
ORACLE®

An Intro to JavaScript Web Apps on Oracle Database

Dan McGhan
Oracle Developer Advocate | JavaScript & HTML5
dan.mcghan@oracle.com || @dmcghan
February, 2015

ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

About me

- Dan McGhan
  - Oracle Developer Advocate
  - Focus on JavaScript and HTML5
- Contact Info
  - dan.mcghan@oracle.com
  - @dmcghan
  - jsao.io

Agenda

1. Why JavaScript?
2. Creating a JavaScript App
3. Using the Oracle Database Node.js Driver
4. Reactive/Real-time data demo

Agenda

1. Why JavaScript?
2. Creating a JavaScript App
3. Using the Oracle Database Node.js Driver
4. Reactive/Real-time data demo

JavaScript is…

- Your browser's native language
  - Remember Java applets & Flash?
- Continually improving
  - ES6 brings many powerful constructs and features
- Everywhere
  - Can run on the server thanks to Node.js
  - Countless open source frontend and backend projects
- The world's most popular programming language
  - According to RedMonk

What is AngularJS?

- A JavaScript MVC framework
  - Others include Backbone, Ember, & Knockout
- Created by and maintained by Google
- Typically used to build single page applications (SPA)

Client Side Logic

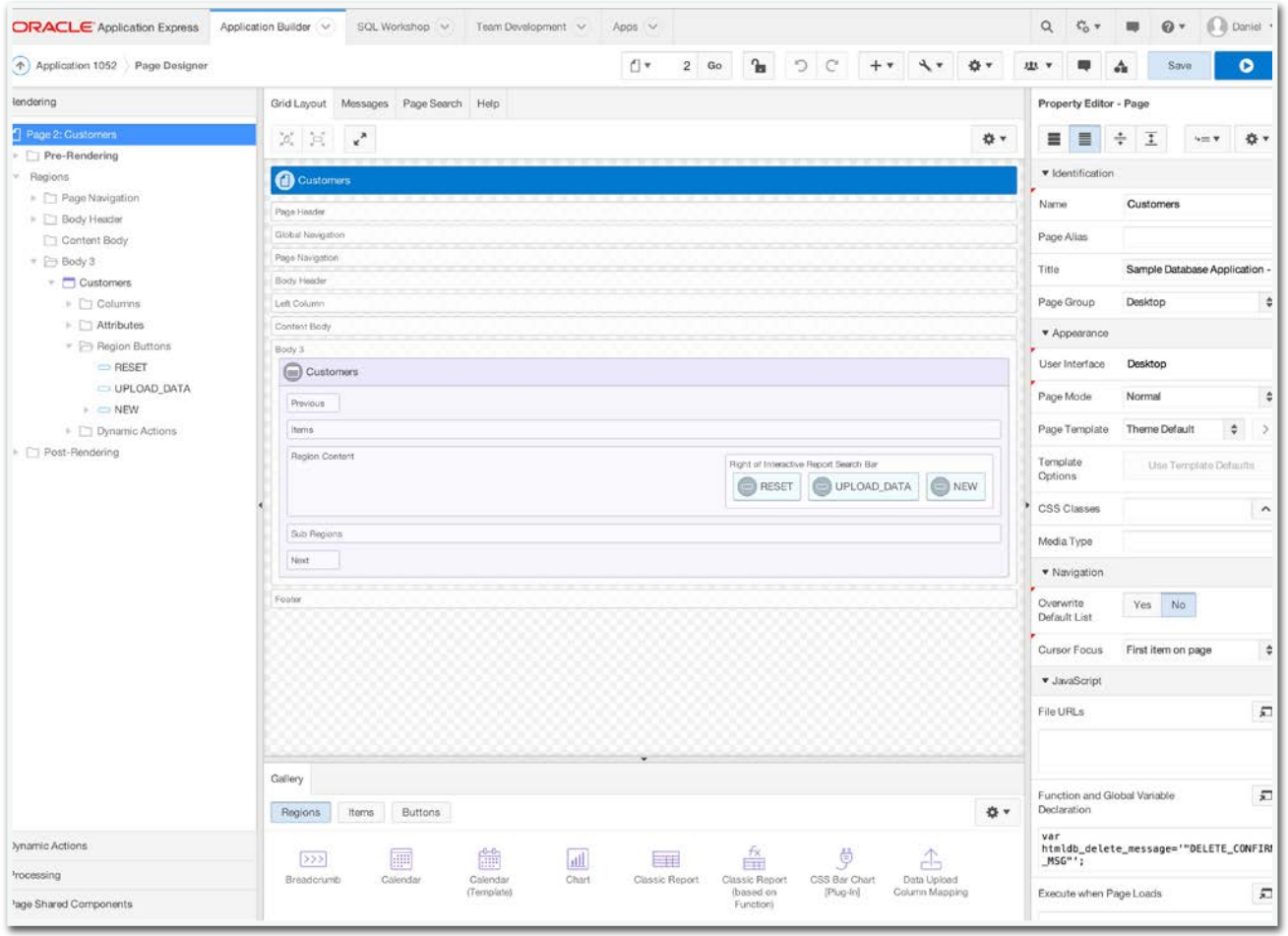
Advantages

- Avoid full page refreshes
- Snappier applications
- Reduced network traffic

Disadvantages

- Increased development time
- Data can’t be trusted
  - must re-validate server side

When is Client Side Logic Useful?



