7 Versions of One Web Application

Yakov Fain, Farata Systems

FARATA THE EXPERT CONSULTANCY

We build applications. Every app is unique.

We create it. You own it.

Quick Show of Hands

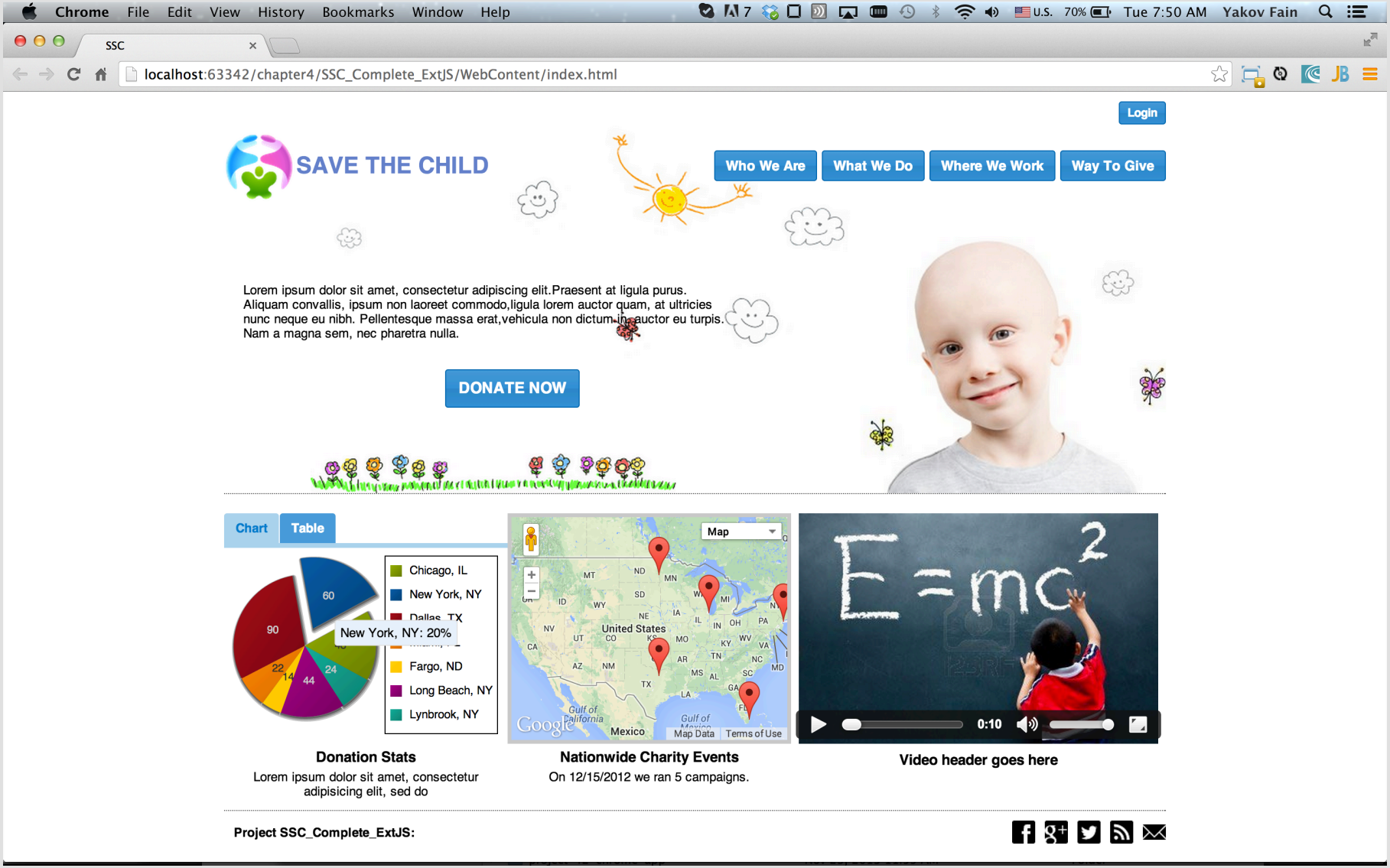
Raise you hand if you know…

Show of Hands

Actually, don't…

I'm not going to change my presentation anyway.

We'll Review 7 Versions of the UI of this Single Page Application

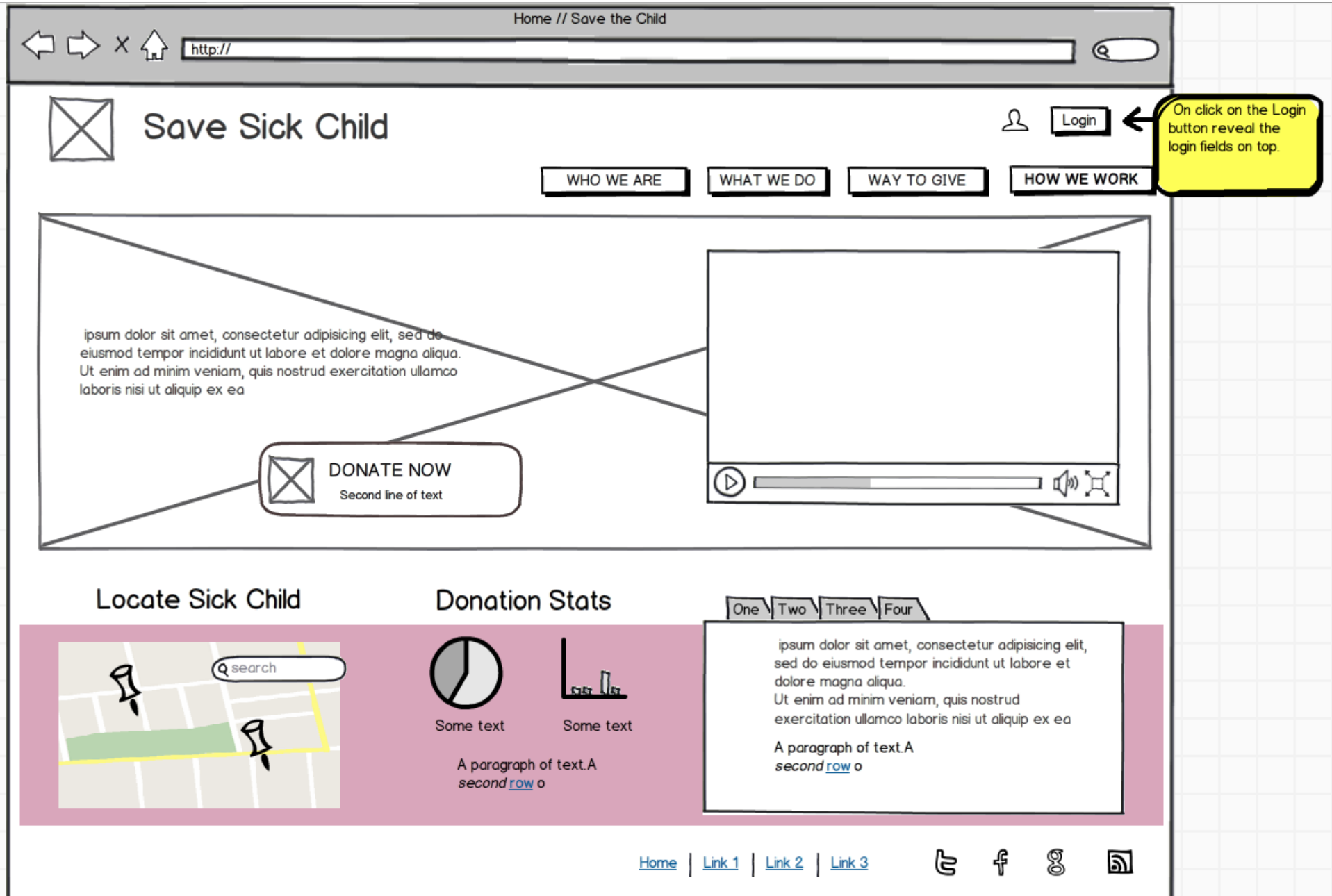


SPA: One Web page, AJAX calls bring the data as needed, CSS hides/shows HTML elements

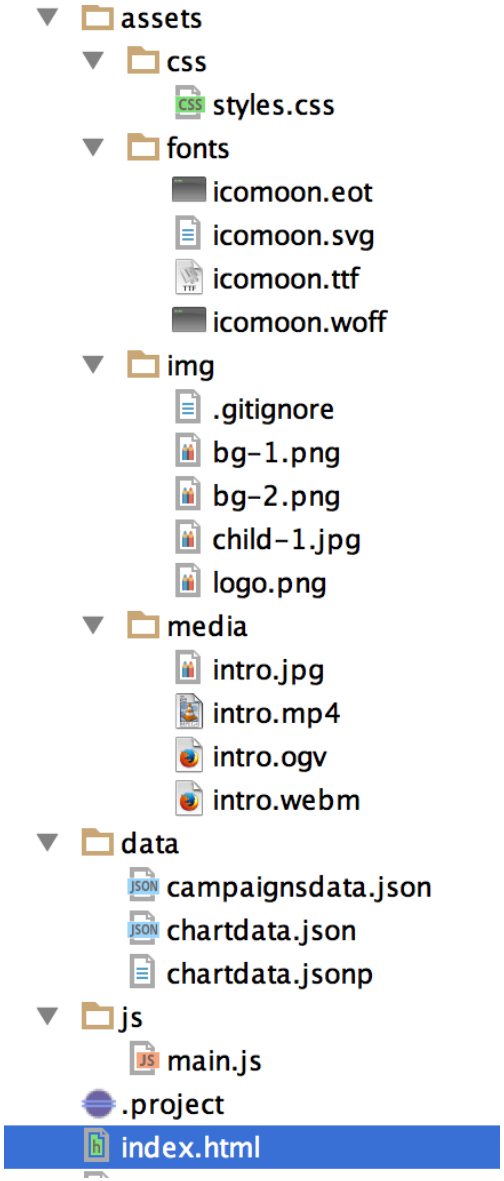
… and the 7 versions are…

1. HTML/AJAX
2. HTML + Responsive Web Design
3. With jQuery library
4. With Ext JS framework
5. Modularizing HTML5
6. With jQuery Mobile
7. With Sencha Touch

