablets und Desktops [4]

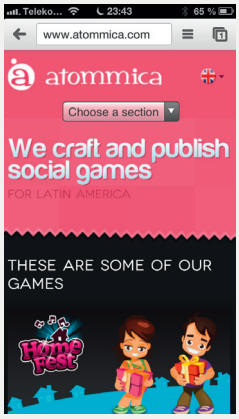

Abb. 7.
Select Menü für kleine Smartphone Bildschirme [4]

gemeinsamen Code-Grundlage formuliert, sodass die Applikation selbst weiterhin als eine einzige Anwendung betreut werden kann. Dies sorgt für die gewünschte Wirtschaftlichkeit bei der Umsetzung und eine erweiterte Gerätekompatibilität.

3.2. Navigation-Patterns

Ein weiteres Beispiel für ein responsives Konstrukt ist die bildschirmoptimierte Darstellung der Menüstruktur. Dabei sind u.a. folgende Lösungsansätze bei responsiven Applikationen weit verbreitet [4].

3.2.1. Der „Es bleibt wie es ist" Ansatz

Dieser Ansatz stellt lediglich die Erreichbarkeit der Navigation sicher, ohne diese konkret für den Einsatz auf Touchgeräten zu optimieren und ist bereits mit wenigen CSS-Anpassungen umsetzbar. Leider geht dabei häufig viel Platz im Kopfbereich der Applikation verloren. **[Abb. 3], [Abb. 4]**

3.2.2. Footer-Navigation

Um dem Inhalt mehr Platz einzuräumen, wird bei dieser Variante die Navigation in den Footer der Website verschoben. **[Abb. 5]** Problematisch bei dieser Variante ist die Tatsache, dass die Navigation sehr versteckt wirkt und u. U. lange „Scrollwege" bis ans Ende der Seite erforderlich sind. Für ein zentrales Navigationselement ist dieser Ansatz daher eher weniger zu empfehlen.

3.2.3. „Select" Menü

Diese Darstellung der Navigationseinträge in einem Dropdown-Menü ist durch ihre einfache Verwendbarkeit häufig anzutreffen. Die Realisierung geschieht durch das Einbinden eines JavaScript Plugins, das die Menüliste in **Select** und **Option** Elemente umwandelt. **[Abb. 6], [Abb. 7]** Nachteile dieser Visualisierungsform sind zum Einen das vom Betriebssystem gesteuerte visuelle Design des Select-Controls und zum Anderen die ungewohnte Darstellung in Form eines Dropdown-Menüs [4].

3.2.4 „Toggle" Menü

Für das „Toggle" Menü Konzept gibt es viele verschiedene Darstellungsmöglichkeiten. Generell öffnet sich eine Menüstruktur, die beliebig auf dem Bildschirm positioniert werden kann. Nachfolgende Abbildungen zeigen mögliche Darstellungsvarianten:

Einfacher „Toggle"

Die mobile Website der „Starbucks"-Kette positioniert das Menü innerhalb des Contents im Zentrum des Bildschirms. **[Abb. 8]**

Verdecktes Menü

Die „dconstruct" Website verfolgt den Ansatz, das Menü von oben auf der gleichen Ebene wie den Inhaltsbereich hineinfahren zu lassen und den Inhaltsbereich um die benötigte Platzmenge hinunterzuschieben. **[Abb. 9]**

Einschub von links

Die mobile Website der „Sycamore School" schiebt das Menü als eine eigene Ebene von links nach rechts über den eigentlichen Content. **[Abb. 10]**

Off-Canvas

Die Off-Canvas-Technik schiebt den Inhaltsbereich der Webseite nach links oder rechts nahezu vollständig aus dem sichtbaren Bereich hinaus. **[Abb. 11]** Diese hochwertig wirkende „Toggle" Menü-Variante ist z.B. auf der mobilen „Facebook" Website anzutreffen.

4. Responsives UI Design in der Anwendung

In diesem Kapitel wird die grundsätzliche Signifikanz sowie Strategien und Vorgehensweisen bei der Konzipierung responsiven Designs vorgestellt und dessen Besonderheiten in einem User-Centered Designprozess (kurz: UCD) verdeutlicht.