Complex, highly interactive UI

Agenda

1. Why JavaScript?
2. Creating a JavaScript App
3. Using the Oracle Database Node.js Driver
4. Reactive/Real-time data demo

Prerequisites

- Oracle Database
  - 11g or 12c
  - Access to HR
- Node.js installed
- Oracle Database Node.js Driver installed “globally”
- Detailed instructions here
  - https://jsao.io/2015/01/up-and-running-with-node-js-oracle/

Start your engines!

- Create a directory to hold the project
- Add a “public” sub-directory
- Add an index.html file to public

A dead simple web server with http

```
1  var http = require('http');
2
3  http.createServer(function(req, res) {
4      res.writeHead(200, {'Content-Type': 'text/plain'});
5      res.end('Hello World!');
6  }).listen(3000);
7
8  console.log('Server running at http://localhost:3000/');
```

A dead simple web server with Express

```
1  var express = require('express');
2  var app = express();
3
4  app.use('/', function(req, res) {
5      res.end('Hello World.');
6  });
7
8  app.listen(3000, function() {
9      console.log('Server running at http://localhost:3000/');
10 });
```

Serve static files in “public” with serve-static

```
1  var express = require('express');
2  var app = express();
3  var serveStatic = require('serve-static');
4
5  app.use('/', serveStatic(__dirname + '/public'));
6
7  app.listen('3000', function() {
8      console.log('Server running at http://localhost:3000/');
9  });
```

Add AngularJS to the project via Bower

- Install Bower via npm
  - Use “-g” for global install
  - May need to run with sudo
- Use Bower to install AngularJS
- Add static file handling for bower_components
- Add script tag to html file

Start using AngularJS

- Create app.js
  - Will contain our AngularJS application logic
  - Start with a module and a controller
- Update index.html to accommodate AngularJS
  - Add ng-app and ng-controller to html element
  - Add script tag that points to app.js
- Move “Hello World” text to the “model”

Add an employee report with static data

- Add an employees array to the model
  - Later we'll fetch the employees from Oracle Database
- Add a table to index.html to display the employees

Agenda

1. Why JavaScript?
2. Creating a JavaScript App
3. **Using the Oracle Database Node.js Driver**
4. Reactive/Real-time data demo

Create a RESTful API to get employees

- Add another end point for “/api/employees”
  - Use the Oracle Database Node.js Driver to fetch employees
- Update app.js to fetch the employees via the API

Consider using ORDS over manual API creation

- ORDS can automate RESTfully enabling tables
- Node can still act as the web server
  - Talks to the database via ORDS rather than the driver
- ORDS can handle sorting, pagination, filtering, etc.

Agenda

1. Why JavaScript?
2. Creating a JavaScript App
3. Using the Oracle Database Node.js Driver
4. Reactive/Real-time data demo

Imagine this scenario

- You're building an app where users should see updates in real-time
  - Think gmail, twitter, a dashboard, whatever!
- You have ~1000 concurrent users all the time
- You could use polling (every 10 seconds?)
  - 1000 users * 6 polls/min * 60 min/hr * 24 hr/day
  - 8,640,000
- Or you could push when the data changes (every 10 seconds?)
  - 6 changes/min * 60 min/hr * 24 hr/day
  - 8,640

But how???

- Node.js
  - Web server, but you knew that already
- Socket.io
  - Makes working with web sockets easy (push changes to client)
- Oracle Database's Continuous Change Notification
  - Object Change Notification (OCN)
  - Query Result Change Notification (QRCH)
  - Allow us to "register" queries and "execute" PL/SQL when changes occur

Recipe for Pushing Data

- Node.js
  - Web server, but you knew that already
- Socket.io
  - Makes working with web sockets easy (push changes to client)
- Oracle Database's Continuous Query Notification (CQN)
  - Object Change Notification (OCN)
  - Query Result Change Notification (QRCH)
  - Allows us to "register" queries and "execute" PL/SQL when changes occur

Summary

- JavaScript is an amazingly powerful language
- Oracle Database is an amazingly powerful database
- Using both yields super awesome apps!


ORACLE®