GWT - Introduction

Dr. Lofi Dewanto
https://lofidewanto.blogspot.de

Agenda

- Webapps - State of the Art
- Components of GWT
  - Java to JavaScript Transpiler
  - Emulated Java Runtime Environment
  - Interoperability Layer to JavaScript
  - Tools
  - User Interface Components

Webapps - State of the Art

- See: Webapps - State of the Art
- State of the Art:
  - SOFEA (Service Oriented Front End Architecture)
  - ROCA (Resource Oriented Client Architecture)

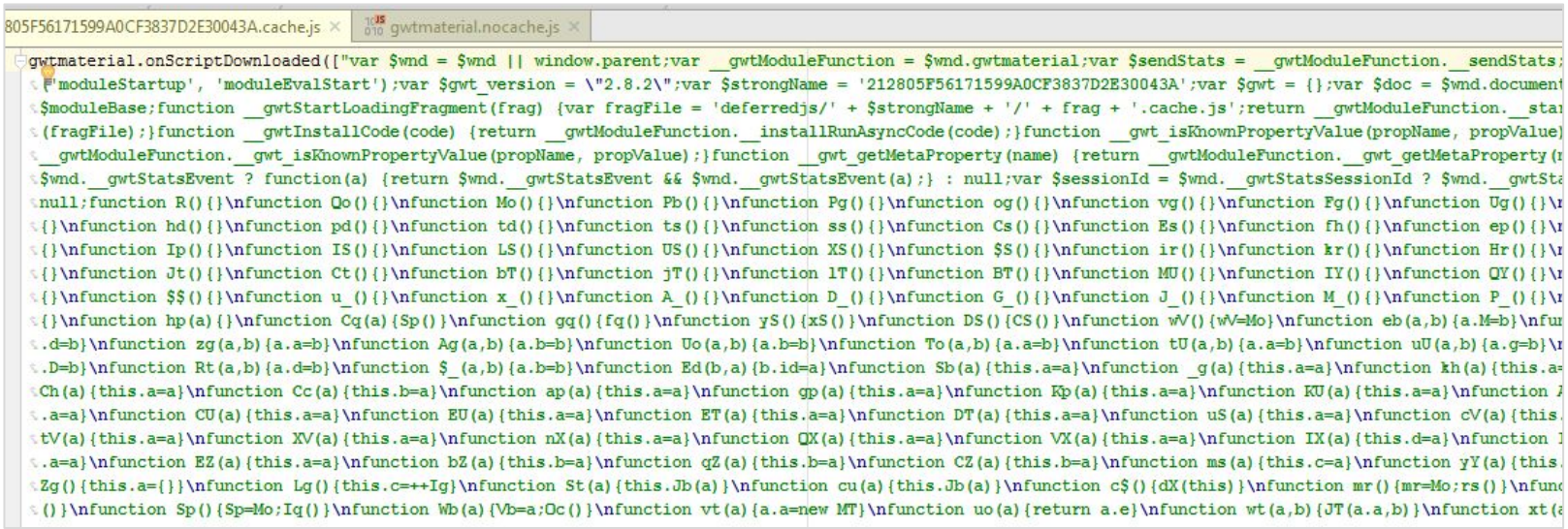
Components of GWT

- Java to JavaScript Transpiler
- Emulated Java Runtime Environment
- Interoperability Layer to JavaScript
- Tools
- User Interface Components