Wireframing with Balsamiq Mockups

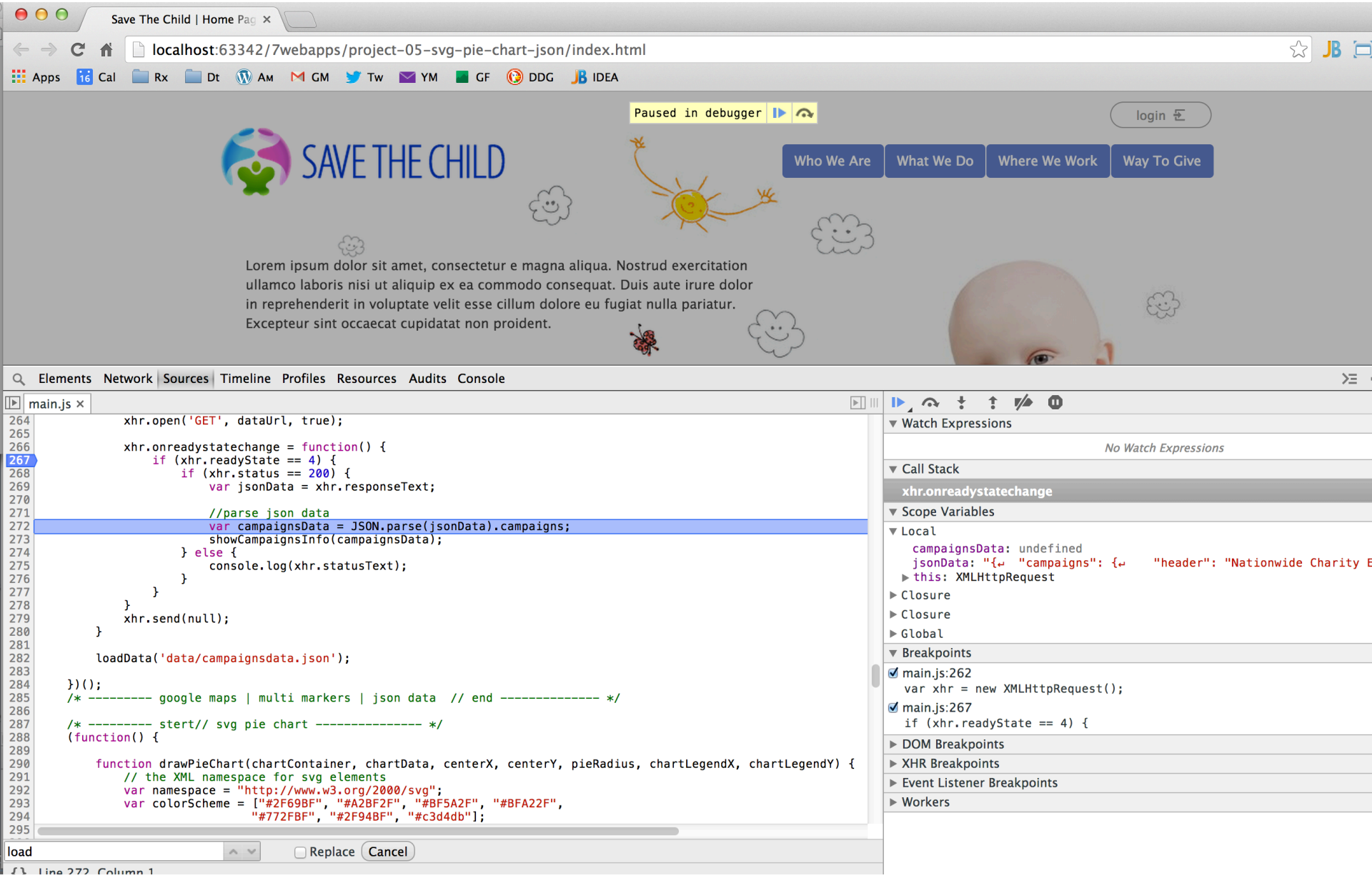


V1: HTML, JS, CSS, AJAX, JSON




```
// Loading data  with AJAX and parsing JSON

function loadData(dataUrl) {
        var xhr = new XMLHttpRequest();

        xhr.open('GET', dataUrl, true);

        xhr.onreadystatechange = function() {
                if (xhr.readyState == 4) {
                        if (xhr.status == 200) {
                                var jsonData = xhr.responseText;

                                //parse json data
                                var campaignsData = JSON.parse(jsonData).campaigns;
                                showCampaignsInfo(campaignsData);
                        } else {
                                console.log(xhr.statusText);
                        }
                }
        }
        xhr.send(null);
}
```

Demo: Debugging in Chrome

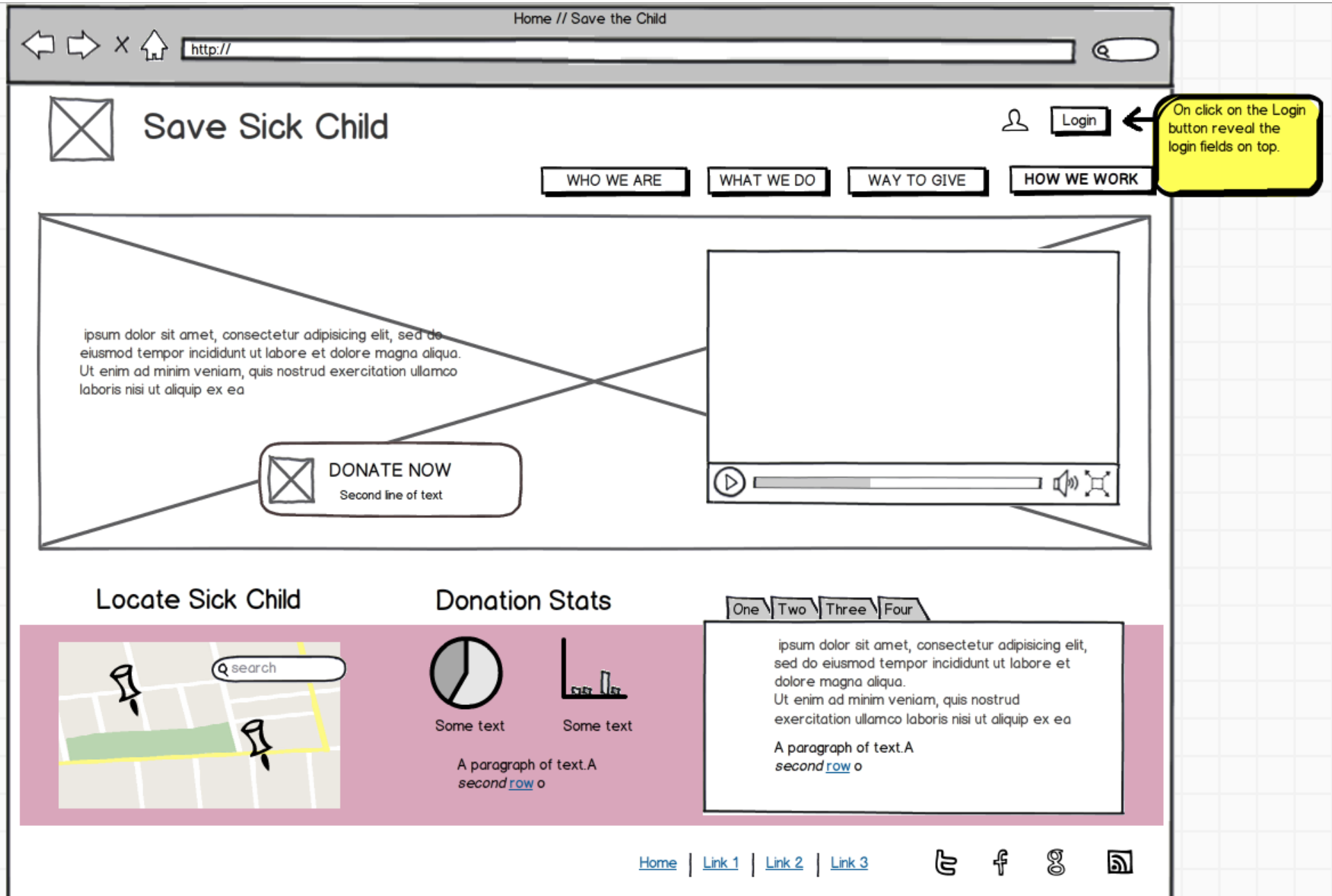


Oops…A Smaller Screen

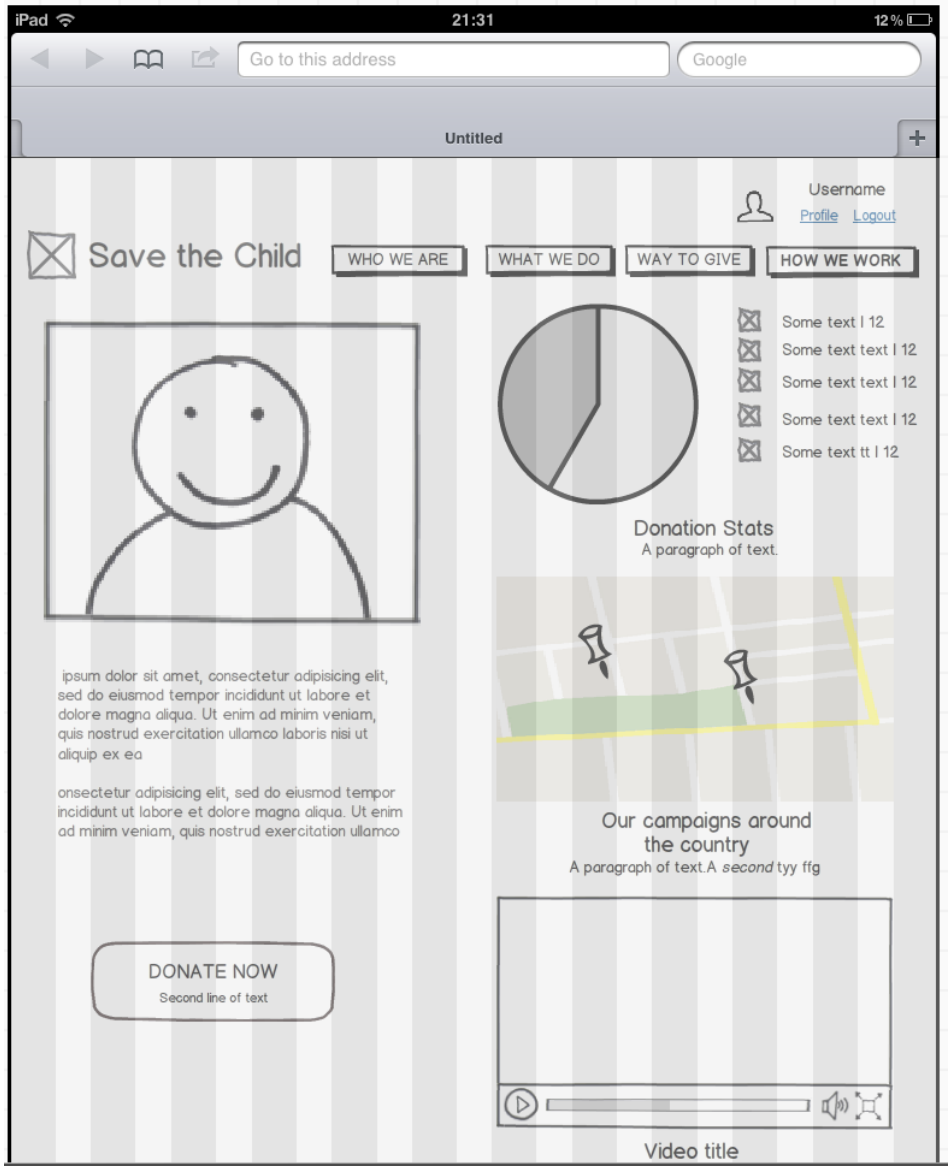
V2: Responsive Web Design(RWD)

