

Programação Móvel

Interface e Ambiente de Desenvolvimento

Prof. Dr. Diego R. Antunes

drantunes@utfpr.edu.br

Departamento de Computação
Universidade Tecnológica Federal do Paraná

Programação Web

ES6

ECMAScript 6

*Ou **ES6** ou **ES2015**, é a nova especificação para o JavaScript. O padrão especifica classes, construtores, funções especiais, import e módulos, entre outros.*

ECMAScript 6

Um problema atual (2016) é que os navegadores não suportam 100% das novas funcionalidades.

ECMAScript 6

Assim, torna-se necessário utilizar alguma ferramenta para traduzir o código ES6 para a versão ES5, utilizada hoje nos navegadores.

Vantagens

- *Facilidade de adaptação para dev backend (java);*
- *Facilidade de leitura;*
- *Simplicidade;*

Babel

Ferramenta que faz a tradução do padrão ES6 para ES5.

<https://babeljs.io/>

Instalação

No diretório do projeto, criamos o arquivo `package.json` para o gerenciador de pacotes `npm`. Se necessário, instalar o [Node.js \(com NPM\)](#)

Instalação

Então, executar os passos descritos no site do Babel.

Exemplo

Pessoa.js

```
1  class Pessoa {
2      constructor(nome){
3          this.nome = nome;
4      }
5      ola(){
6          return this.nome + " disse Olá!";
7      }
8  }
9
10 console.log(new Pessoa("Diego").ola());
```

Exemplo

Então, podemos importar esta classe em nosso HTML e fazer o teste no navegador.

```
14 <body>
15 |
16 |   <script src="src/Pessoa.js"></script>
17 |
18 </body>
```

Exemplo

No Google Chrome este exemplo deve funcionar, e mostrar o Hello Word no console.log.

```
14 <body>
15
16 <script src="src/Pessoa.js"></script>
17
18 </body>
```

Elements Console Sources Network Timeline Profiles Resources Security Audits

top ▼ ☐ Preserve log

Diego disse Olá!

> |

Exemplo

Porém, para compilar para ES5, podemos usar o Babel usando o comando: `npm run build`, conforme as instruções no site oficial

Exemplo

O comando nos dará um javascript nesta forma:

```
1  "use strict";
2
3  var _createClass = function () { function defineProperties(target, props) { for (var i = 0; i < props.length; i++)
4
5  function _classCallCheck(instance, Constructor) { if (!(instance instanceof Constructor)) { throw new TypeError("Ca
6
7  var Pessoa = function () {
8      function Pessoa(nome) {
9          _classCallCheck(this, Pessoa);
10
11          this.nome = nome;
12      }
13
14      _createClass(Pessoa, [{
15          key: "ola",
16          value: function ola() {
```

Exemplo

Não se preocupe com a legibilidade, este é um arquivo compilado apenas para ser usado em produção para garantir a compatibilidade com ES5.

Exemplo

Então carregamos nosso arquivo compilado no HTML

```
14 <body>
15
16   <script src="lib/Pessoa.js"></script>
17
18 </body>
```

Diego disse Olá!

Gulp.js

Eventualmente podemos utilizar ferramentas de produtividade como o Gulp.

Gulp.js

Esta ferramenta possibilita concatenar arquivos, mover entre diretórios, assistir (watch) por mudanças nos arquivos, validação, entre outras tarefas comuns no workflow.

Gulp.js

No site do Babel há instruções para usar o plugin junto com o Gulp. Então podemos configurar um watch para compilar automaticamente os arquivos JS a cada mudança.

Gulp.js

Um exemplo de configuração seguindo as recomendações do site do Babel

```
4  gulp.task("js", function () {
5    return gulp.src("src/*.js")
6      .pipe(babel())
7      .pipe(gulp.dest("lib"));
8  });
9  // configure which files to watch and what tasks to use on file changes
10 gulp.task('watch', function() {
11   gulp.watch('src/**/*.js', ['js']);
12 });
```

Gulp.js

Com o comando `gulp watch`, agora os arquivos são compilados automaticamente a cada mudança.

```
iMac-DRA:exemplos-es6 drantunes$ gulp watch
[07:21:43] Using gulpfile ~/Desktop/exemplos-es6/gulpfile.js
[07:21:43] Starting 'watch'...
[07:21:43] Finished 'watch' after 15 ms
[07:21:50] Starting 'js'...
[07:21:51] Finished 'js' after 496 ms
```

Ferramentas

O Ionic e o Angular já incluem templates e ferramentas configuradas para as tarefas mais comuns de compilação, inicialização de servidor, entre outros.

ES6: var, const, let

*O ES6 insere novas formas de definição de variáveis além do conhecido **var**.*

ES6: var

Um problema comum é que ao declarar uma variável com var, o escopo dela se torna 'global' na função que estamos declarando.

ES6: var

*Portanto, ao testar o código a seguir com false no parâmetro, nós vemos no console o valor **undefined***

```
1 function teste(bool){
2     if(bool){
3         var ok = "OK!";
4         console.log(ok);
5     }
6     else {
7         console.log(ok);
8     }
9 }
```

ES6: var

*Ou seja, a função **teste** entra no **else**, linha 6, mas como não há atribuição para **var ok**, então ela é **undefined**.*

```
1 function teste(bool){  
2     if(bool){  
3         var ok = "OK!";  
4         console.log(ok);  
5     }  
6     else {  
7         console.log(ok);  
8     }  
9 }
```

ES6: var

Assim, é como se o navegador interpretasse o arquivo JS declarando a variável ok no topo da função.

```
2  var ok;
3  if(bool){
4      ok = "OK!";
5      console.log(ok);
6  }
7  else {
8      console.log(ok);
9  }
```

ES6: let

O let permite declarar variáveis com escopo de bloco ou local ao bloco que está declarando.

ES6: let

Agora nosso exemplo executa como deveria executar.

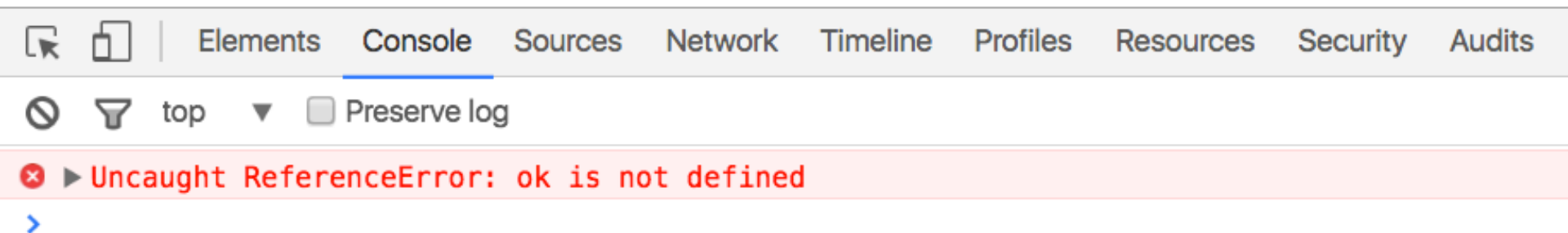
Neste caso, se cair no else o console apresentará um erro, pois a variável ok não está definida.

```
1 function teste(bool){  
2     if(bool){  
3         let ok = "OK!";  
4         console.log(ok);  
5     }  
6     else {  
7         console.log(ok);  
8     }
```

ES6: let

Agora nosso exemplo executa como deveria executar.