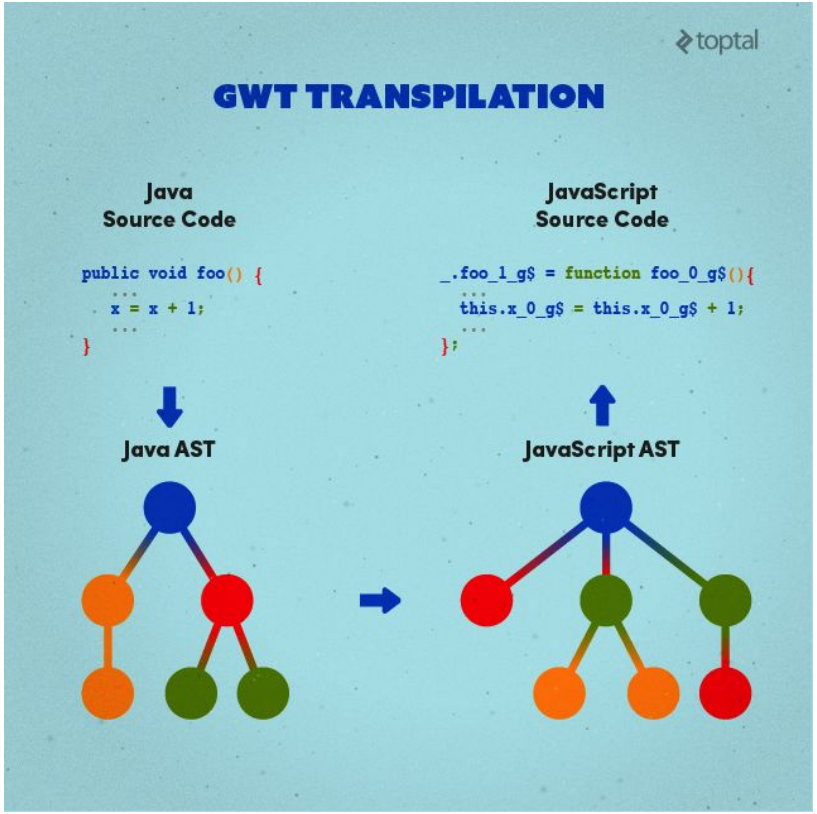
Java to JavaScript Transpiler

Java to JavaScript (JS) Transpiler

- Main component: Java (Source) to JavaScript (Source) transpiler
- Optimize the JS output at compile time
  - Naming the shortest possible name (attribute, method, class)
  - Pruning unused classes, methods and attributes



Java to JavaScript (JS) Transpiler

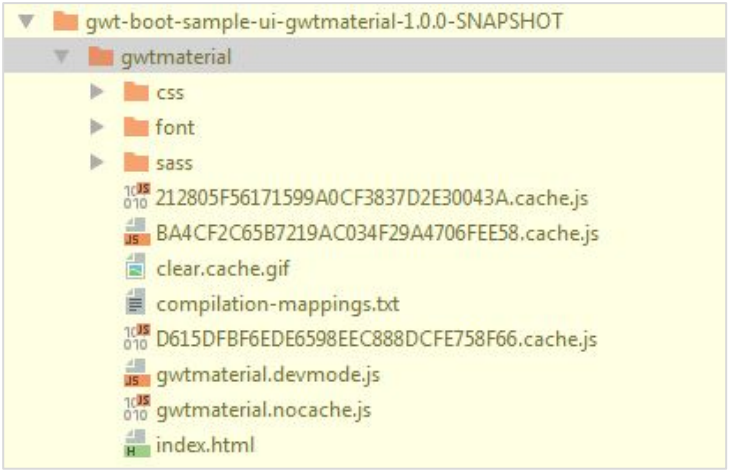


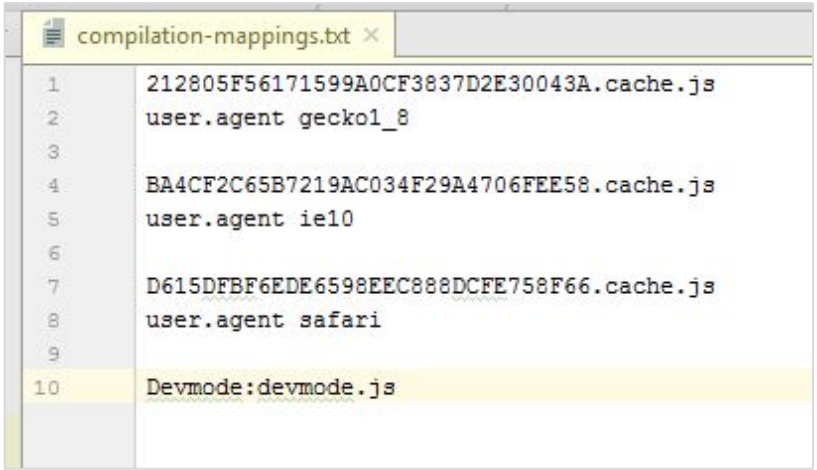
Java to JavaScript (JS) Transpiler

- Inside GWT transpiler: http://bit.ly/2InB0AW

Java to JavaScript (JS) Transpiler

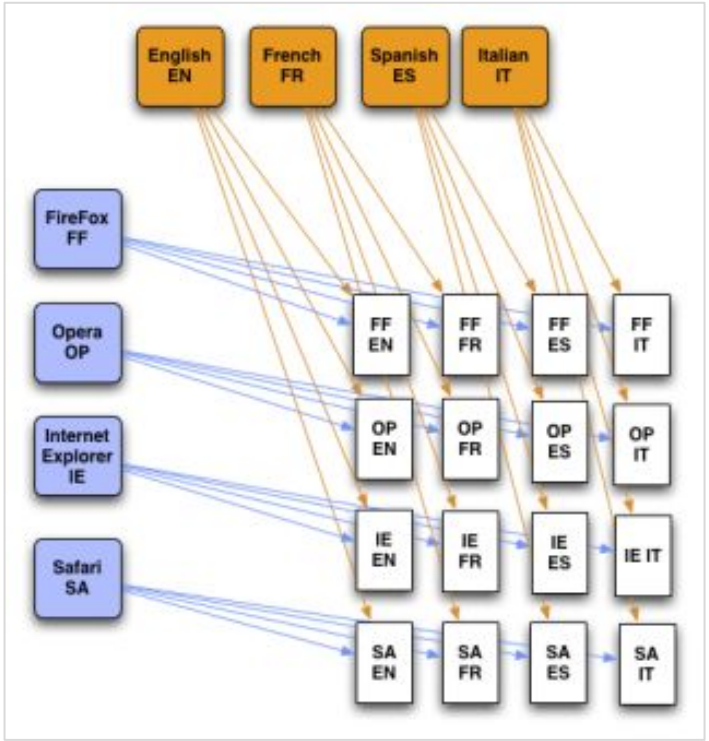
- Perfect caching
  - `*.nocache.js or *.nocache.*`
  - `<hashname>.cache.js or <hashname>.cache.*`