- Separate versions for desktop and mobile?
- How many versions of the UI to create?
- Can we have a single HTML version for all devices?
- CSS *Media Queries* – layouts based on screen width
- Setting CSS *Breakpoints*
- Pros and Cons of RWD

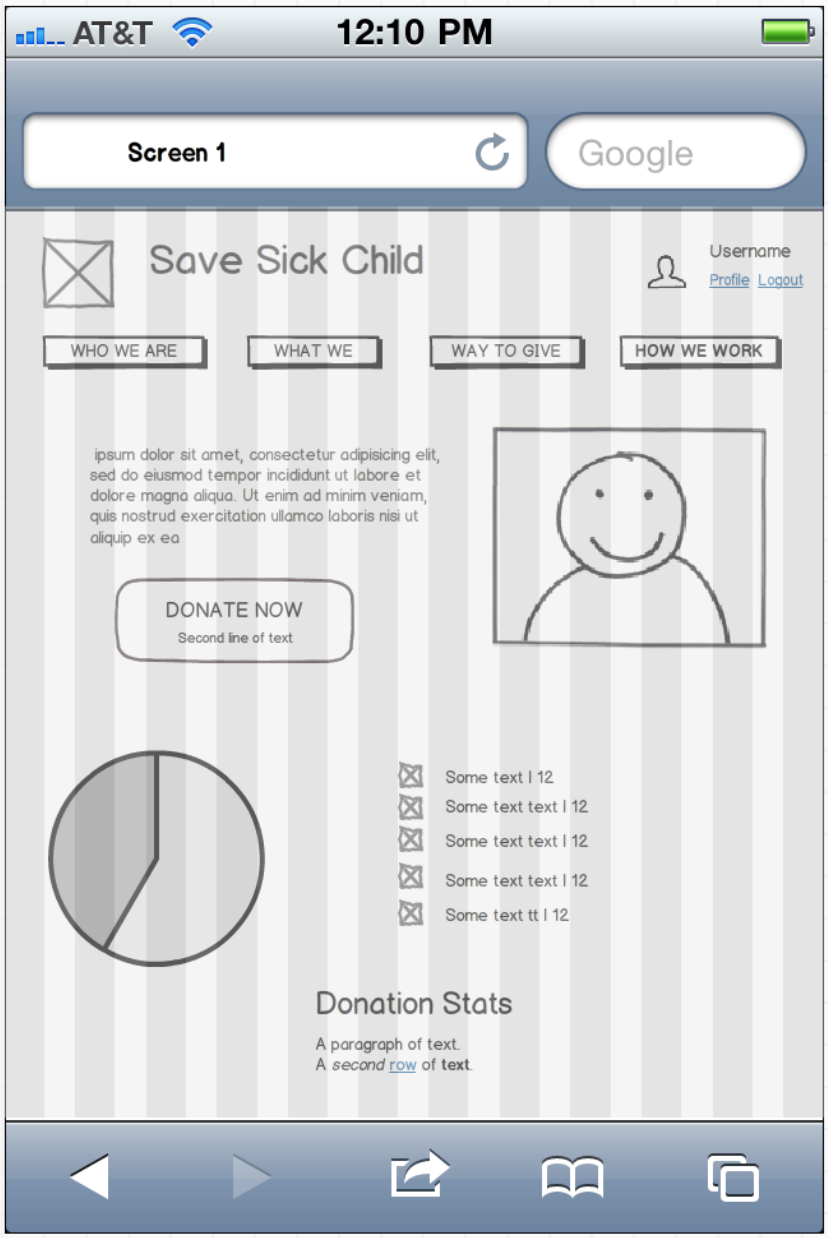
Remember the wireframe for desktops?