Neste caso, se cair no else o console apresentará um erro, pois a variável ok não está definida.



ES6: function

*No ES6 funções não precisam mais ser definidas usando a tag **function**. Exemplo: linha 5*

```
1  class ColecaoTarefas {
2      constructor(tarefas = []){
3          this.tarefas = tarefas;
4      }
5      log(){
6          this.tarefas.forEach( function(tarefa) {
7              console.log(tarefa);
8          });
9      }
10 }
```


ES6: function

Neste caso definimos a função log apenas declarando-a como uma função `log()`{ ...

```
1 class ColecaoTarefas {
2     constructor(tarefas = []){
3         this.tarefas = tarefas;
4     }
5     log(){
6         this.tarefas.forEach( function(tarefa) {
7             console.log(tarefa);
8         });
9     }
10 }
```

ES6: sintaxe arrow

O ES6 implementa o conceito do operador => (arrow).

ES6: sintaxe arrow

Considerando o exemplo anterior, podemos traduzí-lo para o uso do operador arrow:

```
1  class ColecaoTarefas {  
2      constructor(tarefas = []){  
3          this.tarefas = tarefas;  
4      }  
5      log(){  
6          this.tarefas.forEach( function(tarefa) {  
7              console.log(tarefa);  
8          });  
9      }  
10 }
```

ES6: sintaxe arrow

Considerando o exemplo anterior, podemos traduzí-lo para o uso do operador arrow:

```
1  class ColecaoTarefas {
2      constructor(tarefas = []){
3          this.tarefas = tarefas;
4      }
5      log(){
6          this.tarefas.forEach( (tarefa) => {
7              console.log(tarefa);
8          });
9      }
10 }
```

ES6: sintaxe arrow

Nos casos onde o corpo (conteúdo) da função é somente uma linha, como no exemplo anterior, podemos simplificar ainda mais a sintaxe.

ES6: sintaxe arrow

Por exemplo:

```
1  class ColecaoTarefas {  
2      constructor(tarefas = []){  
3          this.tarefas = tarefas;  
4      }  
5      log(){  
6          this.tarefas.forEach( (tarefa) => console.log(tarefa) );  
7      }  
8  }
```

ES6: sintaxe arrow

A sintaxe => também deixa implícito o return quando ele é necessário. Vejamos o seguinte exemplo.

```
1  let nomes = ["Diego", "Andre", "Laura"];
2
3  nomes = nomes.map(function(nome){
4      return "#" + nome + "#";
5  });
6
7  console.log(nomes);
```

ES6: sintaxe arrow

Ou seja, ao usar a função map para alterar o array, poderíamos usar a sintaxe =>, pois o return já está implícito.

```
1 let nomes = ["Diego", "Andre", "Laura"];
2
3 nomes = nomes.map(function(nome){
4     return "#" + nome + "#";
5 });
6
7 console.log(nomes);
```


ES6: sintaxe arrow

Modificando para ES6, temos:

```
1  let nomes = ["Diego", "Andre", "Laura"];  
2  
3  nomes = nomes.map(nome => "#" + nome + "#");  
4  
5  console.log(nomes);|
```

ES6: parâmetros default

Agora, podemos especificar valores default para parâmetros de funções, quando estes são nulos.

ES6: parâmetros default

Neste caso, chamamos a função teste sem parâmetros. Assim ela é executada com um valor padrão para a variável palavra.

```
1 function teste(palavra = "palavra padrão"){  
2     console.log(palavra);  
3 }  
4 teste();  
5 teste("teste");
```

ES6: template strings

Uma nova maneira de concatenar strings e variáveis.

ES6: template strings

Exemplo: um template HTML dentro do JS pode ser complexo. Além disso, não é possível usar múltiplas linhas.

```
1 let nome = "João";  
2  
3 let template = '<div class="alert"><span>' + nome + '</span></div>';  
4  
5 console.log(template);
```

ES6: template strings

Com template strings temos o seguinte:

```
1 let nome = "João";  
2  
3 let template = `4  
5 console.log(template);
```

ES6: template strings

Parece simples, mas agora também podemos usar múltiplas linhas se necessário.

```
1 let nome = "João";  
2  
3 let template = `  
4     <div class="alert">  
5         <span> ${nome} </span>  
6     </div>  
7 `;
```

ES6: Outras funcionalidades

Podemos ainda usar classes, herança, construtores, funções estáticas e módulos – imports (próxima aula).

Framework vs. Biblioteca

SPA

*Single Page Applications: são aplicações desenvolvidas no lado do cliente (e.g. **Gmail**) em que o usuário pode navegar e utilizar sem necessitar de reload na página.*

SPA

Quais componentes um framework deve possuir para facilitar o desenvolvimento de SPA?

Funcionalidades

Funcionalidades

A maioria dos frameworks JS atuais disponibilizam alguns componentes / funcionalidades padrão, necessários para o desenvolvimento de SPA.

Two Data Binding

One e Two – Data Binding

Um termo que se tornou popular com a versão 1 do framework AngularJS, o Two Data Binding é um conceito fundamental em uma SPA.

Two Data Binding

Basicamente, o framework cria um mapeamento em tempo real e dinâmico de uma propriedade. Ou seja, um link é realizado entre o model e a view pelo controller.

Two Data Binding

Desta forma, a interface se torna “reativa”, ou seja, quando a propriedade sofre uma modificação, esta modificação é percebida em tempo real na view (HTML).

Two Data Binding

Da mesma maneira, se realizarmos uma alteração na propriedade na view (HTML), o model é automaticamente atualizado.

Two Data Binding

Veremos diversos exemplos destes ao trabalhar com Angular.

Componentes

Componentes

Esta funcionalidade possibilita a criação e o uso de elementos HTML personalizados.

Componentes

Os componentes ficaram populares no AngularJS. Eles possibilitam a criação de um elemento customizado com seus próprios atributos, eventos e valores.

Componentes

Por exemplo, é possível criar um componente chamado lista-alunos, que recebe uma lista e mostra a lista de alunos no HTML.

Componentes

Exemplo:

```
<body>
```

```
  <lista-alunos fonte="lista.json"></lista-alunos>
```

```
</body>
```


Componentes

Basicamente, cada framework descreve uma forma específica de construir tais componentes.

Elementos básicos

- *nome do elemento (tag);*
- *propriedades e atributos (data binding);*
- *controlador (controller);*
- *dependências (outros componentes);*
- *template HTML (estrutura HTML do elemento);*

Componentes

É importante entender o conceito de componente, pois Angular e Ionic são orientados a componentes, ou seja, cada “módulo” ou funcionalidade da nossa aplicação será criada como um componente.

Componentes

*Por exemplo, um componente de **Login** é composto de um template HTML (view), as anotações de classe, o controller, os atributos e métodos e o data-binding.*

Componentes

O componente Login também pode utilizar outros componentes como Botão, Barra de Tarefas, entre outros.

