

# HTML5 Live-Cooking

Entwicklungsumgebung & Werkzeuge



# HTML5, JavaScript, CSS

- Unterschiede DEV und Prod
  - Dev: wartbarer Code, Debugging, kurze Roundtrips, trotzdem nah an der Produktiv-Umgebung
  - Prod: kompakter, maschinenlesbarer Code, mit kurzen Ladezeiten



Transformation von Code schon während der Entwicklung und erst recht beim Build





# Yeoman

- Scaffolding für Web Projekte auf Basis von Node.js, npm, Bower, Grunt
- Generatoren
  - erstellen die Projektstruktur
  - konfigurieren Grunt und Bower
  - installieren Komponenten
- `npm install -g yo`  
`npm install -g generator-angular`  
`yo angular`





# Bower

- Paketmanager für beliebige Ressourcen (JavaScript, CSS, Fonts, ...)
- bevorzugt aus Git oder SVN
- Versionen über Tags nach Semantic Versioning <http://semver.org/>
- `bower install angular --save`  
`bower install angular-mocks --save-dev`
- `bower.json`

<http://bower.io/>





# Bower

```
{  
  "name": "angular-animate",  
  "version": "1.2.15",  
  "main": "./angular-animate.js",  
  "dependencies": {  
    "angular": "1.2.15"  
  }  
}
```

[https://github.com/angular/bower-angular-animate/blob/v1.2.15/  
bower.json](https://github.com/angular/bower-angular-animate/blob/v1.2.15/bower.json)





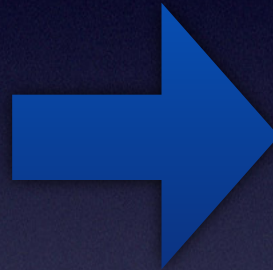
# Grunt

- Task-Runner für Web-Projekte in JavaScript
- Task?
- Installation von Modulen über npm
- Konfiguration: Gruntfile.js
- Einige Beispiele...



# Grunt - compass

```
@mixin square($size) {  
  width: $size;  
  height: $size;  
}  
  
.bild {  
  @include square(200px);  
}  
  
.hinweis {  
  @include square(20%);  
}
```



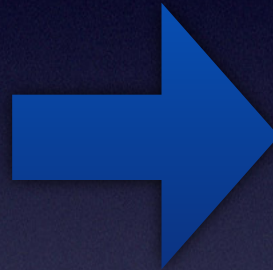
```
.bild {  
  width: 200px;  
  height: 200px;  
}  
  
.hinweis {  
  width: 20%;  
  height: 20%;  
}
```

Ermöglicht die Benutzung von SASS während der Entwicklung und kompiliert scss nach css



# Grunt - autoprefixer

```
[draggable] {  
  user-select: none;  
}
```



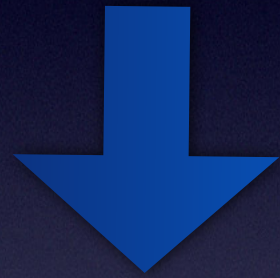
```
[draggable] {  
  -webkit-user-select: none;  
  -moz-user-select: none;  
  -ms-user-select: none;  
  user-select: none;  
}
```

Analysiert CSS-Dateien und ergänzt  
Herstellerpräfixe auf Basis der Can I Use  
Datenbank ([www.caniuse.com](http://www.caniuse.com))



# Grunt - ngmin

```
app.factory('cryptoService', function($http, $q){  
    ...  
});
```



```
app.factory('cryptoService', ['$http' , '$q' , function($http, $q){  
    ...  
}] );
```

Analysiert JavaScript-Dateien und sorgt dafür,  
dass der Angular Dependency Injection -  
Mechanismus das Minifizieren übersteht.