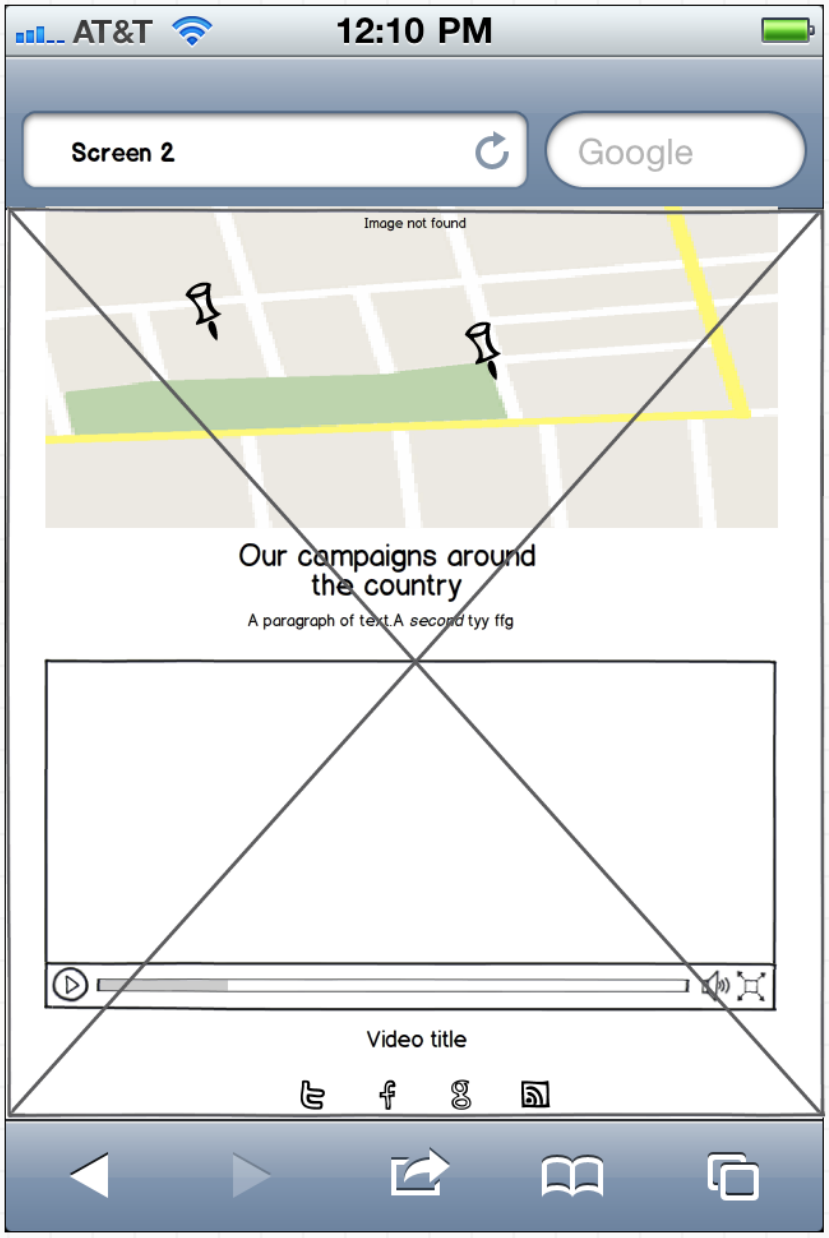


Wireframing for a table in portrait

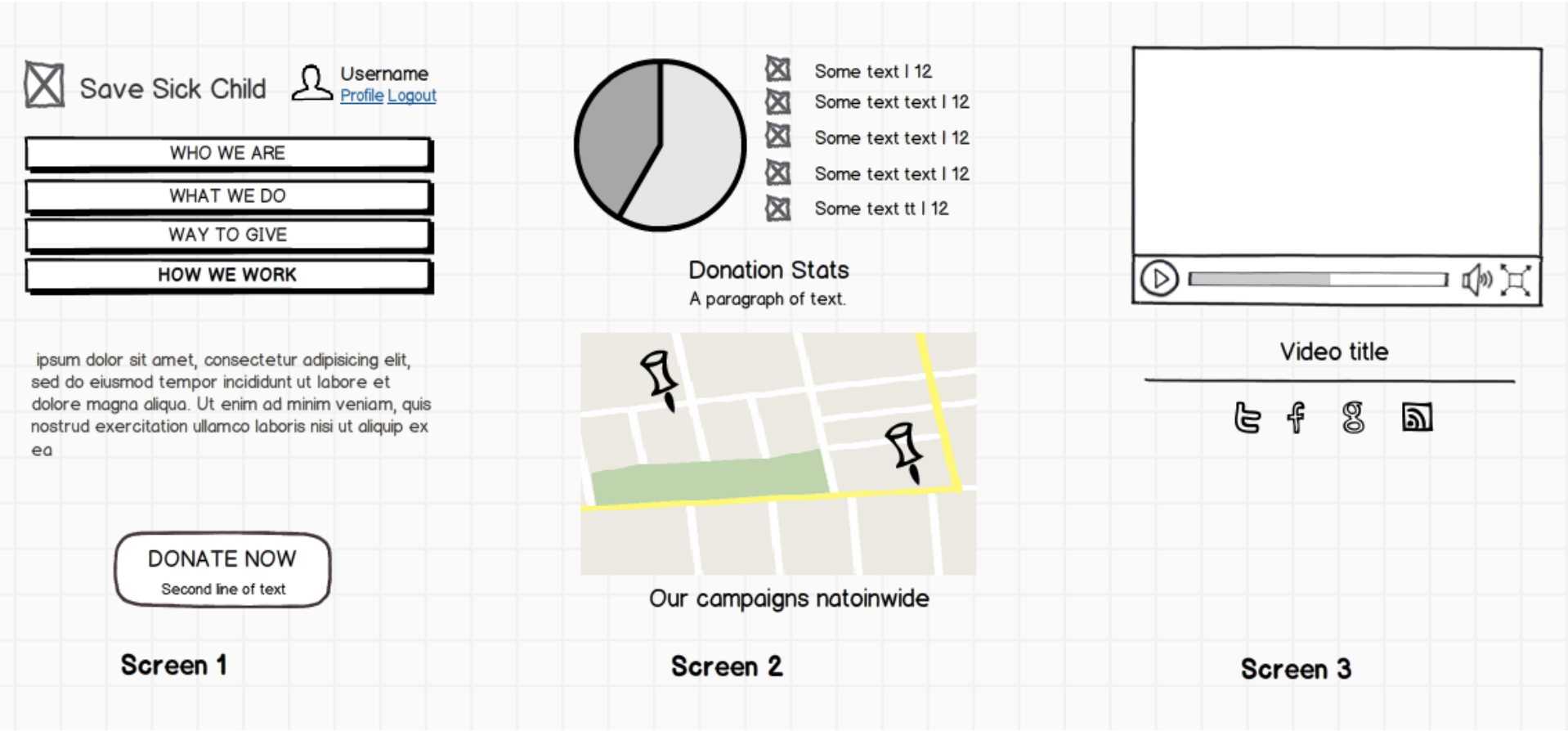


Wireframing for large phones





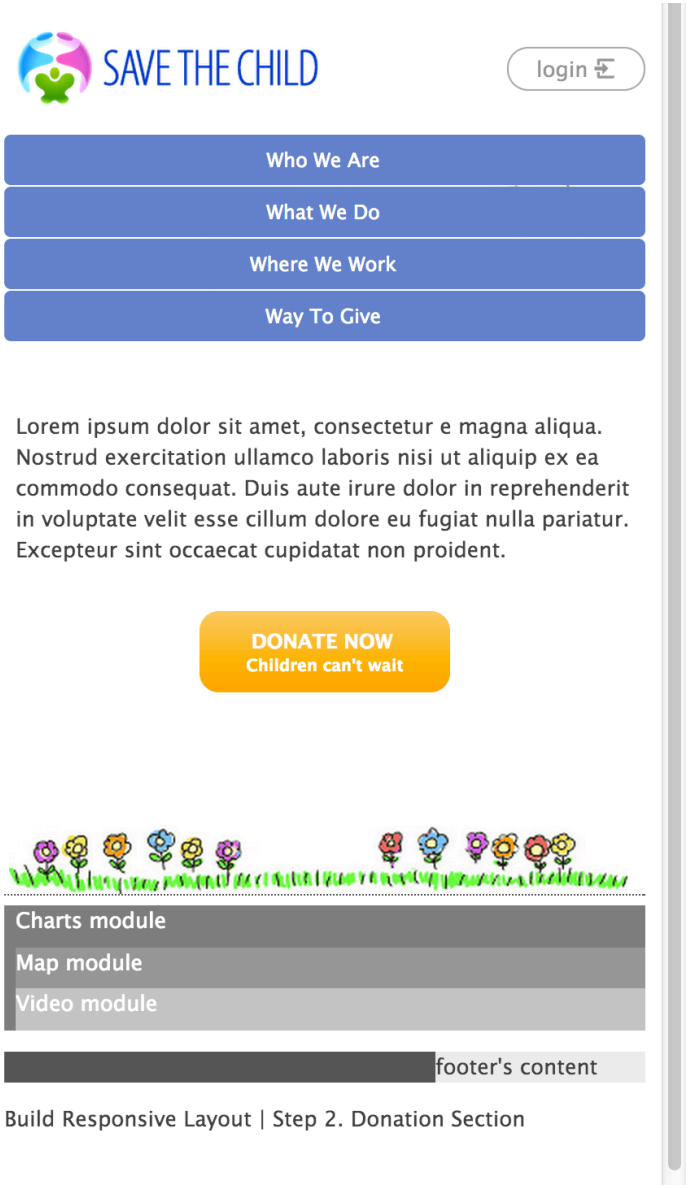
Wireframing for smaller phones



These are the wireframes for 3 phone screens

V2: Demo

1. Basic Media Queries
2. Responsive Header
3. Responsive Donation
4. Responsive Final



RWD Pros and Cons

- RWD is good for publishing info. Mobile frameworks can be a better choice for interactive apps
- RWD allows to have a single app code base
- Mobile versions of an app may need limited functionality and specific navigation
- RWD means larger traffic (heavy CSS) – no good for slower connections
- Mobile frameworks offer more native look and feel of the UI

V3: With jQuery Library

- Learning is easy for designers – mostly HTML.
- Still, you have to design and style your HTML
- 40 – 50% of top Web sites use it (see bultwith.com)
- Light-weight addition to your app – 33Kb gzipped, minified
- Shorter than in JS syntax for DOM Browsing
- $() – can pass it a String, can pass it a function
- There are thousands plugins in jQuery Plugin Registry

DOM Querying & Function Chaining

```
function showLoginForm() {
    $('#login-link, #login-form, #login-submit').toggle();
}

$('#login-link').on('click', showLoginForm);

function showAuthorizedSection() {
    $('#authorized, #login-form, #login-submit').toggle();
}

function logIn() {
    var userNameValue = $('#username').val();
    var userNameValueLength = userNameValue.length;
    var userPasswordValue = $('#password').val();
    var userPasswordLength = userPasswordValue.length;
```

An AJAX call in jQuery

```
function loadData(dataUrl) {
    $.ajax({
        url : dataUrl,
        type : 'GET',
        dataType : 'json'
    }).done(function(data) {

        showCampaignsInfo(data.campaigns);

    }).fail(function(jqXHR, textStatus) {
        console.log('Error status code:' + jqXHR.status);
        if (textStatus === 'parsererror') {
            console.log('Requested JSON parse was failed.');
        } else if (textStatus === 'abort') {
            console.log('Ajax request was aborted.');
        }
    });
}

loadData('data/campaignsdata.json');
```

The shortcut methods: $.get(), $.post(), $.getJSON()

V3: Demo with jQuery

V4: With Sencha Ext JS Framework

- Rich library of enterprise-grade components, e.g. grids, charts
- Steep learning curve – has no HTML, but new JS-based syntax
- Cool code generator Sencha CMD
- MVC support
- The “weight” of the app substantially increases
- If you decided to go with Ext JS, there is no easy way out
- Doesn’t support Responsive Web Design
- There’s a pier framework for mobile app: Sencha Touch

Ext JS: index.html and app.js

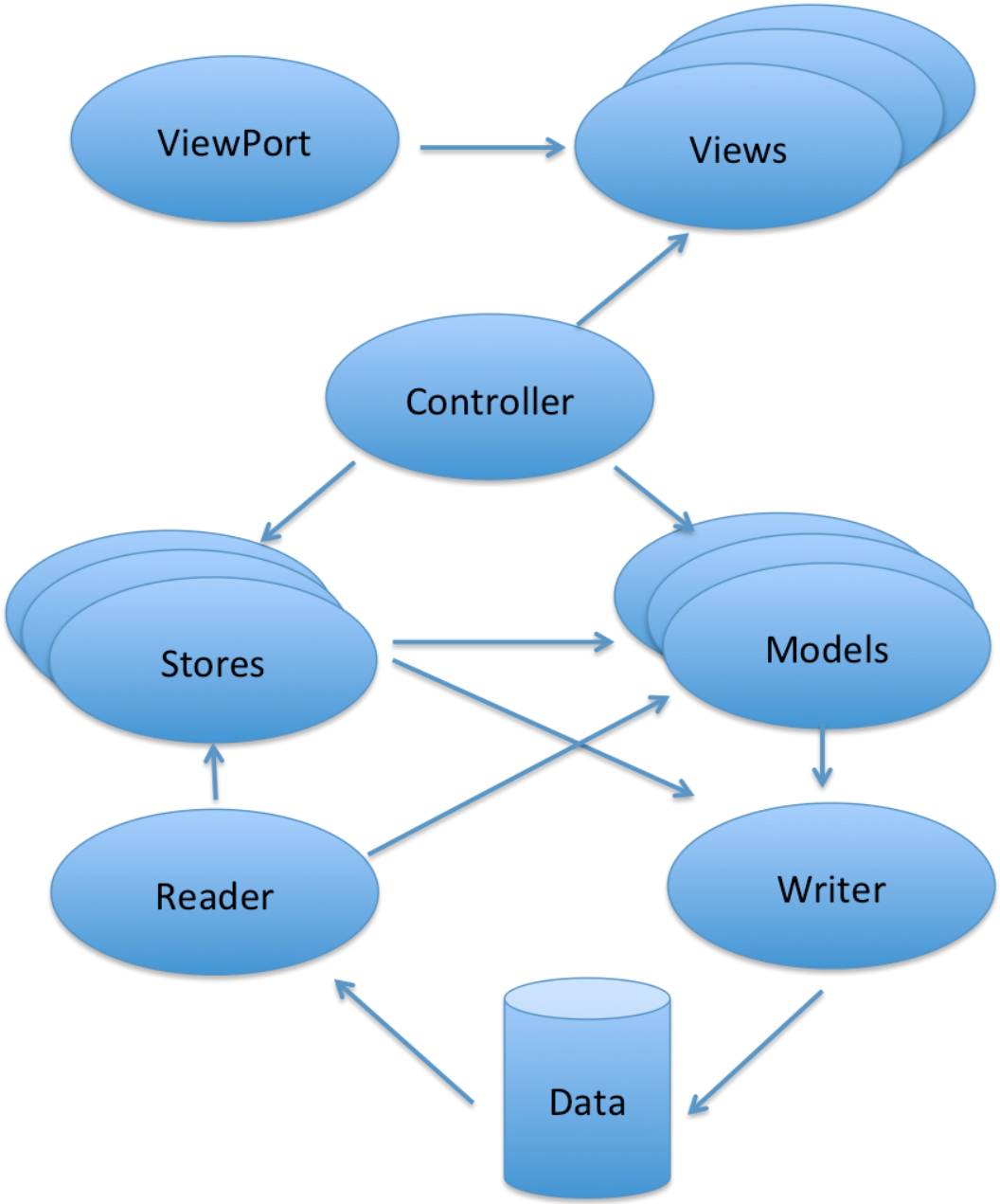
```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="UTF-8">
    <title>SSC</title>

        <link rel="stylesheet" href="bootstrap.css">
        <script src="ext/ext-dev.js"></script>
        <script src="bootstrap.js"></script>
      <script src="app/app.js"></script>
</head>
<body></body>
</html>
```

Ext JS: index.html and app.js

```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="UTF-8">
    <title>SSC</title>

        <link rel="stylesheet" href="bootstrap.css">
        <script src="ext/ext-dev.js"></script>
        <script src="bootstrap.js"></script>
    <script src="app/app.js"></script>
</head>
<body></body>
</html>
```

```
Ext.application({
    name: 'SSC',

    views: [
        'CampaignsMap',
        'DonateForm',
        'DonorsPanel',
        'Header',
        'LoginBox',
        'VideoPanel',
        'Viewport'
    ],

    stores: [
        'Campaigns',
        'Donors'
    ],

    controllers: [
        'Donate'
    ],

    autoCreateViewport: true
});
```

