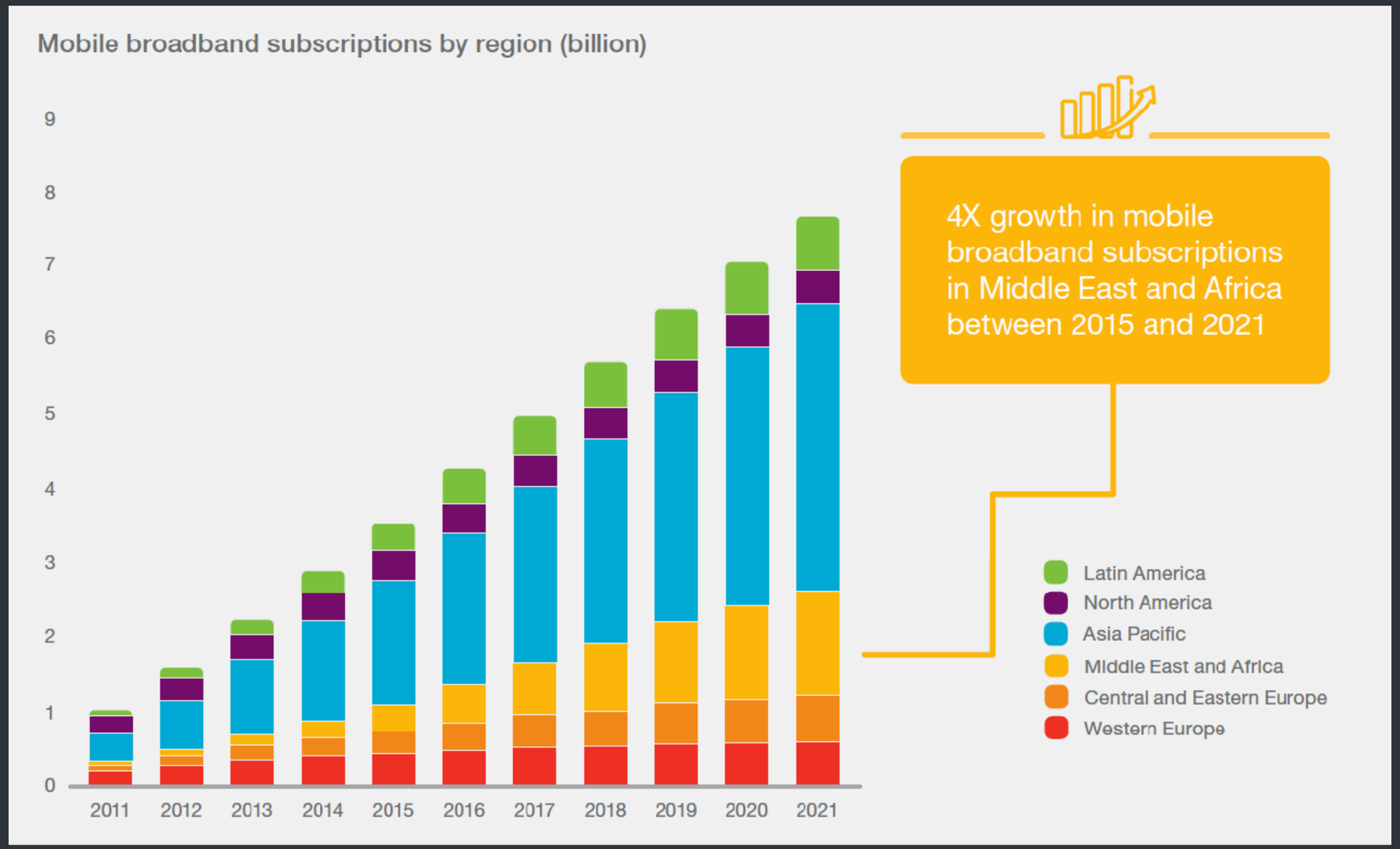
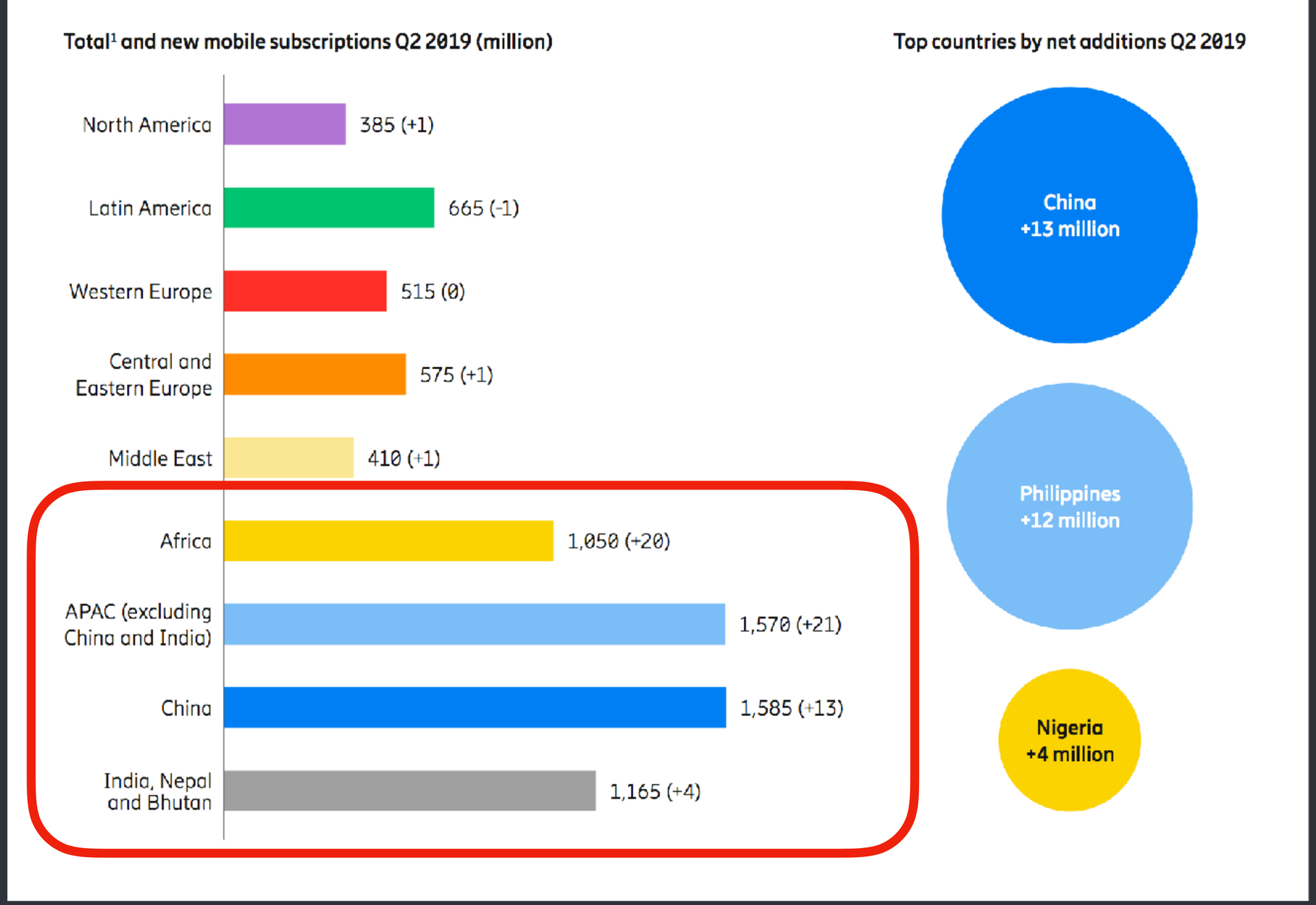
Mobile Web Animation

Lisi Linhart



Ericsson Mobility Report



Ericsson Mobility Report

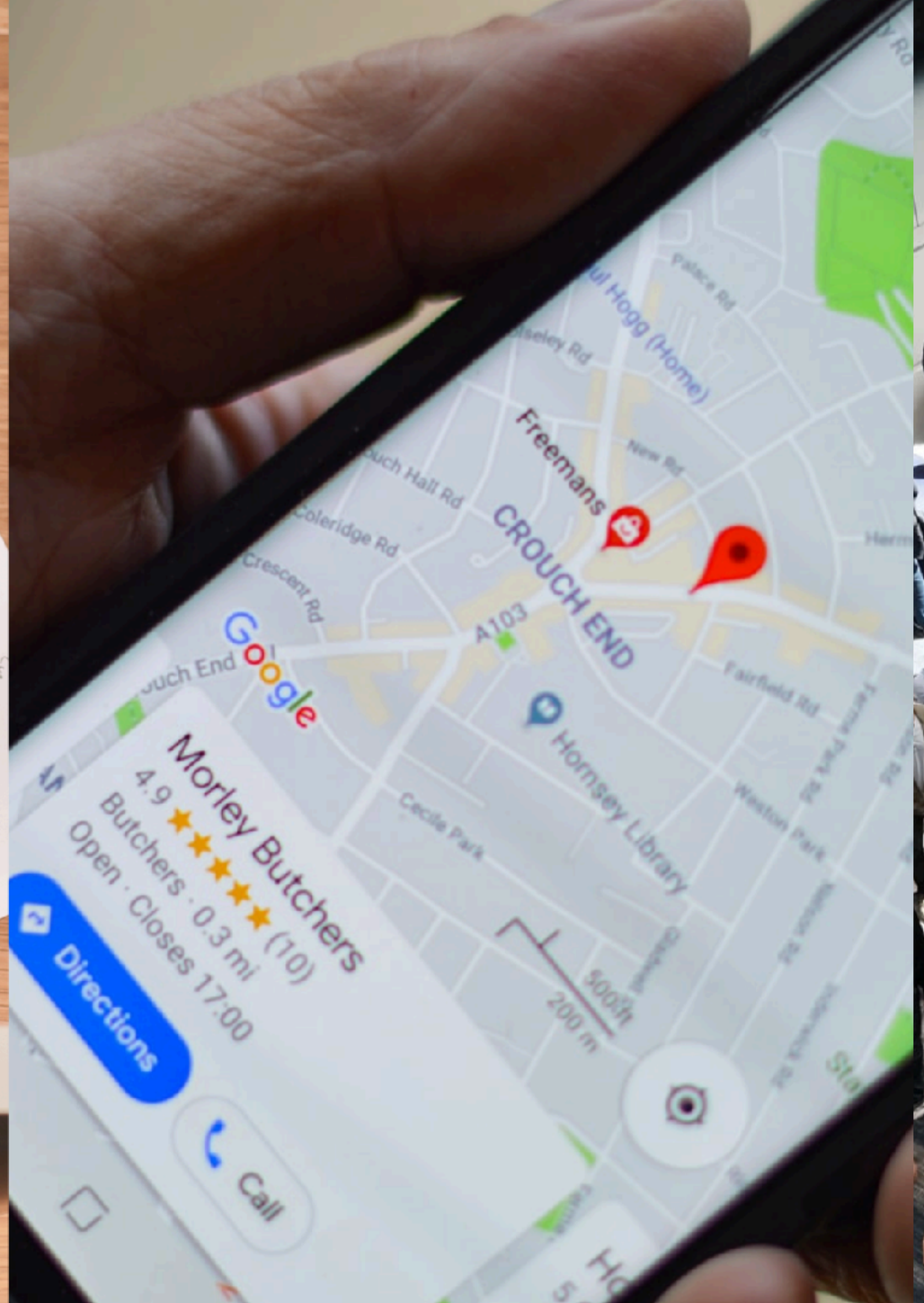
Existing research identifies several avenues through which mobile phones, and smartphones in particular, may contribute to the fortunes of disadvantaged populations.

Mobile phones and inequality - Will Marler

Mobile phones are considered separately as tools which are used to take advantage of digital resources, strengthen and grow personal networks and enhance coordination and mobility in everyday life.

Mobile phones and inequality - Will Marler

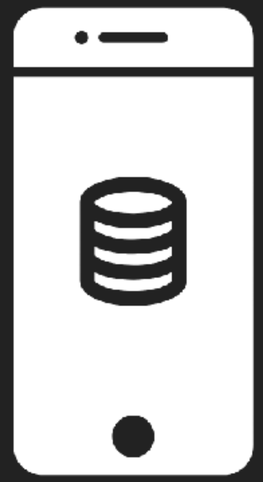

MAP
VIDEO
LIKE?
KEYBOARD
LISTS
HEADER

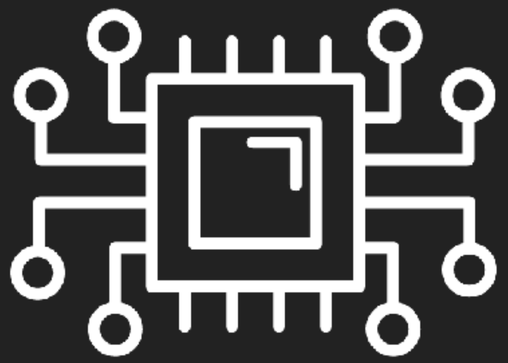

Freemans
CROUCH END
A103
Google
Morley Butchers
4.9
(10)
Butchers · 0.3 mi
Open · Closes 17:00
Directions
Call
Hornsey Library
Cecile Park
Fairfield Rd
Coleridge Rd
Crescent Rd
500ft
200 m

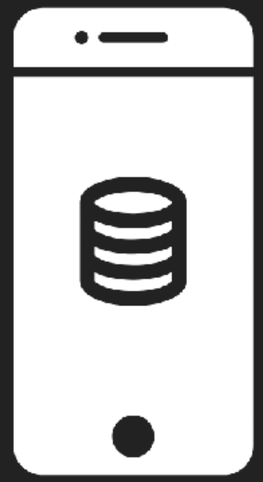

UNDERSTAND
EVERY USER'S
JOURNEY WITH
INSTANT INSIGHTS

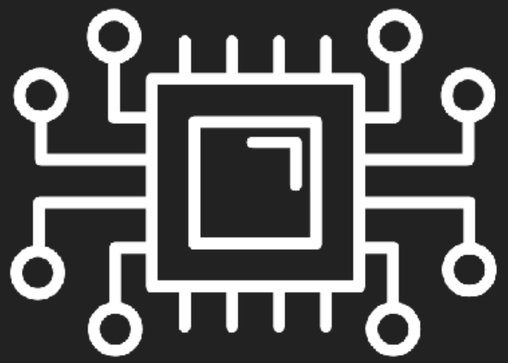
The notion is that a gap has emerged between those who access the Internet by phone and those who do so by computer known as the “device divide”.
The researchers find that those marginalized by race, income, and education are more likely to depend on a smartphone for Internet access.

Mobile phones and inequality - Will Marler









my.headspace.com/dis

PACKS SINGLES KIDS ANIM

CONTACT US

OUR COMMUNITY

BLOG

GET THE APP

© 2019 HEADSPACE INC

TERMS & CONDITIONS

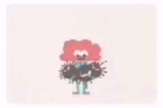
Home discover profile

happy

Happiness
10-20 MIN • COURSE

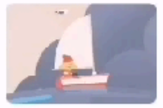
Happiness of Others
1 MIN • ANIMATION

Happiness x NBA
10-20 MIN • COURSE

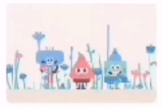


Skillful Compassion
<1 MIN • ANIMATION

Precious Human Life
1 MIN • ANIMATION



Shared Human Condition
1 MIN • ANIMATION



Loving Kindness
<1 MIN • ANIMATION

- not all apps need to be standalone
- Web APIs allow access to device features (BlueTooth, USB, Camera, Geo-location, …)
- no need for app / play store admission
- no separate development effort for Android / iOS
- often smaller in size
- higher conversion (no need for app install)

Touch Interaction and Animation in Mobile Web Interfaces

Usability Engineering

JAKOB NIELSEN

SunSoft
2550 Garcia Avenue
Mountain View, California

Morgan Kaufmann
AN IMPRINT OF ACADEMIC PRESS
A Harcourt Science and Technology Company
San Diego San Francisco New York Boston
London Sydney Tokyo

Engineering Animations in User Interfaces

Thomas Mirlacher[1, 2], Philippe Palanque[2], Regina Bernhaupt[1, 2]
[1]ruwido, Köstendorferstraße 8, A-5020 Neumarkt, Austria
[2]ICS-IRIT, University of Toulouse, 118 Route de Narbonne, F-31062, Toulouse, France
thomas.mirlacher@ruwido.com, palanque@irit.fr, regina.bernhaupt@ruwido.com

ABSTRACT
Graphical User Interfaces used to be static, graphically representing one software state after the other. However, animated transitions between these static states are an integral part in modern user interfaces and processes for both their design and implementation remain a challenge for designers and developers.

This paper proposes a Petri net model-based approach to support the design, implementation and validation of animated user interfaces by providing a complete and unambiguous description of the entire user interface including animations. A process for designing interactive systems focusing on animations is presented, along with a framework for the definition and implementation of animation in user interfaces. The framework proposes a two levels approach for defining a high-level view of an animation (focusing on animated objects, their properties to be animated and on the composition of animations) and a low-level one dealing with detailed aspects of animations such as timing and optimization. A case study (in the domain of interactive Television) elaborating the application of the presented process and framework exemplifies the contribution.

Author Keywords
Animation; Interaction Design; Software Engineering Methods and Processes; User Interface Design.

ACM Classification Keywords
D.2.2 [Software] Design Tools and Techniques - Computer-aided software engineering (CASE), H.5.2 [Information Interfaces and Presentation]: User Interfaces - Interaction styles.

INTRODUCTION
While today most parts of a graphical user interface are static, animations are being increasingly used. This increase of use, especially for areas away from desktop applications, can be attributed to the fact that resources remain available after the system's main functions are executed. Areas where animations can be found include domains such as education [5,33], representation of life-like behavior on simulated objects [8,29] or support understanding of dynamic systems (e.g. programs) [17,31] and for GUIs [1].

Using animations in these application domains can decrease cognitive load [12] and increase usability, as they visually support the users in understanding the system's behavior and evolution from the current state to the future one. With animations, user interface evolutions can be represented in a "natural" way (similar to state changes occurring in an animated and continuous way in real life) possibly conveying additional information pertinent for the users' tasks.

While animations might increase usability, they also increase the complexity both for the specification and implementation of the software part of interactive systems, and therefore the probability of occurrence of faulty or undesired behaviors. For instance, the time-based nature of animations makes them hard to specify and hard to define and assess the detailed temporal behavior prior to implementation.

This paper proposes a Petri net model-based approach to bridge the gap between a multidisciplinary team responsible for the design, implementation and validation of animated user interfaces by providing a complete and unambiguous description language for defining animated interfaces. The connection of the animated behavior with the rest of the user interface is presented, making it possible to embed animations in a seamless way.

The following section starts with an overview of the engineering process representing explicit phases dealing with animation design and prototyping. It describes the engineering process from the requirements gathering to a fully animated user interface running on a dedicated execution platform. Section 2 contains an introduction of animations and classifies animations according to their type and goals and describes the properties that can be manipulated when designing and implementing animations. Section 3 together with Section 4 present the low and high fidelity prototyping of animations. Section 5 is dedicated to the motivation for modeling as well as the the formal modeling of animations. It presents the description technique used and how to handle the modeling of every component of the framework. Section 6 describes the application of the approach with a real-size case study from the Interactive Television (iTV) domain. The last section concludes the paper emphasizing the advantages and limitations of the approach and outlines future work.

EICS'12, June 25–26, 2012, Copenhagen, Denmark.

World Academy of Science, Engineering and Technology
International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:2, No:2, 2008

Animated Versus Static User Interfaces: A Study of Mathsigner™

Scott Dyer, and Nicoletta Adamo-Villani

International Science Index, Computer and Information Engineering Vol:2, No:2, 2008 waset.org/Publication/9299

***Abstract*—**In this paper we report a study aimed at determining the effects of animation on usability and appeal of educational software user interfaces. Specifically, the study compares 3 interfaces developed for the Mathsigner™ program: a static interface, an interface with highlighting/sound feedback, and an interface that incorporates five Disney animation principles. The main objectives of the comparative study were to: (1) determine which interface is the most effective for the target users of Mathsigner™ (e.g., children ages 5-11), and (2) identify any Gender and Age differences in using the three interfaces. To accomplish these goals we have designed an experiment consisting of a cognitive walkthrough and a survey with rating questions. Sixteen children ages 7-11 participated in the study, ten males and six females. Results showed no significant interface effect on user task performance (e.g., task completion time and number of errors); however, interface differences were seen in rating of appeal, with the animated interface rated more 'likeable' than the other two. Task performance and rating of appeal were not affected significantly by Gender or Age of the subjects.

***Keywords*—**Animation, Animated interfaces, Educational Software, Human Computer Interaction, Multimedia.

I. Introduction

A few research studies have reported the cognitive and affective benefits of incorporating the principles of animation in software user interfaces. In this paper we add to the relatively small body of literature in this area of research by presenting a study aimed at investigating usability and appeal differences of three versions of the Mathsigner™ interface (two static and one animated).

Mathsigner™ [1] is a 3D animation-based program for teaching math concepts and related American Sign Language (ASL) signs to deaf and hearing children in grades K-3. The software is currently under development and undergoing a series of formative evaluations. The results of this study will be used to inform the final decision on which interface design to adopt.

Manuscript received January 30th, 2008. This work was supported in part by NSF-RDE grant#0622900, by the College of Technology at Purdue University (grant#00006585), and by the Envision Center for Data Perceptualization at Purdue University.
Scott K. Dyer is a graduate student in the department of Computer Graphics Technology at Purdue University, West Lafayette, IN 47906 USA (phone: 219-588-2125 e-mail: skdyer@purdue.edu).
Nicoletta Adamo-Villani is Assistant Professor in the department of Computer Graphics Technology at Purdue University West Lafayette, IN (e-mail: nadamov@purdue.edu).

The paper is organized as follows: In section II we (a) discuss the 12 principles of animation, (b) present a summary of previous work on animated interfaces, and (c) describe the Mathsigner™ application and its evaluation methodology. In section III we present the user study, and in section IV we report the findings. Discussion of results and conclusive remarks are included in section V.

II. Background

A. The 12 Principles of Animation

"Between the 1920's and the late 1930's animation grew from a novelty to an art form at the Walt Disney Studio" [2]. Thanks to Disney, the growth and development of traditional animation helped produce 12 fundamental principles [3] that are still used and applied today. These principles include: Squash and Stretch; Timing; Anticipation; Staging; Follow Through and Overlapping Action; Straight Ahead Action and Pose-To-Pose Action; Slow In and Out; Arcs; Exaggeration; Secondary Action; Appeal; Solid Drawing.

We have implemented 5 of these 12 principles in our animated interface:(1) *stretch and squash*- Techniques that define object rigidity and mass by manipulating or distorting its shape during an action; (2) *follow through and overlapping action*- Upon ending one action, establishing a relationship to the next action; (3) *slow-in slow-out*- The spacing of the in-between frames to achieve acceleration and deceleration at the beginning and ending of the action, respectively; (4) *Anticipation* – Preparing the audience for an action to come; and (5) *Arcs* -The curved path of action for movement.

The selection of these 5 principles was based partially on research by Chang and Ungar [4], and was motivated by the following considerations. Changes of shapes (e.g., stretch and squash) give objects solidity and provide users with selection feedback. Follow-through and anticipation make inter-connections between different interface states logical and clear, while slow-in-slow-outs provide smooth transitions between them. Motion that describes arcs is visually pleasing and can increase the overall appeal of the interface.