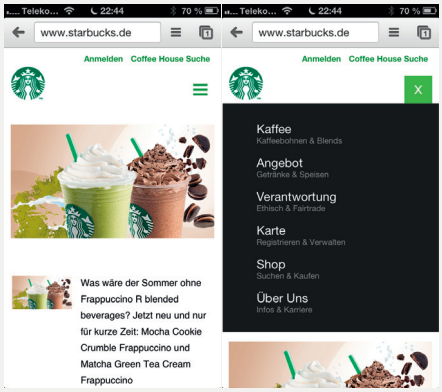

Abb. 8.
Integrierte Menüdarstellung im Inhaltsbereich auf der Starbucks-Website [11]

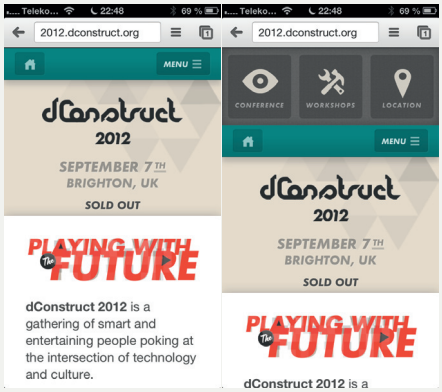

Abb. 9.
Von oben einfahrender Navigationsbereich [12]

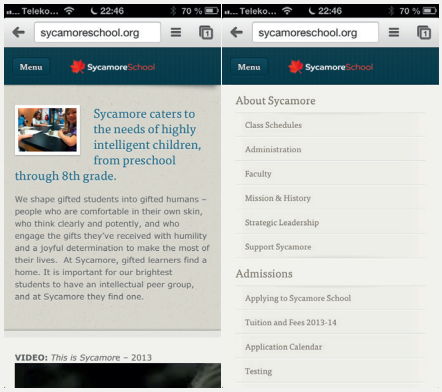

Abb.10.
Von links nach rechts einfahrende Navigationsebene [13]

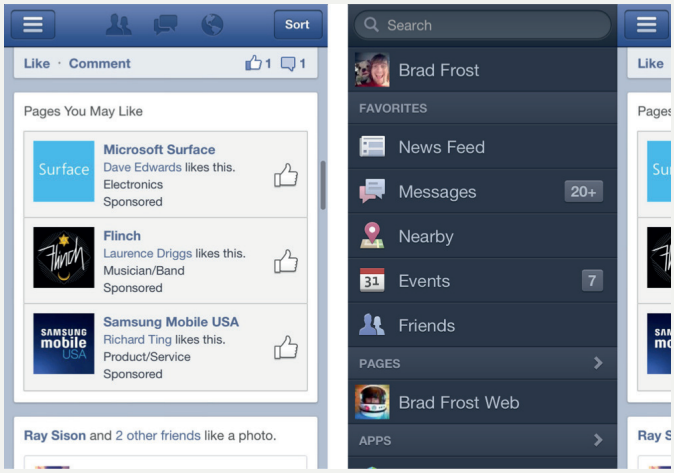

Abb. 11.
Off-Canvas Navigationsansatz [3]

4.1. Der Designprozess

Der im Folgenden vorgestellte responsive Designprozess setzt auf den klassischen UCD Prozess auf und unterteilt sich in seine typischen Phasen und deren Aufgaben. In diesem Abschnitt werden lediglich die **Besonderheiten des responsiven Designprozesses** herausgestellt und näher erläutert.

Analyse

Innerhalb der Analysephase gilt es, verstärkt umfassende Informationen über die primär zu unterstützenden Bildschirmgrößen und die zugrunde liegenden Gerätekonzepte zu sammeln. Dieses Wissen über Gerätetypen, Gerätehersteller, Ausgabegrößen und Betriebssystemkonzepte bietet die Grundlage jedes responsiven Designs und sollte, um über das gesamte Projekt hinweg den Überblick zu behalten, an zentraler Stelle dokumentiert und visualisiert werden, ähnlich wie es mit „Personas" gehandhabt wird. Der Nutzungskontext ist hier ein zentraler Bestandteil der Analyse, da sich hier häufig Rückschlüsse auf die benötigten Geräteklassen ableiten lassen.