Ext JS MVC



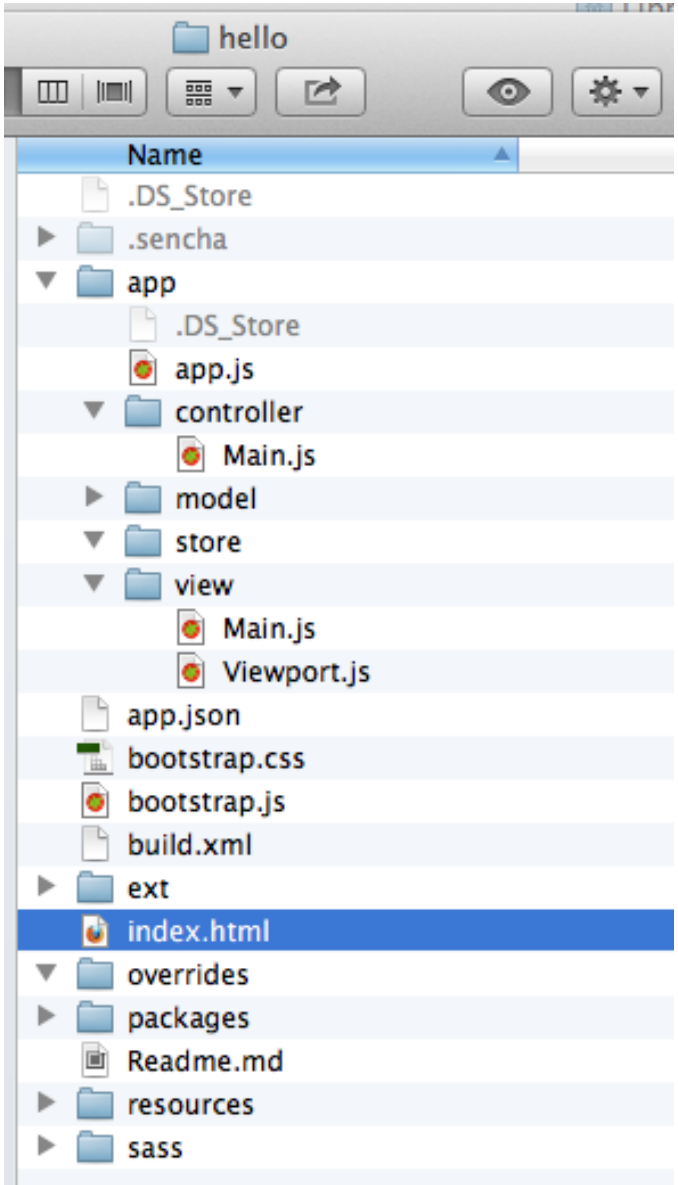
V

C

M

Generating a project with Sencha CMD

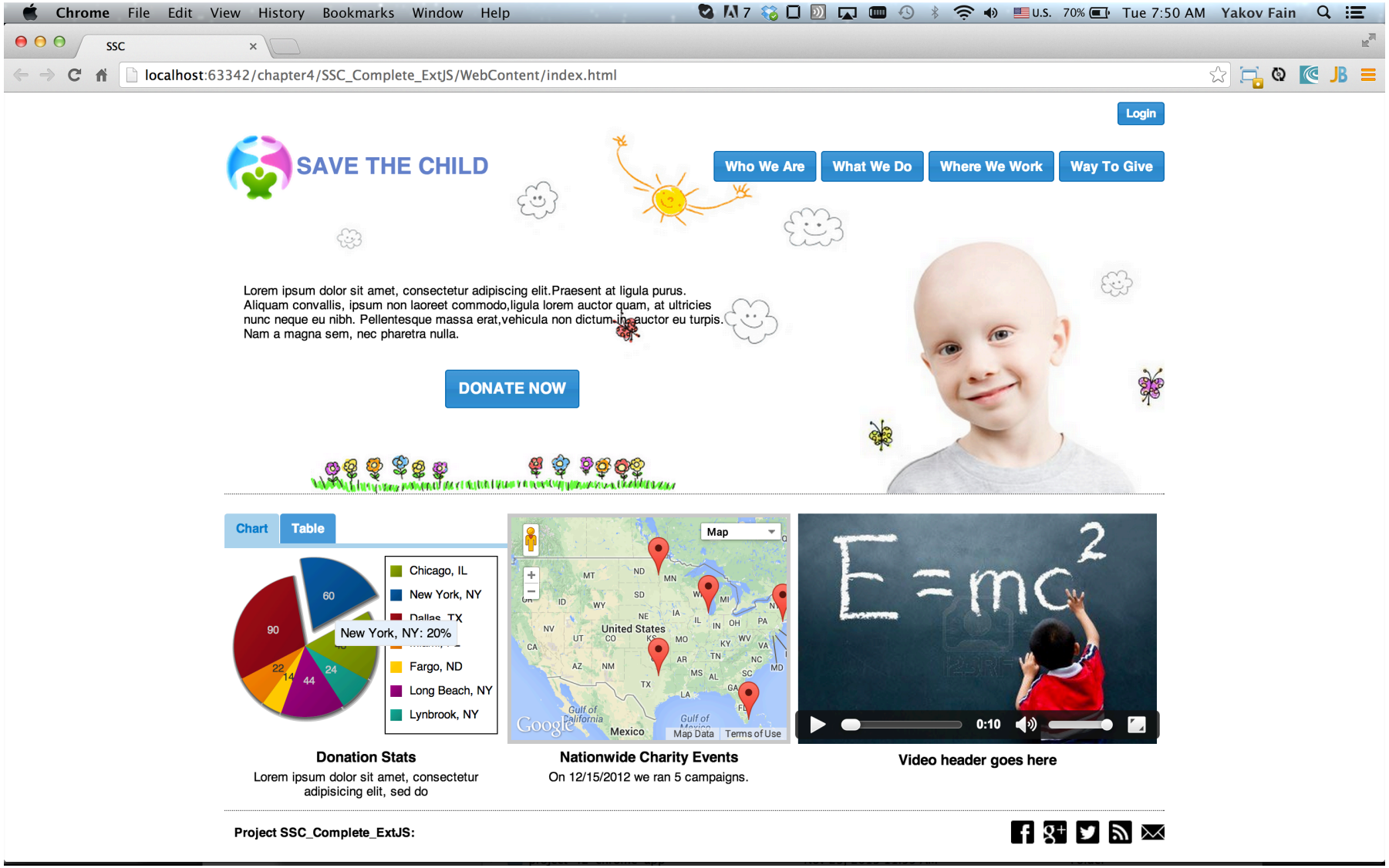
sencha -sdk /Library/ext-4.2 generate app HelloWorld /Users/yfain11/hello



The View Fragment: DonateForm.js

```
Ext.define('SSC.view.DonateForm', {
  extend: 'Ext.form.Panel',
  xtype: 'donateform',
  requires: [
    'Ext.form.RadioGroup',
    'Ext.form.field.*',
    'Ext.form.Label'
  ],

  layout: {
    type: 'hbox',
    align: 'stretch'
  },

  bodyStyle: {
    backgroundColor: 'transparent'
  },

  defaults: {
    margin: '0 50 0 0'
  },

  items:[{
    xtype: 'container',
    layout: 'vbox',
```

V4: Demo with Ext JS



V5: Modularizing UI

- Large apps should be modularized to avoid loading all code at once.
- Multiple <script> tags may depend on each other and have to be loaded in certain order.
- Need to be able to specify dependencies between the modules.

  Need to avoid polluting global scope and name conflicts.

  Manually writing Modules doesn't solve these issues.
- Today: CommonJS and Async Module Definition (AMD) specs
- Tomorrow: ECMAScript 6 spec (a.k.a. Harmony) defines modules.

One way of implementing Module Pattern

```
(function(global) {
    "use strict";
    var module = {};

    var privateVariable = 42;
    var privateLogin = function(userNameValue, userPasswordValue) {
        if (userNameValue === "admin" && userPasswordValue === "secret") {
            return privateVariable;
        }
    };

    module.myConstant = 1984;
    module.login = function(userNameValue, userPasswordValue) {
        privateLogin(userNameValue, userPasswordValue);
        console.log("rest of login implementation is omitted");
    };

    module.logout = function() {
        console.log("logout implementation omitted");
    };

    global.loginModule = module;
})(this);
```

Passing inside the module a reference to the global window object

CommonJS is an effort to standardize JS APIs.

CommonJS Modules defines 3 variables for modules:

- requires
- exports
- module

Node.js framework implements CommonJS Modules spec and provides these global variables.

Code Sample With CommonJS

```
var loginModule = {};
var privateVariable = 42;

var ldapLogin = require("login/ldap");
var otherImportantDep = require("modules/util/strings");

var privateLogin = function(userNameValue, userPasswordValue) {
    if (userNameValue === "admin" && userPasswordValue === "secret") {
        ldapLogin.login(userNameValue, userPasswordValue);
        return privateVariable;
    }
};

loginModule.myConstant = 1984;
loginModule.login = function(userNameValue, userPasswordValue) {
    privateLogin(userNameValue, userPasswordValue);
    console.log("login implementation omitted");
};

loginModule.logout = function() {
    console.log("logout implementation omitted");
};

exports.login = loginModule;
// or
module.exports = loginModule;

loginModule.printMetadata = function(){
    console.log(module.id);
    console.log(module.uri);
};
```

CommonJS Pros and Cons

Pros:

- Simple API

Cons:

- Mainly for the server-side JavaScript. Web browsers don't have require, export, and module variables.
- The require method is synchronous.
- CommonJS API is suitable for loading JS files, but can't load CSS and HTML.