Java to JavaScript (JS) Transpiler

- Separation of
  - User agent (IE, Firefox, Safari)
  - Locale (EN, DE, …)
- Reducing the download size of JS

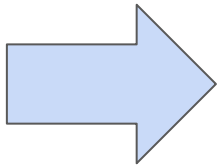


Java to JavaScript (JS) Transpiler

- Dead-for-now (DFN) code-Splitting “on demand”

```
public class Hello implements EntryPoint {
  public void onModuleLoad() {
    Button b = new Button("Click me", new ClickHandler() {
      public void onClick(ClickEvent event) {
        Window.alert("Hello, AJAX");
      }
    });

    RootPanel.get().add(b);
  }
}
```

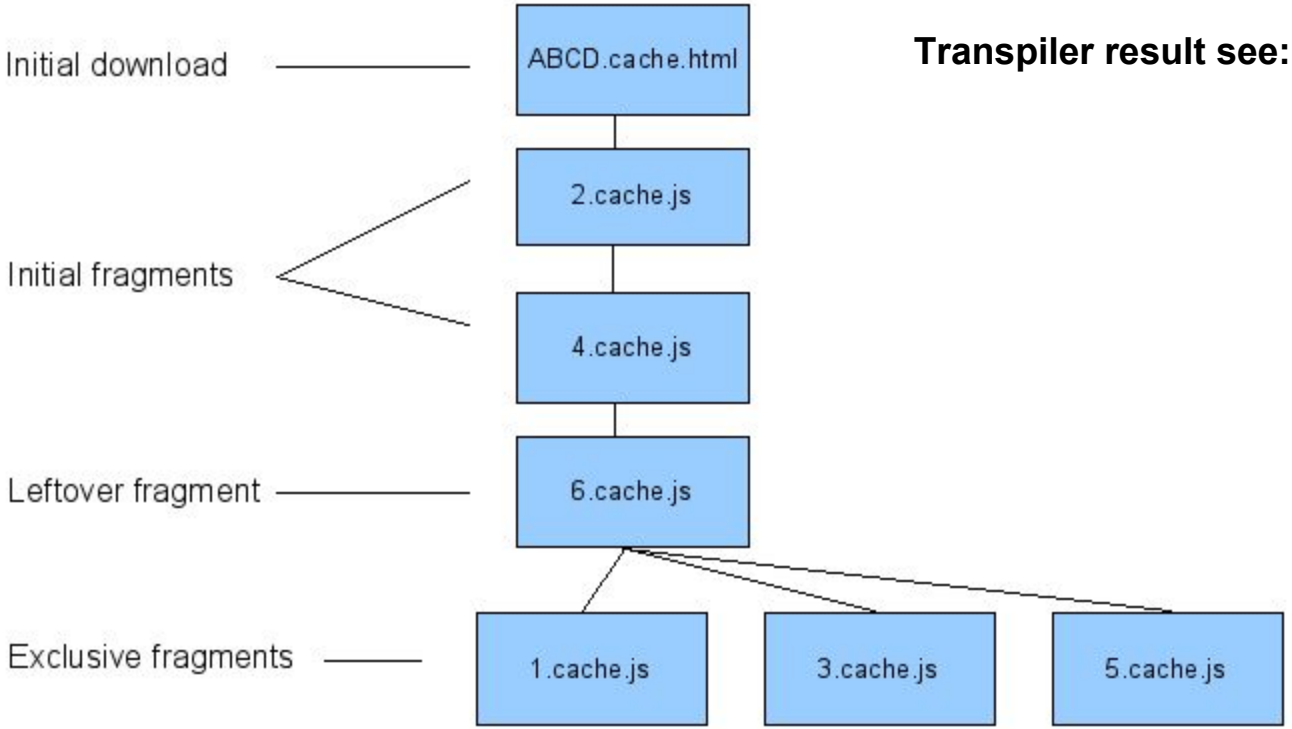


```
public class Hello implements EntryPoint {
  public void onModuleLoad() {
    Button b = new Button("Click me", new ClickHandler() {
      public void onClick(ClickEvent event) {
        GWT.runAsync(new RunAsyncCallback() {
          public void onFailure(Throwable caught) {
            Window.alert("Code download failed");
          }

          public void onSuccess() {
            Window.alert("Hello, AJAX");
          }
        });
      }
    });

    RootPanel.get().add(b);
  }
}
```