Services

Serviços

Para que um componente se comunique com um servidor ou API, um serviço (Service) deve ser utilizado para encapsular as regras de negócio para o acesso aos dados.

HTTP Resource

HTTP Resource

Consiste em um módulo responsável pelos métodos de acesso aos recursos HTTP de um servidor.

HTTP Resource

Ou seja, consiste em uma estrutura para utilização de AJAX para consumo e uso de Web Services (REST API).

Router

Router / Rotas

Consiste em um módulo para permitir a criação de rotas entre páginas (links), mas sem recarregar a página. Ou seja, possibilita a navegação em uma SPA.

Router / Rotas

Dentre as características interessantes estão as rotas aninhadas (Nested Routes), que possibilitam criar sub-páginas ou sub-navegação.

Router / Rotas

Exemplos rotas principais:

/aluno

/aluno/1

Exemplos de rotas aninhadas:

/aluno/1/disciplinas

Router / Rotas

Cada framework implementa de uma maneira específica. Mas também pode ser utilizado um componente de rota de terceiros (um plugin).

Templates

Templates HTML

Juntamente com Componentes e Rotas, o framework geralmente disponibiliza uma forma de criar templates HTML dinâmicos.

Templates HTML

Estes templates podem ser especificados no próprio arquivo JS ou em um arquivo HTML e carregado via AJAX.

Templates HTML

Atualmente, muitos frameworks tem definido os templates diretamente no arquivo JS, usando strings ou template strings (ES6), para evitar o tráfego AJAX para cada template HTML.

Templates HTML

Também existem alguns módulos, tais como o JSX (usado pelo Facebook React) que permite escrever HTML puro dentro de um arquivo JS. Então, o componente compila este código HTML para JS.

Exemplo JSX

Code

```
var Nav, Profile;
// Input (JSX):
var app = <Nav color="blue"><Profile>click</Profile></Nav>;
// Output (JS):
var app = React.createElement(
  Nav,
  {color:"blue"},
  React.createElement(Profile, null, "click")
);
```

Interface e Componentes

Interface e Componentes

Na maioria dos casos, os frameworks JS se concentram nas funcionalidades e no desempenho.

Interface e Componentes

Porém, muitas vezes, s frameworks disponibilizam componentes visuais para interação com o usuário. Este é o caso do Ionic.

Frameworks JS



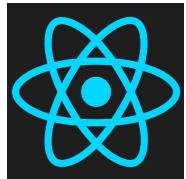
vuejs.org



angular.io



emberjs.com



facebook.github.io/react



knockoutjs.com



backbonejs.org

Como escolher um framework?

- 1 *Facilidade de uso, necessidades, etc.*
- 2 *Estude os frameworks.*
- 3 *Compare os frameworks;*



Helping you **select** an MV* framework

[Download](#)[View on GitHub](#)[Blog](#)

Introduction

Developers these days are spoiled with choice when it comes to **selecting** an **MV* framework** for structuring and organizing their JavaScript web apps.

Backbone, Ember, AngularJS... the list of new and stable solutions continues to grow, but just how do you decide on which to use in a sea of so many options?

To help solve this problem, we created **TodoMVC** - a project which offers the same Todo application implemented using MV* concepts in most of the popular JavaScript MV* frameworks of today.

[Follow](#) [Tweet](#) [G+1](#)

Examples

[JavaScript](#)[Compile-to-JS](#)[Labs](#)

These are examples written in pure JavaScript.

[Backbone.js^R](#)[AngularJS^R](#)[Ember.js^R](#)[KnockoutJS^R](#)[Dojo^R](#)[Knockback.js^R](#)[CanJS^R](#)[Polymer^R](#)[React^R](#)[Mithril^R](#)[Ampersand^R](#)[Flight^R](#)[Vue.js^R](#)[MarionetteJS^R](#)[TroopJS +
RequireJS^R](#)

^R = App also demonstrates routing

Real-time

[SocketStream^R](#)[Firebase +
AngularJS^R](#)

TodoMVC

- [TodoMVC](#)
- *Desenvolve um Todo (lista de tarefas) nos frameworks mais utilizados / populares.*
- *Como se trata das mesmas funcionalidades, você pode comparar o código fonte em cada framework.*

Angular



www.angular.io

Angular

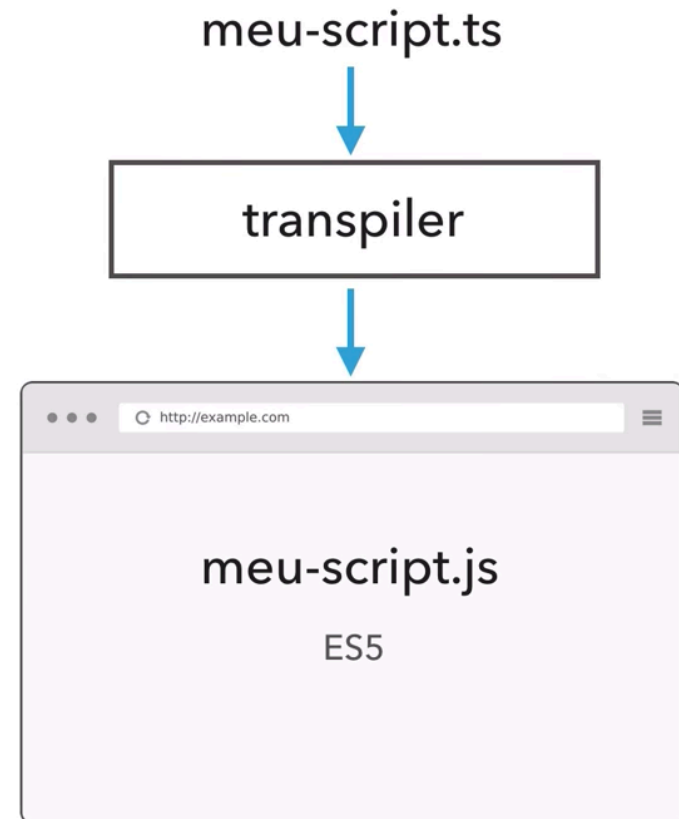
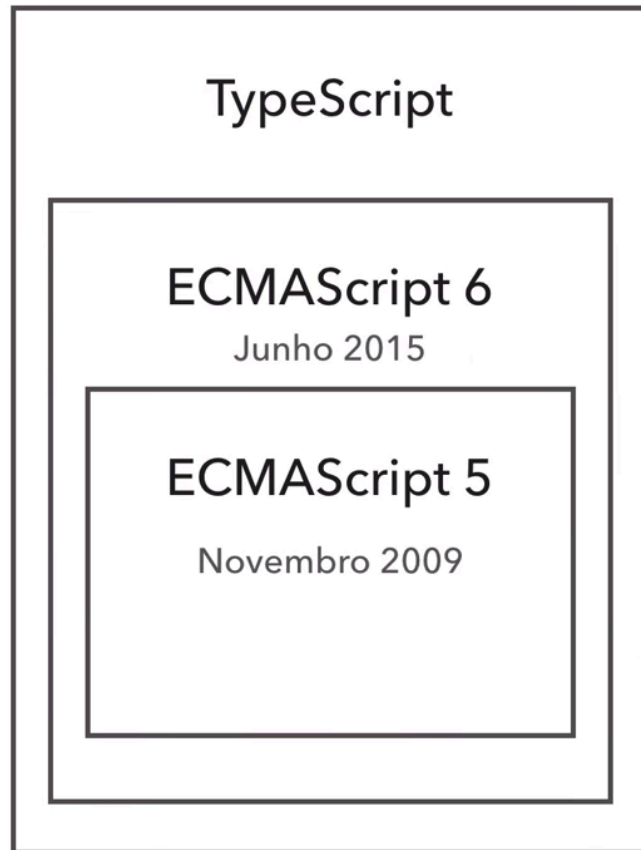
- *Versão 2 do famoso Framework AngularJS;*
- *Utiliza as tecnologias recentes do ES6;*
- *Melhoria de performance em relação às versões 1.X;*
- *Focado tanto em Web quanto Mobile;*