Design

Auf Grundlage des gewonnenen Verständnisses über technische Rahmenbedingungen und gerätespezifische Interaktionskonzepte aus der Analysephase beginnt der UI-Designer in dieser Phase mit seinen ersten Überlegungen zu einem passenden flexiblen UI-Konzept. Dabei lässt sich bei der Neukonzeption einer responsiven Applikation keine generelle Empfehlung für den „Mobile First" oder „Desktop First" Ansatz aussprechen. Nach dem „Desktop First" Ansatz wird zunächst die maximale Zielgröße, meist die des Desktop-PCs, betrachtet. Durch die größere Fläche können hierbei mehrere Designelemente im Blick behalten und verschiedene Darstellungsvarianten ausprobiert werden. Mit dem „Mobile First" Ansatz hingegen wird, beginnend mit der kleinsten anzunehmenden Geräteklasse, die Reduktion auf die wesentlichen Elemente gefördert. Es muss aufgrund der kontextabhängigen Funktionen bei jedem responsiven Designprojekt neu zwischen „Mobile First" und „Desktop First" abgewägt werden. Beeinflussen zum Beispiel besondere Funktionen der mobilen Endgeräte, wie u.a. die Ausrichtung (Porträt oder Landscape-Modus) oder die Geo-Lokalisierung das UI, sollte eher der „Mobile First" Ansatz verfolgt werden. Sind solche Funktionen eher unwichtig, ist es mit der „Desktop First" Variante einfacher, Antworten auf Fragen wie „Wie ordne ich meine Inhalte an?" oder „Welchen Gesamteindruck möchte ich erhalten?" zu bekommen [4]. In manchen Fällen ist bereits eine Desktop-Lösung vorhanden, wodurch sich diese Fragestellung erübrigt.

Prototyping

In der Praxis hat sich außerdem gezeigt, dass Hilfsmittel wie Photoshop oder Fireworks aufgrund ihrer fixen Größenangaben bei responsiven Designs schnell an ihre Grenzen stoßen. Aus diesem Grund ist es zu empfehlen, dass der UI-Designer seine Ideen zu einer responsiven Layoutstruktur schon frühzeitig in leichtgewichtige HTML-Prototypen überführt. Diese sehr einfachen Prototypen dienen lediglich dem Zweck, das Layout und sein Verhalten auf mehreren unterschiedlichen Bildschirmgrößen zu testen. Dabei empfiehlt es sich, so viele unterschiedliche Bildschirmgrößen wie möglich, jedoch zumindest die typischen Größen genauer unter die Lupe zu nehmen.

Das Testen der leichtgewichtigen HTML-Prototypen auf den verschiedenen Endgeräten gibt dabei dem UI-Designer unmittelbar Hinweise darauf, wie er das UI „umgestalten" muss, um eine möglichst optimale Platzausnutzung und Bedienbarkeit zu gewährleisten. Die dabei gewonnenen Erkenntnisse sollten wiederum in den Prototypen eingearbeitet werden, um durch diese iterative Vorgehensweise zwischen Grobkonzeption und Testen ein stabiles Layoutkonzept zu entwickeln. Erst danach wird mit der Feinkonzeption, wie zum Beispiel dem Erstellen der benötigten Controls und ihrer Positionierung in den einzelnen Layoutmodellen begonnen. Auch dabei gilt es, sich stets an ein frühes und kontinuierliches Testen zu halten, da nur so alle Bildschirmgrößen gleichermaßen berücksichtigt werden können. Das Ergebnis der Konzeption beinhaltet das flexible UI-Layout und die darin positionierten Controls, Media Daten und sonstige Inhalte.