B. Previous Work on Animated User Interfaces

A few research studies report the benefits of implementing animation in software user interfaces [5-8]. The two main benefits highlighted by all the studies are (1) a reduced user's cognitive load and (2) a higher level of appeal. Cognitive benefits result primarily from the fact that animation allows

The Communicative Functions of Animation in User Interfaces

David Novick
The University of Texas at El Paso
El Paso, TX 79968-0518
novick@utep.edu

Joseph Rhodes
Georgia Institute of Technology
Atlanta, Georgia 30332
jrhodes@gatech.edu

Wervyn Wert
Georgia Institute of Technology
Atlanta, Georgia 30332
wervyn@gmail.com

ABSTRACT
To develop a model that relates the purpose of the communication to the nature of the animation, we surveyed existing user interfaces that use animation, analyzed these uses with respect to type of animation and communicative function, and considered ambiguous or otherwise difficult cases. From this analysis, we constructed a matrix with appropriateness/inappropriateness values for all combinations of communicative functions and animation types covered by our survey. To illustrate how the model could be applied to graphical user interfaces and to assess the model's plausibility, we used the model to develop two versions of a user interface for an MP3 player.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *graphical user interfaces (GUI), style guides, Theory and methods.*

General Terms
Design, Human Factors, Standardization

Keywords
Animation, design patterns

1. INTRODUCTION

Most people equate animated content with useless content.
—Jakob Nielsen [17]

Nielsen aimed his maxim at designers of user interfaces who use Flash with, as he put it, no purpose beyond annoying people. But designers who seek to use animation in a more integrated way, as an inherent part of the user interface rather than as something merely to catch the eye, find scant support in the interface-design literature. Developers can rely on resources that provide design patterns that, incidentally, use animation, but they are largely on their own if they seek more generally applicable, model-based advice. What do different kinds of animation communicate to the user? When should a designer consider introducing animation into an interface, or changing the nature of an existing animation in an interface? Answers to these sorts of questions, which touch on fundamental but relatively unexplored issues in the design of communication, require a more comprehensive and systematic connection between the form of different animations and what they communicate. In this paper we claim that the key communicative functions of animation in user interfaces can be characterized by a model that relates the purpose of the communication to the nature of the animation. Such a model could support designers who seek to use animation to expand the effectiveness and usability of user interfaces.

In this paper, then, we review the state of art with respect to models of animation in user interfaces, survey the uses of animation in existing and proposed user interfaces, present an initial model that describes the appropriateness of different animation types for different communicative functions, and illustrates how the model could be applied to graphical user interfaces by presenting two versions of a user interface for an MP3 player. We conclude with a discussion of the strengths and limitations of the model.

2. REVIEW OF RELATED RESEARCH

Definitions of animation in user interfaces have ranged from specific to general. Vodislav [25] distinguished three categories of animation: (1) demonstration of program behavior, such as algorithm animation and data visualization, (2) feedback for users' actions to provide more realistic interaction, and (3) predefined animation, unaffected by either the user or the program. The latter category, the most conservative view, defines animation such as videos presented in help systems [e.g., 10, 21]. And to the three categories could be added a newer and often more annoying one: animation as a visual attractor [8], such as the elements of on-line advertisements that try to grab the viewer's attention—the very function that was the target of Nielsen's comment.

In this paper, we define animation broadly, with a view toward building as general a model as possible. As expressed by Thomas and Calder [22 (p. 199)], the "goal is to apply animation to the interface itself—to enhance or augment the effectiveness of human interaction with applications that present a graphical interface." Animation as an inherent part of the user interface, as opposed to a predefined animation, can reduce the user's cognitive load by enabling the user to follow and understand changes in the interface's appearance; that is, the interface uses animation to replace sudden changes with smooth transitions [5, 6]. These transitions can be spatial or qualitative. For example, developers of toolkits for interfaces with this sort of animation have provided motion, scaling, and change of color [13], and change of position and change of shape [9].

SIGDOC'11, October 3–5, 2011, Pisa, Italy.

Turning Function Calls Into Animations

Thibault Raffaillac
Inria, Lille, France
thibault.raffaillac@inria.fr

Stéphane Huot
Inria, Lille, France
stephane.huot@inria.fr

Stéphane Ducasse
Inria, Lille, France
stephane.ducasse@inria.f

TRACT
ated transitions are an integral part of modern interac-
frameworks. With the increasing number of animation
rios, they have grown in range of animatable features.
ot all transitions can be smoothed: programming sys-
limit the flexibility of frameworks for animating new
s, and force them to expose low-level details to program-
. We present an ongoing work to provide system-wide
ation of objects, by introducing a delay operator. This
tor turns setter function calls into animations. It offers
erent way to express animations across frameworks,
acilitates the animation of new properties.

CONCEPTS
**tware and its engineering → Graphical user in-
ce languages; • Computing methodologies → An-
on; • Human-centered computing → *User interface
amming*;**

WORDS
ation, programming constructs, expressing time, user
face design

Reference format:
ult Raffaillac, Stéphane Huot, and Stéphane Ducasse. 2017.
g Function Calls Into Animations. In *Proceedings of EICS '17,
, Portugal, June 26-29, 2017,* 6 pages.
://doi.org/10.1145/3102113.3102134

NTRODUCTION
he development of computers, animations have been

views on data [12, 19, 27]; or the animation of virtu
ters in video games by interpolation among motion
keyframes [6, 29]. Animations are considered usef
interfaces to help following changes [26], and in
tions to build a mental map of spatial information
can also convey meaning in data visualization [1
telling [18], as well as many other roles in user inte

Interaction frameworks have been evolving ove
support a greater range of uses, by developing mo
ways to animate elements in user interfaces. Wh
systems would animate a few properties, such as
or color, with different functions for each, moder
contain too many to scale this way. In particular, CS
animatable properties[3], and Core Animation has 29
with this increasing numbers of animatable proper
frameworks rely on naming strategies to refer to p
– like "position", "scale", "color" – instead of ha
specific function for each. This improves flexibility
ing animated properties at runtime, and reduces A
also gives an implicit contract that *any* property ca
or at least one which would make sense to the pro

This flexibility has a price though. The animati
tom properties and types requires frameworks to p
advanced API that exposes low-level details of the
tion systems, including timers and threads. This
larger animation APIs, and cumbersome syntaxes
the complex techniques often used to animate prop
names. It also creates a steep learning curve fro
advanced API, which is likely to force programme
to existing animatable properties whenever possib

Does Animation Help Users Build Mental Maps of Spatial Information?

Benjamin B. Bederson
Computer Science Department
bederson@cs.umd.edu
Human-Computer Interaction Lab
Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742

Angela Boltman
College of Education
aboltman@umiacs.umd.edu
Human-Computer Interaction Lab
Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742

Abstract

We examine how animating a viewpoint change in a spatial information system affects a user's ability to build a mental map of the information in the space. We found that animation improves users' ability to reconstruct the information space, with no penalty on task performance time. We believe that this study provides strong evidence for adding animated transitions in many applications with fixed spatial data where the user navigates around the data space.

1. Introduction

During the past decade, researchers have explored the use of animation in many aspects of user interfaces. In 1984, the Apple Macintosh used rudimentary animation when opening and closing icons. This kind of animation was used to provide a continuous transition from one state of the interface to another, and has become increasingly common, both in research and commercial user interfaces. Users commonly report that they prefer animation, and yet there has been very little research that attempts to understand how animation affects users' performance.

A commonly held belief is that animation helps users maintain object constancy and thus helps users to relate the two states of the system. This notion was described well by Robertson and his colleagues in their paper on "cone trees".

1.1. Animation takes time

One potential drawback of adding animation to an interface or visualization is that animation, by definition, takes time. This brings up a fundamental trade-off between the time spent animating and the time spent using the interface. At one extreme with no animation, system response can be instantaneous. Users spend all of their time using the system. However, the user may then spend some time after an abrupt transition adjusting to the new representation of information and relating it to the previous representation.

At the other extreme, each visual change in the interface is accompanied by a smooth transition that relates the old representation to the new one. While developers of animated systems hope that this animation makes it easier for users to relate the different states of the system, there is clearly a trade-off in how much time is actually spent on the transition. If the transition is too fast, users may not be able to make the connection, and if the transition is too long, the users' time will be wasted. The ideal animation time is likely to be dependent on a number of factors, including task type, and the user's experience with the interface and the data. In pilot studies and our experience building animated systems, we have found that animations of 0.5 – 1.0second seem to strike a balance. Others have found one second animations to be appropriate [9 p. 185].

In the worse case, animations can be thought of as an

Emerging Guidelines for Communicating with Animation in Mobile User Interfaces

Daniel Liddle
Purdue University
Department of English
500 Oval Drive, Lafayette IN
(765) 494-3740
dliddle@purdue.edu

ABSTRACT
Animations are a widely used design element in mobile user interfaces; however, it is unclear how these uses of animation support or challenge prior research. Therefore, I examine the current animation guidelines published by Microsoft, Apple, and Google, and then compare these guidelines to seven design principles proposed by previous studies (solidity, exaggeration, reinforcement, attachment, reluctance, smoothness, anticipation). I then examine the animation types (e.g., rotation and fading) used by a specific mobile application. By comparing current practices to previous research, I argue that decades-old design principles can be used to understand and clarify the frequent injunction to design animation that is "realistic" and/or "lively." I also call attention to emerging terms and concepts that are not covered by prior research, providing possible threads for future study.

CCS Concepts
• **Human-centered computing ~ User interface design** • Human-centered computing ~ Graphical user interfaces • Human-centered computing ~ Mobile phones • Computing methodologies ~ Animation • Software and its engineering ~ Design patterns

Keywords
Animation; motion design; user experience design; mobile design; direct manipulation; motion graphics

1. INTRODUCTION

Just as there are good and bad uses of color, so there are good and bad uses of animation.

-Thomas and Caulder [1, p. 220]

Not long ago, animated graphics were commonly understood to be one of the top mistakes in web design. As recently as 2005, user experience specialist Jakob Neilsen argued that designers should

the interface, and add a sense of delight to the user experience [3]. Animated design elements are also increasingly employed in the design of mobile user interfaces, where they are used to provide feedback for gesture-based operations and signal the spatial organization of the interface [4].

Despite these areas of development, much of the research on animation and usability has focused solely on the design of animation within particular desktop applications [5,1,4]. Given the increasing use of animation in mobile interfaces, it is necessary to connect prior research with contemporary practices and the specific constraints of mobile UI in order to develop an updated set of best practices.

In this paper I work toward these updated practices by reviewing the current guidelines published by Microsoft, Apple, and Google, examining the way their guidelines offer differing sets of general principles and specific practices. I then compare the general principles to those proposed by Chang and Ungar [6] and Thomas and Caulder [1], assessing how the traditional guidelines for animation are re-contextualized and challenged by contemporary practices. Finally, I compare the animation types and communicative functions discussed by Novik et al. [5] with the use of animation in the Tumblr application developed for the Android platform. Through this process I offer a tentative outline of the animation types and communicative functions that may require renewed attention in light of the growing attention to mobile interface design. I conclude with possible directions for future research.

2. REVIEW OF RELATED RESEARCH

Defining animation is no easy task. Some studies refer to animation as purely a phenomena related to spatial movement [1] while others use the term to refer to refer to any qualitative changes in an interface, such as changes in color or opacity [7]. Additionally, there are some studies that refer to all animations as

Transition Animations Support Orientation in Mobile Interfaces Without Increased User Effort

Jonas F. Kraft
Chair of Psychological Ergonomics
Julius-Maximilians-Universität
97074 Würzburg, Germany
kraftuni@gmail.com

Jörn Hurtienne
Chair of Psychological Ergonomics
Julius-Maximilians-Universität
97074 Würzburg, Germany
joern.hurtienne@uni-wuerzburg.de

ABSTRACT
Some animations in mobile user interfaces aim at supporting user orientation by facilitating users to build a mental model of the UI's structure. Possible drawbacks are that animations are time-consuming and that complex and distracting animations may increase users' mental workload. These effects of orientation animations are investigated in an empirical study. Participants either used an animated or a non-animated version of a mobile movie recommender app. The results imply that animations can support users in building more accurate mental models of the app's structure and enhance gesture-based interaction. No additional costs in terms of time or mental workload were incurred when using the animations. Thus, lightweight orientation animations can have large potential benefits at little cost.

Author Keywords
Animation; User Experience; Usability; Orientation; Mental Model; Mental Workload

ACM Classification Keywords
H.5.1 Multimedia Information Systems: [animations]

INTRODUCTION
Animations in mobile user interfaces are used in several ways [20]: animated microinteractions give the user an illusion of a naturally behaving object (e.g. a button looks like it has been pressed); animations show the state of a process to the user (e.g. loading bars); animations can be used decoratively to make the interface more interesting and catch the user's attention; and clarifying animations,

In HCI, animations have been defined as *a series of varying images presented dynamically according to user actions in ways that help the user to perceive a continuous change over time and develop a more appropriate mental model of the task* [10, based on 2, 14]. Some early work on animations in HCI was done in the 1990s. Some academics investigated the effects of animation on decision making and in information visualisation [9, 18]. Others examined effects of animations on the ease of use of a system and other factors of the user experience [2, 17]. The most comprehensive taxonomy of the roles of animations in user interfaces was recently provided by Chevalier et al. [6]. According to them animations can

- Keep users in context – by orienting users during navigation, supporting tracking during layout changes, keeping users up to date, summarising and replaying history.
- Aid teaching – by pointing out affordances, giving previews, explaining how a system works, giving examples, illustrating an algorithm and teach new representations of information.
- Enhance the user experience – by engaging users, providing visual comfort and aesthetics, visualising activities and progress, revealing and hiding content, providing virtual tours and feedback on input mechanisms.
- Aid data encoding – by revealing data relationships, conveying uncertainty or randomness and encoding object attributes and emotions.

Animation

Animation

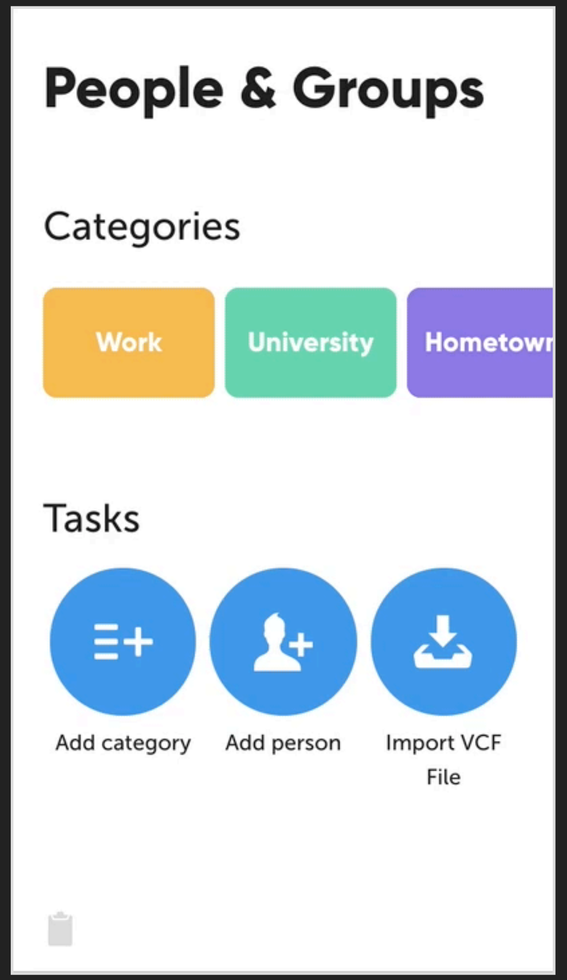
Touch
Interaction

Animation

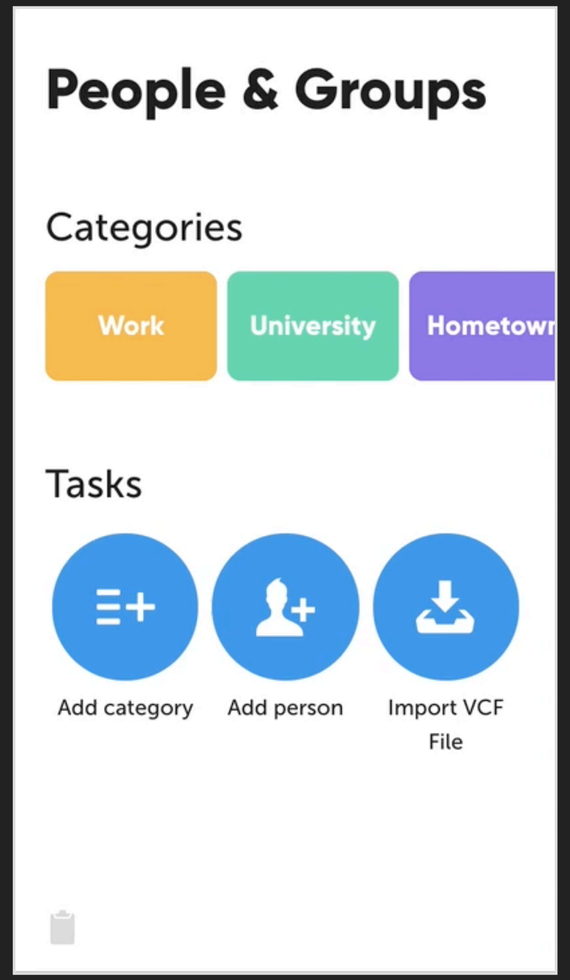
Touch
Interaction

Mobile Context
& Usability


Travelling
Hanna
diving instructor
Kate
from South Africa
Matt
loves vintage clothing
Tom
met in Vietnam
ADD PERSON
Name (required)
NEXT
Work
Astrid
plays rugby in her free time
Elias
always grumpy, but likes board games
Heidi
ADD CATEGORY
Category Name
Category Color
DONE
People & Group
Categories
Work
University
Tasks
Add category
Add person



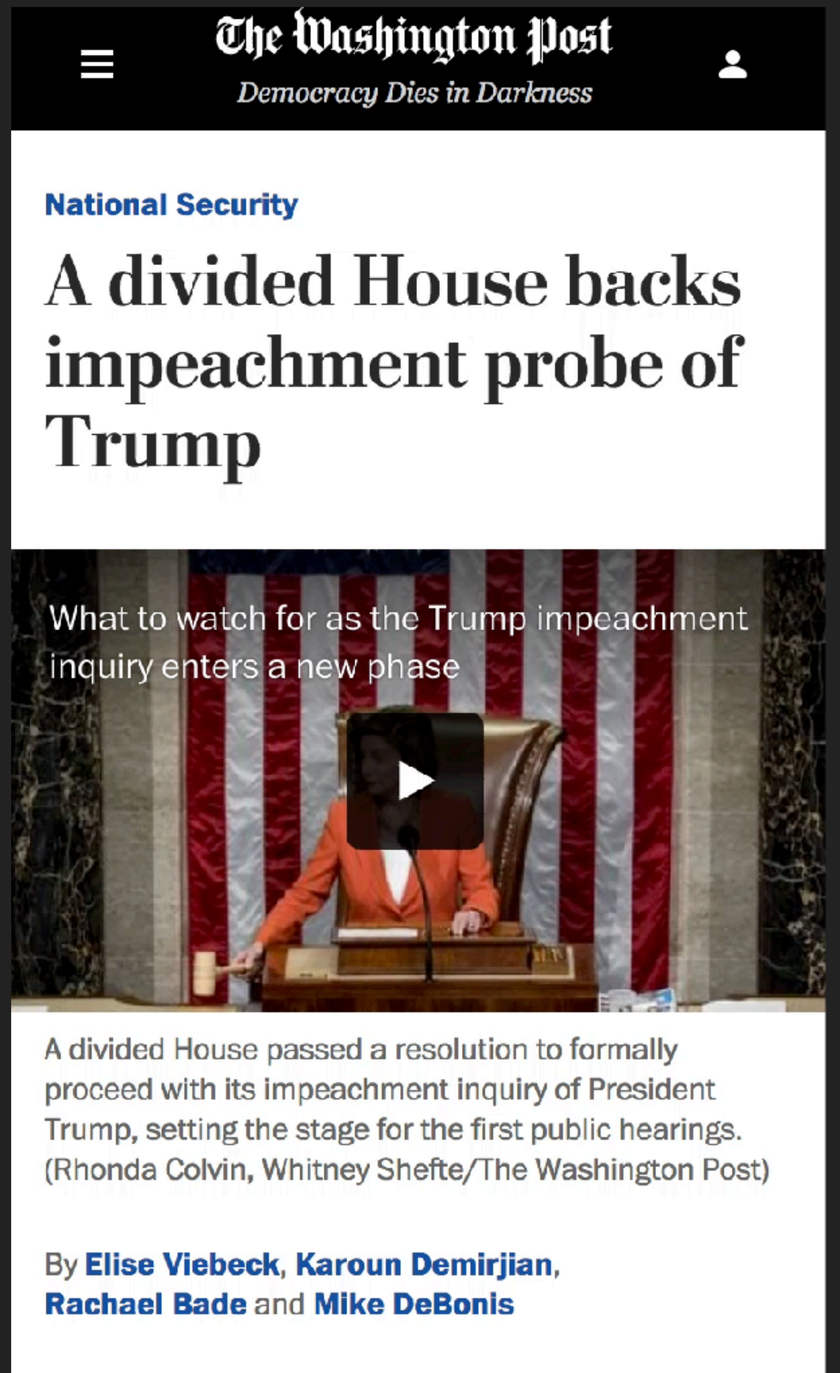
bit.ly/pwa-static



bit.ly/pwa-animated

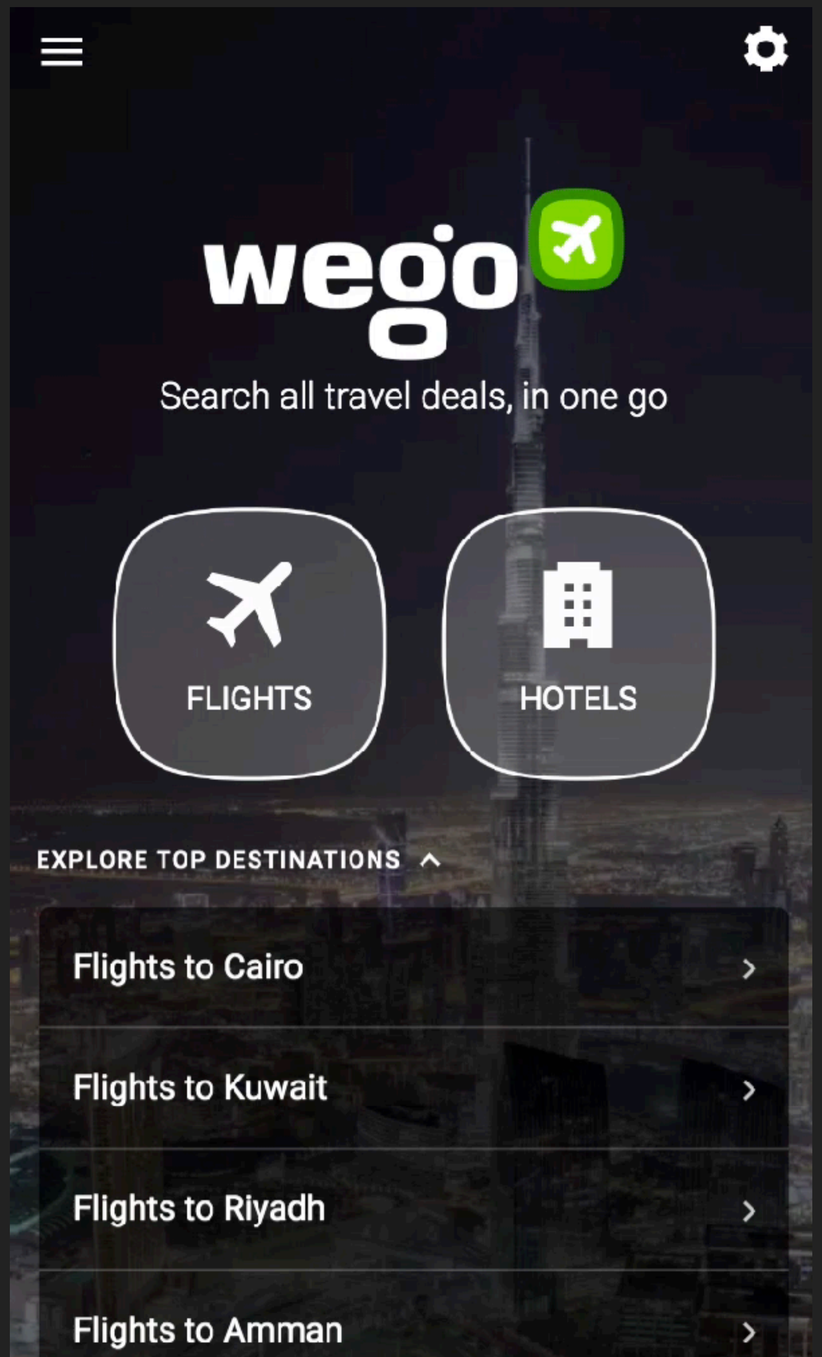
What are the problems with the mobile web?