Angular

Além das funcionalidades do ES6, o Angular utiliza o padrão TypeScript, que define um “super conjunto” de funcionalidades para o JavaScript.

<https://www.typescriptlang.org>

TypeScript



Angular

Por exemplo, com o TypeScript é possível utilizar tipagem de dados, anotações em componentes, importação de módulos, entre outros.

Angular

Então, uma ferramenta de compilação fica encarregada de traduzir o TypeScript (.ts) em JavaScript convencional (*.js)*

Angular 2

Como o Angular 2 é uma versão escrita em ES6 e TypeScript, não é necessário aprender AngularJS 1.x. Angular 2 é um framework completamente novo.

Angular 2

Devido a estas características e pelo Angular conter a maioria das funcionalidades e componentes de um framework JS, o Ionic adotou o Angular 2 em sua plataforma para interação, rotas e componentes.

Ionic + Angular

Então, o Ionic utiliza os recursos do Angular em sua plataforma para resolver os problemas comuns relacionados ao JavaScript, Ajax e Eventos de Interface.

Quem mantém?

Engenheiros e desenvolvedores do Google e Microsoft.

Cuidado

É importante ter atenção ao pesquisar bibliotecas e plugins para Angular. Você deve pesquisar sempre por Angular 2 e não AngularJS (os plugins não serão compatíveis).

Angular e Ionic

Como o Angular é utilizado como motor de funcionamento do Ionic, vamos aprender os detalhes específicos de Angular juntamente com Ionic.

Angular e Ionic

Ou seja, vamos aprender Angular e Ionic ao mesmo tempo para evitar essa dependência. O importante é compreender os conceitos e componentes que formam o Angular.

Angular e Ionic RC

Ambos os frameworks estão em fase de Release Candidate. Em breve devem lançar a versão final. Portanto, pequenas mudanças ainda podem ocorrer.

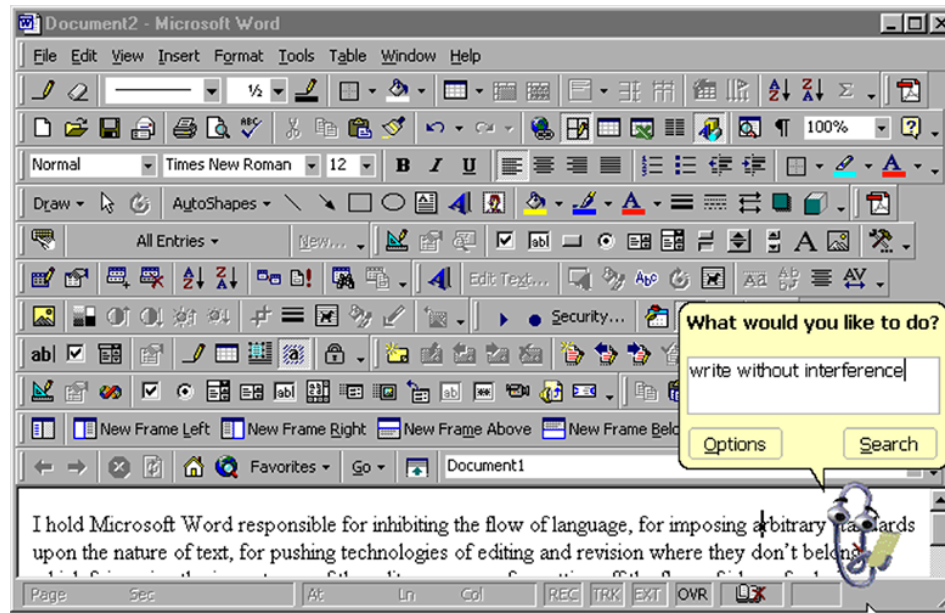
Interface para Mobile

Projeto de Interface

Além dos conceitos clássicos de IHC sobre usabilidade, corretude, legibilidade, entre outros, precisamos ter alguns cuidados ao criar interfaces para dispositivos móveis.

Atenção do Usuário

Em dispositivos móveis é importante manter o menor número de elementos de interface quanto possível.



Atenção do Usuário

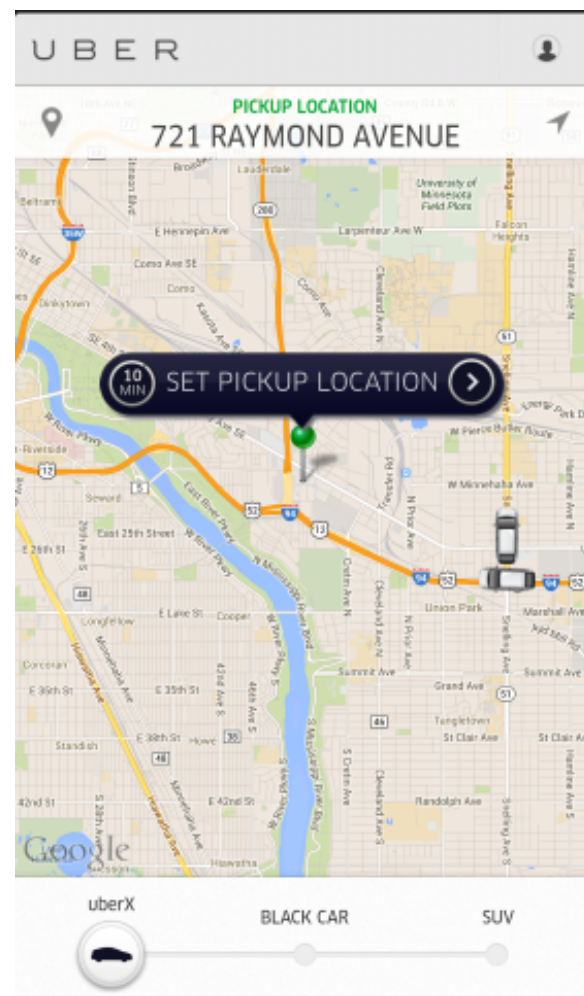
Isto é necessário, pois o tamanho da tela é reduzido, assim, cada elemento (botão, texto, imagem) é mais um elemento para prender a atenção do usuário e tirar o foco das ações que realmente importam.

Dica

Mantenha uma única ação primária por tela. Ou seja, uma única ação que executa uma atividade principal nesta tela.