Validierung

Die responsiven Besonderheiten der Validierungsphase sind in dem notwendigen erweiterten Testumfang wiederzufinden. Das Design sollte dabei auf möglichst allen unterschiedlichen Bildschirm- und Plattformtypen getestet werden, um ein möglichst allumfassendes Bild des responsiven UI zu erhalten. Es empfiehlt sich, die Validierungsergebnisse schriftlich zu dokumentieren und bei Bedarf in das bisher entstandene Applikationslayout zu überführen.

Spezifizierung

Für die Spezifizierung responsiven Designs bietet es sich an, die klassischen Spezifizierungsdokumente mit den in der Designphase erstellten HTML-Prototypen anzureichern, um so den Entwicklern die Möglichkeit zu geben, das UI und sein Verhalten besser zu verstehen. Viele responsive UI-Besonderheiten sind schwierig textuell zu beschreiben und können besser anhand von Prototypen verdeutlicht werden.

4.2. Responsives Design anhand eines praktischen Beispiels

Die im Folgenden demonstrierten praktischen Beispiele für responsive UI-Elemente entstammen einer Webapplikation der SAP AG, die Manager bei ihren Verwaltungs- und Planungsaufgaben unterstützt. Aufgrund der agilen Arbeitswelt der Manager eignet sich eine responsive Umsetzung der Applikation ganz besonders. Als primäre Zielplattformen galt es dabei, die mobilen iOS Geräte wie iPad und iPhone besonders im Auge zu behalten. Desweiteren sollten mobile Android- und Windows-Geräte unterstützt werden, allerdings mit einer geringeren Priorität.

Master-Detail-Konzept

Das im Landscape Mode eines Tablets angewandte Master-Detail-Konzept mit einem feststehenden Master-Bereich auf der linken Seite des Screens und dem Detail-Bereich rechts davon funktioniert auf allen querformatigen Tabletausrichtungen als ein plattformübergreifendes Interaktionskonzept gleichermaßen gut. Dieses einfache, im mobilen Kontext sehr häufig vorkommende UI-Pattern kommt auch in dieser Web-Applikation zum Einsatz. **[Abb. 12]**

Im Portrait Mode wird der Master-Bereich ausgeblendet und der Detail-Bereich zu einem Eintrag nimmt den vollständigen Bildschirm des Gerätes ein. **[Abb. 13]** Die „Shopping Cart"-Einträge lassen sich durch einen Button in der linken oberen Ecke ein- und ausblenden. **[Abb. 14]**

Auf den kleineren Bildschirmen der Smartphones wäre eine analoge Transformation dieses Tablet-Layouts aufgrund des eingeschränkten verfügbaren Platzes nicht sinnvoll, weshalb dieses UI-Pattern eine separate Darstellungsvariante von Master- und Detail-Bereich erfordert. **[Abb. 15], [Abb. 16]** Für diese Darstellungsvariante ist jedoch eine neue Navigationsmöglichkeit im responsiven Design vorzusehen, um von dem Detail-Bereich wieder zurück in den Master-Bereich wechseln zu können.

Fly-Out vs. Fullscreen-Konzept

Eine Methode, detailliertere Informationen zu einem UI-Element anzuzeigen, ist die Verwendung eines Fly-Out Containers, der über dem Content-Bereich liegt. **[Abb. 17], [Abb. 18]** Dieses Konzept, das weitestgehend auf Geräten mit größeren Bildschirmen, wie Tablets und stationären Geräten zu finden ist, ist für die kleinen Smartphone-Screens nicht immer sinnvoll und es empfiehlt sich, ab einer bestimmten Informationsmenge einen eigenen Screen für die Informationen vorzusehen. **[Abb. 19], [Abb. 20]** Hierzu sind die oben erwähnten „Media Queries" und flexiblen Größenangaben vonnöten, damit sich das Layout an alle Smartphone-Bildschirmgrößen anpasst.