not reactive
to touch
interaction



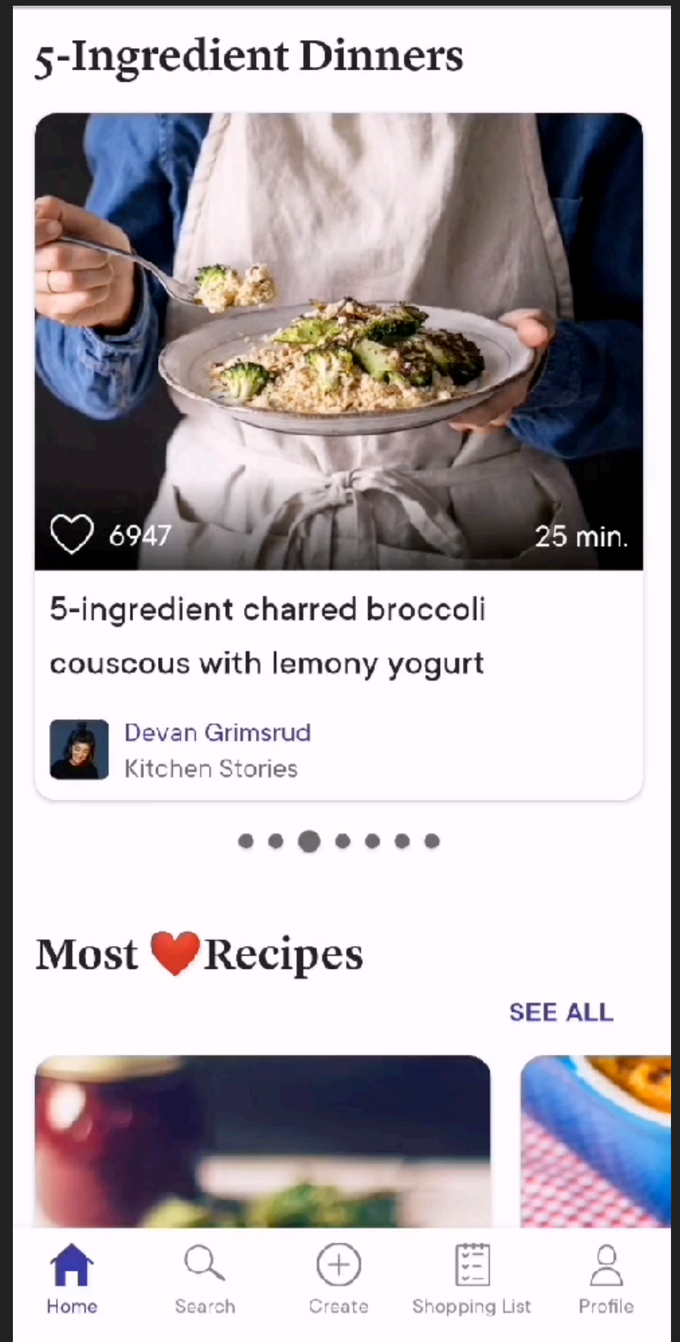
👇

reactive to
touch
interaction

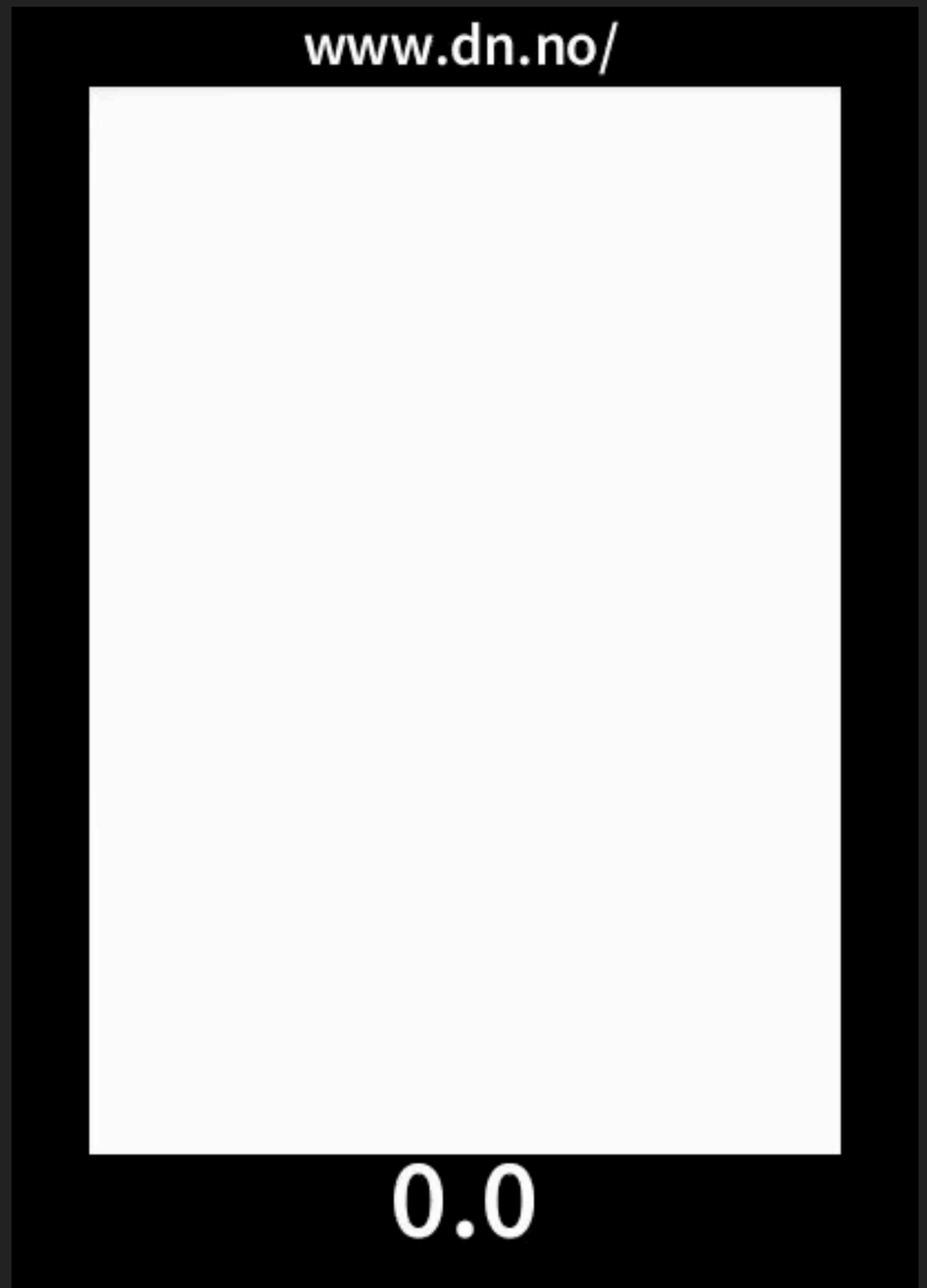


user loses
orientation

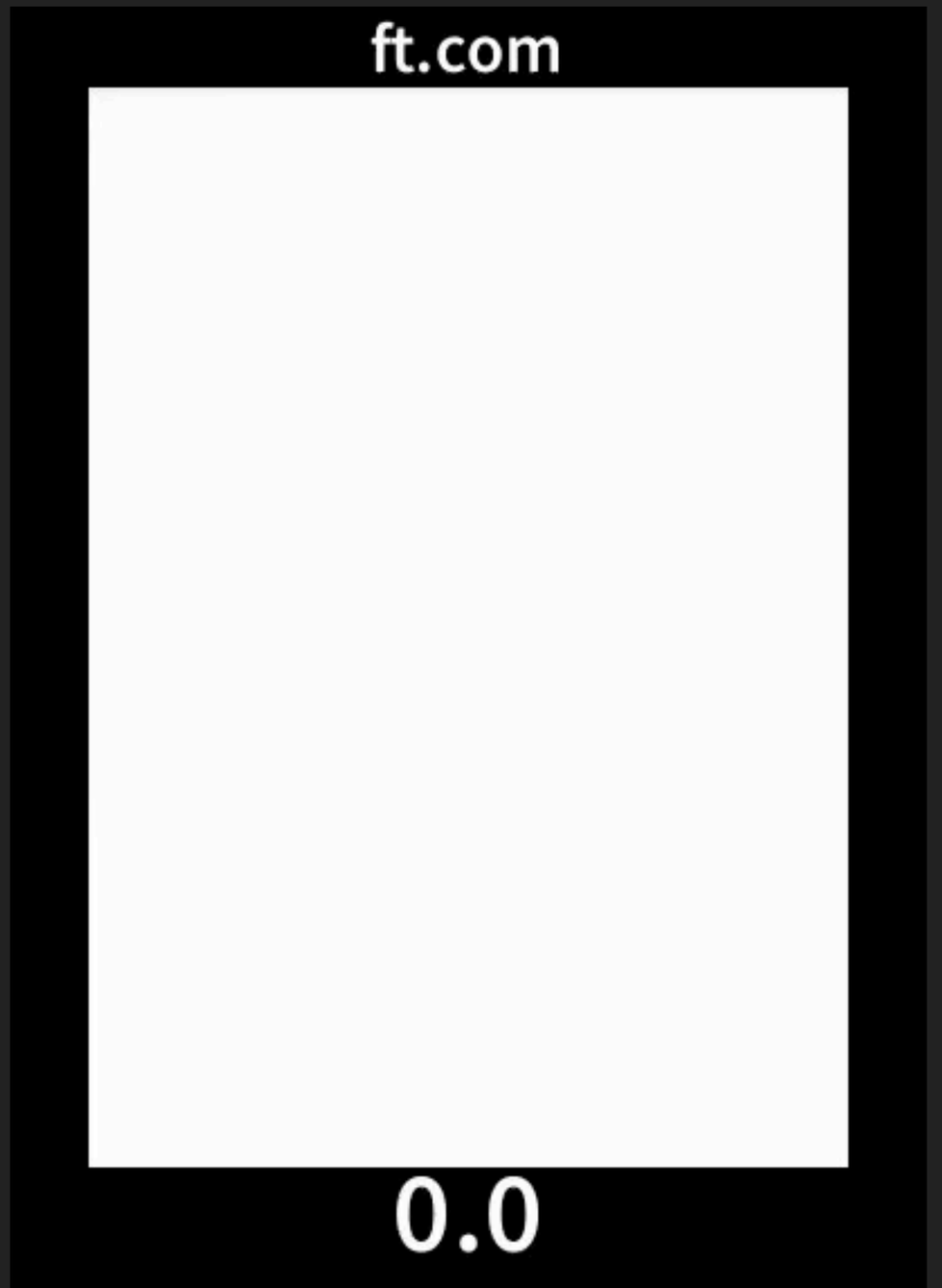
users understand why content changed and can reorient themselves



slow
performance
or network
connection



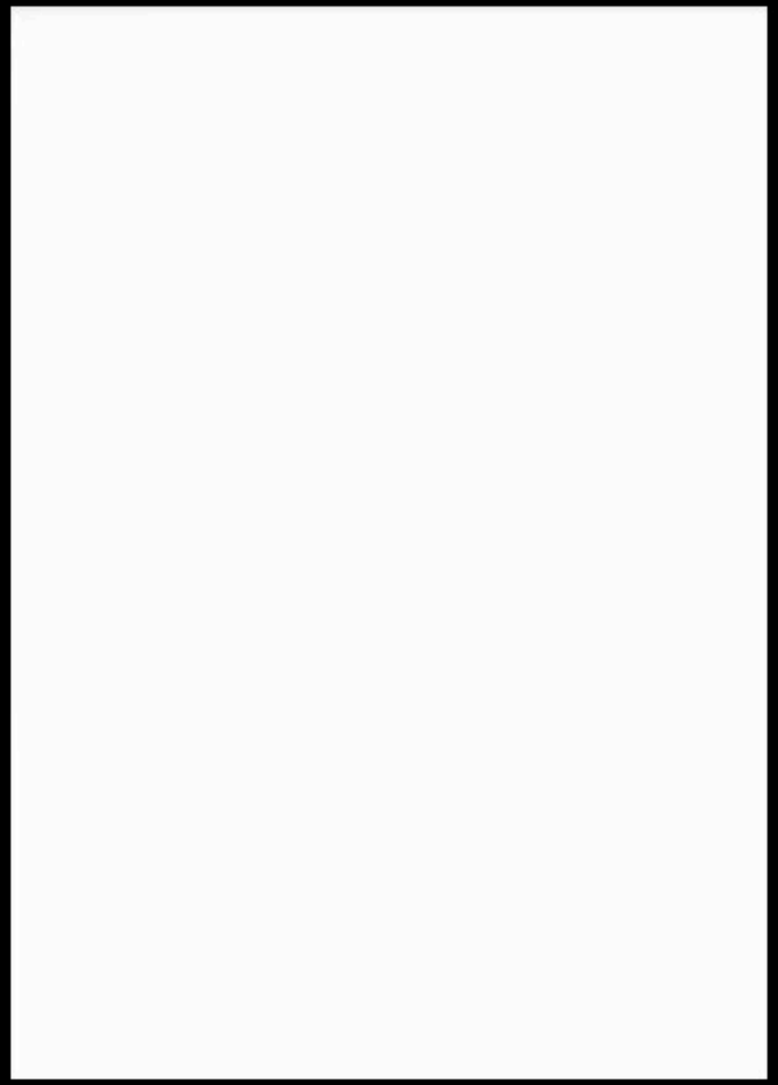
loads most important parts quickly



www.dn.no/

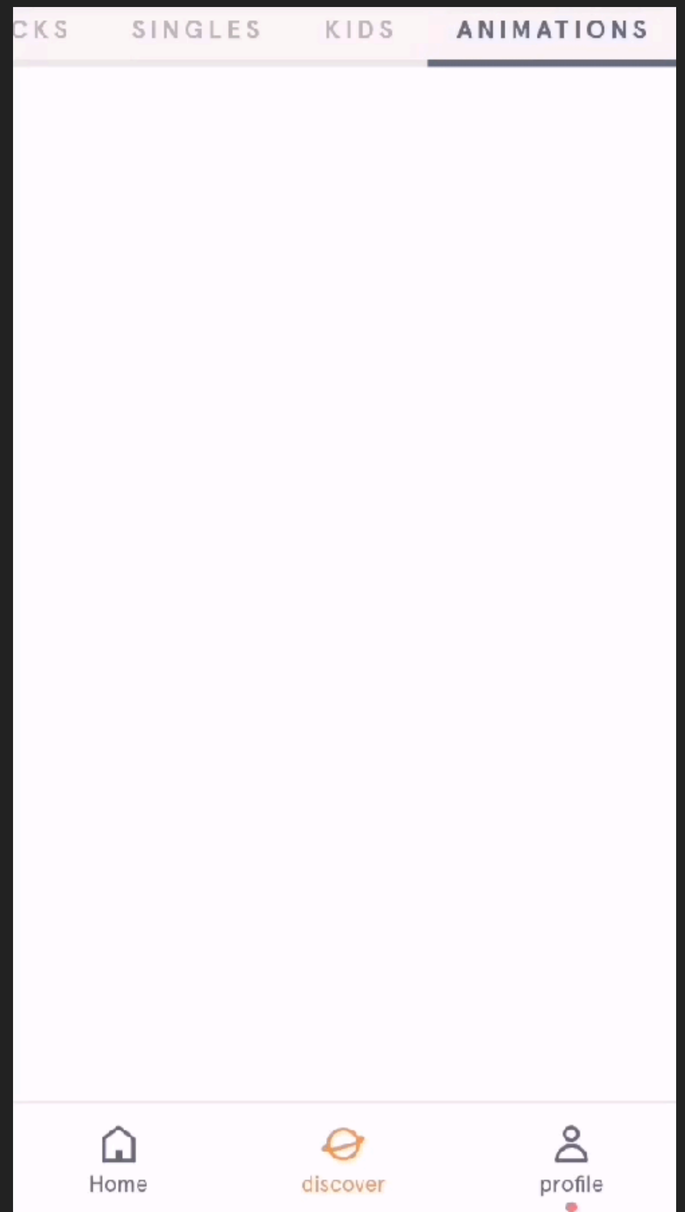
0.0

ft.com



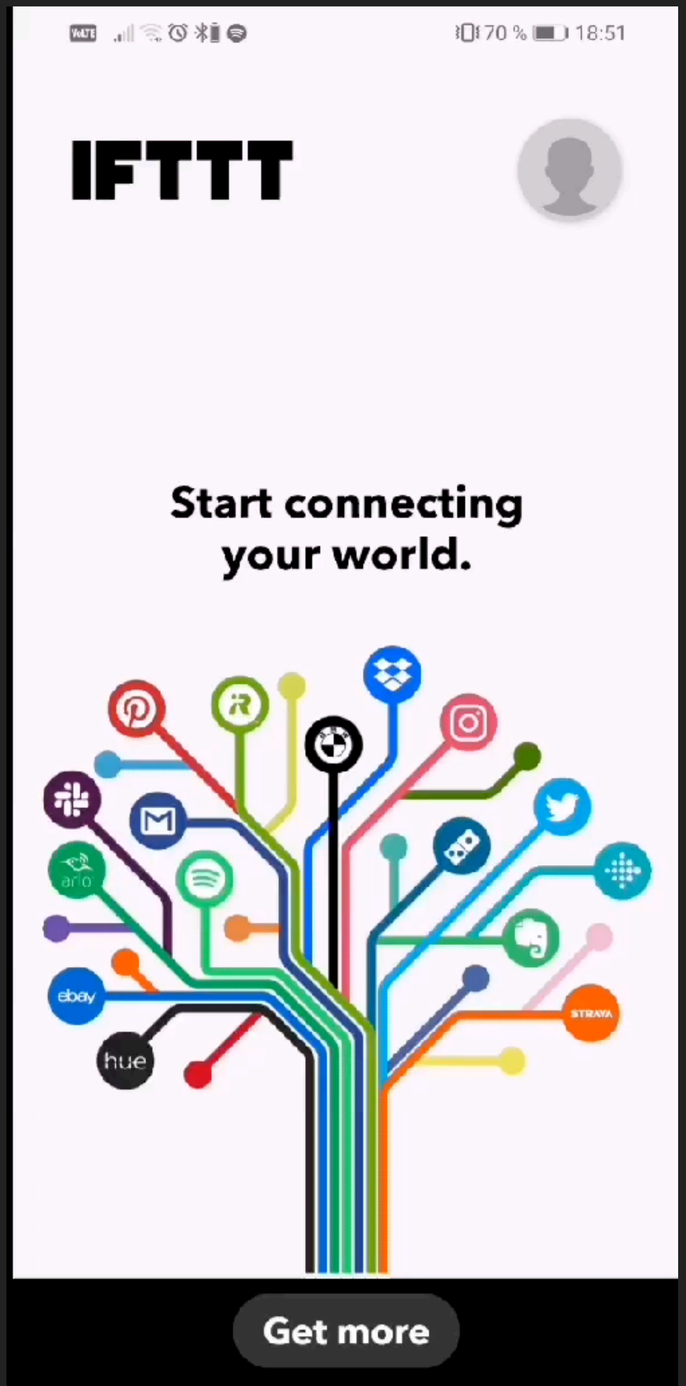
0.0

user doesn’t
feel in control



🎮

application is behaving in a predictable way



👇 touch

 navigation

⏰ wait times

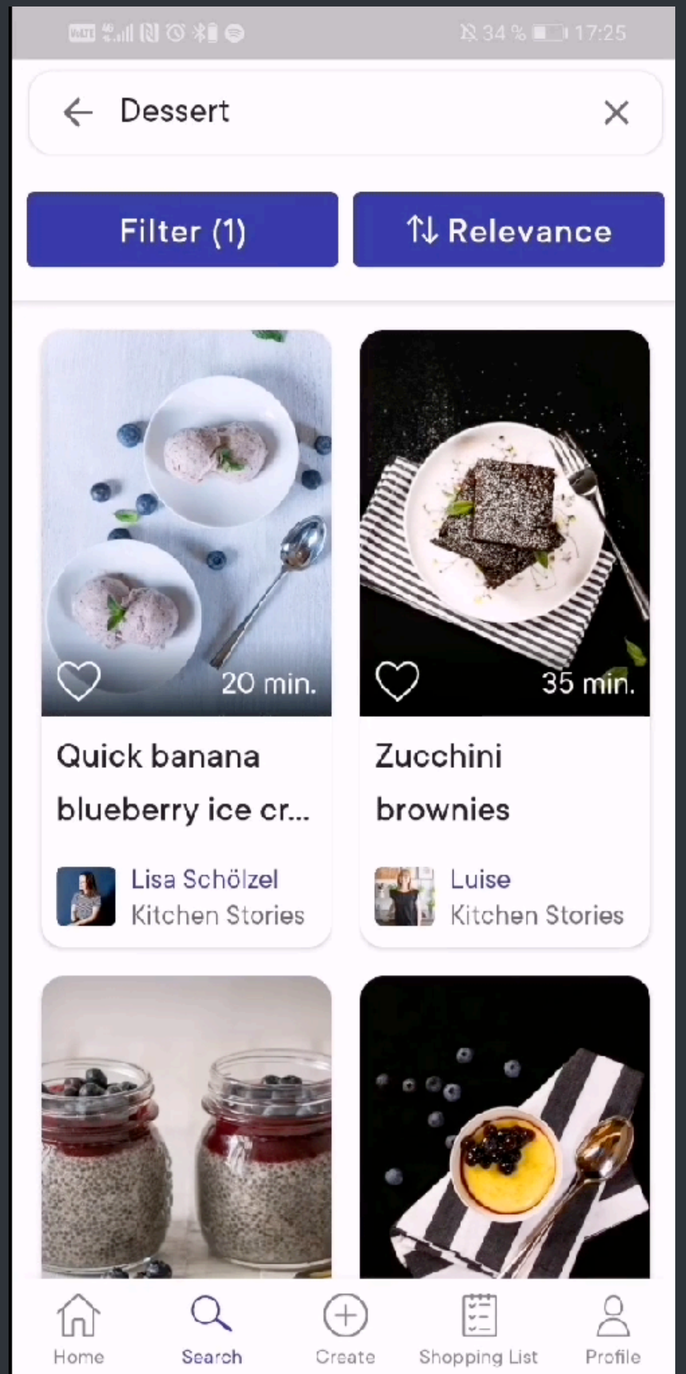
🎮 control

What do native applications do differently?