Java to JavaScript (JS) Transpiler



Transpiler result see: Report

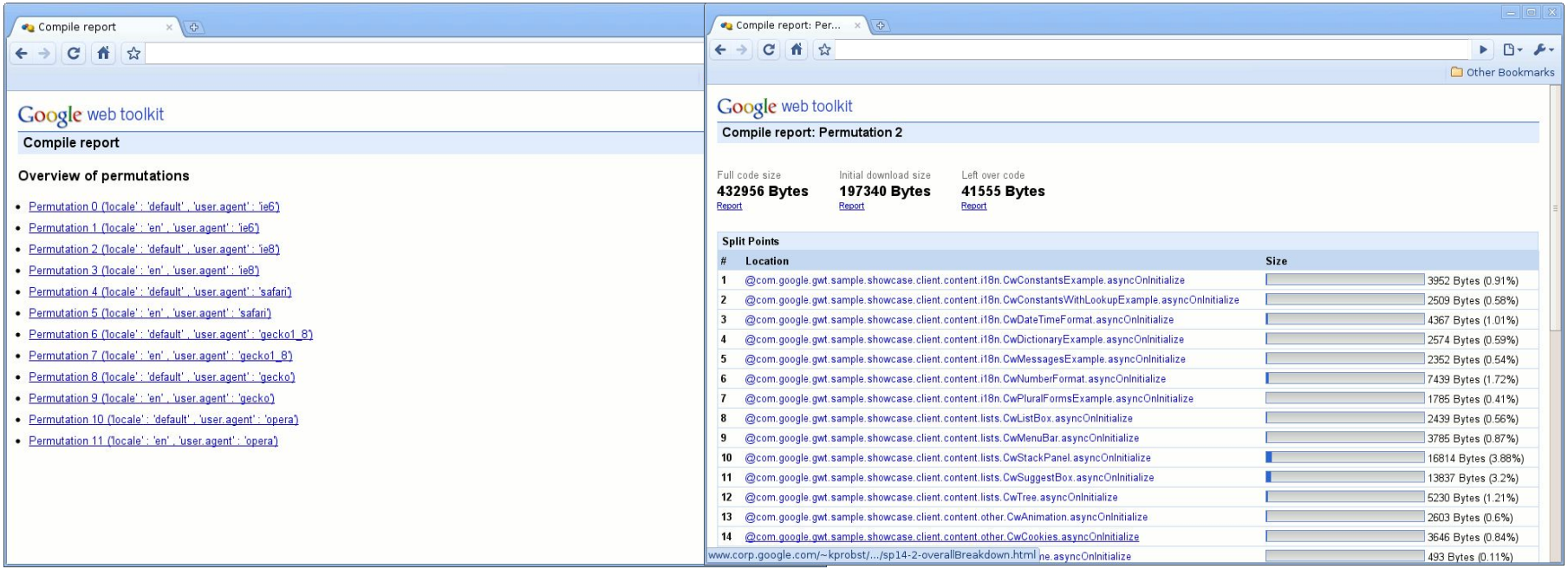
Java to JavaScript (JS) Transpiler

- Compiler CLI
  - com.google.gwt.dev.Compiler
- Compiler options (2.8.x)

```
> java -cp gwt-dev.jar com.google.gwt.dev.Compiler
Missing required argument 'module[s]'
Google Web Toolkit 2.8.0
Compiler [-logLevel (ERROR|WARN|INFO|TRACE|DEBUG|SPAM|ALL)] [-workDir dir] [-X[no]closureFormattedOutput] [-[no]compileReport] [-X

where
  -logLevel                    The level of logging detail: ERROR, WARN, INFO, TRACE, DEBUG, SPAM or ALL (defaults to INFO)
  -workDir                     The compiler's working directory for internal use (must be writeable; defaults to a system temp d
  -X[no]closureFormattedOutput EXPERIMENTAL: Enables Javascript output suitable for post-compilation by Closure Compiler (defaul
  -[no]compileReport           Compile a report that tells the "Story of Your Compile". (defaults to OFF)
  -X[no]checkCasts             EXPERIMENTAL: DEPRECATED: use jre.checks.checkLevel instead. (defaults to OFF)
  -X[no]classMetadata          EXPERIMENTAL: Include metadata for some java.lang.Class methods (e.g. getName()). (defaults to ON
  -[no]draftCompile            Compile quickly with minimal optimizations. (defaults to OFF)
  -[no]checkAssertions         Include assert statements in compiled output. (defaults to OFF)
  -XfragmentCount              EXPERIMENTAL: Limits of number of fragments using a code splitter that merges split points.
  -XfragmentMerge              DEPRECATED (use -XfragmentCount instead): Enables Fragment merging code splitter.
  -gen                         Debugging: causes normally-transient generated types to be saved in the specified directory
  -[no]generateJsInteropExports Generate exports for JsInterop purposes (defaults to OFF)
  -XmethodNameDisplayMode      EXPERIMENTAL: Specifies method display name mode for chrome devtools: NONE, ONLY_METHOD_NAME, ABB
  -Xnamespace                  Puts most JavaScript globals into namespaces. Default: PACKAGE for -draftCompile, otherwise NONE
  -optimize                    Sets the optimization level used by the compiler.  0=none 9=maximum.
  -[no]saveSource              Enables saving source code needed by debuggers. Also see -debugDir. (defaults to OFF)
  -setProperty                 Set the values of a property in the form of propertyName=value1[,value2...].
  -style                       Script output style: DETAILED, OBFUSCATED or PRETTY (defaults to OBFUSCATED)
  -[no]failOnError             Fail compilation if any input file contains an error. (defaults to OFF)
  -[no]validateOnly            Validate all source code, but do not compile. (defaults to OFF)
  -sourceLevel                 Specifies Java source level (defaults to 1.8)
  -localWorkers                The number of local workers to use when compiling permutations
  -[no]incremental             Compiles faster by reusing data from the previous compile. (defaults to OFF)
  -war                         The directory into which deployable output files will be written (defaults to 'war')
  -deploy                      The directory into which deployable but not servable output files will be written (defaults to 'W
  -extra                       The directory into which extra files, not intended for deployment, will be written
  -saveSourceOutput            Overrides where source files useful to debuggers will be written. Default: saved with extras.
and
  module[s]                    Specifies the name(s) of the module(s) to compile
```