# Gruntfile.js

```
module.exports = function(grunt) {  
  
  require('load-grunt-tasks')(grunt);  
  
  grunt.initConfig({  
    uglify: {  
      options: {  
        banner: '/*! <%= grunt.template.today("yyyy-mm-dd") %> */\n',  
      },  
      build: {  
        src: 'src/*.js',  
        dest: 'build/*.min.js'  
      }  
    }  
  });  
  
  grunt.registerTask('default', ['uglify']);  
};
```



# Demo

Entwicklungsumgebung & Werkzeuge



# HTML5 Live-Cooking

Album erstellen mit Angular





# AngularJS

- OpenSource JavaScript Framework(!)
- MVC für CRUD Single Page WebApps
- Fokus: Entwicklung & Testen einfach gestalten

<http://angularjs.org/>





# AngularJS Feature

- DSL in HTML (nur Direktiven greifen auf das DOM zu)
- Dependency Injection (u.a. Provider, Factories, Services organisiert in Modulen)
- View und Controller kommunizieren über ein \$scope Objekt
- 2-Way-Data-Binding (ohne die Model-Objekte zu „verunreinigen“)
- Formular-Validierung



# Promises

- JavaScript läuft in einem Thread im Browser
- Aber: asynchrone Aufrufe (Timer, HTTP, Dateioperationen, Browser-Events ...)
- Herkömmlich: Callbacks



# Callbacks

```
loadImageFormDisk(file, {
  success: function(img){
    scaleImage(img, {
      success: function(scaledImage){
        readExifData(file, {
          success: function(exifData){
            post2Backend(scaledImage, exifData, {
              success: function(savedData){
                },
                error: function(...){}
            });
          },
          error: function(error){...}
        });
      },
      error: function(error){...}
    });
  },
  error: function(err){...}
});
```

- Lesbar?
- Erweiterbar?
- Parallelisierbar?



# Promises

```
var promise = loadImageFormDisk(file).then(function(img){
    return scaleImage(img).then(function(scaledImage){
        return readExifData(file).then(function(exifData){
            return post2Backend(scaledImage, exifData);
        });
    });
});
```

```
promise.then(function(savedData){
    // alles gut
}, function(error){
    // Fehler
});
```

- \$q-Service in Angular
- \$q.all()
- \$http und \$timeout sind *thenable*
- besser?



# Promises

```
var mapResult2Album = function(result){  
    return backendMapper.mapAlbum(result.data);  
};  
  
var decryptAlbum = function(album){  
    return cryptoService.decrypt(album);  
};  
  
var success = function(album){  
    ...  
}  
var onError = function(err){  
    ...  
}  
  
$http.get(url)  
    .then(mapResult2Album)  
    .then(decryptAlbum)  
    .then(success)  
    .catch(onError);
```