- onboarding animation
- animated transitions
- fading in new content gradually

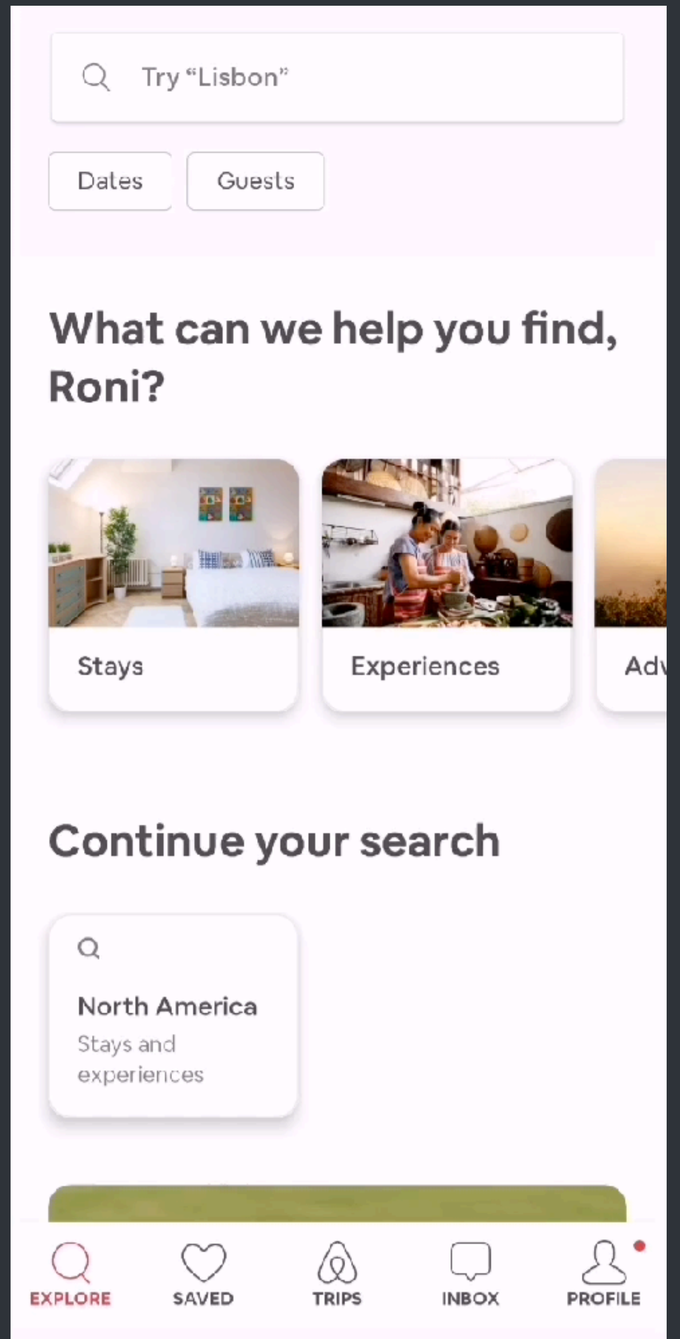
🎮 👇 🗺️

- navigation animation
- indicating change in content



🎮 👇 ⏰

- spinners / loaders
- hiding content in swappable areas to the side
- fading in new content
- navigation animation



- loaders
- showing progress
- animated transitions
- fading in new content gradually

FABULOUS

Animation in PWAs

NAVIGATION & CONTEXT

“When animations were present, more participants had the impression of interacting with a continuous space rather than with separate screens.”

Jonas F. Kraft, Jörn Hurtienne

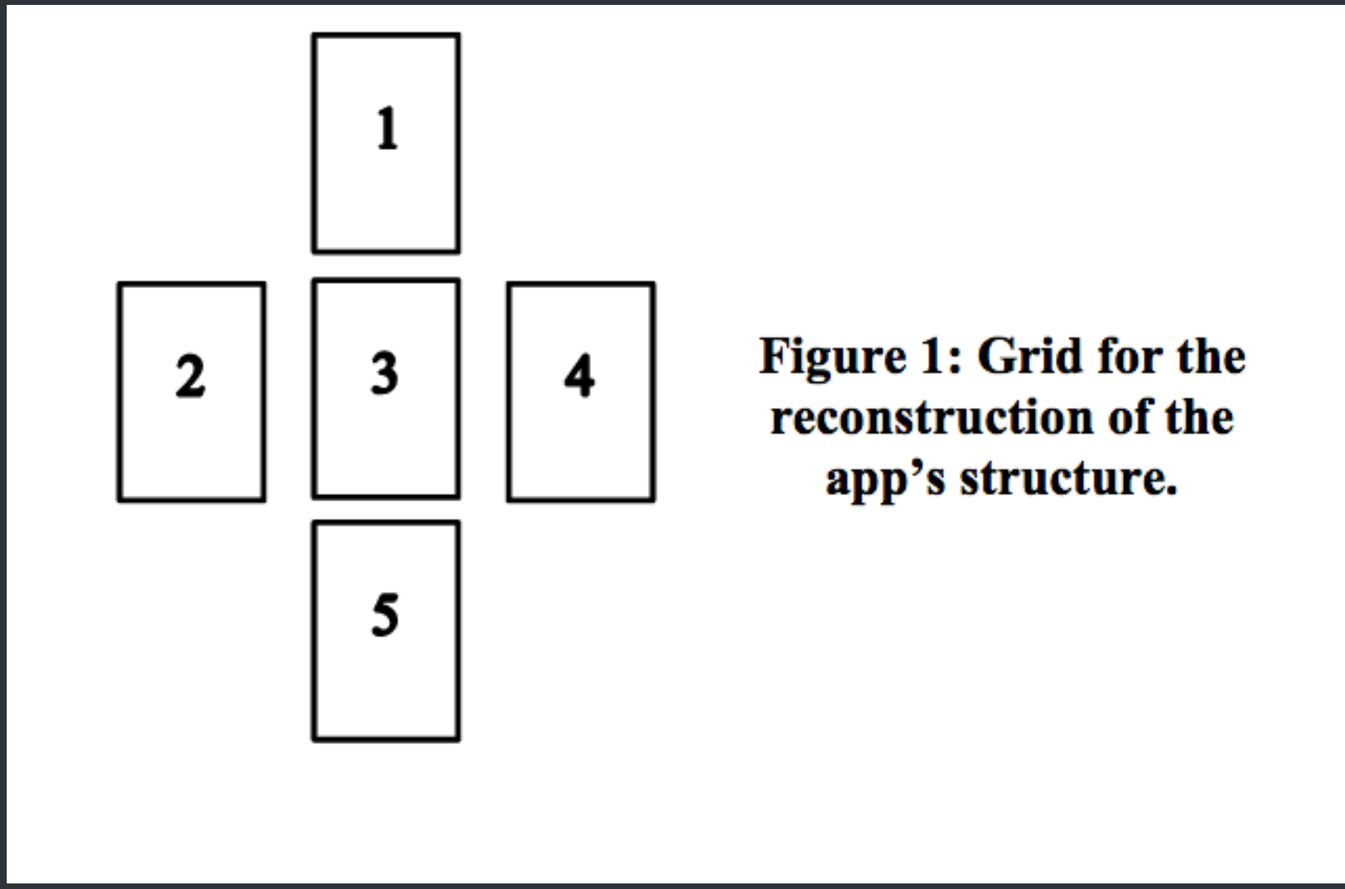
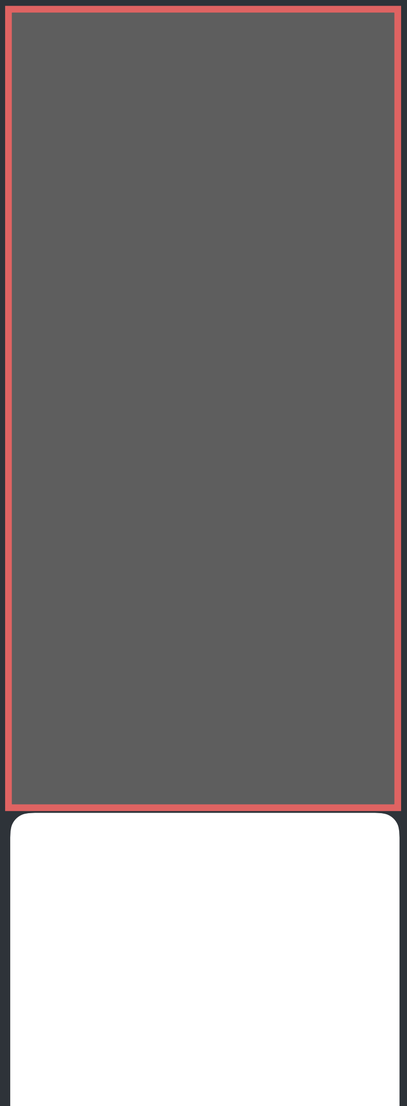


Figure 1: Grid for the reconstruction of the app’s structure.

Transition animations support orientation in mobile interfaces without increased user effort

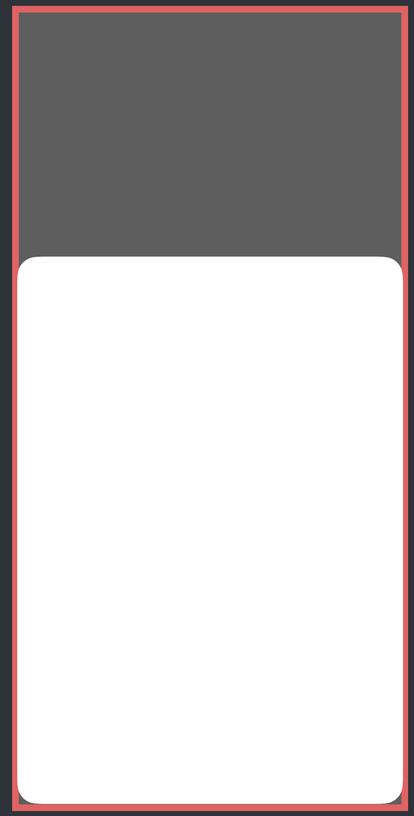

23:12
100%
Lorem Ipsum Dolor Sit Amet Consectetur
by Jaison Justus, 16 Jun 2016
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Lorem Ipsum Dolor Sit Amet Consectetur
by Jaison Justus, 16 Jun 2016
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in

```css
.bottom-nav {
  position: fixed;
  width: 100%;
  top: 100%;
  left: 0;

  transition: transform 0.2s
    cubic-bezier(0.39, 0.575, 0.565, 1);
}
```

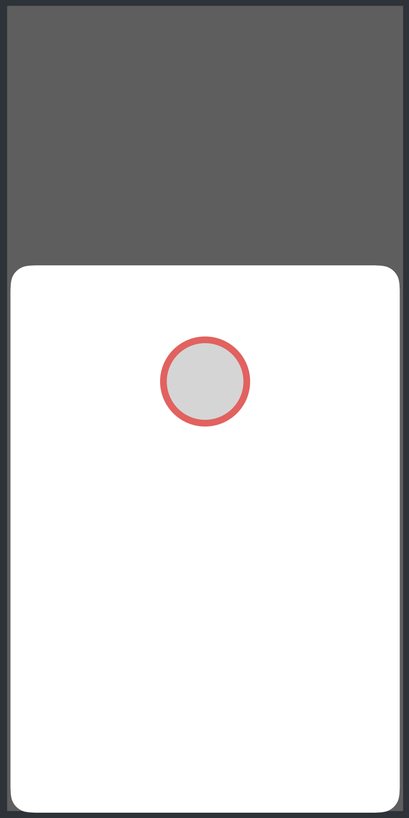


```css
.bottom-nav {
  position: fixed;
  width: 100%;
  top: 100%;
  left: 0;

  transition: transform 0.2s
    cubic-bezier(0.39, 0.575, 0.565, 1);
}
```

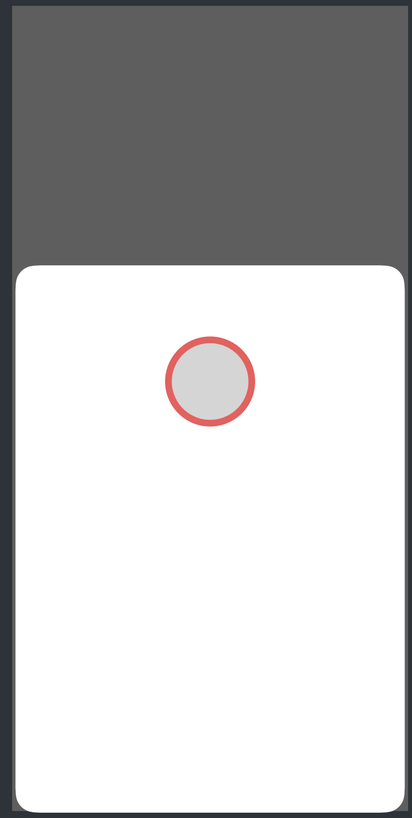
```css
.bottom-nav.visible {
  transform: translateY(-100%);
}
```



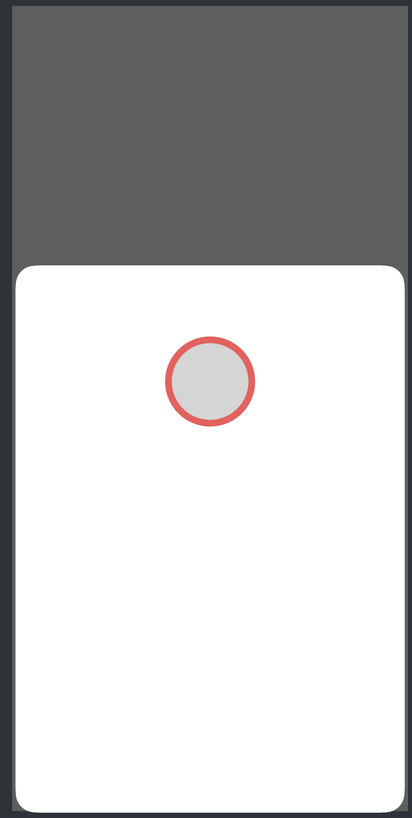
```
bottomNav.addEventListener('touchstart', handleGestureStart, true);
bottomNav.addEventListener('touchmove', handleGestureMove, true);
bottomNav.addEventListener('touchend', handleGestureEnd, true);
bottomNav.addEventListener('touchcancel', handleGestureEnd, true);
```



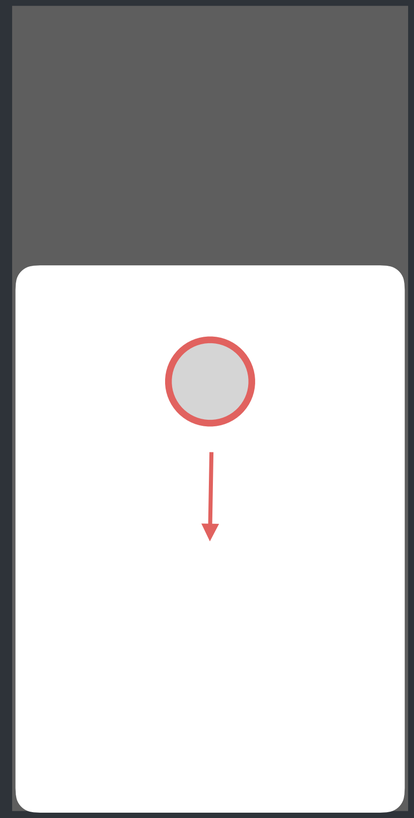
```
handleGestureStart = (evt) ⇒ {
  evt.preventDefault();

  if(evt.touches && evt.touches.length > 1) {
    return;
  }

  initialTouchPos = getGesturePointFromEvent(evt);

  navBar.style.transition = 'initial';
};
```



```
function getGesturePointFromEvent(evt) {
    let point = {};

    if(evt.targetTouches) {
      // Prefer Touch Events
      point.x = evt.targetTouches[0].clientX;
      point.y = evt.targetTouches[0].clientY;
    } else {
      // Either Mouse event or Pointer Event
      point.x = evt.clientX;
      point.y = evt.clientY;
    }

    return point;
  }
```

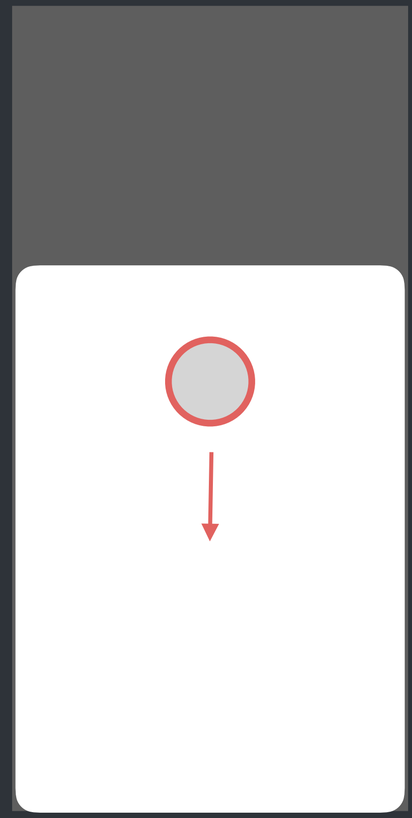


```
handleGestureMove = (evt) ⇒ {
  evt.preventDefault();

  if(!initialTouchPos) {
    return;
  }

  lastTouch = getGesturePointFromEvent(evt);

  if(rafPending) {
    return;
  }

  rafPending = true;

  window.requestAnimFrame(() ⇒ update(lastTouch.x, lastTouch.y));
};
```

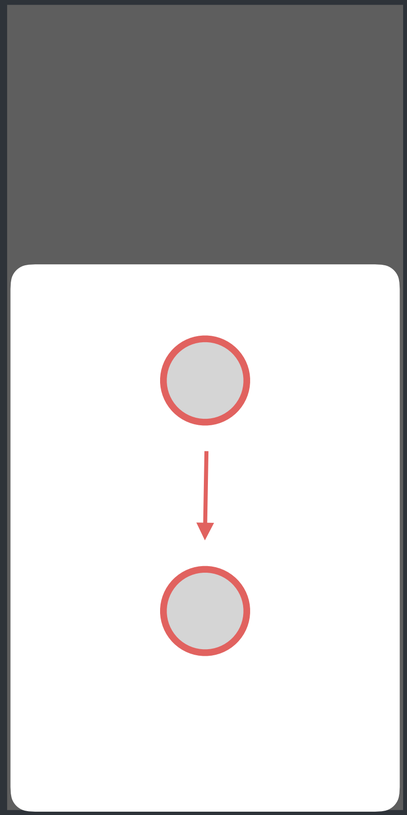


```
function update() {
  if(!rafPending) {
    return;
  }

  let differenceInY = initialTouchPos.y - lastTouchPos.y;
  let newYTransform = currentYPosition - differenceInY;
  navBar.setProperty('--y', newYTransform);

  rafPending = false;
}
```

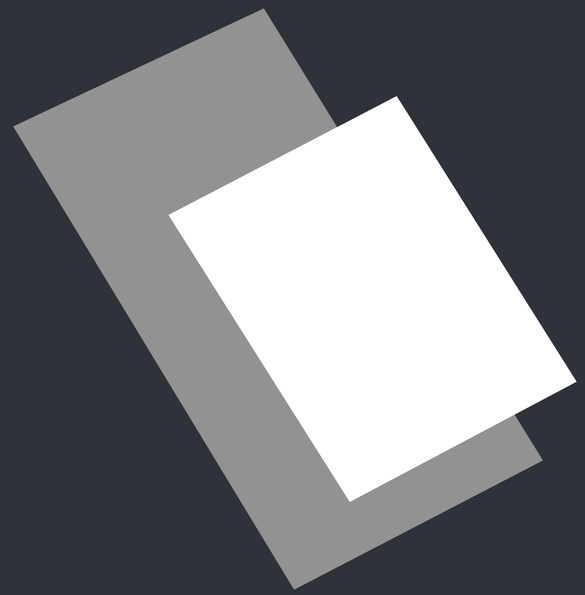
```
.bottom-nav {
    transform: translateY(var(--y));
}
```



```
// Handle end gestures
handleGestureEnd = (evt) => {
  evt.preventDefault();

  if(evt.touches && evt.touches.length > 0) {
    return;
  }

  rafPending = false;
  navBar.style.transition = 'transform 0.3s ease-in-out';
}
```



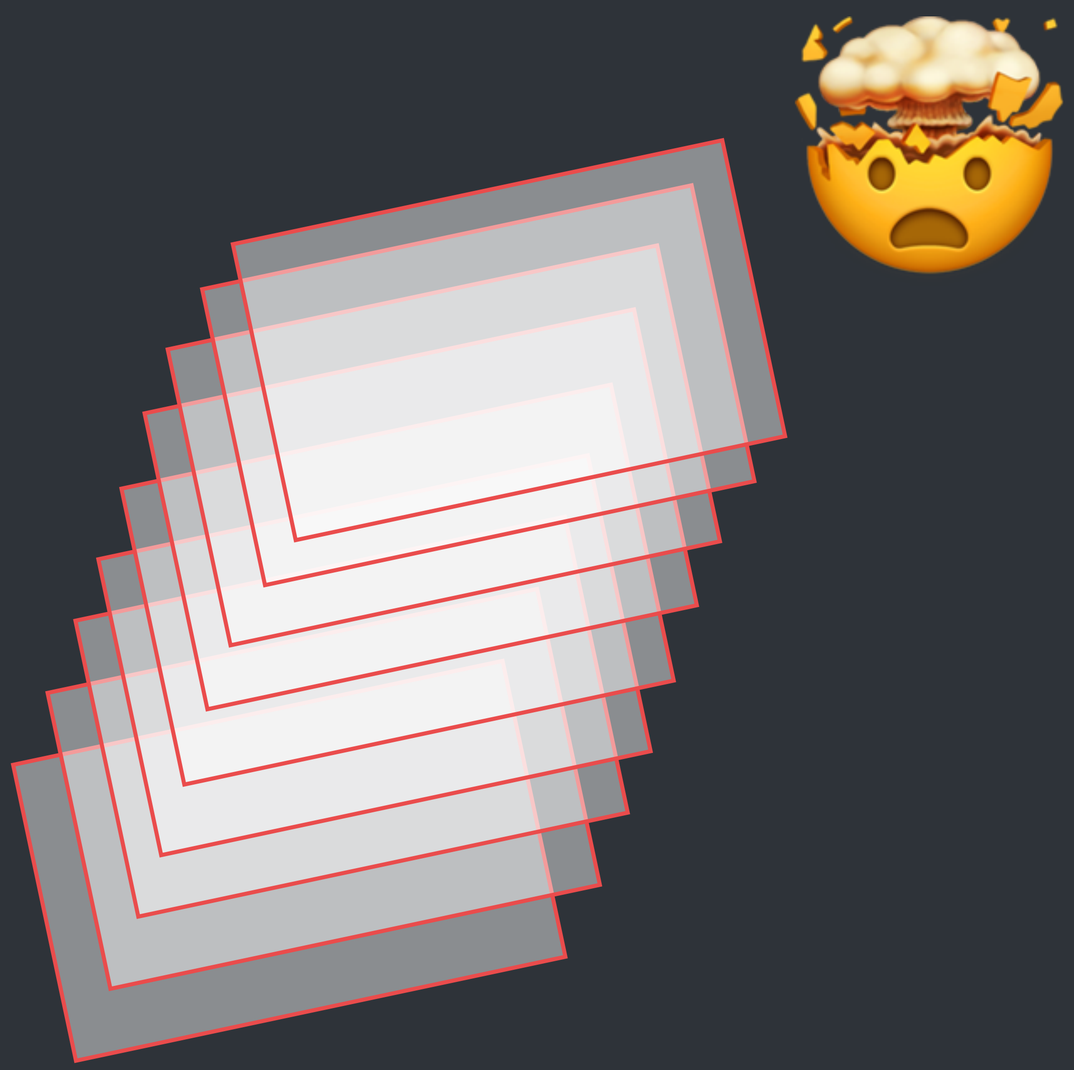
```css
.bottom-nav {
    will-change: transform;
}
```



Creating new layers

```
.foo {
 will-change: transform;
 transform: translateZ(0);
 backface-visibility: hidden;
}
```