Asynchronous Module Definition (AMD)

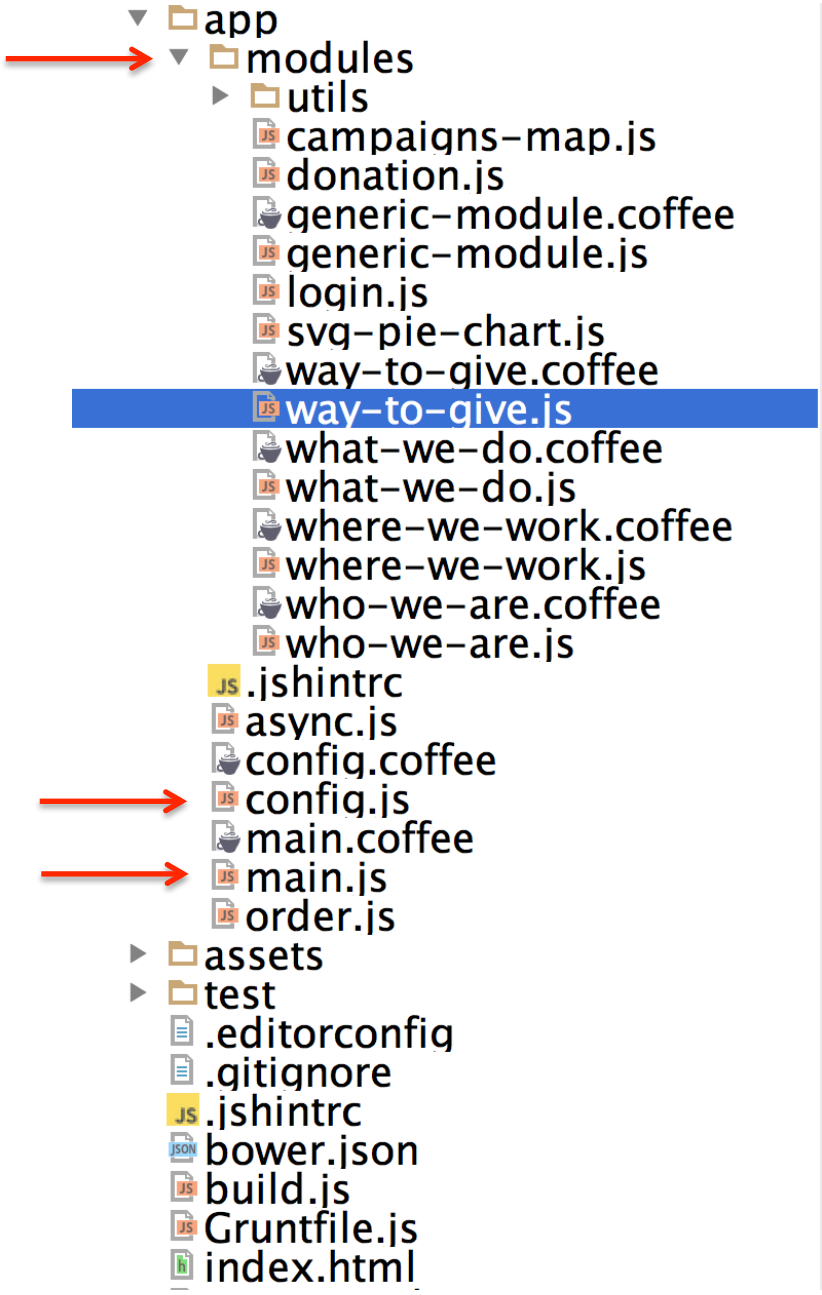
AMD is a proposal for async loading of both the module and its dependencies *in Web browsers*.

You provide define and require functions:

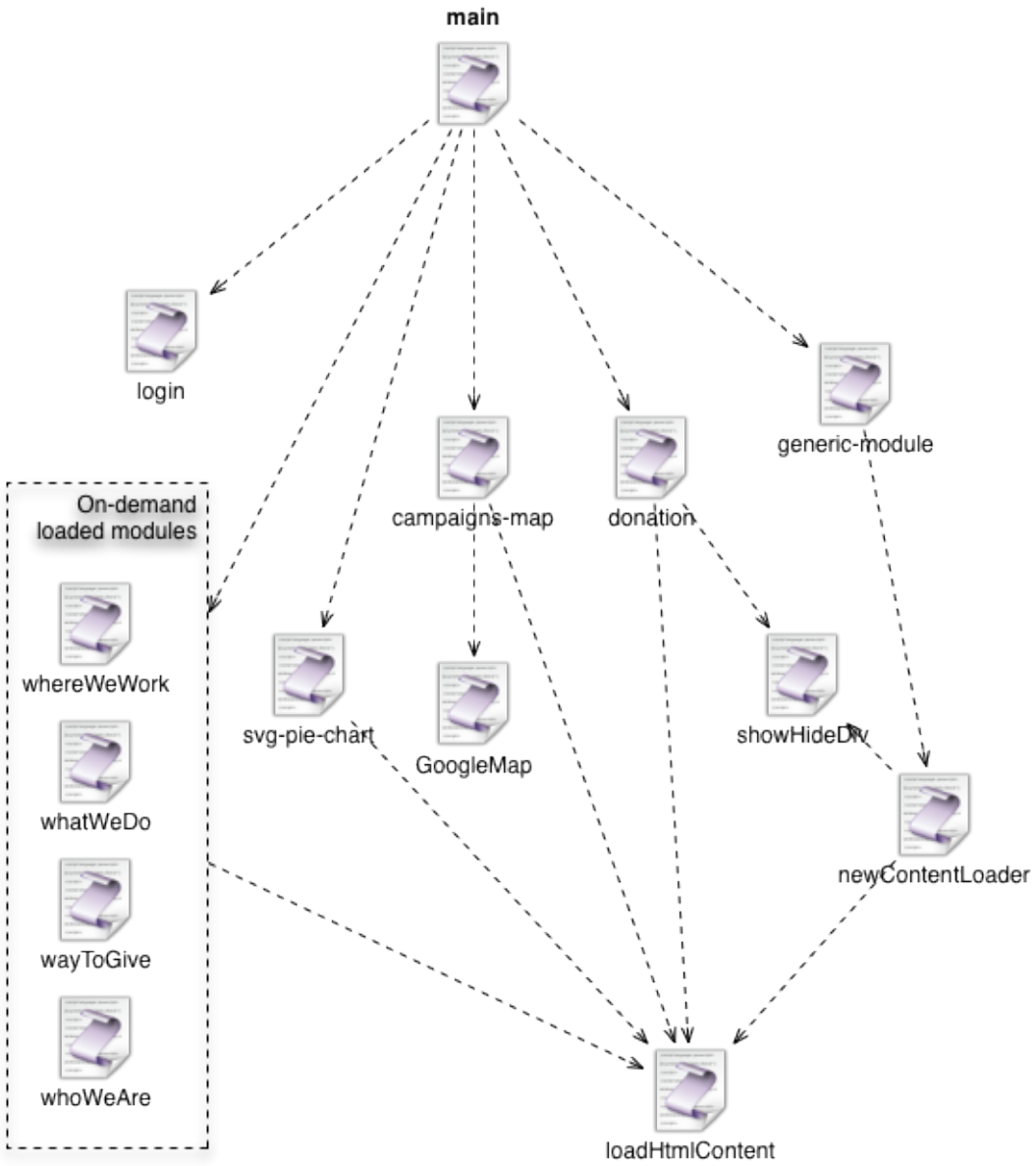
```
define(
    module_id,             // optional
    [dependencies],
    function (){
     // This function runs once when the module and its dependencies are loaded
    }
);


require(["main"], function() {
    console.log("The module main is loaded");
});
```

The function define defines the module and returns it once it's needed.
The require executes the given function checking that the dependencies were loaded.

Save The Child Modularized With RequireJS



V5 Demo: With RequireJS



config.js

```
require.config({
  paths: {
    'login': 'modules/login',
    'donation': 'modules/donation',
    'svg-pie-chart': 'modules/svg-pie-chart',
    'campaigns-map': 'modules/campaigns-map',
    'showHideDiv': 'modules/utils/show-hide-div',
    'loadHtmlContent': 'modules/utils/load-html-content',
    'newContentLoader': 'modules/utils/new-content-loader',
    'who-we-are': 'modules/who-we-are',
    'what-we-do': 'modules/what-we-do',
    'where-we-work': 'modules/where-we-work',
    'way-to-give': 'modules/way-to-give',
    'bower_components': "../bower_components",
    'jquery': '../bower_components/jquery/jquery',
    'main': 'main',
    'GoogleMap': '../bower_components/requirejs-google-maps/dist/GoogleMap'
  }
});


  require(["main"], function() {});
```

main.js (fragment)

```
define(['order!login', 'order!donation', 'order!campaigns-map', 'order!svg-pie-chart', 'modules/generic-module'],
  function() {
   return (function() {
    var initModule, lazyLoadingEventHandlerFactory, module, modulesConfig,
        way_to_give, what_we_do, where_we_work, who_we_are, _i, _len;

    way_to_give = document.getElementById("way-to-give");
    what_we_do = document.getElementById("what-we-do");
    who_we_are = document.getElementById("who-we-are");
    where_we_work = document.getElementById("where-we-work");
    modulesConfig = [
      {
        moduleId: "whoWeAre",
        button: who_we_are,
        containerId: "who-we-are-container",
        htmlContentUrl: "assets/html-includes/who-we-are.html"
      }, {
        moduleId: "whatWeDo",
        button: what_we_do,
        containerId: "what-we-do-container",
        htmlContentUrl: "assets/html-includes/what-we-do.html"
      }, {
        moduleId: "whereWeWork",
        button: where_we_work,
        containerId: "where-we-work-container",
        htmlContentUrl: "assets/html-includes/where-we-work.html"
      }, {
        moduleId: "wayToGive",
        button: way_to_give,
        containerId: "way-to-give-container",
        htmlContentUrl: "assets/html-includes/way-to-give.html"
      }
    ];
    lazyLoadingEventHandlerFactory = function(moduleId, containerId, htmlContentPath) {
      return function(event) {
        var module;
```

“Way To Give” Module Definition

```
define(["newContentLoader"], function(contentLoader) {
  var wayToGive;

  console.log("way-to-give module is loaded");
  wayToGive = function() {
    return {
      rendered: false,
      render: function() {
        var dataUrl, newContainerID, whatWeDoButton;

        whatWeDoButton = "way-to-give";
        newContainerID = "way-to-give-container";
        dataUrl = "assets/html-includes/way-to-give.html";
        contentLoader.getNewContent(whatWeDoButton, newContainerID, dataUrl);
        return console.log("way-to-give module is rendered");
      },
      init: function() {
        return console.log("way-to-give init");
      }
    };
  };
  return wayToGive;
});
```

V6: With jQuery Mobile

- Easy to learn. Built on top of jQuery Core library
- HTML5 data- custom non-visible attribute:

```
<div data-role="page" id="Stats">
```

- The UI shows one page at a time
- Light-weight (90Kb gZipped)

Multi-Page Template

The content of multiple pages is located in one file.
When the app starts, only the first page is displayed

```
<body>
  <!--  Page 1   -->
     <div data-role="page" id="Donate" >
                ...
     </div>

  <!--  Page 2   -->
     <div data-role="page" id="Stats" >
                ...
     </div>
</body>
```