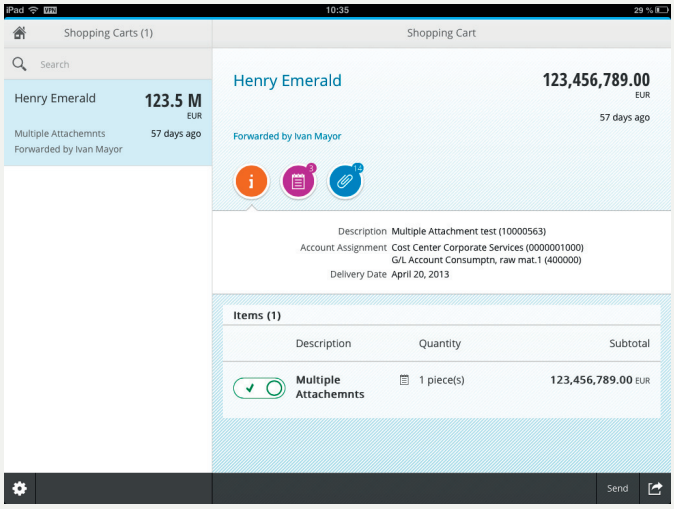

Abb. 12.
Shopping Cart im Landscape Mode auf dem iPad [14]

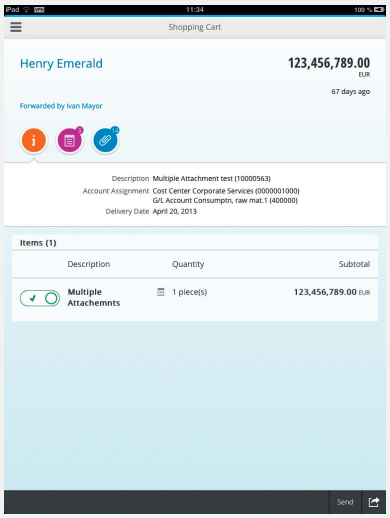

Abb. 13.
Shopping Cart Element-Detailansicht auf dem iPad im Portrait Mode [14]

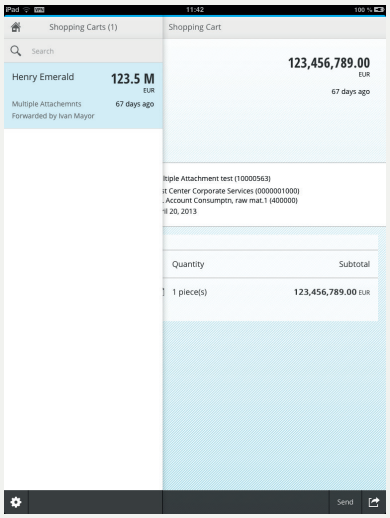

Abb. 14.
Shopping Cart im Portrait Mode auf dem iPad mit eingeblendetem Master-Bereich [14]

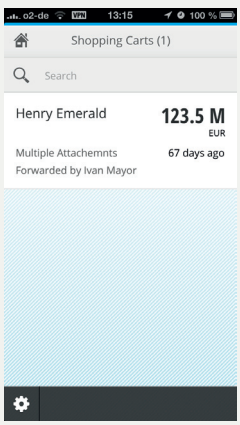

Abb. 15.
Shopping Cart Master-Screen auf dem iPhone [14]

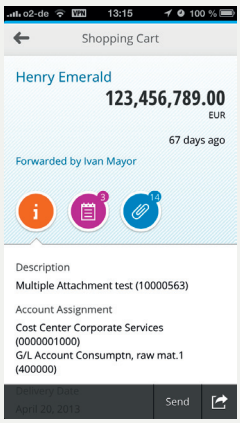

Abb. 16.
Shopping Cart Detailseite auf dem iPhone [14]