1. will-change property
2. 3D Transform
3. animated 2D transforms
4. being on top of a compositing layers
5. animated CSS filters



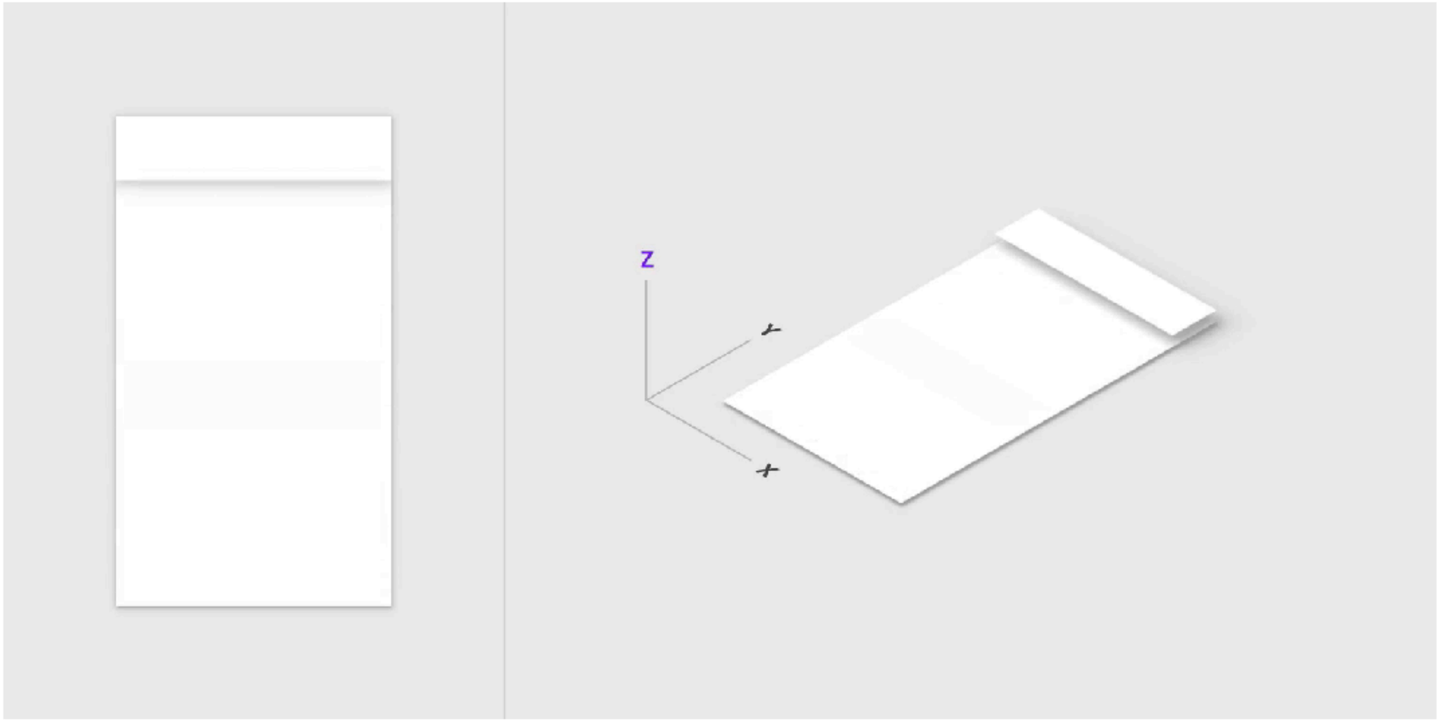
don’t create too many layers since they consume memory

remove them if the animation is finished


23:12
100%
Lorem Ipsum Dolor Sit Amet Consectetur
by Jaison Justus, 16 Jun 2016
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Lorem Ipsum Dolor Sit Amet Consectetur
by Jaison Justus, 16 Jun 2016
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in

Parent-child transitions

From a parent screen, an embedded child element lifts up on touch and expands in place, using **Standard easing**. The motion both draws focus to the child screen (which is the destination of the interaction), while reinforcing the relationship between parent and child screens.

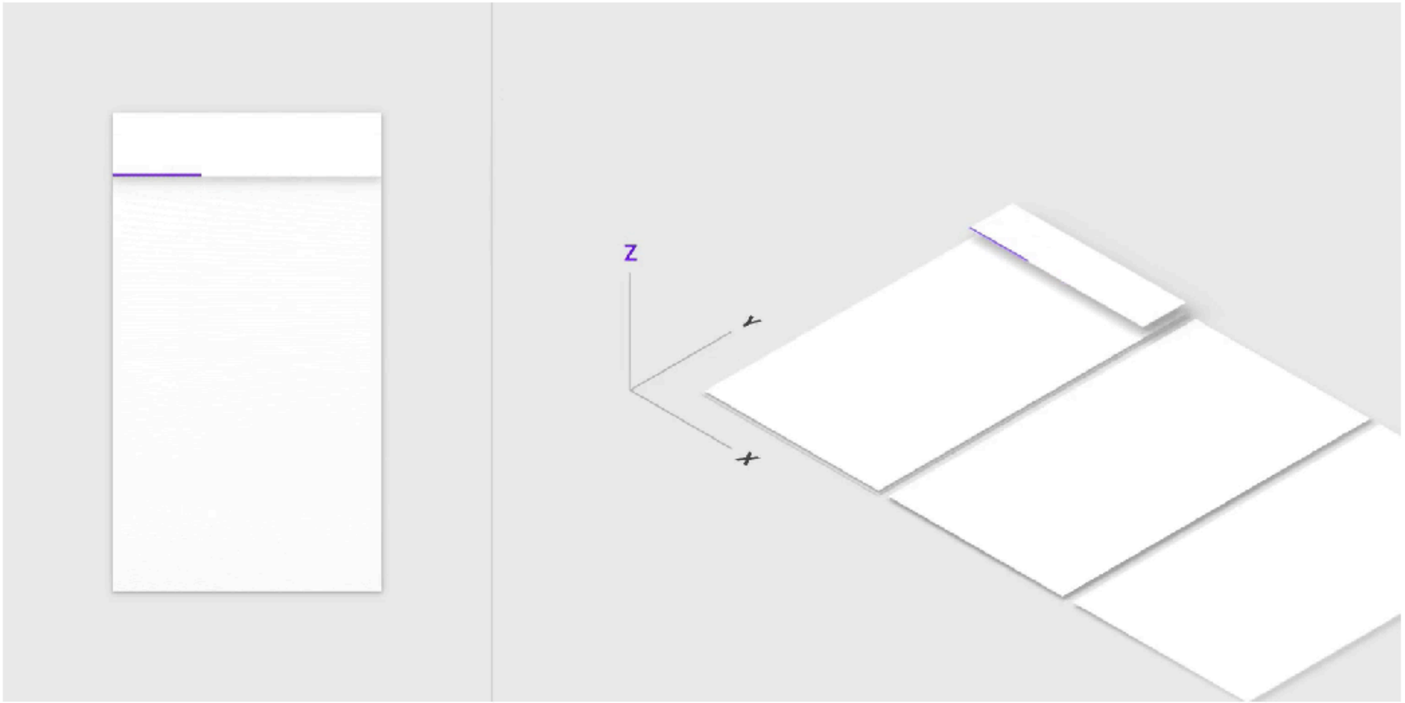


A parent-to-child transition

material.io/design/navigation/navigation-transitions.html

Sibling transitions

Screens that share the same parent (such as photos in an album, sections of a profile, or steps in a flow) move in unison to reinforce their relationship to one another. The peer screen slides in from the one side, while its sibling moves off screen in the opposite direction.

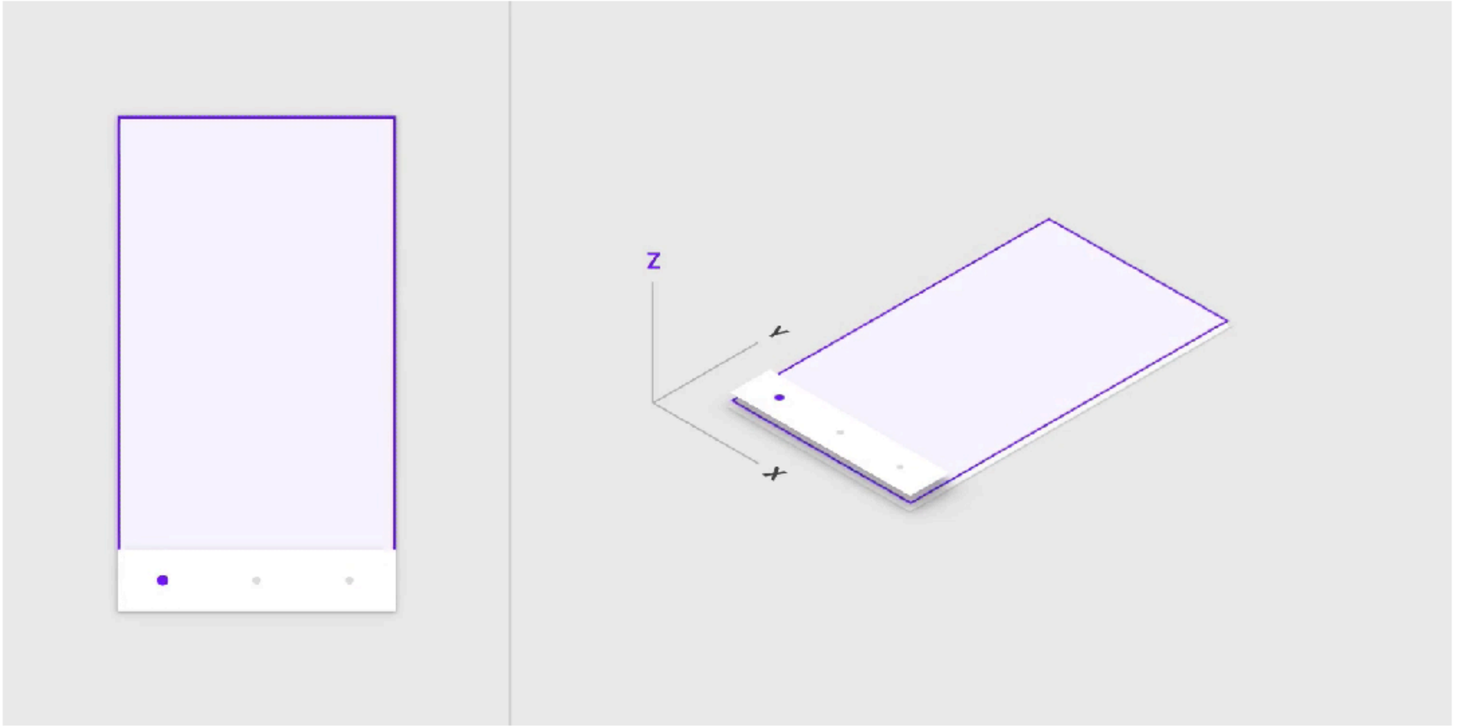


Tabs indicate they are related by sitting at the same elevation and moving together.

material.io/design/navigation/navigation-transitions.html

Top-level transitions

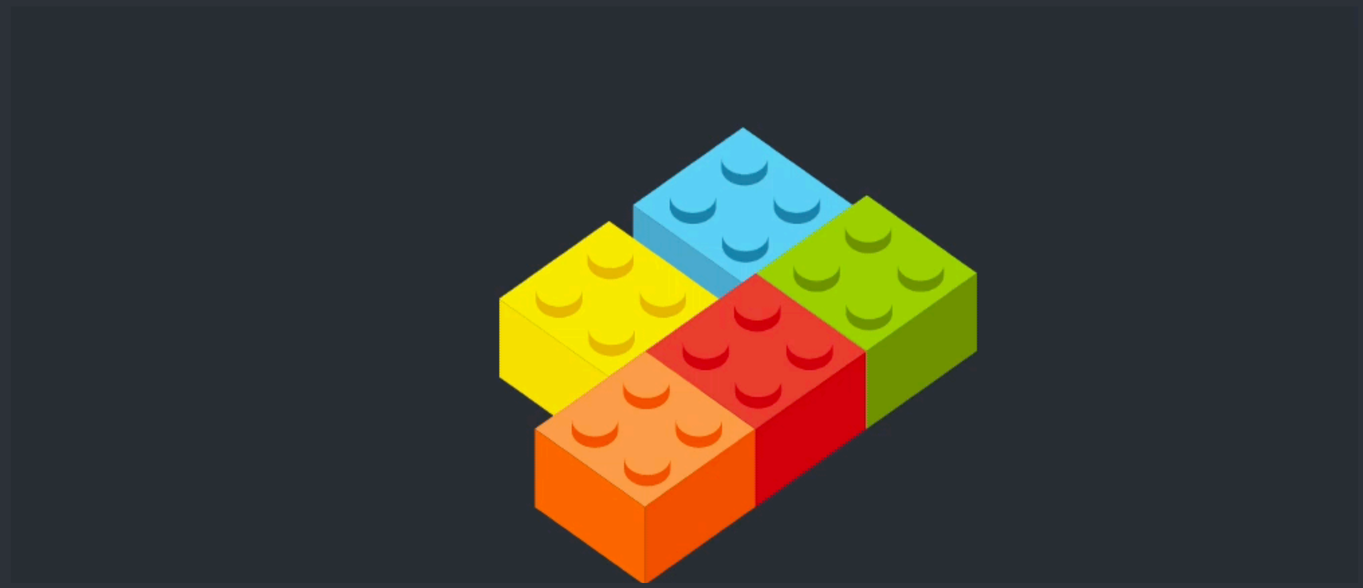
At the top level of an app, destinations are often grouped into major tasks (and the tasks may not relate to one another). These screens transition in place by changing values such as opacity and scale.



Destinations in a bottom navigation bar transition in place.

material.io/design/navigation/navigation-transitions.html

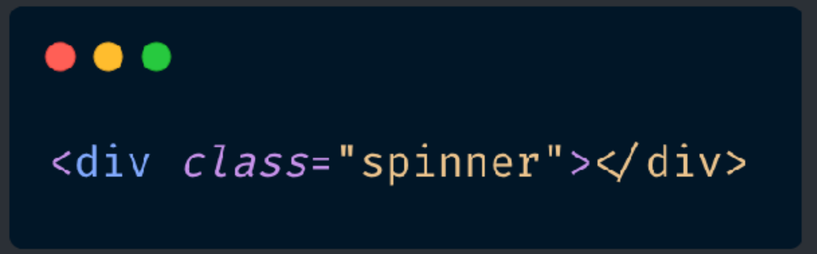
ACTIVITY & FEEDBACK



Emphasize branding

animation:
codepen.io/chrisgannon/pen/yXmbMg

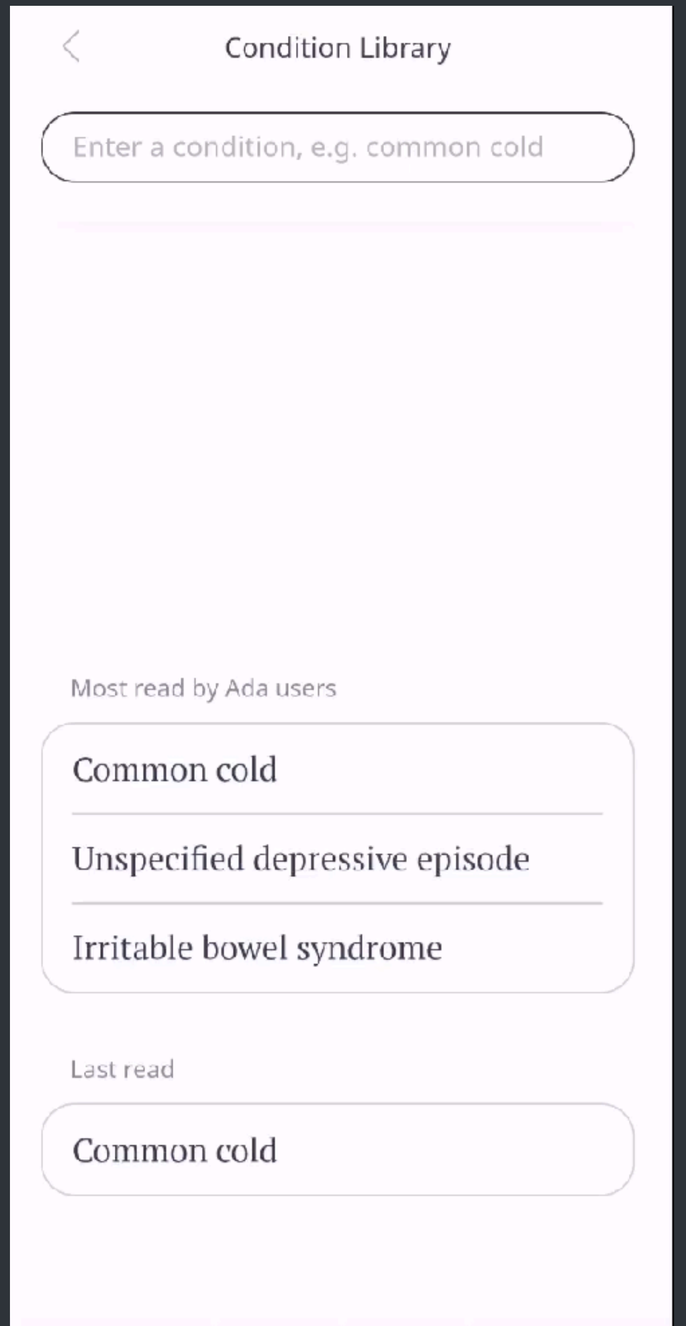

00:01



```css
.spinner {
  width: 40px;
  height: 40px;
  margin: 100px auto;
  background-color: #333;
  border-radius: 100%;
  animation: scaleout 1.0s infinite ease-in-out;
}

@keyframes scaleout {
  from {
    transform: scale(0);
  }
  to {
    transform: scale(1.0);
    opacity: 0;
  }
}
```

```html
<div class="spinner"></div>
```

DISPLAY CLUTTER


Condition Library
Enter a condition, e.g. common cold
Most read by Ada users
Common cold
Unspecified depressive episode
Irritable bowel syndrome
Last read
Common cold

Display clutter can be defined as the decrement of performance and cost of attention, particularly, visual search decrements, that result from the interaction between display-based and user-based factors.

Nadine Moacdieh, Nadine Sarter

Display Clutter: A Review of Definitions and Measurement Techniques

Display clutter can be defined as the decrement of performance and cost of attention, particularly, visual search decrements, that result from the interaction between display-based and user-based factors.

Nadine Moacdieh, Nadine Sarter

Display Clutter: A Review of Definitions and Measurement Techniques

Los Angeles Times LOG IN

NOV. 2, 2019 56°F

Illegal drones ground water-dropping helicopters at critical moment in Maria fire battle

More Coverage

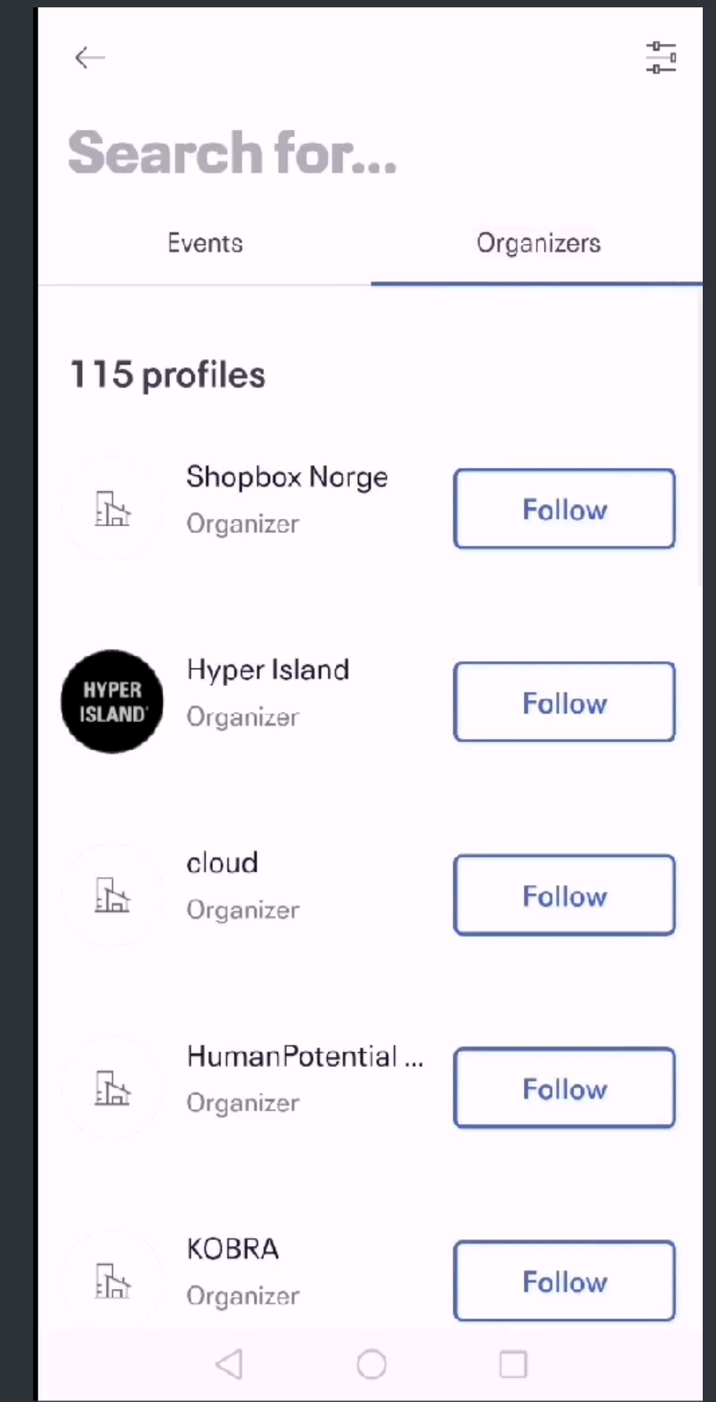
Watch the moment the Maria fire in Ventura County ignited

Firefighters finally getting a handle on wildfires burning across California

Whole menus are reduced to single icons, and additional pages and features are often located in swipe-able areas to the left or right of the visible screen.

Daniel Liddle

Emerging Guidelines for Communicating with Animation in Mobile User Interfaces



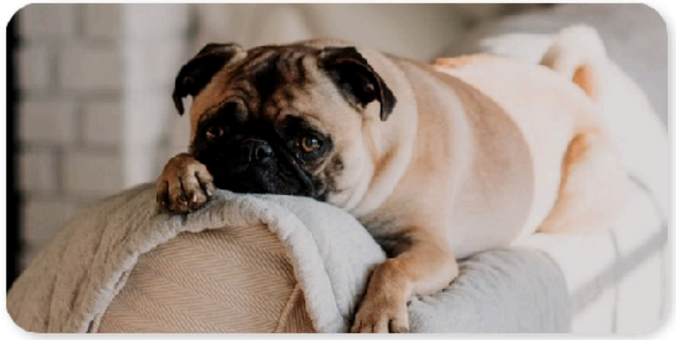
Animation, then, makes up for the lack of visible navigation tools by establishing and reinforcing metaphorical relationships between visible elements.

By encouraging users to navigate with a swipe instead of an arrow or button, we can save screen space.