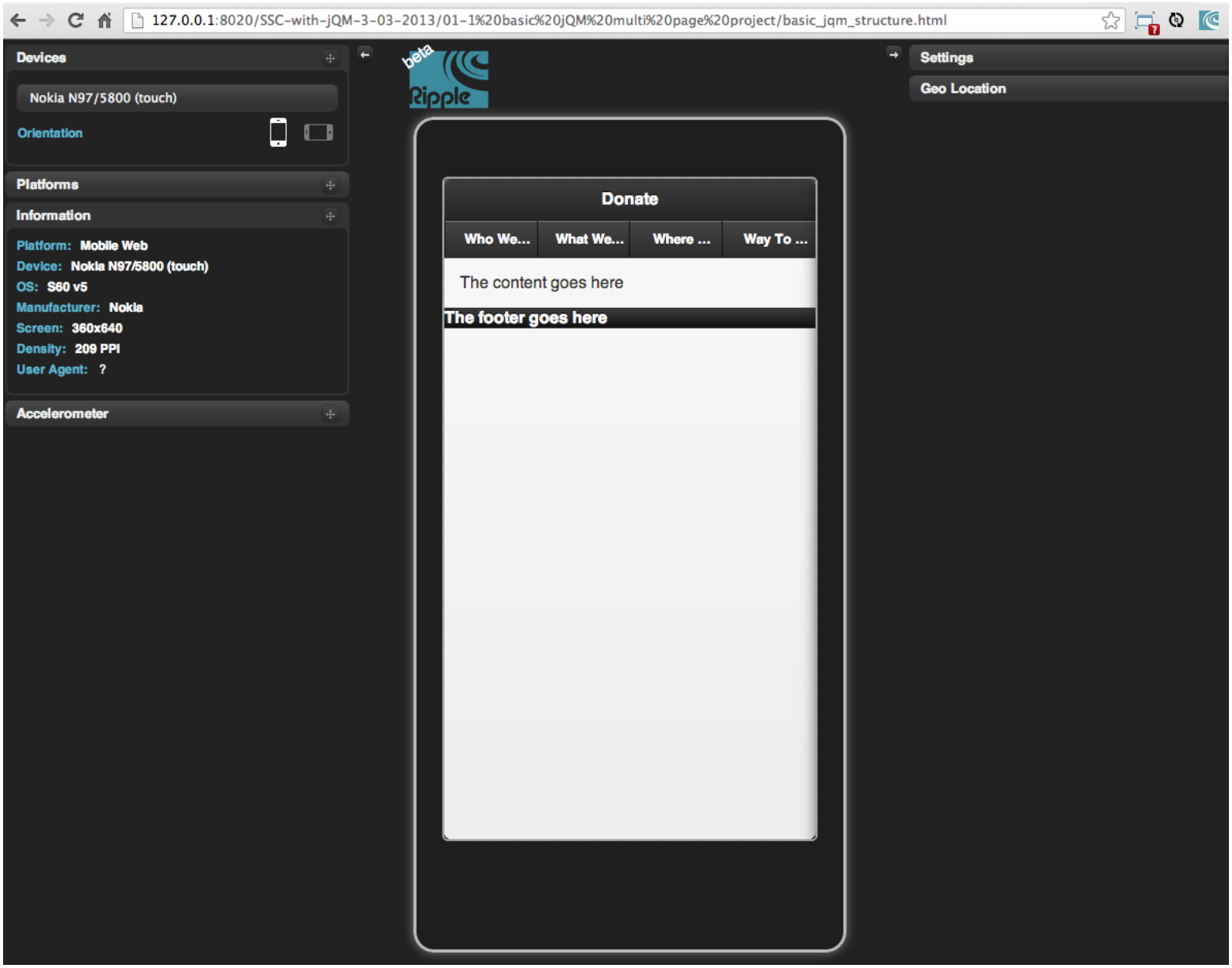
Multi-Page Template (cont.)

```
<body>
  <!--  Page 1    -->
     <div data-role="page" id="Donate">

           <div data-role="header" >...</div>
           <div data-role="content" >...</div>
           <div data-role="footer" >...</div>

     </div>

  <!--  Page 2    -->
     <div data-role="page" id="Stats">
           ...
     </div>
</body>
```

Navigation Bar

```
<div data-role="navbar">
        <ul>
            <li>
                <a href="#Who-We-Are">Who We Are</a>
            </li>
            <li>
                <a href="#What-We-Do">What We Do</a>
            </li>
            <li>
                <a href="#Where-We-Work">Where We Work</a>
            </li>
            <li>
                <a href="#Way-To-Give">Way To Give</a>
            </li>
        </ul>
</div>
```

Ripple Emulator



The Back Button

```
<div data-role="page" id="Stats" data-add-back-btn="true">

    <div data-role="header" >
        <h1>Statistics</h1>
    </div>

    Statistics will go here

</div>
```

Single-Page Template

An HTML file contains the content of a single page.

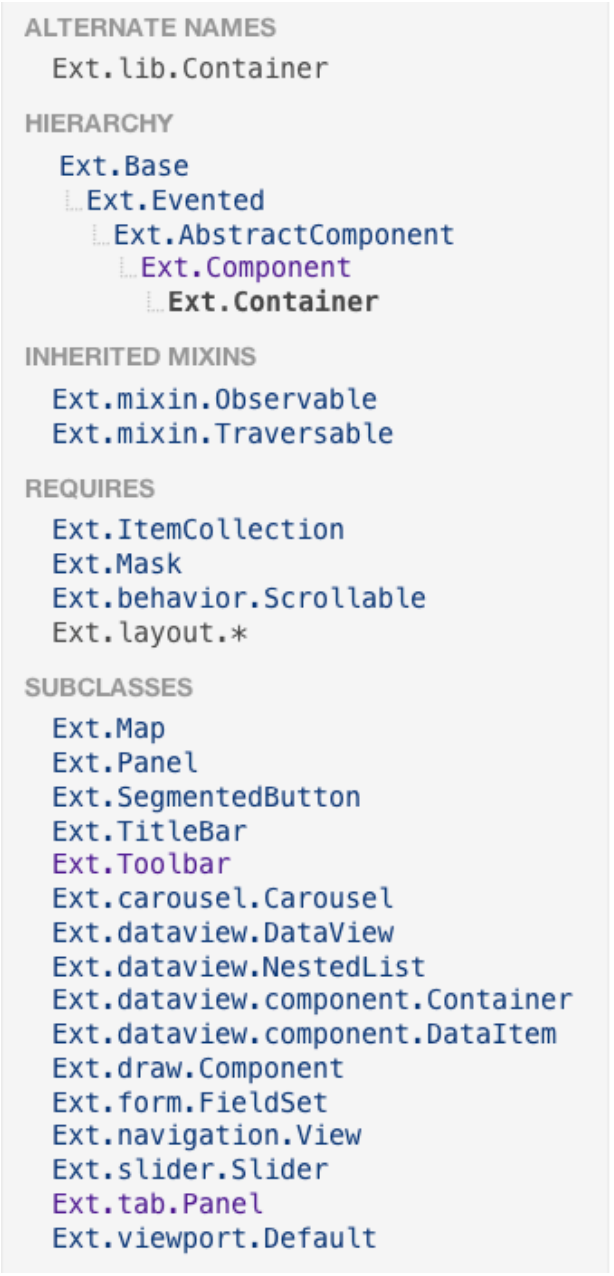
```
<div data-role="navbar">
        <ul>
          <li>
            <a href="page-1.html" data-transition="slideup">Page #1</a>
          </li>
          <li>
            <a href="#" class="ui-state-persist">Page #2</a>
          </li>
          <li>
            <a href="page-3.html" data-transition="slideup">Page #3</a>
          </li>
          <li>
            <a href="page-4.html" data-transition="slideup">Page #4</a>
          </li>
        </ul>
      </div>
```

V6: Demo With jQuery Mobile

- assets
  - css
    - app-styles.css
    - jqm-icon-pack-3.0.0-fa.css
  - fonts
    - FontAwesome.otf
    - fontawesome-webfont.eot
    - fontawesome-webfont.ttf
    - fontawesome-webfont.woff
  - img
    - child-1.png
    - logo-20x20.png
    - thumb-01.jpg
    - thumb-02.jpg
  - media
    - intro.jpg
    - intro.mp4
    - intro.ogv
    - intro.webm
- data
  - campaignsdata.json
  - chartdata.json
- js
  - app-main.js
- pages
  - about.html
  - donate.html
  - donation-submit-result.html
  - events.html
  - login.html
  - media.html
  - photo-app.html
  - share.html
  - share-via-facebook.html
  - share-via-google-plus.html
  - share-via-twitter.html
  - stats.html
  - way-to-give.html
  - what-we-do.html
  - where-we-work.html
  - who-we-are.html
- index.html

V7: With Sencha Touch

- Sencha Touch is a smaller brother of Ext JS
- It comes with mobile versions of lists, forms, toolbars, buttons, charts, audio, video, carousel etc.
- Jumpstart with generating the app with Sencha CMD.
- Package the Sencha Touch app as native