Exemplo: Uber

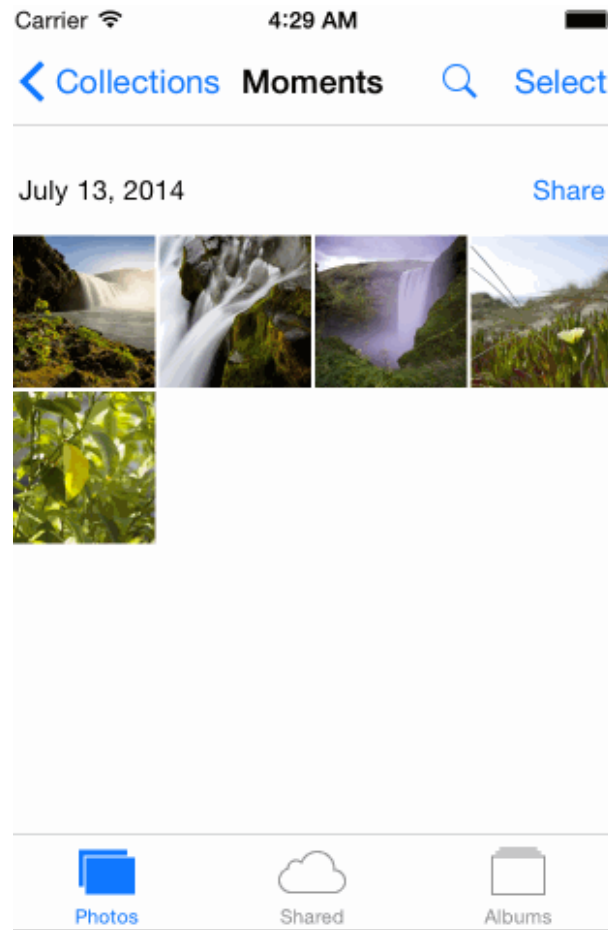
Para solicitar um carro o usuário não precisa informar seu endereço atual, o aplicativo fornece a localização e há apenas um botão (ação) para o usuário executar.



Navegação Evidente

É importante manter uma navegação simples e completamente intuitiva para o usuário. Ou seja, o usuário deve sentir que a navegação é “natural”.

Exemplo: Tabs no iOS



Navegação Evidente

- *Use metáforas e labels consistentes e contextuais;*
- *Seja consistente no padrão de navegação escolhido;*
- *Não esconda a navegação do usuário;*
- *Não mude a localização da navegação. Use um padrão.*

Design Centrado ao Toque

A maioria dos dispositivos móveis hoje é touchscreen. Isto significa que os elementos de interface responsáveis por disparar eventos a partir da interação com o usuário devem ser amigáveis ao toque.

Design Centrado ao Toque



Don't.



Do.

Design Centrado ao Toque

Além do tamanho também é importante conhecer os tipos de toque disponíveis e entender como cada um pode ser melhor usado para manter uma maior consistência entre diferentes aplicativos.

Design Centrado ao Toque

Uma boa referência é o guia Touch Gesture Reference Guide, que mostra e descreve os gestos multitoque e como utilizá-los na prática.

<http://www.lukew.com/ff/entry.asp?1071>

Legibilidade de Texto

Da mesma forma que elementos de toque, os textos devem ser legíveis ao usuário independente do tamanho da tela do dispositivo.

Legibilidade de Texto

A idéia é que o texto seja legível a uma certa distância do dispositivo que você está usando. Por isso o tamanho das fontes deve variar de acordo com a tela.

Legibilidad de Texto

Heading

Sub-Headline

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Don't.

Heading

Sub-Headline

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla eleifend.

Do.

Visibilidade

Torne os elementos de interface claramente visíveis ao usuário. Isto inclui botões, textos, imagens, ícones, etc.

Visibilidade

Heading

Sub-Headline

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend.

Don't.

Heading

Sub-Headline

Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend egestas nisl vehicula nec. Nullam varius est dui, nec accumsan lectus posuere ut. Nullam viverra purus laoreet euismod tempor.

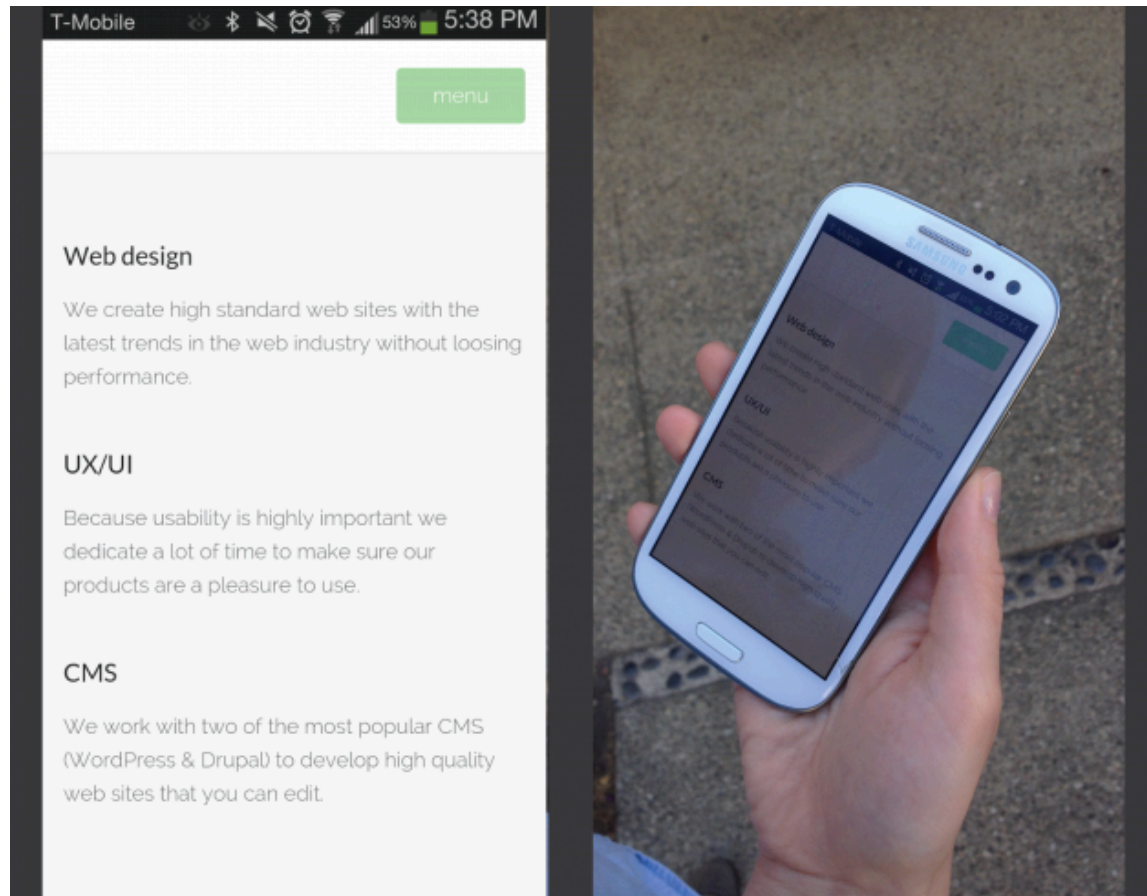
Adipiscing elit. Sed neque nisl, blandit vel ipsum eu, imperdiet blandit lectus. Morbi tristique urna ut volutpat ornare. Curabitur semper vitae urna ac tempus. Duis vehicula elit nulla, eleifend.