Daniel Liddle

Emerging Guidelines for Communicating with Animation in Mobile User Interfaces

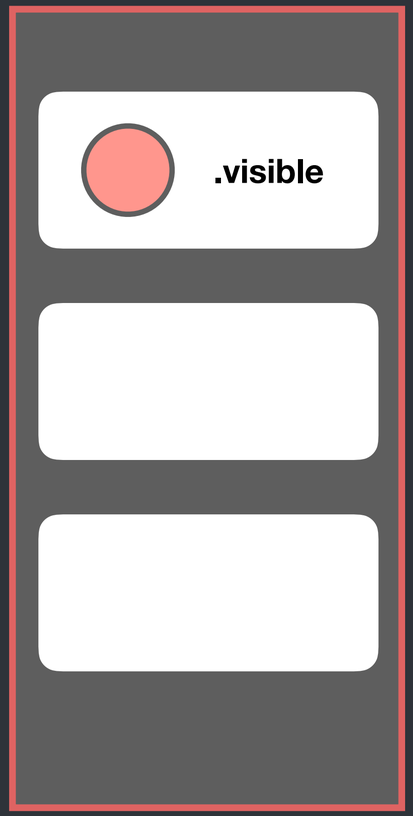
Hidden Content Animation



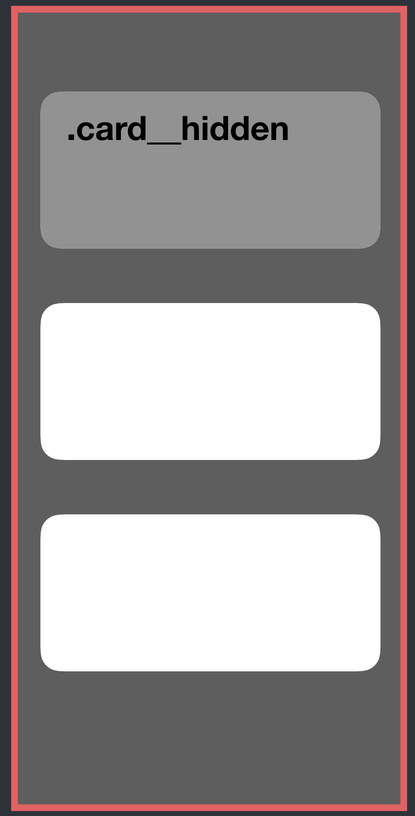
codepen.io/lisilinhart/full/GRRyxxY

```
<section class="card">
  <img class="card__image" src=" ... " >
  <div class="card__hidden">
    <h3 class="card__heading">Labrador</h3>
    <p class="card__text">The Labrador Retriever, or just Labrador,
        is a large breed of retriever-gun dog. The Labrador is the
        most popular breeds of dog in Canada, the United Kingdom
        and the United States.
    </p>
  </div>
</section>
```

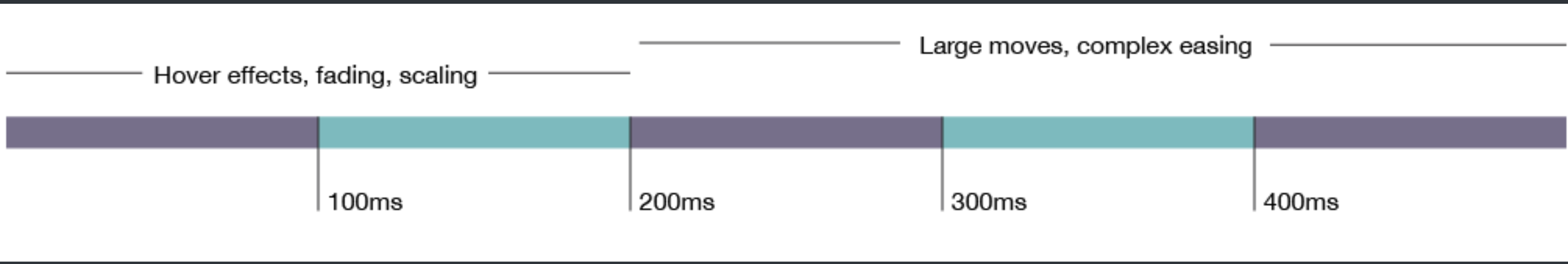
```
const cardEls = [...document.querySelectorAll(".card")];

cardEls.map(card => {
    card.addEventListener("pointerdown", handleStart, true);
    card.addEventListener("touchstart", handleStart, true);
    card.addEventListener("mousedown", handleStart, true);
});

function handleStart(evt) {
    const target = evt.currentTarget;
    target.classList.toggle("visible");
}
```



```css
.card__hidden {
  position: absolute;
  top: 0;
  left: 100%;
  transition: transform 0.35s cubic-bezier(0.47, 0, 0.745, 0.715);
}

.card.visible .card__hidden {
  transform: translateX(-100%);
}
```



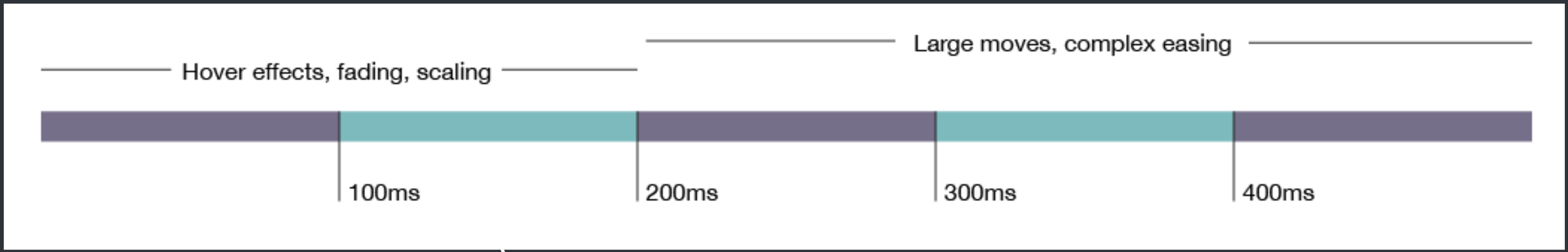
TIMING



valhead.com/2016/05/05/how-fast-should-your-ui-animations-be

TIMING

slow transitions
are less distracting



fast animations
are more likely to attract
attention

valhead.com/2016/05/05/how-fast-should-your-ui-animations-be/

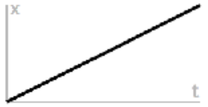
EASINGS.NET

Easing functions specify the rate of change of a parameter over time.

Objects in real life don't just start and stop instantly, and almost never move at a constant speed. When we open a drawer, we first move it quickly, and slow it down as it comes out. Drop something on the floor, and it will first accelerate downwards, and then bounce back up after hitting the floor.

This page helps you choose the right easing function.

linear



open source

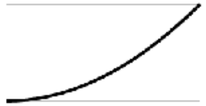
css+js

easeInSine

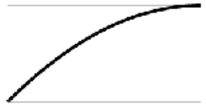
easeOutSine

easeInOutSine

easeInQuad

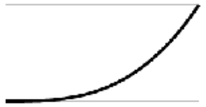


easeOutQuad



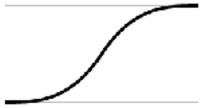
easeInOutQuad

easeInCubic

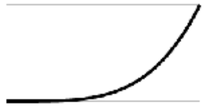


easeOutCubic

easeInOutCubic



easeInQuart



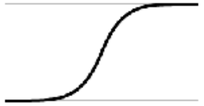
easeOutQuart

easeInOutQuart

easeInQuint

easeOutQuint

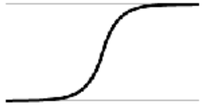
easeInOutQuint



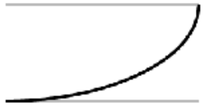
easeInExpo

easeOutExpo

easeInOutExpo



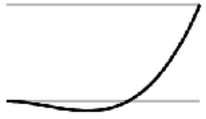
easeInCirc



easeOutCirc

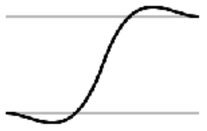
easeInOutCirc

easeInBack

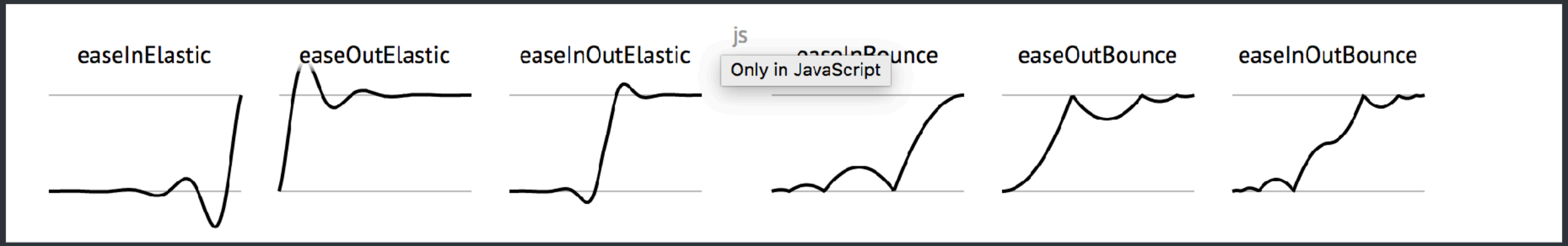


easeOutBack

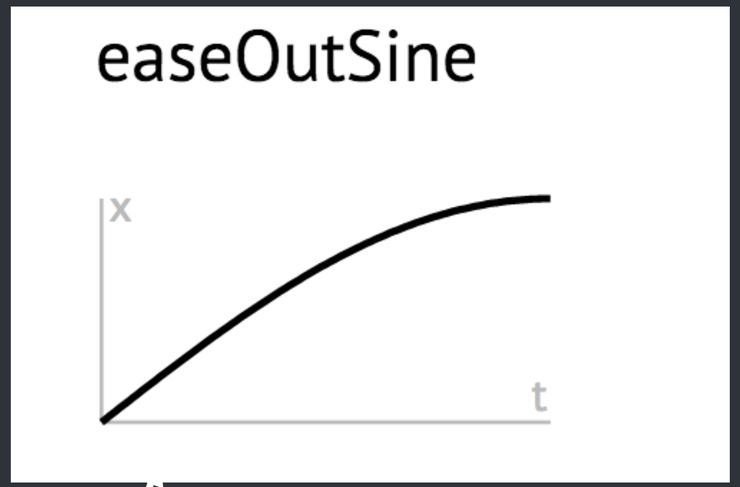
easeInOutBack



EASINGS.NET

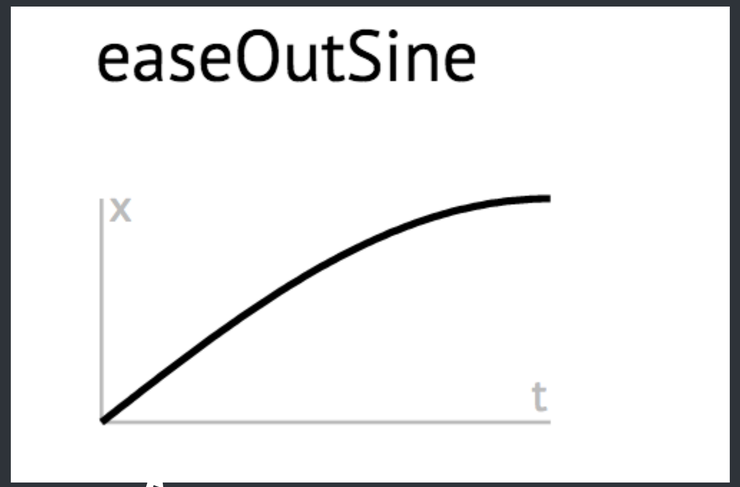

easeInElastic
easeOutElastic
easeInOutElastic
js
Only in JavaScript
easeOutBounce
easeInOutBounce

EASING

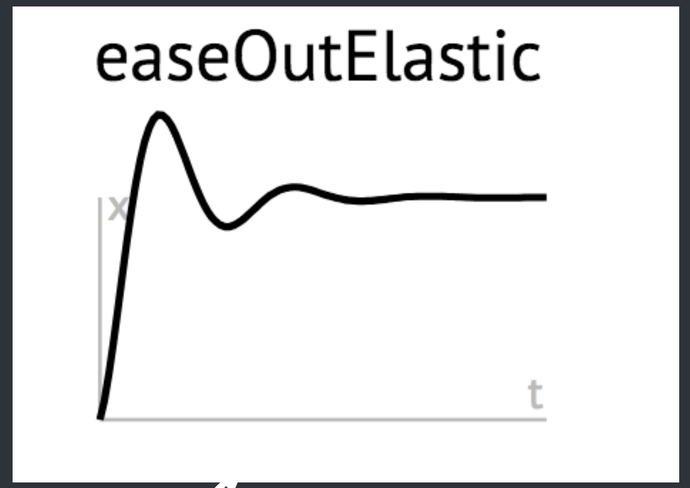


easeout feels more
reactive and instantaneous

EASING

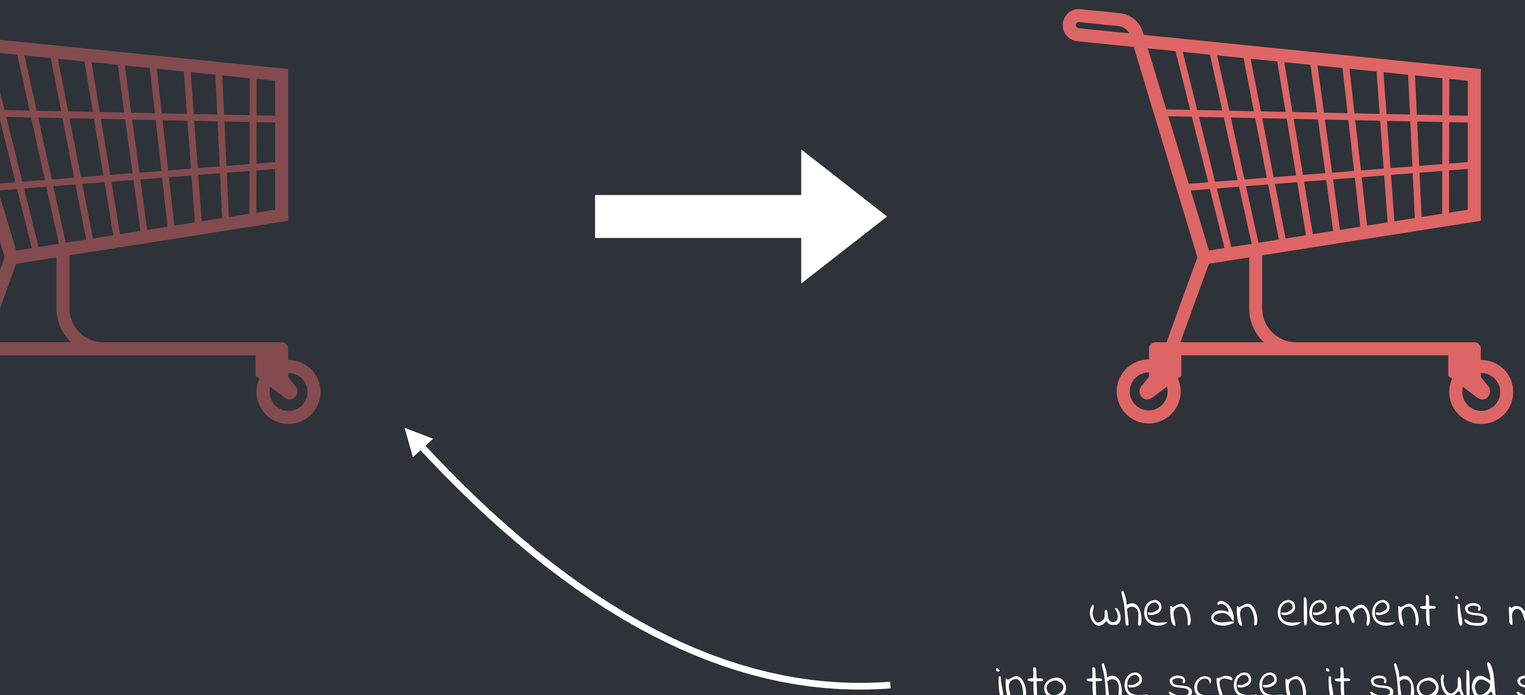


easeout feels more
reactive and instantaneous

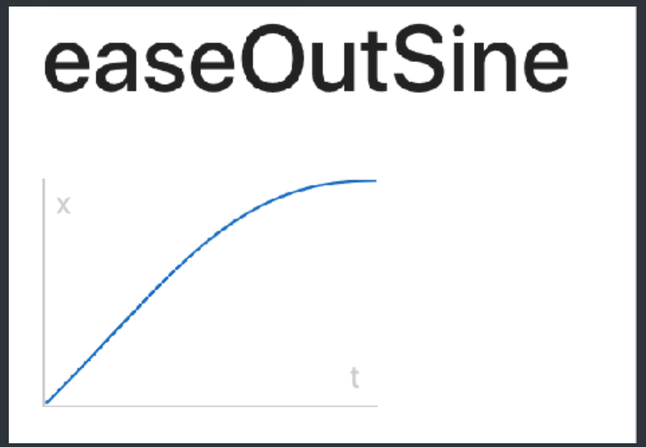


needs more time,
because it's a more
complex curve

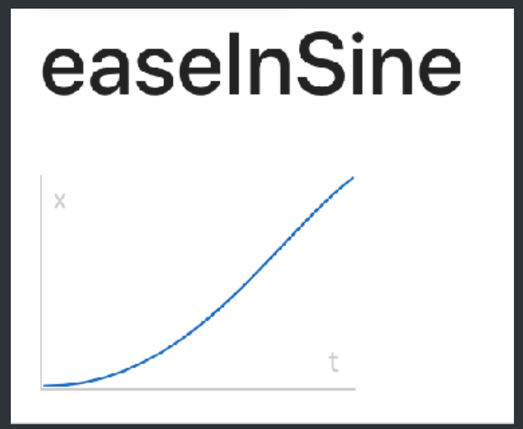
EASING



when an element is moving
into the screen it should start quick
and slow down towards the center



EASING

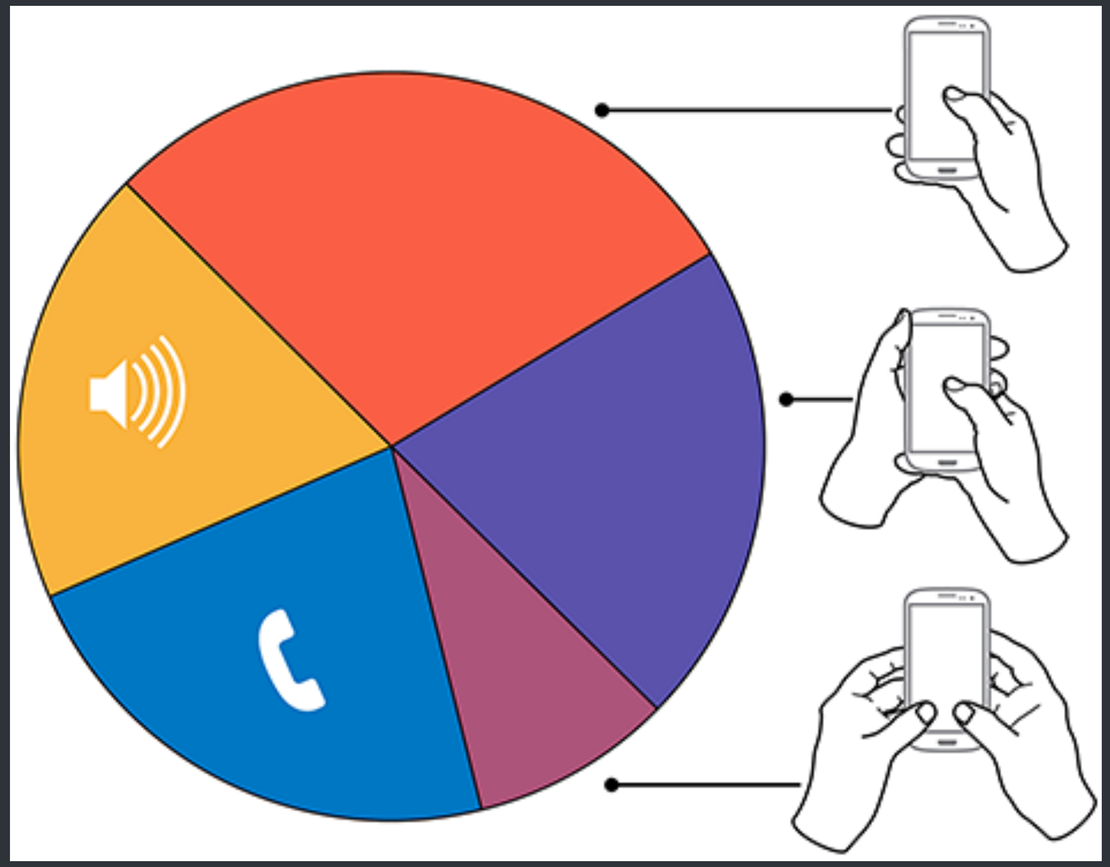


when an element is moving
out of the screen it should start slow
and speed up towards the end

Touch & Animation

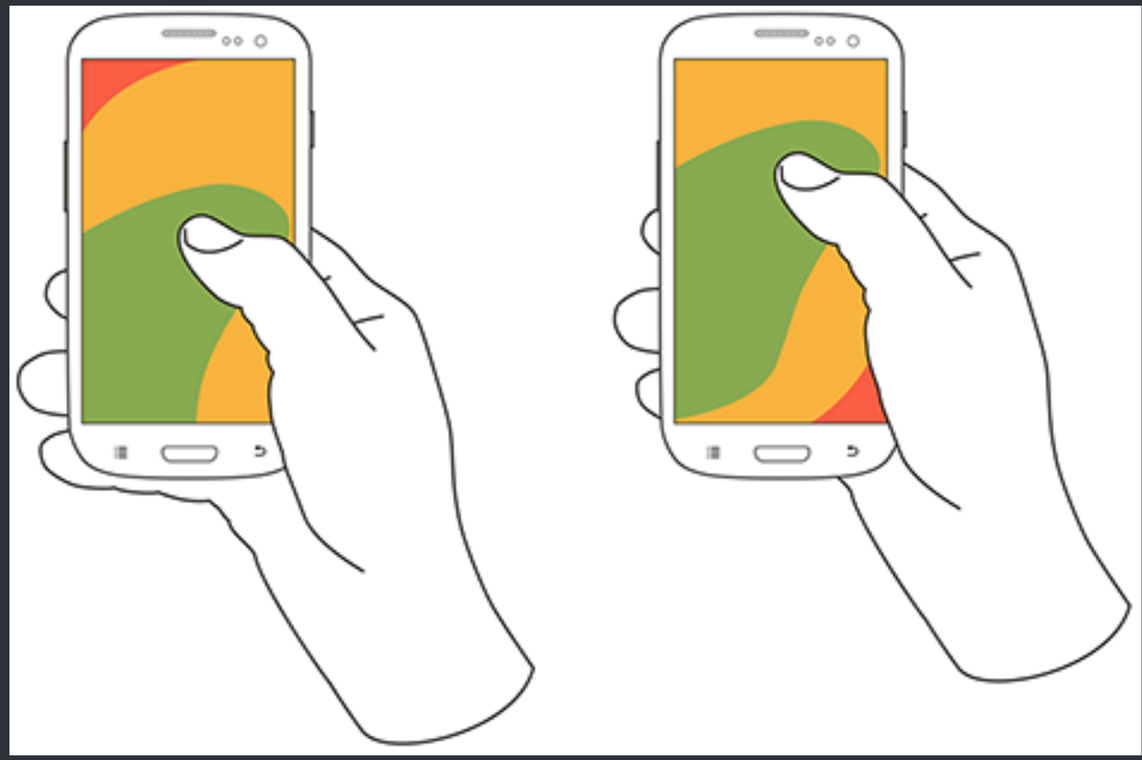
The users who we observed touching their phone's screens or buttons held their phones in three basic ways:

one handed — 49%
cradled — 36%
two handed — 15%

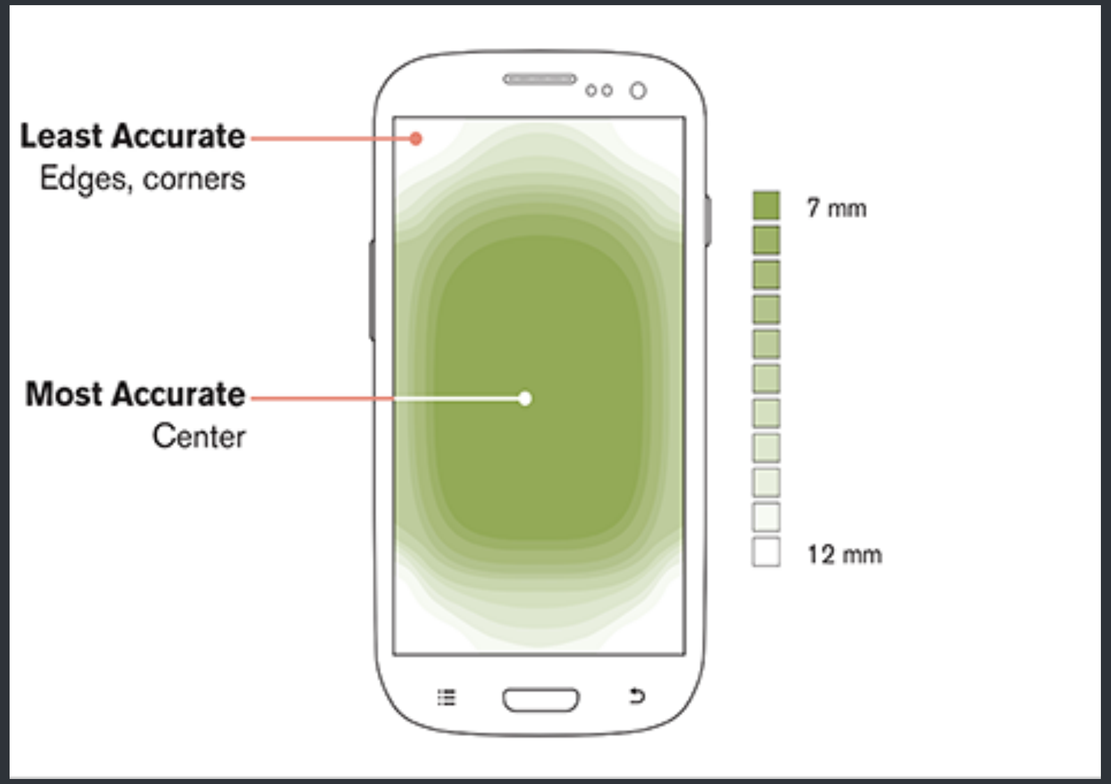


Study: How do users hold their phone

The 49% of users who use just one hand typically hold their phone in a variety of positions.