Story of your Compile Report (SoyC)



Bootstrap JS with GWT

- Host HTML file with JS to bootstrap the JS

```
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <link type="text/css" rel="stylesheet" href="Hello.css">
    <title></title>
  </head>
  <body>

<script type="text/javascript" language='javascript' src='hello/hello.nocache.js'></script>
    <!-- Along with page title and table headers defined -->
  </body>
</html>
```

Bootstrap JS with GWT

- Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transiti
<html>
  <body onload='alert("w00t!")'>
    <img src='bigImageZero.jpg'></img>
    <script source='externalScriptZero.js'></script>
    <img src='bigImageOne.jpg'></img>
    <img src='reallyBigImageTwo.jpg'></img>
    <script src='myApp/myApp.nocache.js'></script>
    <script src='externalScriptOne.js'></script>
  </body>
</html>
```

Bootstrap JS with GWT - Principles

- `<script>` tags always block evaluation of the page until the script is fetched and evaluated.
- `<img>` tags do not block page evaluation.
- Most browsers will allow a maximum of two simultaneous connections for fetching resources.
- The `body.onload()` event will only fire once all external resources are fetched, including images and frames.

Bootstrap JS with GWT - Principles

- The GWT selection script (i.e. `myApp/myApp.nocache.js`) will be fetched and evaluated like a normal script tag, but the compiled script will be fetched asynchronously.
- Once the GWT selection script has started, its `onModuleLoad()` can be called at any point after the outer document has been parsed.

Bootstrap JS with GWT - Example Sequence

- The HTML document is fetched and parsing begins.
- Begin fetching `bigImageZero.jpg`.
- Begin fetching `externalScriptZero.js`.
- `bigImageZero.jpg` completes (let's assume). Parsing is blocked until `externalScriptZero.js` is done fetching and evaluating.
- `externalScriptZero.js` completes.
- Begin fetching `bigImageOne.jpg` and `reallyBigImageTwo.jpg` simultaneously.
- `bigImageOne.jpg` completes (let's assume again). `myApp/myApp.nocache.js` begins fetching and evaluating.

Bootstrap JS with GWT - Example Sequence

- `myApp/myApp.nocache.js` completes, and the compiled script (`<hashname>.cache.js`) begins fetching in a hidden IFRAME (this is non-blocking).
- `<hashname>.cache.js` completes. `onModuleLoad()` is not called yet, as we're still waiting on `externalScriptOne.js` to complete before the document is considered 'ready'.
- `externalScriptOne.js` completes. The document is ready, so `onModuleLoad()` fires.
- `reallyBigImageTwo.jpg` completes.
- `body.onload()` fires, in this case showing an `alert()` box.

Bootstrap JS with GWT - Remember...

- You want to put the GWT selection script as early as possible within the body, so that it begins fetching the compiled script before other scripts (because it won't block any other script requests).
- If you are going to be fetching external images and scripts, you want to manage your two connections carefully.
- `<img>` tags are not guaranteed to be done loading when `onModuleLoad()` is called.
- `<script>` tags are guaranteed to be done loading when `onModuleLoad()` is called.

Emulated Java Runtime Environment

Emulated Java Runtime Environment (JRE)

- The Emulated JRE is by no means a full re-implementation of the Java JRE, but is rather a sort of selection of classes and methods that can be useful (and usable) client-side.
- The functionalities that are in the Java JRE but which you will not find inside the Emulated JRE fall into **three categories**.

Emulated Java Runtime Environment (JRE)

1. **Things that cannot be ported client-side**. For instance, java.lang.Thread or java.io.File cannot be implemented in a browser with the same semantics of Java. The browser page is single-threaded and has no direct access to the filesystem.
2. **Things that could be implemented** but that would “cost too much” in terms of code size, performance, or dependencies, and which the community thus prefers not to have inside GWT. Included in this category, for instance, is Java reflection (java.lang.reflect) which would require the transpiler to keep class information for each type, and that would cause the size of the compiled JavaScript to balloon.
3. **Things nobody had interest** in and therefore have not been implemented.

Emulated Java Runtime Environment (JRE)

- Write your own implementation using `<super-source>`
- List of all implemented JRE classes
  - http://www.gwtproject.org/doc/latest/DevGuideCodingBasicsCompatibility.html
  - http://www.gwtproject.org/doc/latest/RefJreEmulation.html

Interoperability Layer to JavaScript

Interoperability Layer to JS

- JSNI (Java Script Native Interface)
  - **Deprecated**
  - Similar to JNI (Java Native Interface)
- JsInterop (Java Script Interoperability)
  - From GWT 2.7.x
  - Annotation-based

JSNI

- Example