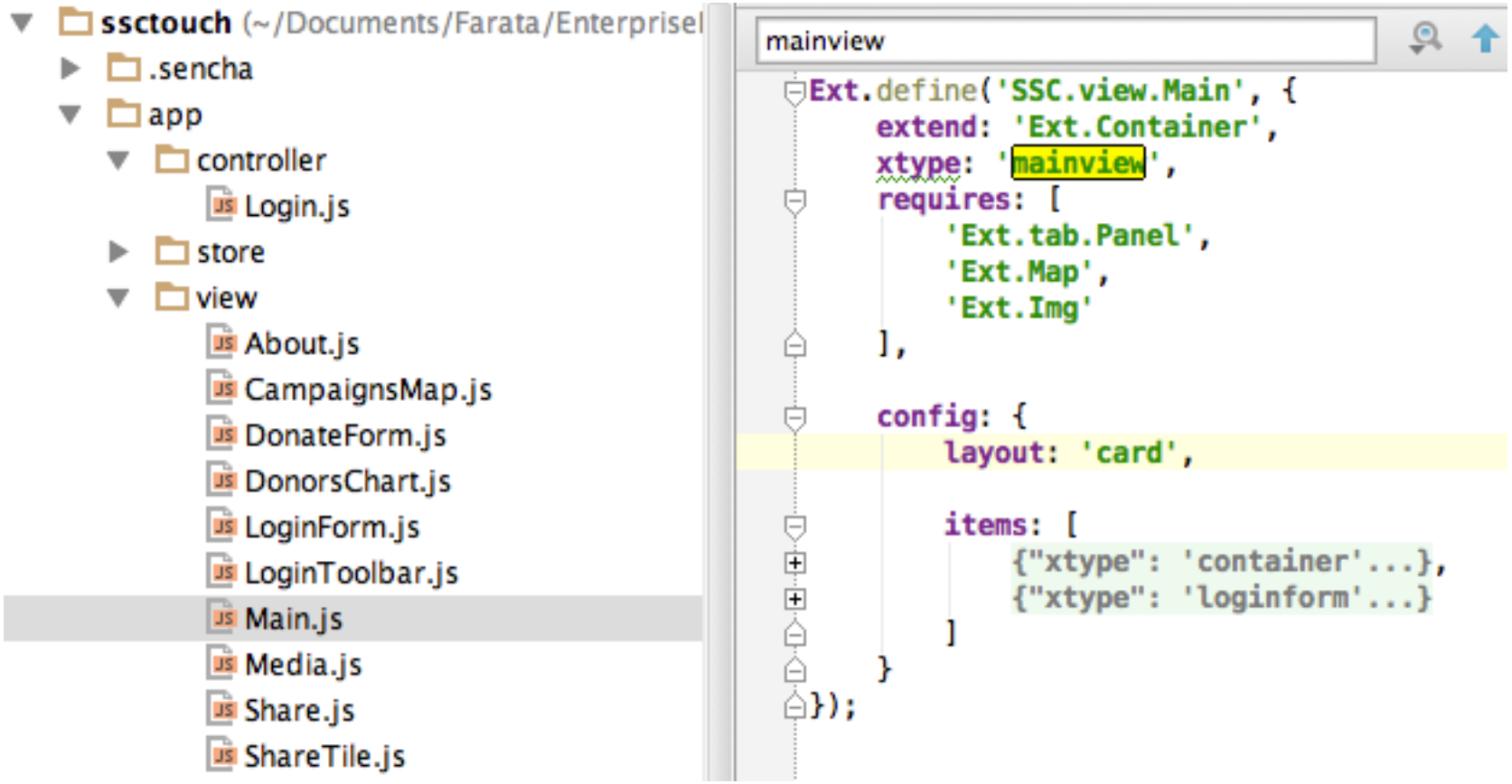


The app.JS

```
Ext.application({
    name: 'SSC',

    requires: [
        'Ext.MessageBox'
    ],

    views: [
        'About',
        'CampaignsMap',
        'DonateForm',
        'DonorsChart',
        'LoginForm',
        'LoginToolbar',
        'Main',
        'Media',
        'Share',
        'ShareTile'
    ],

    stores: [
        'Campaigns',
        'Countries',
        'Donors',
        'States',
        'Videos'
    ],x
```

```
    controllers: [
        'Login'
    ],

    launch: function() {
        // Destroy the #appLoadingIndicator element
        Ext.fly('appLoadingIndicator').destroy();

        // Initialize the main view
        Ext.Viewport.add(Ext.create('SSC.view.Main'));
    },

    onUpdated: function() {
        Ext.Msg.confirm(
            "Application Update",
            "This application has just successfully been updated
to the latest version. Reload now?",
            function(buttonId) {
                if (buttonId === 'yes') {
                    window.location.reload();
                }
            }
        );
    }
});
```

The Landing Page



Login Controller

```
Ext.define('SSC.controller.Login', {
    extend: 'Ext.app.Controller',

    config: {
        refs: {
            mainView: 'mainview',
            loginButton: 'button[action=login]',
            loginForm: 'loginform',
            cancelButton: 'loginform button[action=cancel]'
        },

        control: {
            loginButton: {
                tap: 'showLoginView'
            },
            cancelButton: {
                tap: 'cancelLogin'
            }
        }
    },

    showLoginView: function () {
        this.getMainView().setActiveItem(1);
    },

    cancelLogin: function () {
        this.getMainView().setActiveItem(0);
    }

});
```

Donors Store

```
Ext.define('SSC.store.Donors', {
    extend: 'Ext.data.Store',

    config: {
        fields: [
            { name: 'donors',   type: 'int' },
            { name: 'location', type: 'string' }
        ],

        data: [
            { donors: 48, location: 'Chicago, IL' },
            { donors: 60, location: 'New York, NY' },
            { donors: 90, location: 'Dallas, TX' },
            { donors: 22, location: 'Miami, FL' },
            { donors: 14, location: 'Fargo, ND' },
            { donors: 44, location: 'Long Beach, NY' },
            { donors: 24, location: 'Lynbrook, NY' }
        ]
    }

});
```

jQuery Mobile or Sencha Touch?

Use jQuery Mobile if:

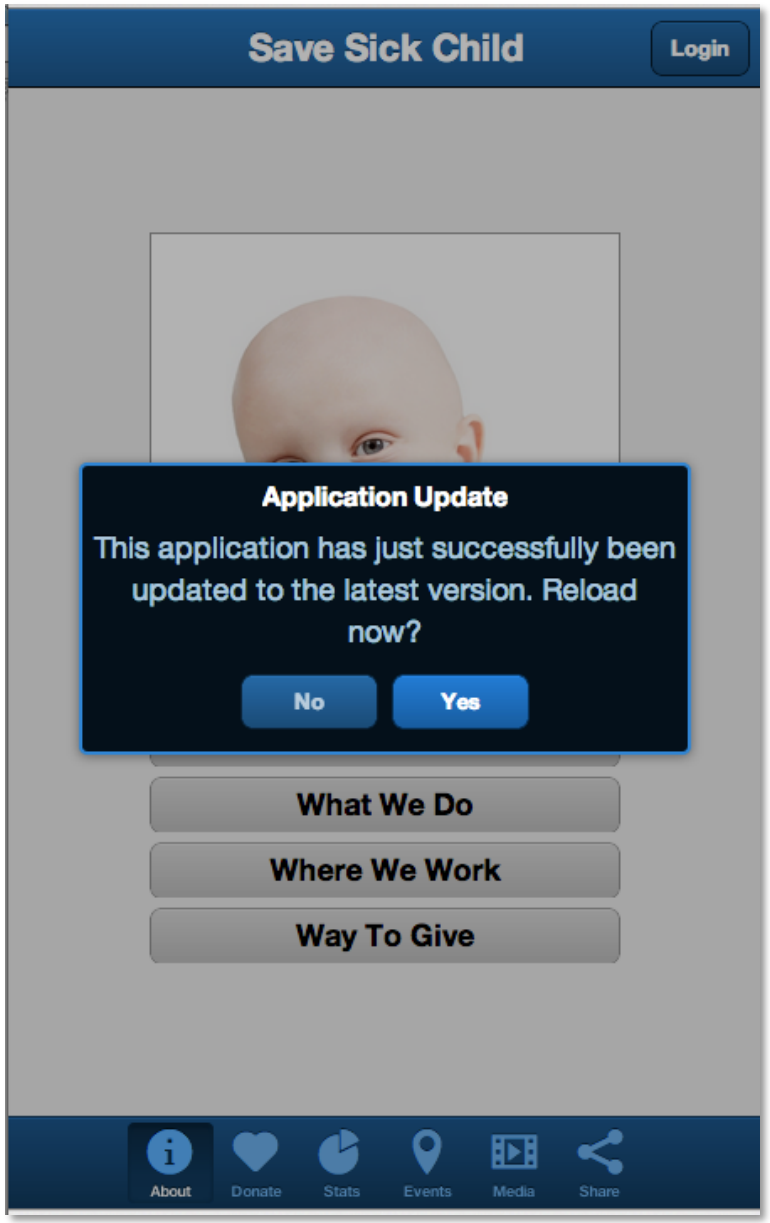
- You are afraid of being locked up with any one vendor.
- You need your application to work on most of the mobile platforms.
- You prefer declarative UI and hate debugging JavaScript.

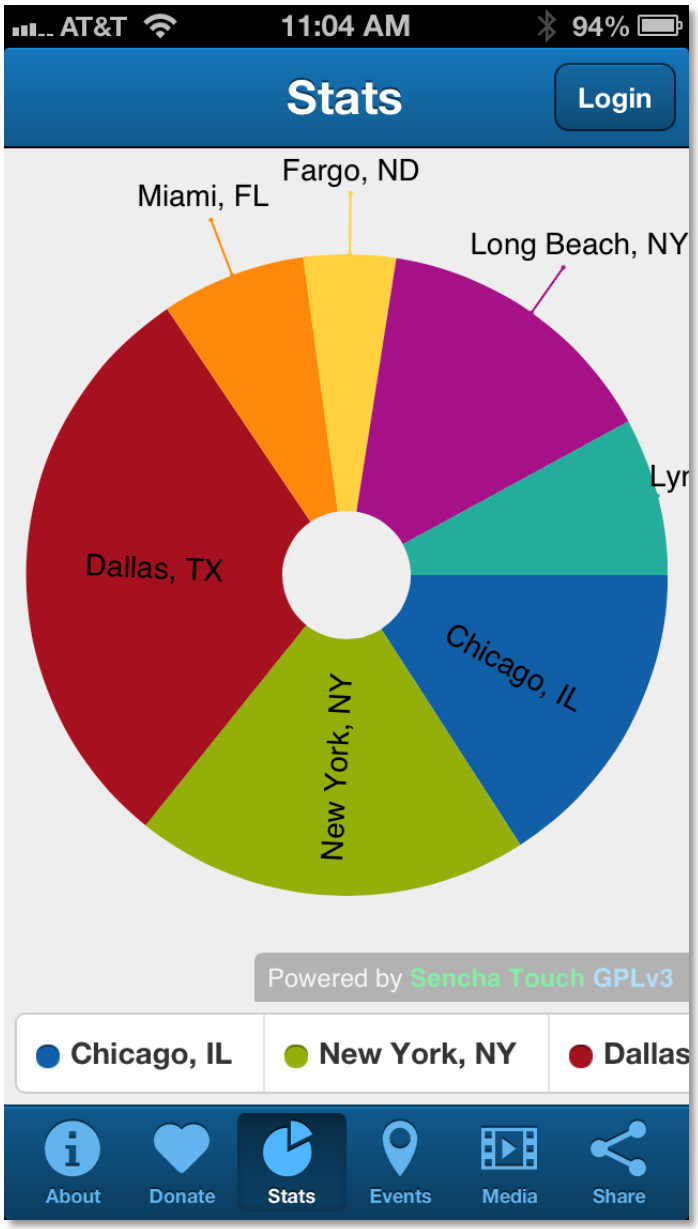
jQuery Mobile or Sencha Touch?

Use Sencha Touch if:

- You like to have a rich library of pre-created UI.
- Your application needs smooth animation.
- You are into MVC
- You want to package your application as a native one.
- You want your application to look as close to the native ones as possible.

V7: Demo With Sencha Touch





Where to go next?

Google Dart Language

https://www.dartlang.org

Links

- Drafts of the Enterprise Web Development book:

  http://enterprisewebbook.com

- 7 versions of the Save The Child app:

  http://savesickchild.org

- Farata Systems: http://faratasystems.com

- My Twitter: @yfain

- This presentation: http://slidesha.re/1fePxul