Do.

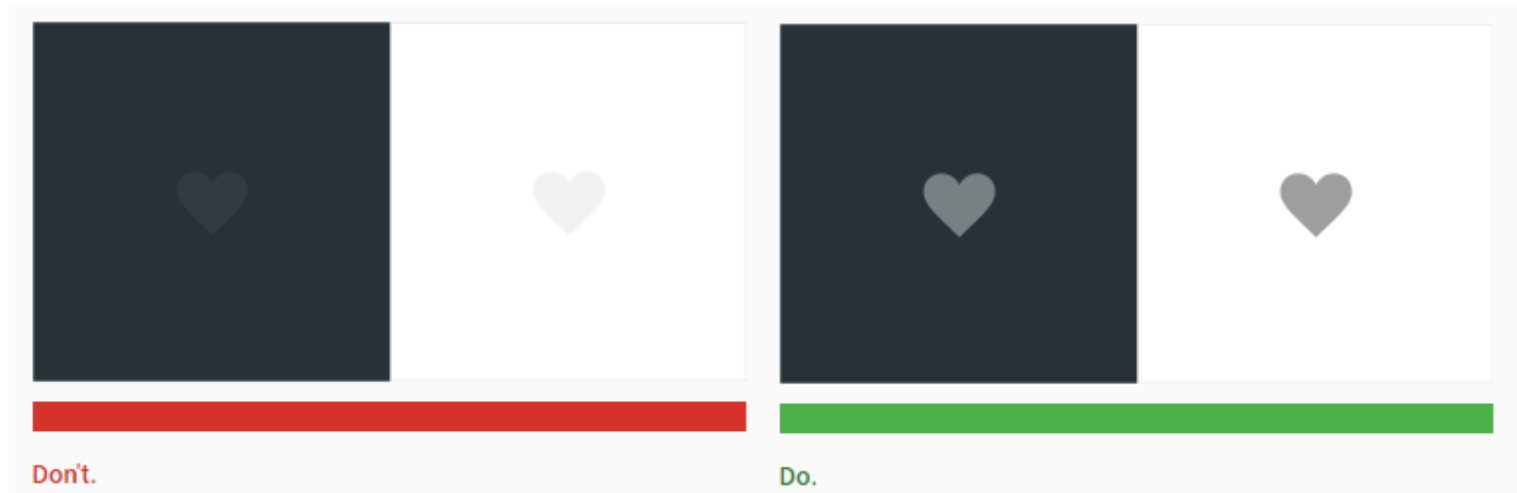
Visibilidade

Também é importante avaliar o contraste da tela para a visualização da informação. O brilho da tela pode impactar a percepção correta da informação.

Visibilidad



Visibilidade



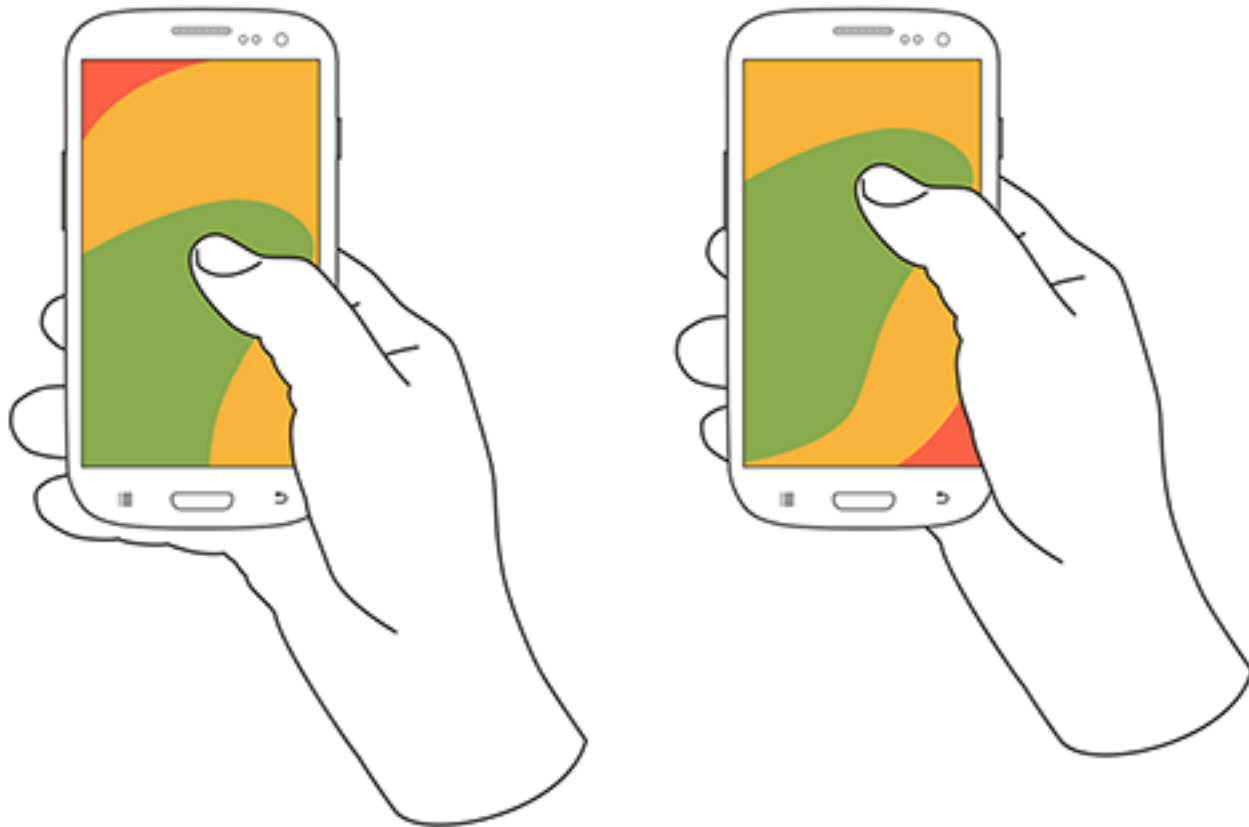
Controles ao Alcance

É importante focar no design de controles baseado nas posições mais comuns das mãos do usuário ao utilizar um dispositivo móvel.