```
public static native void alert(String msg) /*-{
  $wnd.alert(msg);
}-*/;
```

JsInterop

- Example

```
@JsType(namespace = JsPackage.GLOBAL, name = "window", isNative = true)
public class Window {
    public static native void alert(String message);
}
```

Tools

Tools

- Web browser
  - Chrome with source maps support for debugging
  - FF with source maps support for debugging
- IDE
  - Eclipse / STS
    - With SDBG (Super Dev Mode Debugger) plugin: https://sdbg.github.io
  - IntelliJ
- Maven with GWT Maven Plugin
  - https://tbroyer.github.io/gwt-maven-plugin
- GWT Boot
  - One-stop-shop for all the GWT and related technology that you need without having to hunt through sample code and copy paste loads of dependency descriptors.

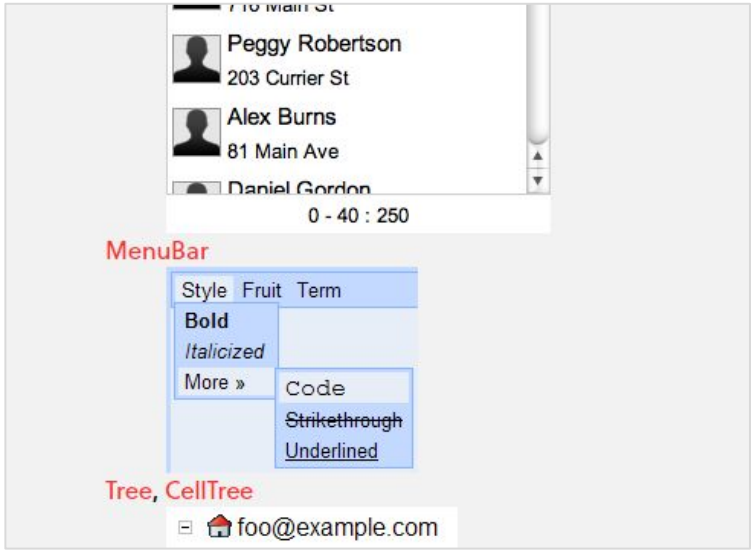
User Interface Components

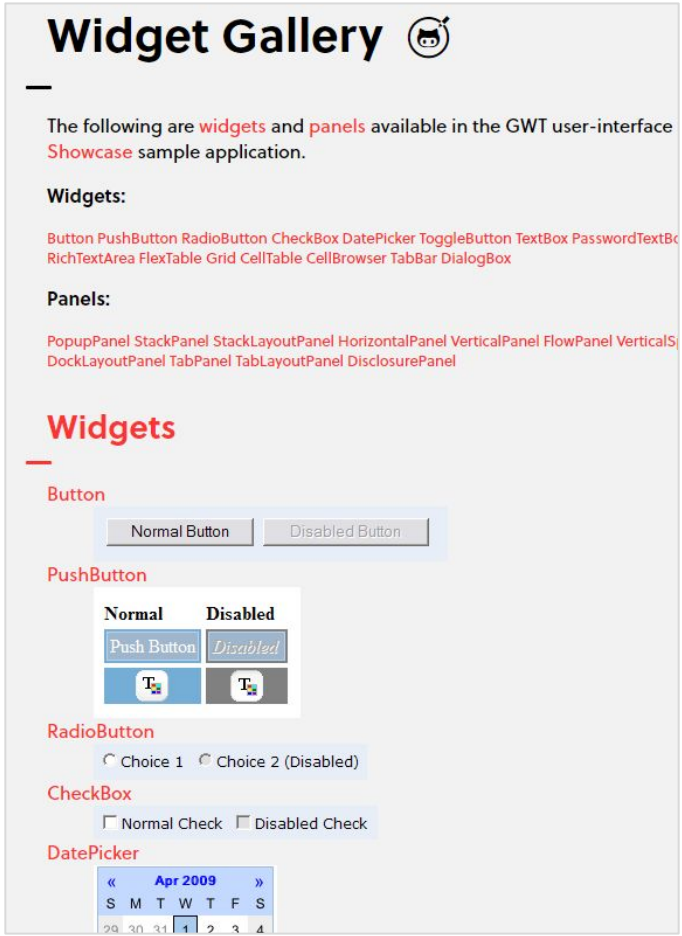
GWT UI Components