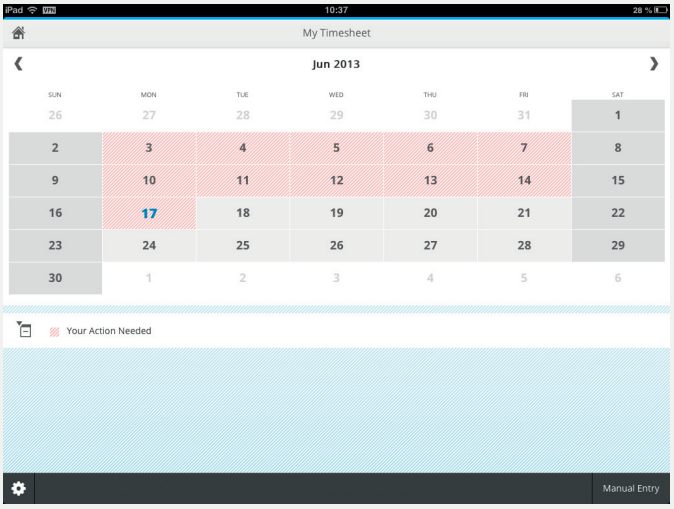

Abb. 17.
Kalenderdarstellung auf dem iPad [14]

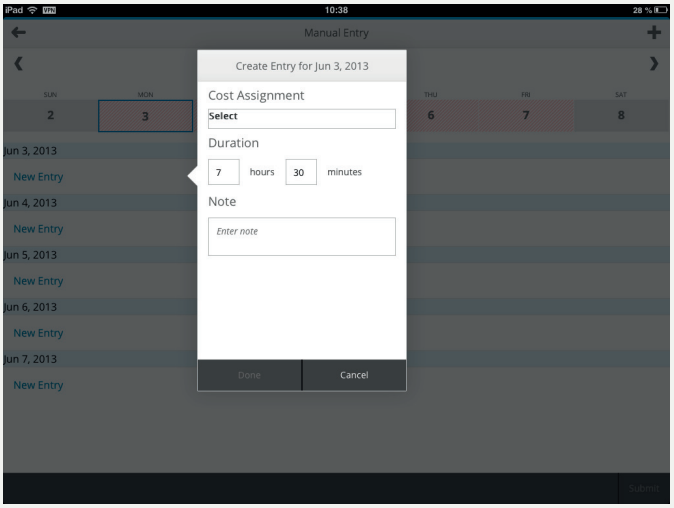

Abb. 18.
Kalenderdarstellung auf dem iPad mit geöffnetem Fly-Out [14]

Abb. 19.
Kalenderdarstellung auf dem iPhone [14]

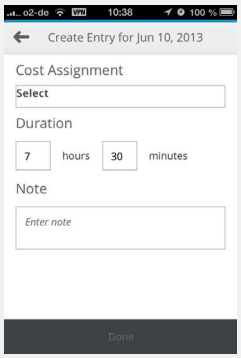

Abb. 20.
Separater Screen zum Anlegen eines „Cost Assignments“ [14]

4.3. „Dos and Don’ts“ in responsiven Designprojekten

In diesem Abschnitt werden „Dos and Don'ts“ in responsiven Designprojekten aufgeführt, die durch die Projekterfahrungen entstanden sind. Die hier aufgelisteten Ratschläge sind als grundlegende praktische Empfehlungen für responsive Designprojekte zu verstehen. **[Tab. 1]**

5. Fazit

Wie das vorangegangene Kapitel 4 zeigt, ist der Gestaltungsprozess für ein responsives Design eine aufwendige, komplexe Aufgabe, die das gleichzeitige Konzipieren eines flexiblen Layouts für verschiedene Bildschirmgrößen erfordert. Die kontinuierliche Verbreitung [5] und der wachsende Einsatz mobiler Geräte in Unternehmen machen Cross-Platform-Lösungen jedoch unabdingbar. Der Trend, Mitarbeitern zu erlauben, ihr (privates) mobiles Gerät geschäftlich zu nutzen („Bring your own Device“) verschärft das Problem der Gerätefragmentierung zusätzlich.

Die gemeinsame Codebasis eines responsiven Designs trägt hierfür zwar zum Einen zu einer wirtschaftlichen, also kostengünstigen Implementierungsphase bei und hält den zukünftigen Wartungsaufwand gering, erzeugt aber zum Anderen einen sehr hohen Aufwand in den oben genannten Designprozessen. Diese Balance aus Aufwänden in Design und Entwicklung erscheint symptomatisch für Projekte dieser Art und muss für jede Anwendung neu austariert werden.