Controles ao Alcance

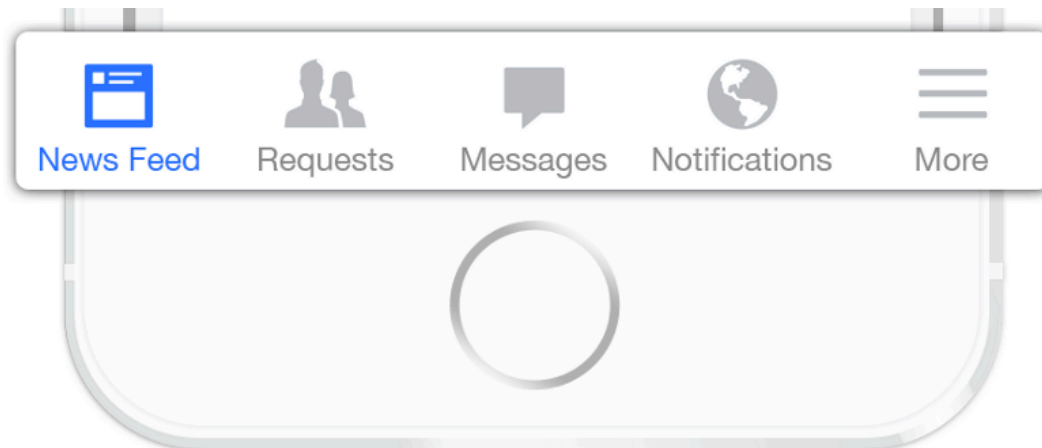
Uma pesquisa sobre estas posições das mãos durante a interação foi realizada por Steve Hooper e constatou que 49% das pessoas utilizam o celular com uma mão e usam o polegar para interação.

Controles ao Alcance

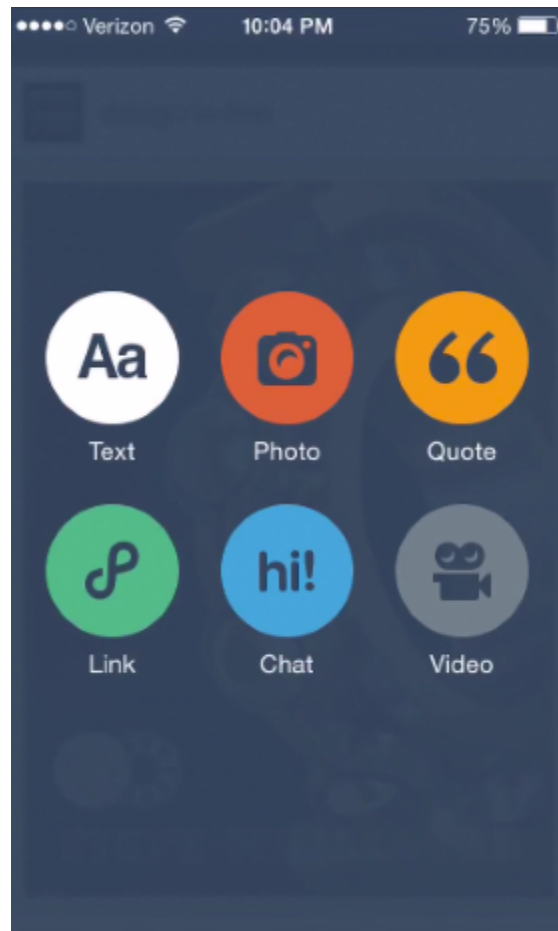


Controles ao Alcance

Neste exemplo, a área verde é mais fácil de alcançar. A área amarela é mais difícil e a vermelha é a zona que faz o usuário mudar a forma de segurar o dispositivo.

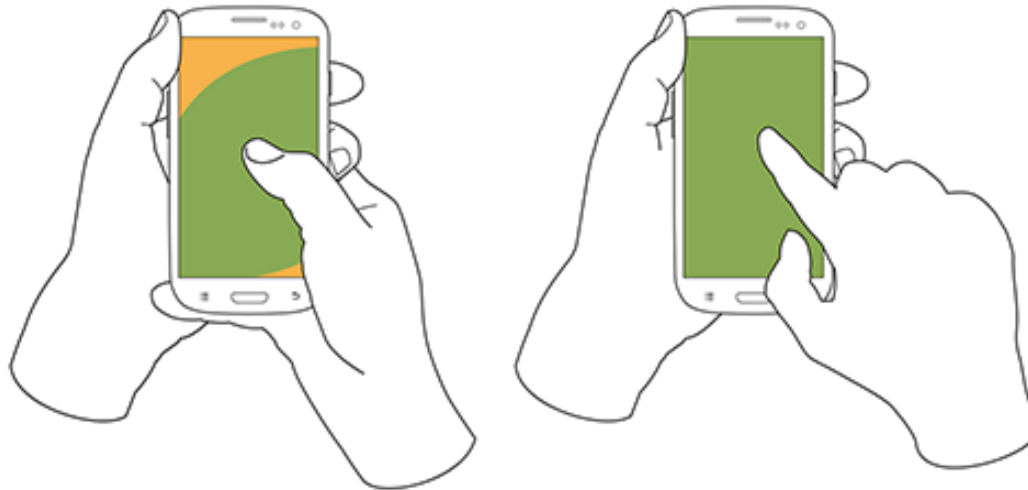


Um bom exemplo



Controles ao Alcance

Na pesquisa, 36% seguram o celular com uma mão e interagem com a outra mão, alcançando uma área maior.

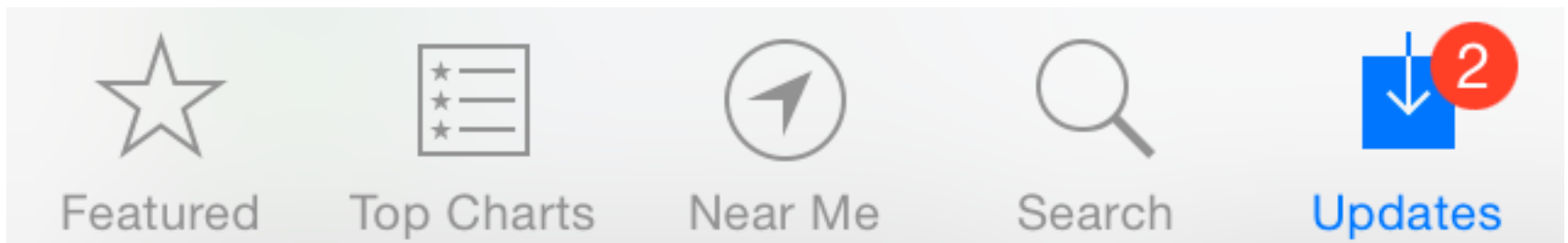


Bottom Navigation

- *Cuidado na escolha dos ícones;*
- *Selecione somente as ações importantes;*
- *Mantenha uma única cor para abas inativas;*
- *Mantenha uma cor diferente para aba ativa;*

Bottom Navigation

Você pode usar “badges” para indicar notificações ou mudanças em uma determinada aba.

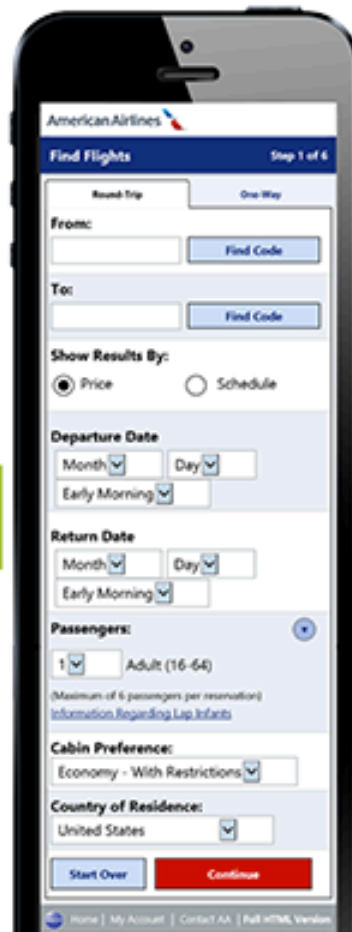


Quantidade de Input

É importante reduzir a quantidade de informações requeridas para o usuário. Ou seja, é necessário diminuir o número de campos e simplificar a forma de solicitar os dados.