Study: How do users hold their phone



People can read content best at the center of the screen and often scroll content to bring the part they're reading to the middle of the screen if they can.

People are better at tapping at the center of the screen, so touch targets there can be smaller.

Article: Design for Fingers, Touch, and People

LUKEW IDEATION + DESIGN

CORE GESTURES Basic gestures for most touch commands

Tap

Briefly touch surface with fingertip

Double tap

Rapidly touch surface twice with fingertip

Drag

Move fingertip over surface without losing contact

Flick

Quickly brush surface with fingertip

Pinch

Touch surface with two fingers and bring them closer together

Spread

Touch surface with two fingers and move them apart

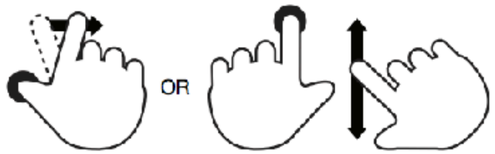
Press

Touch surface for extended period of time

Press and tap

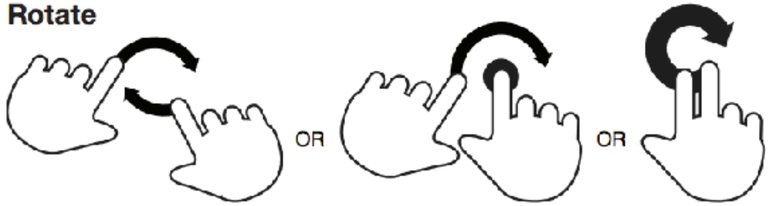
Press surface with one finger and briefly touch surface with second finger

Press and drag



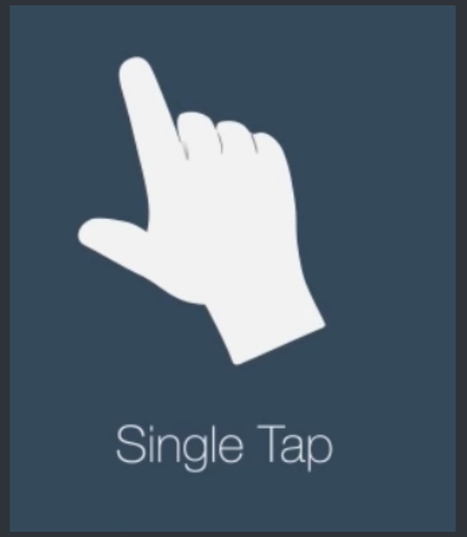
Press surface with one finger and move second finger over surface without losing contact

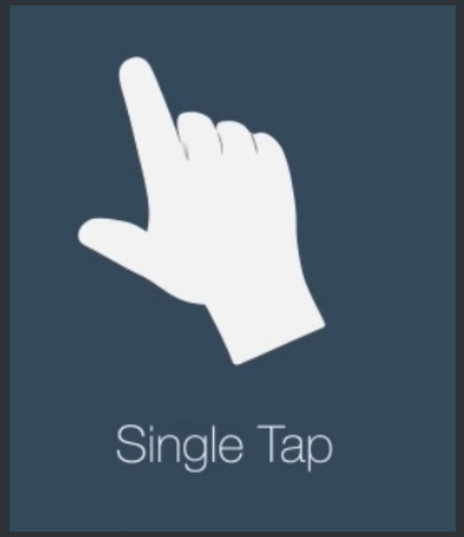
Rotate

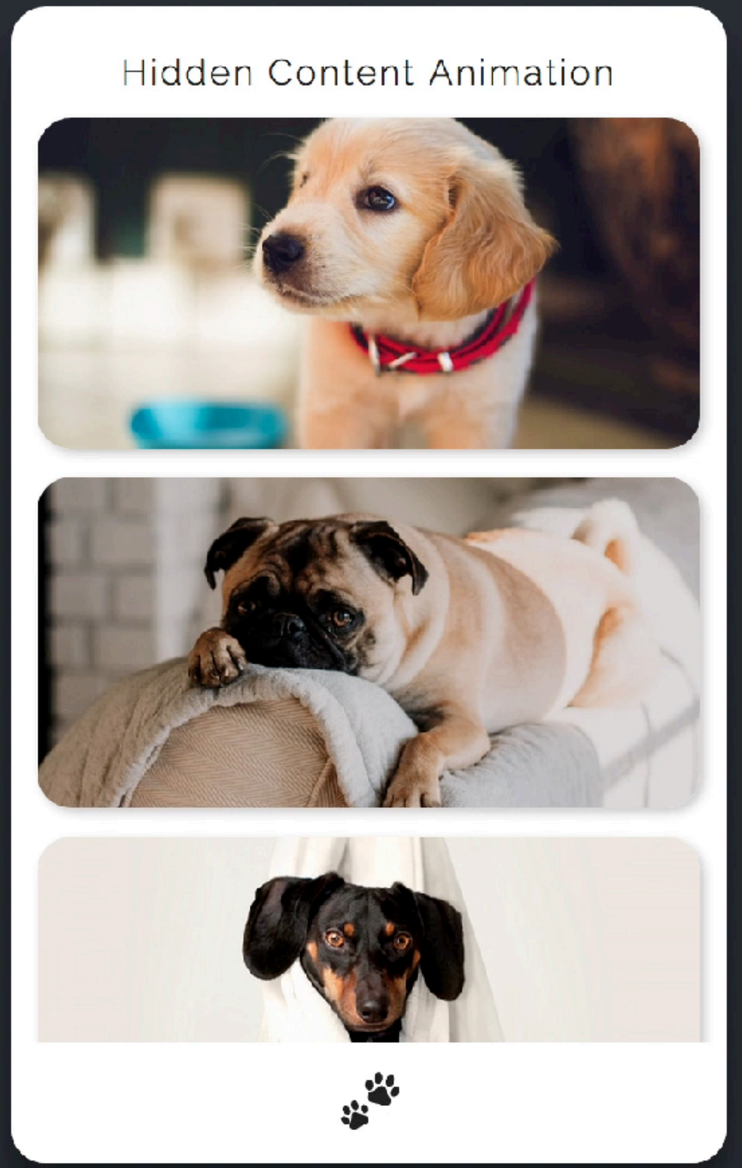


Touch surface with two fingers and move them in a clockwise or counterclockwise direction

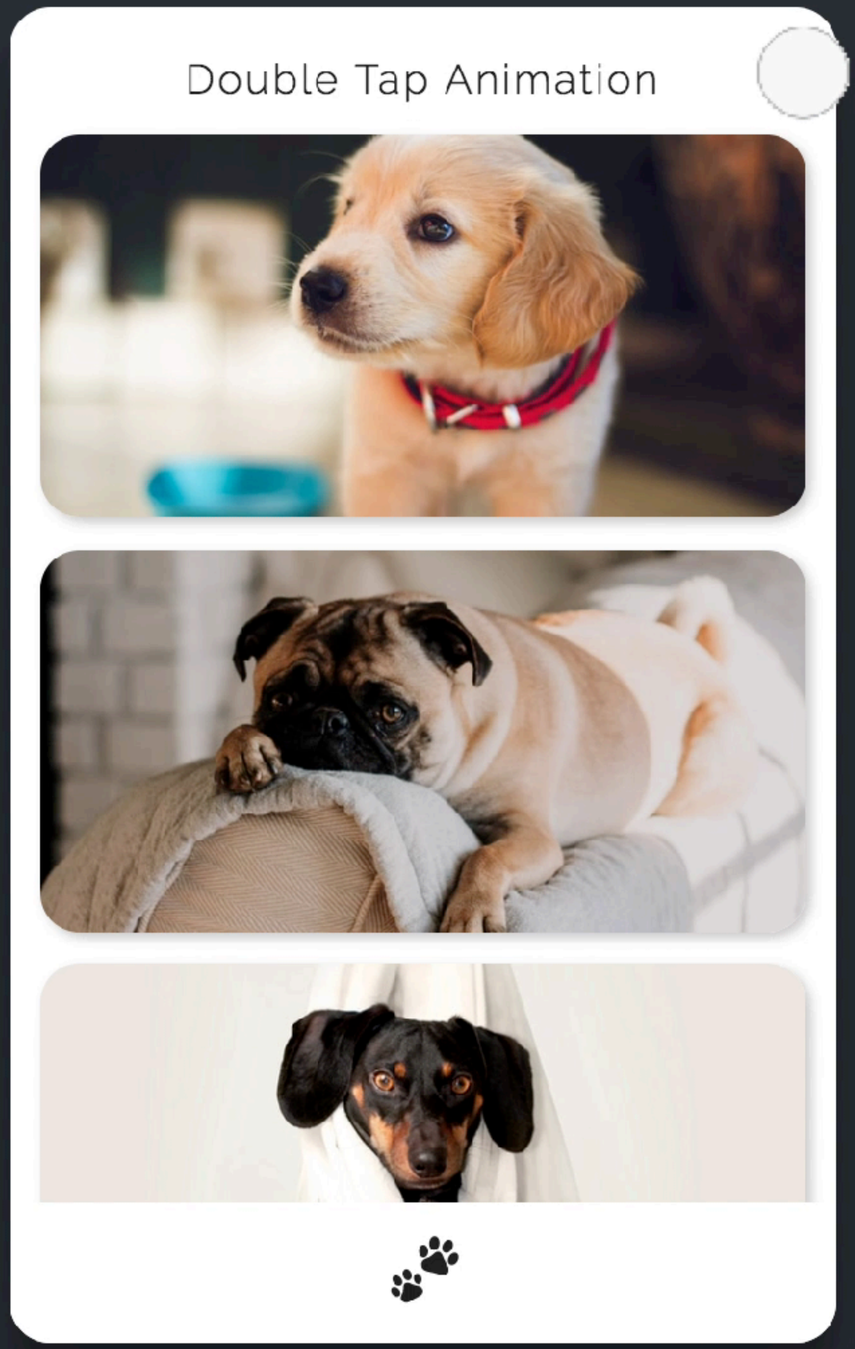
Source: Touch Gesture Reference Guide


Single Tap


Single Tap


Hidden Content Animation


Double Tap



codepen.io/lisilinhart/full/XWWZzYK

```css
.card:after {
    content: "❤️";
    transform: scale(0);
    transition: transform 0.3s
      cubic-bezier(0.39, 0.575, 0.565, 1);
}

.card.liked:after {
          animation: liked 0.8s
            cubic-bezier(0.445, 0.05, 0.55, 0.95)
            forwards;
}
```

codepen.io/lisilinhart/full/XWWZzYK

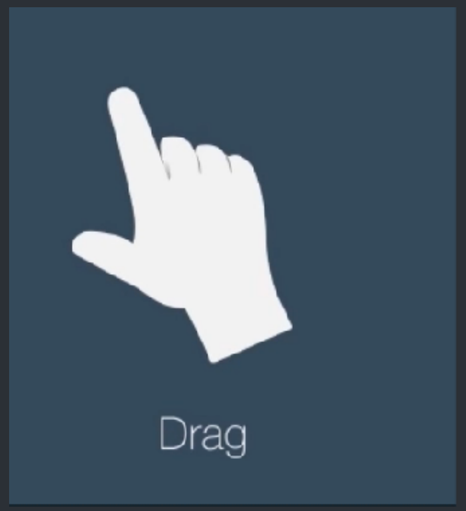
```css
.card.liked:after {
          animation: liked 0.8s
            cubic-bezier(0.445, 0.05, 0.55, 0.95)
            forwards;
}
```

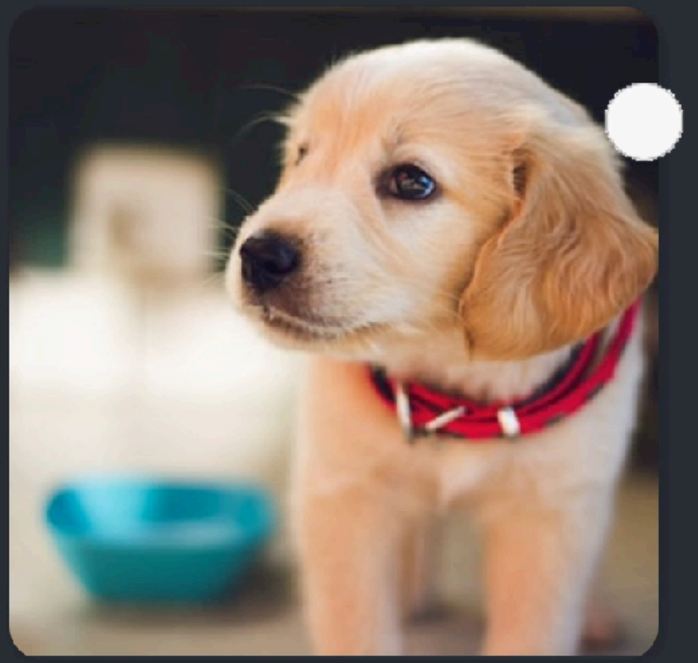
```css
@keyframes liked {
  0% {
    transform: scale(0) translate(0);
  }
  45% {
    transform: scale(1) translate(0);
  }
  65% {
    transform: scale(0.2) translate(0);
  }
  100% {
    transform: scale(0.2) translate(220%, 200%);
  }
}
```

codepen.io/lisilinhart/full/XWWZzYK

```
card.addEventListener("touchend", handleDoubleTap, true);
let lastTap = 0;

function handleDoubleTap(evt) {
    const target = evt.currentTarget;
    evt.preventDefault();

    let currentTime = new Date().getTime();
    let tapLength = currentTime - lastTap;

    if (tapLength < 500 && tapLength > 0) {
        target.classList.toggle("liked");
    }

    lastTap = currentTime;

}
```

codepen.io/lisilinhart/full/XWWZzYK





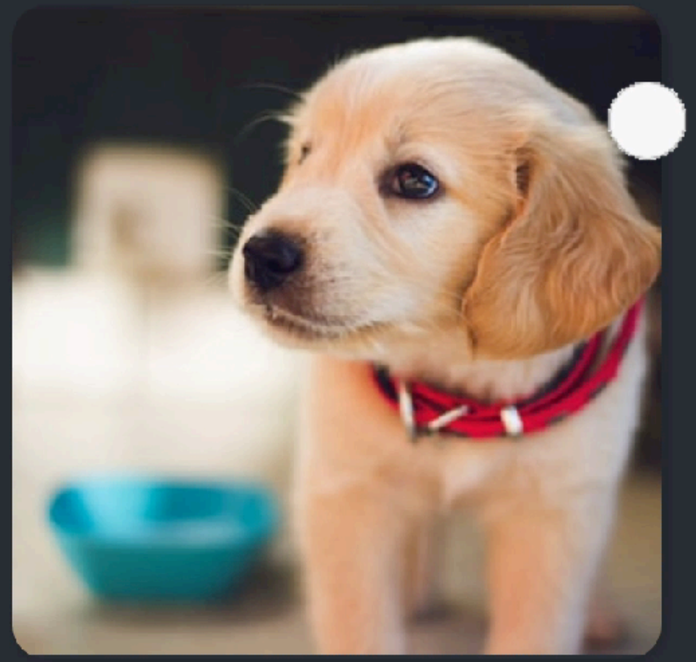
codepen.io/lisilinhart/full/gOOeONB

HAMMER.JS

Docs Support Contribute

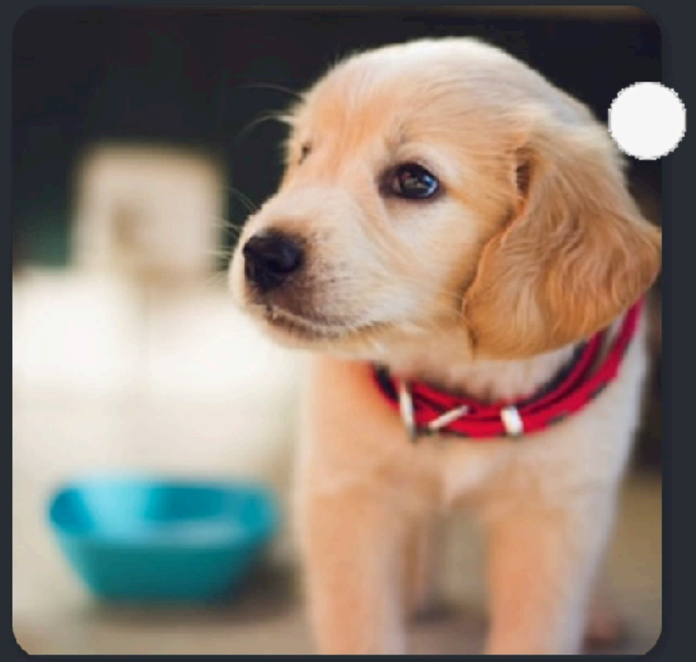
Add touch gestures to your webapp.

Hammer.min.js
v2.0.8 — 7.34 kB gzipped



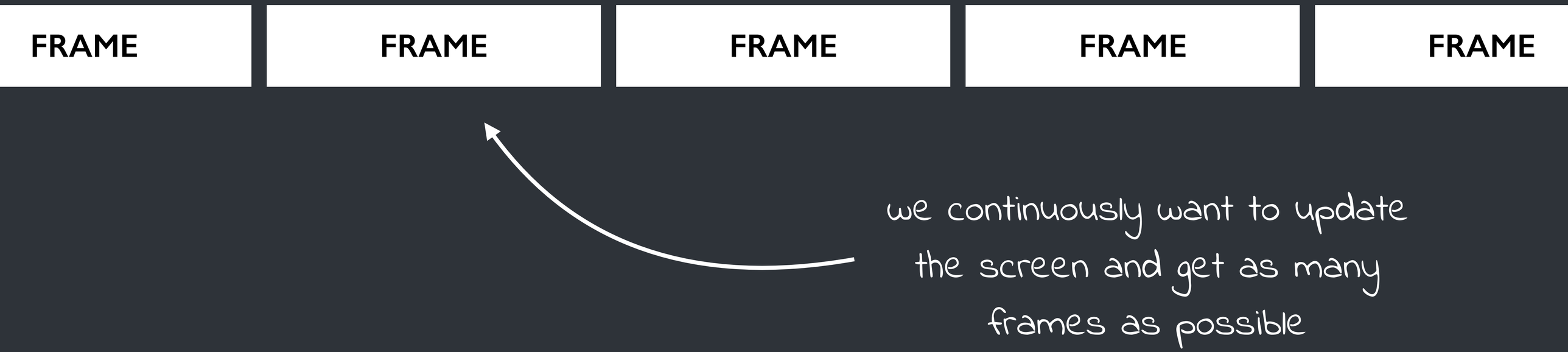
codepen.io/lisilinhart/full/gOOeONB

```
const el = document.querySelector(".container");
let reachedEnd = false; // as far as we can swipe
let rAF; // requestAnimationFrame Variable

const mc = new Hammer(el);
mc.get("pan").set({ direction: Hammer.DIRECTION_VERTICAL });
mc.on("pan", (evt) => {
    if(!reachedEnd) {
        let deltaY = evt.deltaY;
        let rotate = deltaY / window.innerHeight;

        request = requestAnimationFrame(() => update(deltaY, rotate))
    }
});
```