Abschließend lässt sich die Anfangsfrage „Responsives UI Design – Wirkungsvolles Mittel gegen zunehmende Gerätefragmentierung?“ zwar mit „Ja“ beantworten, allgemeingültig ist diese Antwort jedoch aufgrund des variablen Aufwand-Nutzen-Verhältnisses nicht. Bei Applikationen, die ein komplexes User Interface erfordern, muss genau zwischen diesem Verhältnis abgewogen werden, da die Kosten, die beim Umsetzen des Designs entstehen,

Dos	Don'ts
– Sorge für ein teamweites Wissen und Verständnis über responsives Design.	– Einsatz von fixen Layouting-Tools (z.B. Photoshop oder Fireworks)
– Durchdringe deine Zielplattformen und lerne ihre User Experience im Details kennen.	– Entwicklungsbeginn vor Abschluss und Abnahme des Designs
– Teste dein Layout mit leichtgewichtigen Prototypen so früh und auf so vielen Plattformen wie möglich.	– 1:1 Übertragung des Designs von Desktop auf Mobile
– Stelle die Prototypen den Entwicklern in der anschließenden Entwicklungsphase zur Verfügung.	– Verzicht auf relevante Inhalte aufgrund von Platzmangel. Inhalte sollten in diesem Falle anderweitig zugänglich gemacht werden.
– Sprich mit langjährigen Benutzern der Plattformen über deine Layout-Entwürfe.	
– Dokumentiere das Verhalten des Layouts durch anschauliche Mittel.	

Tab. 1.
Dos and Don'ts

das Verhältnis von Aufwand und Nutzen negativ beeinflussen. Um die ohnehin sehr umfangreiche Konzeptionsphase nicht ausufern zu lassen, ist zu empfehlen, responsives Design bei überschaubaren und nicht zu komplexen Applikationen anzuwenden, denn nur bei solchen Anwendungen können die Aufwendungen für die Designphase möglichst gering und die positiven Effekte, die durch die gemeinsame Codebasis in der Implementierung und Wartung auftreten, klein gehalten werden.

Wegen dieser Verlagerung ist für die gezielte Vorhersage der Darstellungsfaktoren und die sinnvolle automatische Anpassung der Applikation die enge Zusammenarbeit von Designern und Entwicklern zwingend vonnöten. Das hochgesteckte Ziel liegt in einer homogenen (Marken-) Darstellung des User Interface bei gleichzeitig größtmöglicher Gerätekompatibilität und Wirtschaftlichkeit.

Literatur

1. Ethan Marcotte, Responsives Web Design, A Book Apart ,(2011)
2. W3C, Media Queries http://www.w3.org/TR/2012/REC-css3-mediaqueries-20120619/ (27. Juni 2013)
3. Peter-Paul Koch, Stephanie Rieger et al, The Mobile Book, Smashing Magazine (2012)
4. Christoph Zillgens, Responsive Webdesign, Carl Hanser Fachbuchverlag, (2012)
5. Gartner, Inc., Worldwide PC, Tablet and Mobile Phone Combined Shipments, http://www.gartner.com/newsroom/id/2408515 (27. Juni 2013)
6. Luke Wroblewski, Multi Device Layout Patterns, http://www.lukew.com/ff/entry.asp?1514 (27.Juni 2013)
7. Wee Nudge, http://weenudge.com/ (22. August 2013)
8. Tim Wendorff, Native Apps vs. Webapps, http://onlinemarketing.de/news/native-apps-vs-webapps (8. Juli.2013)
9. Javier Lo, http://www.javierlo.com/ (27. Juni 2013)
10. Unicef Deutschland, http://m.unicef.de/ (27. Juni 2013)
11. Starbucks, http://www.starbucks.de/ (27. Juni 2013)
12. ClearLeft, http://2012.dconstruct.org/ (27. Juni 2013)
13. SycamoreSchool, http://sycamoreschool.org/ (27. Juni 2013)
14. SAP AG, https://experience.sap.com/fiori (17. Juli 2013)