Quantidade de Input: Problema

"This is going to be painful."



American Airlines
Find Flights Step 1 of 4

Round Trip One-Way

From: Find Code

To: Find Code

Show Results By:
☒ Price ☐ Schedule

Departure Date
Month Day
Early Morning

Return Date
Month Day
Early Morning

Passengers: Adult (16-64)
(Maximum of 6 passengers per reservation)
[Information Regarding Lap Infants](#)

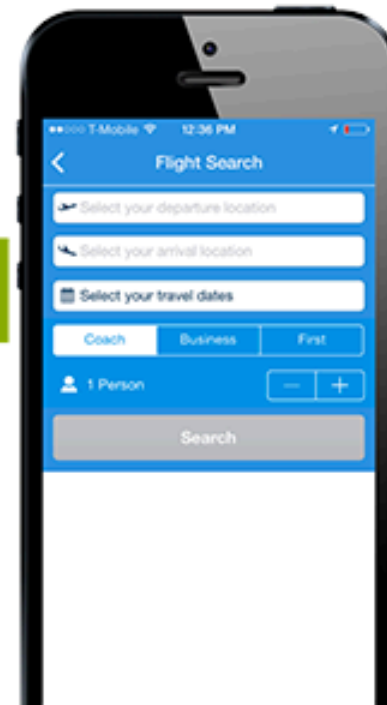
Cabin Preference:
Economy - With Restrictions

Country of Residence:
United States

Start Over Continue

Home | My Account | Contact AA | Full HTML Version

"This will be a breeze."



Flight Search

Select your departure location

Select your arrival location

Select your travel dates

Coach Business First

1 Person

Search

Formas de Input

*Você pode mudar a forma de capturar uma informação.
Por exemplo, ao invés de digitar o usuário poderia clicar
em uma opção de um grupo.*

Formas de Input

What is your spirit animal?

Lion

Tiger

Bear

Bull

Serval

Do

What is your spirit animal?

- Select -

Lion

Tiger

Bear

Bull

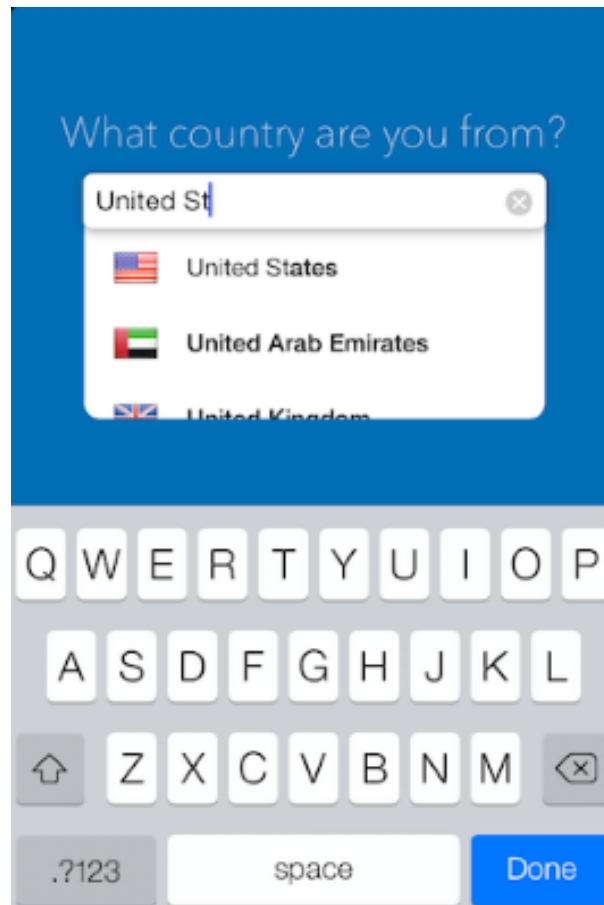
Serval

Don't

Formas de Input

Se você possui muitos elementos, você pode auxiliar o usuário a completar ou filtrar a informação.

Formas de Input



Referências para Design

- *Material Design* ([Google](#));
- *Design Guidelines IOS* ([Apple](#));



www.ionicframework.com

Ionic

É um framework open source que possibilita desenvolver aplicações com boa performance e de alta qualidade utilizando tecnologias web, tais como HTML, CSS e JS.

Ionic disponibiliza

- *Componentes Visuais de Interface;*
- *Componentes para utilizar Recursos Nativos;*
- *Interface de Linha de Comandos para auxiliar no projeto;*
- *Ícones customizados por plataforma;*
- *Forma simples de visualização: Browser ou Ionic View*

Ionic CLI

Interface de Linha de Comando que auxilia a criar projetos Ionic, testar, compilar para Android ou IOS, executar no dispositivo físico ou sincronizar na nuvem com Ionic View.

Ionic View

Aplicativo que possibilita visualizar e testar seus projetos Ionic diretamente no dispositivo físico sem a necessidade de instalação.

Ionic View

*Com o comando **ionic upload**, o CLI envia para a nuvem do Ionic na sua Conta e com isso você pode compartilhar seu aplicativo com outras pessoas.*

Ionic View

Vamos utilizar este conceito para cada aluno compartilhar os trabalhos com o email do professor. Assim será fácil acompanhar e testar os aplicativos.

Glossário

Se você ainda está confuso com tantos termos técnicos relacionados ao Ionic, você pode tirar as dúvidas diretamente no [site](#).

Referências

- <http://ionicframework.com/docs/v2/resources/books-and-courses/>
- <http://ionicframework.com/docs/v2/resources/developer-tips/>
- <http://ionicframework.com/docs/v2/resources/typescript/>
- http://ionicframework.com/docs/v2/resources/editors_and_ides/
- <http://ionicframework.com/docs/v2/resources/using-npm/>
- <http://ionicframework.com/docs/v2/resources/platform-setup/mac-setup.html>
- <http://ionicframework.com/docs/v2/resources/platform-setup/windows-setup.html>

Referências

- Conferir o [Blog](#) do Ionic;
- Vídeos disponíveis no [Site](#);
- Documentação ([Versão 2](#));

Vamos ver os Componentes?

<http://ionicframework.com/docs/v2/components/>

Ambiente

Arquivo em PDF disponível no Moodle

Tarefas

1. *Revisar os conceitos de HTML, CSS e JavaScript;*
2. *Testar os exemplos apresentados;*
3. *Instalar os softwares e pacotes necessários;*
4. *Criar um projeto Ionic de exemplo e rodar no navegador;*
5. *Criar uma conta no Ionic;*
6. *Instalar no seu Android o **Ionic View** (Play Store);*