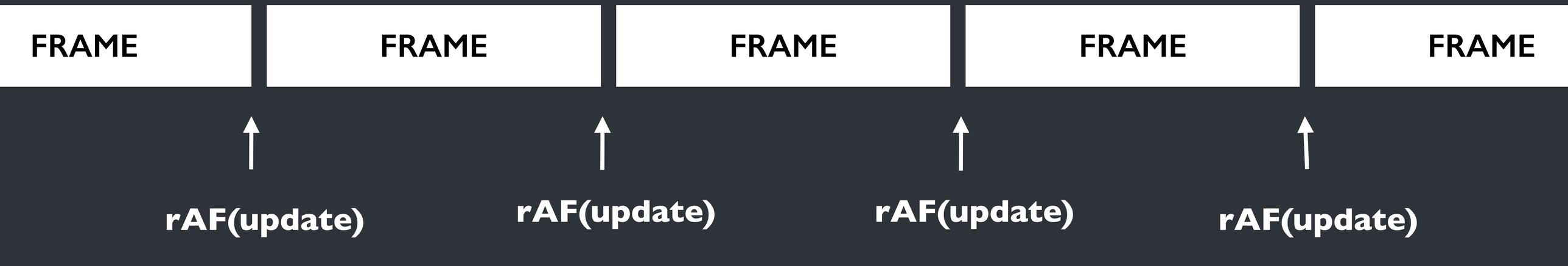


codepen.io/lisilinhart/full/gOOeONB

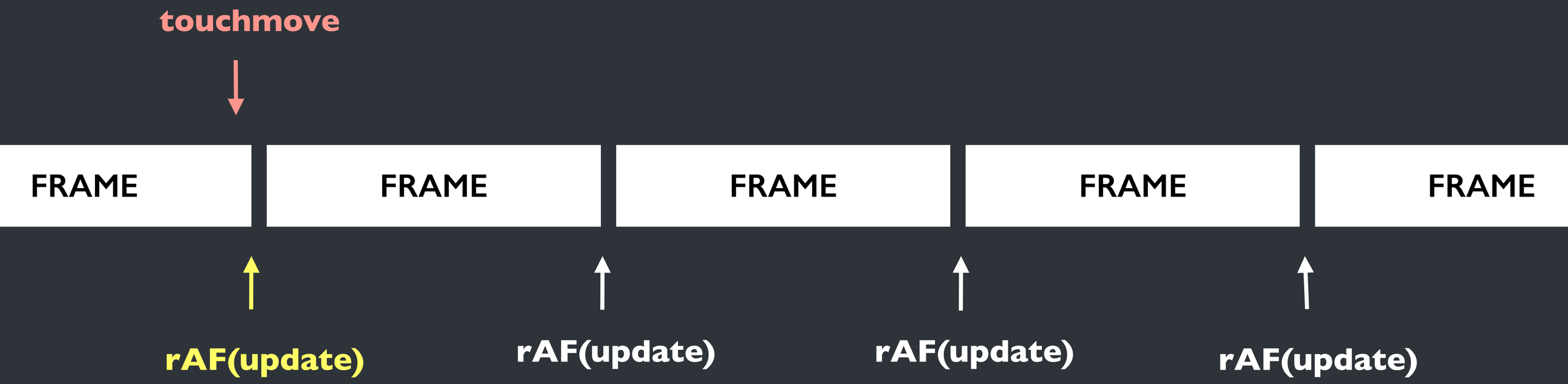
REQUESTANIMATIONFRAME



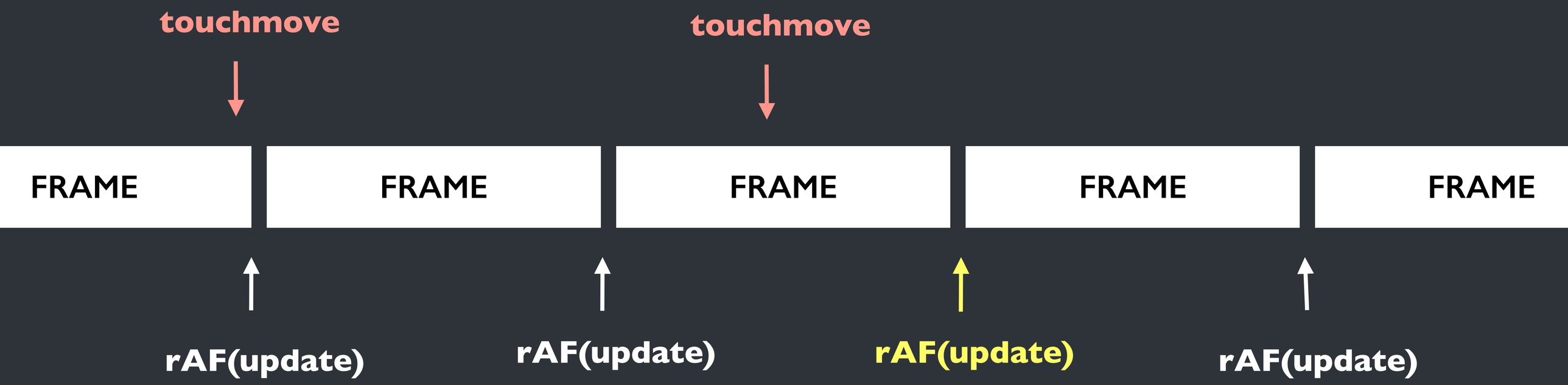
REQUESTANIMATIONFRAME



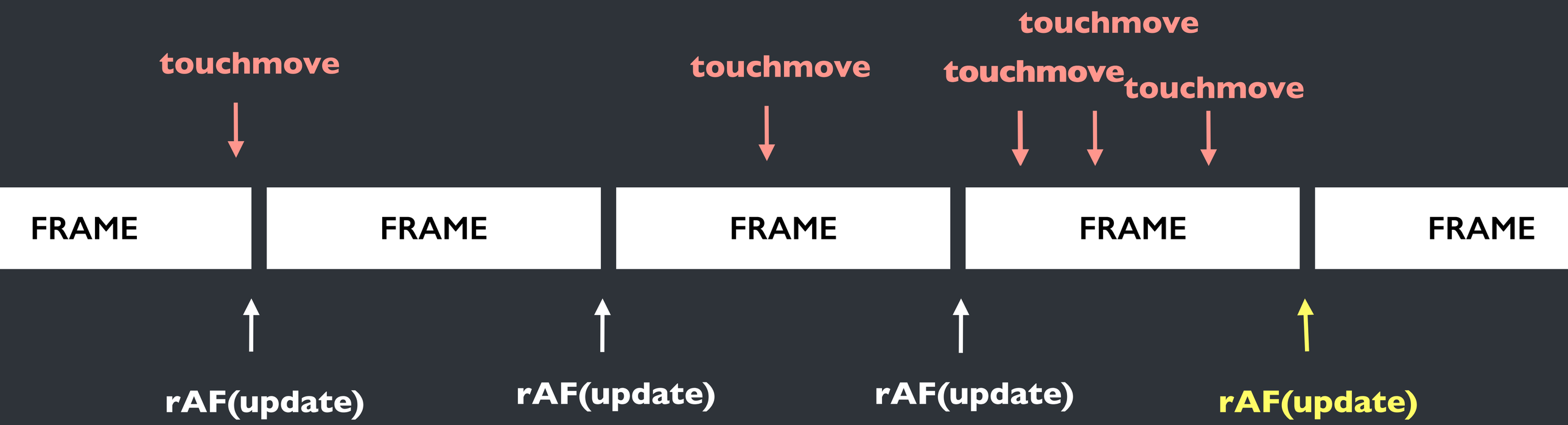
REQUESTANIMATIONFRAME



REQUESTANIMATIONFRAME

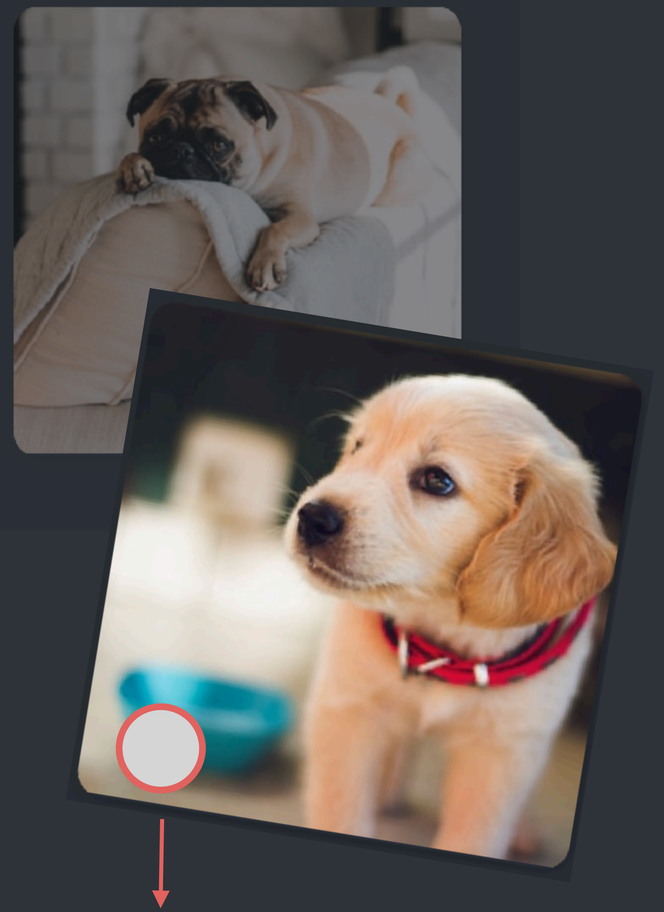


REQUESTANIMATIONFRAME



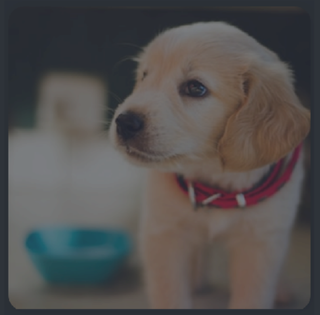
we update the next frame
only with the latest value

```
function update(deltaY, rotate) {
        let currentEl = items[item];
        currentEl.style.setProperty('--y', deltaY);
        currentEl.style.setProperty('--rot', rotate);
        currentEl.style.transitionProperty = "opacity";
```



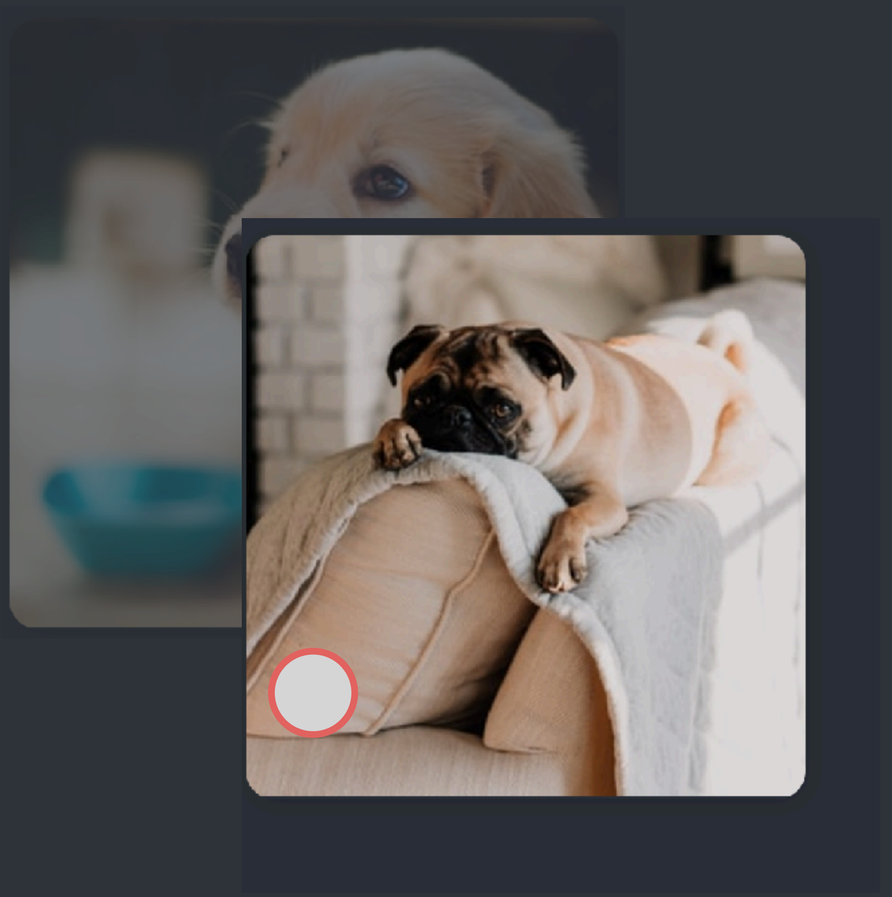
codepen.io/lisilinhart/full/gOOeONB

```
function update(deltaY, rotate) {
        let currentEl = items[item];
        currentEl.style.setProperty('--y', deltaY);
        currentEl.style.setProperty('--rot', rotate);
        currentEl.style.transitionProperty = "opacity";

        if(Math.abs(deltaY) > window.innerHeight * 0.4) {
            currentEl.style.transitionProperty = "opacity, transform";
            el.style.setProperty('--front', item)
            el.style.setProperty('--back', +!item)

            reachedEnd = true;
            item = +!item;
    }
}
```



codepen.io/lisilinhart/pen/gOOeONB

```
mc.on("panend", (evt) => {
    reachedEnd = false;
})

mc.on("panstart", (evt) => {
    otherEl = items[+!item];
    otherEl.style.setProperty('--y', 0);
    otherEl.style.setProperty('--rot', 0);
})
```



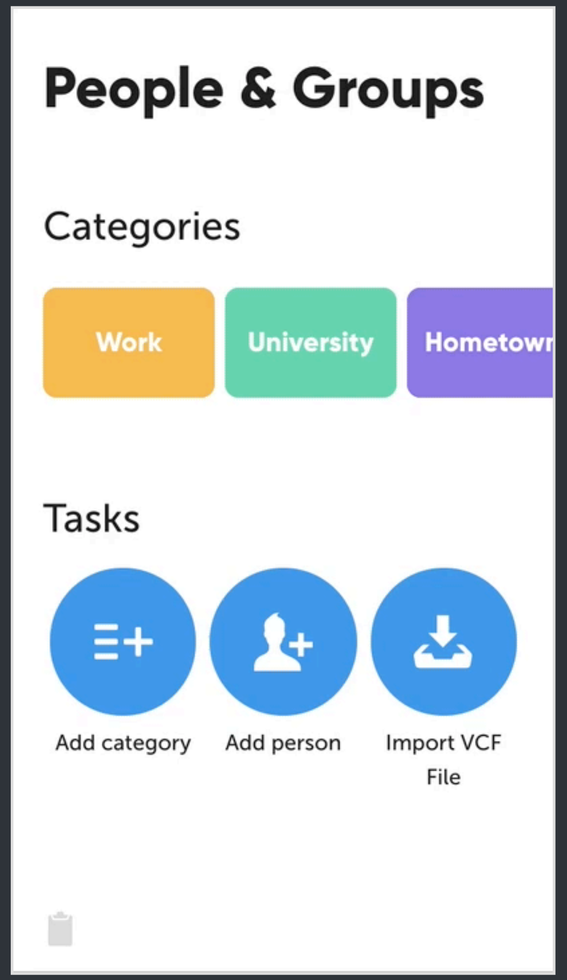
codepen.io/lisilinhart/pen/gOOeONB

Animation

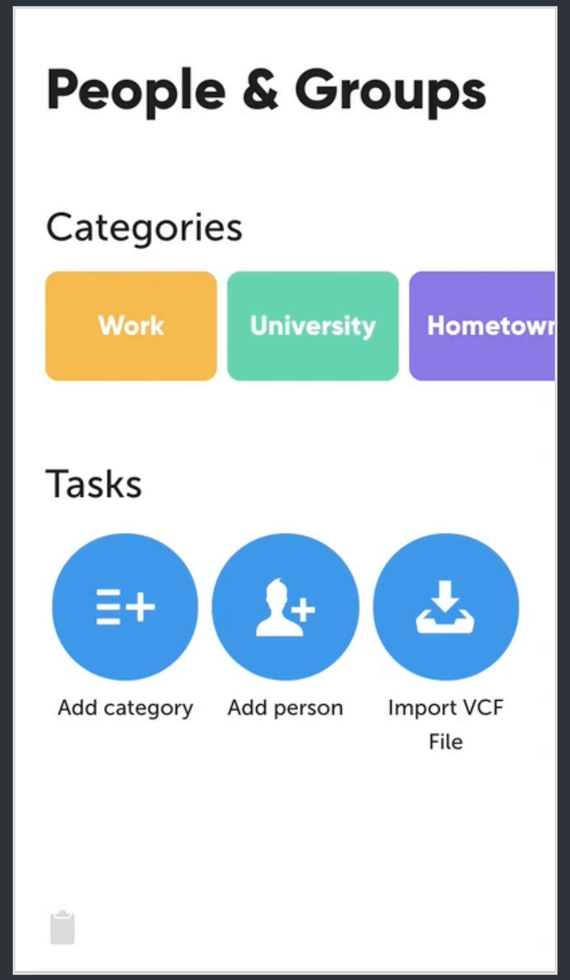
Mobile

TLDR;

Touch



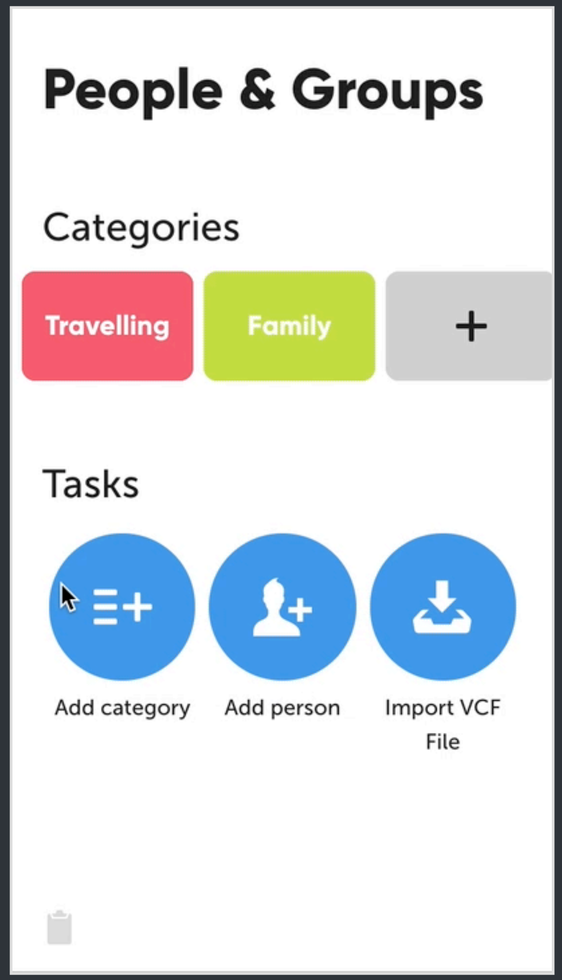
bit.ly/pwa-static

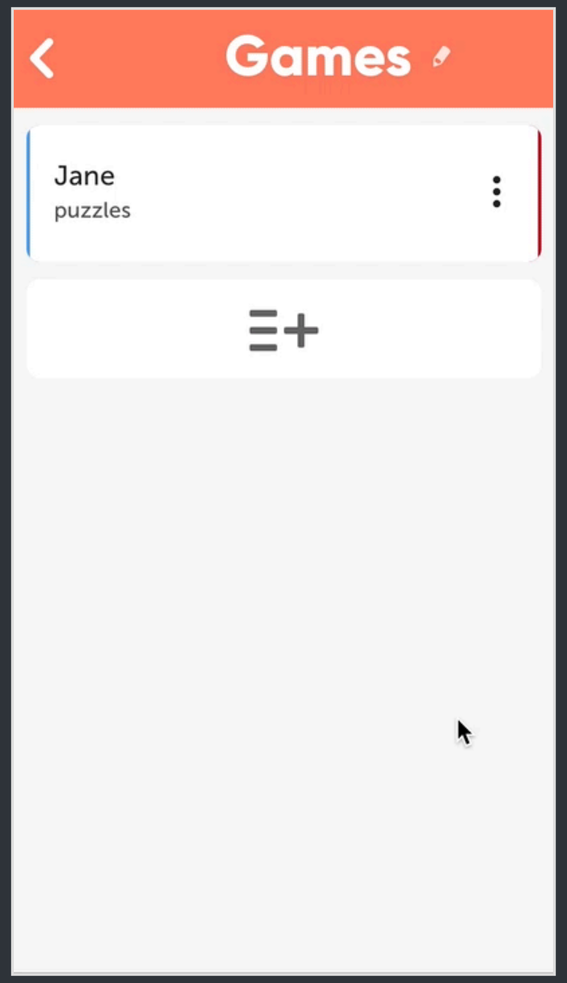


bit.ly/pwa-animated

Users prefer animated interface, when they're done in the right way.

Animation shouldn't delay or annoy the user.

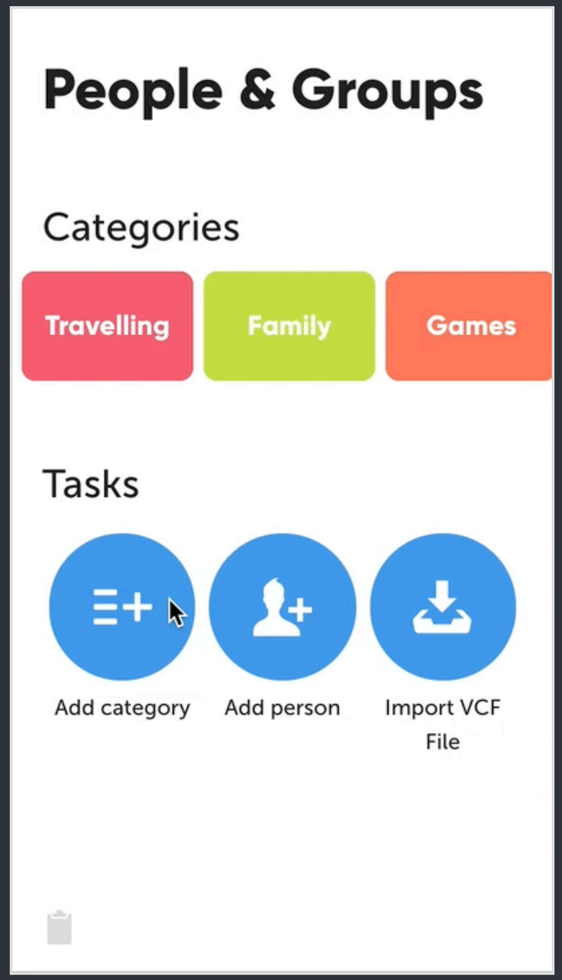


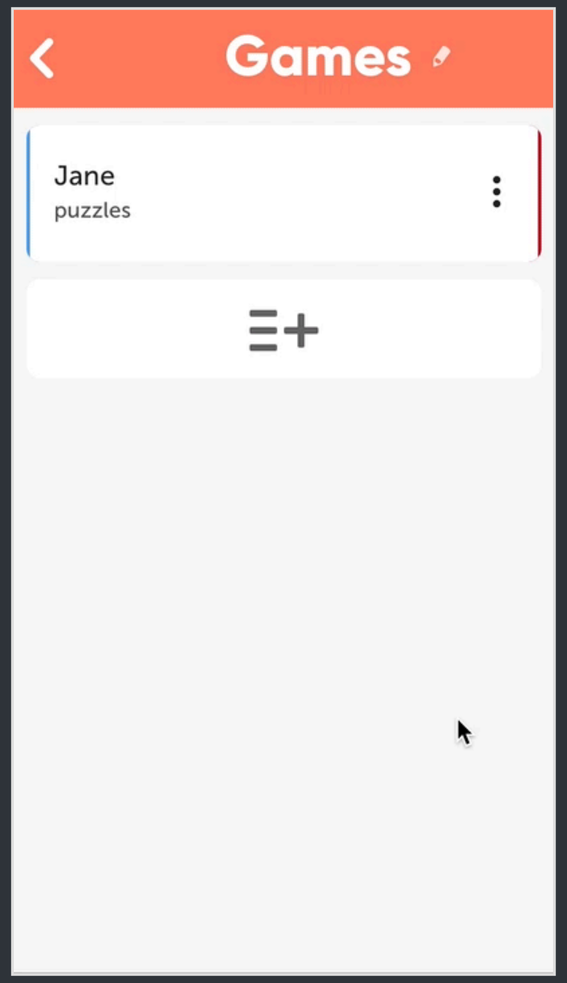


Combing touch gestures with animation, can allow the user to become accustomed to shortcuts through gestures.

These shortcuts allow the user to navigate more quickly in an application (expert user).

A lot of animation processing happens subconciously, which is why well done animations are often **not** noticed by the user.





When the design and UX aspects, which are common in native applications are used in PWAs, users also perceive them more as a an “App” rather than a mobile website.

WHAT TO AVOID?

missing touch feedback

small touch targets

gestures that aren’t understandable

animations that aren’t consistent with the touch movement

WHAT TO AVOID?

hard cuts between screens that are very different

missing transitions when information space updates

missing explanation of relationships between interface elements

WHAT TO AVOID?

empty screens

missing loading animations

missing skeleton screens

long waiting times

WHAT TO AVOID?

no feedback on touch events

blue flashes

long, annoying animations

not considering touch areas

“Users should only notice your animation if you need to attract their attention in that moment. Otherwise, micro-interactions and other transitions should be so seamless, users don’t even notice that there is animation”

Heather Daggett - Animation At work

FH Salzburg
MultiMediaTechnology

Lecturing @

LISILINHART.INFO

Blog

Twitter

@LISI_LINHART