# Demo

Album erstellen mit Angular

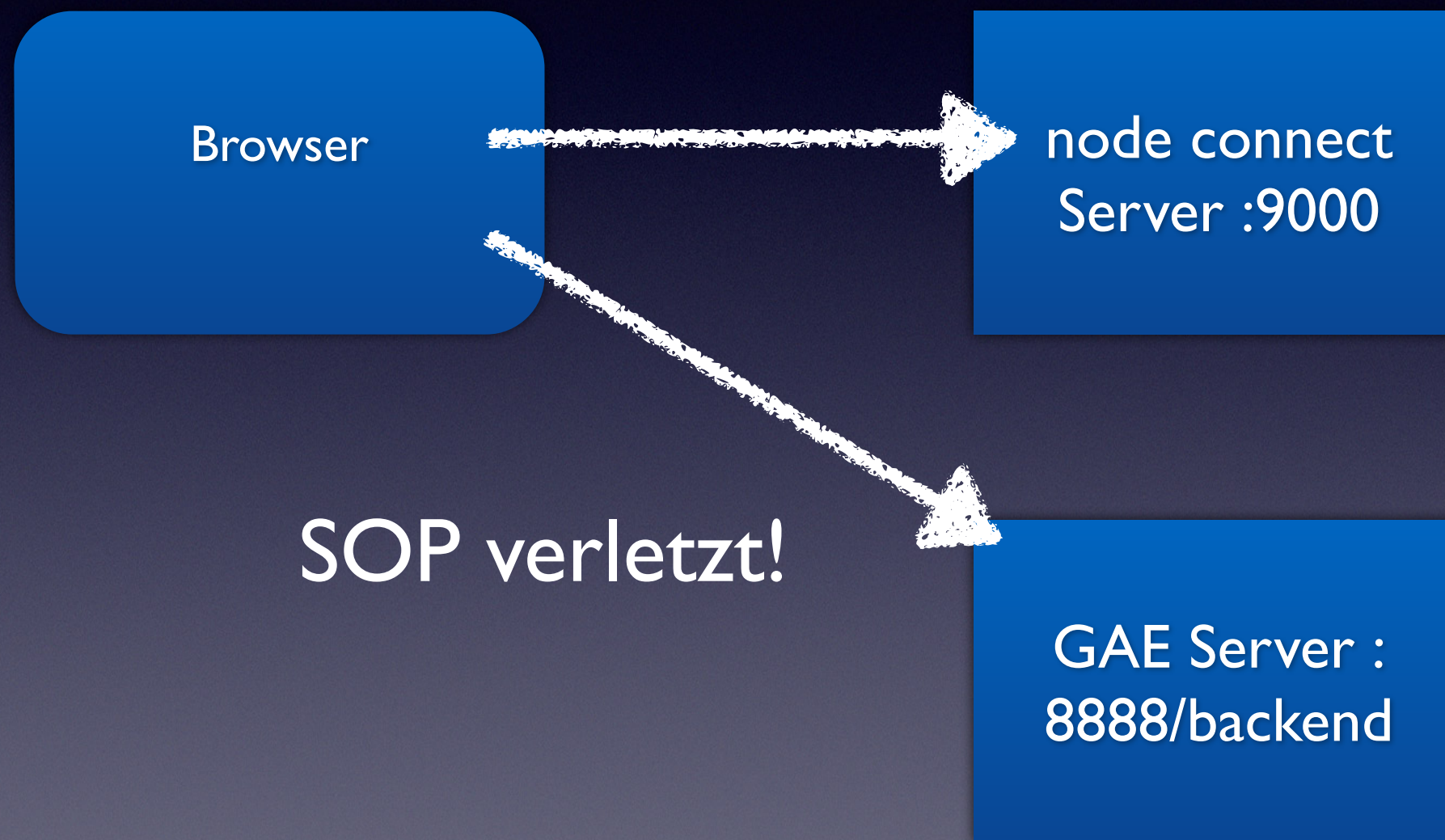


# HTML5 Live-Cooking

„Hochzeit“ mit dem Backend und ein Blick in den  
Maschinenraum

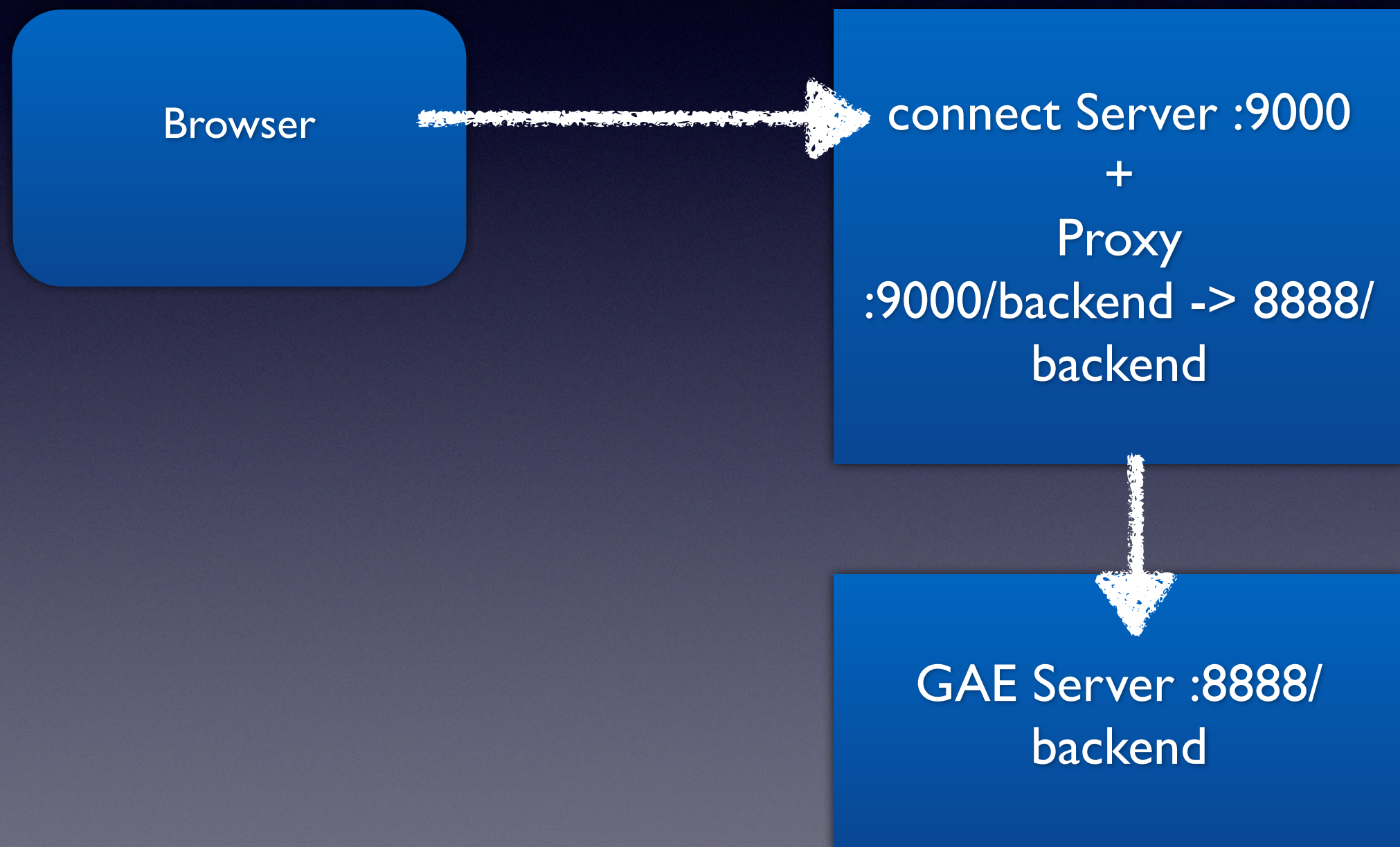


# Anbindung Backend





# Anbindung Backend







# \$http-Service

- XMLHttpRequest + JSONP
- Mock-Version für Unit-Tests
- thenable
- `$http.post(URL, album);`





# Bootstrap

- CSS Framework (+ ein wenig JavaScript - wenn man will)
- Grid System, Formulare, Tabellen, Textsatz
- Komponenten (Glyphicons, Dialoge, Fortschrittsbalken,...)
- Responsive Design über Media-Queries - Grenzen?
- Less aber auch SASS Version(!)

<http://getbootstrap.com/>





# SASS

- Variablen
- Farbtransformation
- Mixins
- Verschachtelungen
- Schleifen & Bedingte Ausdrücke
- Import

<http://sass-lang.com/>




# Demo

Die Single Page WebApp



# JavaScript im Browser

- Javascript im Browser wird in einem Thread ausgeführt
- JavaScript läuft im UI-Thread
- Deshalb: Berechnungsintensive Operationen gehören dort nicht hin!
- Lösungen:
  - Throttle / Script-Ausführung aufteilen
  - Auf dem Server ausführen 
  - Web Worker (!)



# WebWorker

- Führen JavaScript im Hintergrund aus  
`var worker = new Worker('aufgabe.js');`
- Kommunikation über `postMessage` und `EventListener`
- kein Zugriff auf das UI (z.b. Canvas oder gar DOM )
- nicht alle JavaScript Objekte lassen sich als Parameter übergeben (z.B. `file[]`)