- Lot of choices
  - Standard GWT UI included: Panels and Widgets (Button, Grid, Tree, TextField, …)
    - Based on GWT standard UI:
    - GWTBootstrap3 (Open Source)
    - GWTMaterial (Open Source)
    - GXT (commercial)
    - ...
  - Based on new Browser Elemental (“to the metal”HTML5: DOM, WebGL, WebAudio, Shadow DOM, File API, ...)
    - DominoUI (Open Source)
    - VueGWT (Open Source)
    - GWTReact (Open Source)
    - Errai (Open Source)
    - ...

GWT Standard UI Components

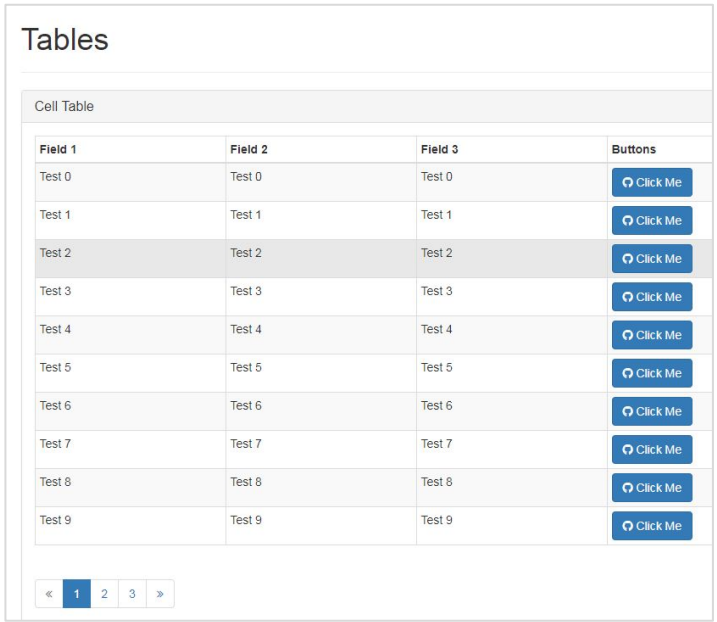
- Integrated in GWT libs
- [Widgets Dokumentation]

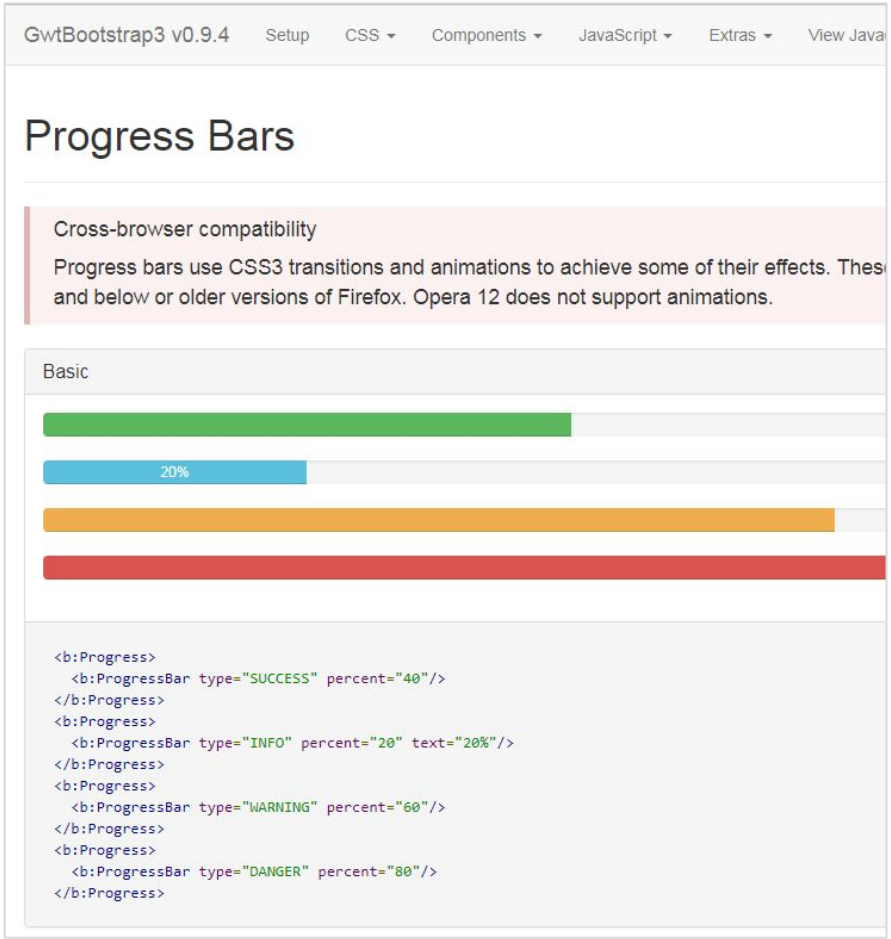




GWTBootstrap3

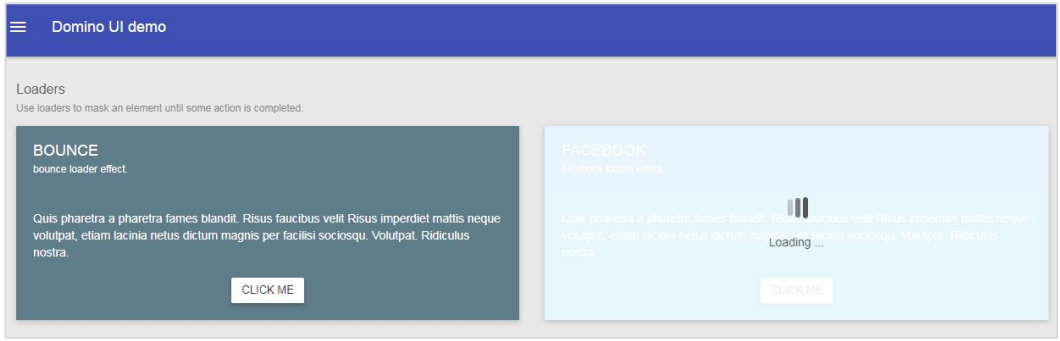
- [Widgets Dokumentation]

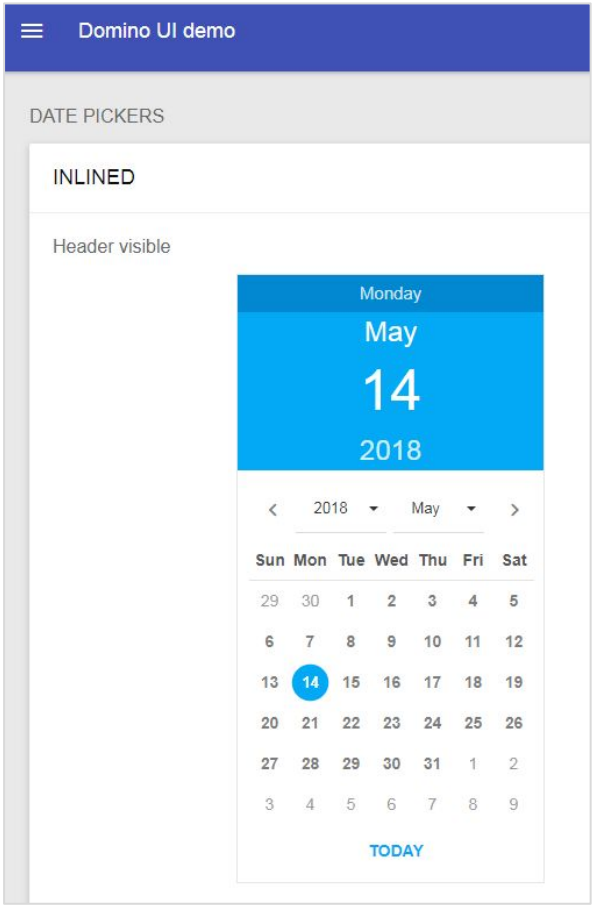




DominoUI

- [Widgets Dokumentation]





References

- https://dzone.com/articles/understanding-gwt-compiler
- http://www.gwtproject.org/doc/latest/DevGuideCompilingAndDebugging.html#DevGuideJavaToJavaScriptCompiler
- http://www.gwtproject.org/doc/latest/DevGuideOrganizingProjects.html#DevGuideBootstrap
- http://www.gwtproject.org/doc/latest/DevGuideCodeSplitting.html
- https://www.toptal.com/front-end/javascript-front-ends-in-java-with-gwt
- http://bit.ly/2InB0AW
- http://www.gwtproject.org/doc/latest/RefJreEmulation.html
- http://www.gwtproject.org/doc/latest/DevGuideCodingBasicsCompatibility.html
- http://www.gwtproject.org/doc/latest/DevGuideCompileReport.html
- http://www.gwtproject.org/articles/fragment_merging.html
- http://www.gwtproject.org/doc/latest/DevGuideCodingBasicsJSNI.html
- http://www.gwtproject.org/doc/latest/DevGuideCodingBasicsJsInterop.html
- http://www.g-widgets.com/2017/06/29/quick-tip-debugging-a-gwt-application-using-chrome-dev-tools
- https://www.sencha.com/products/gxt/#overview