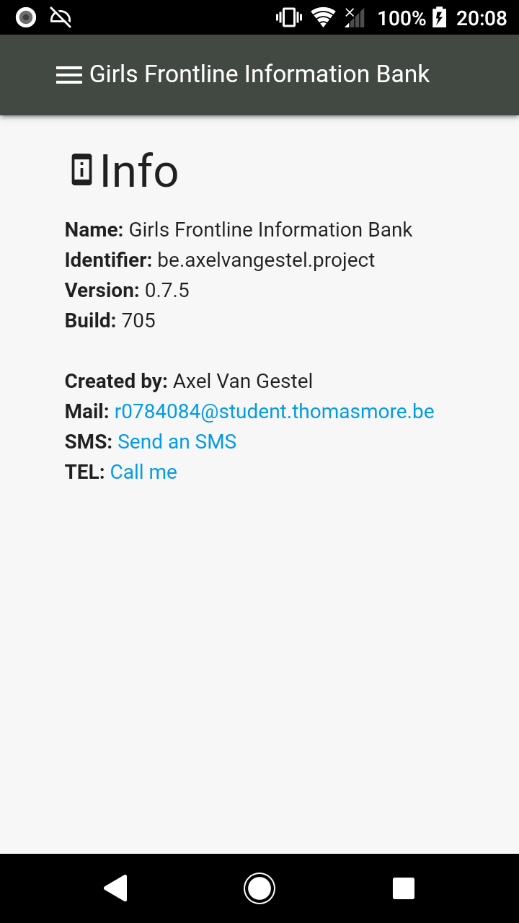
**Cordova-project 2APPAI – 2OHO - 3IoT**

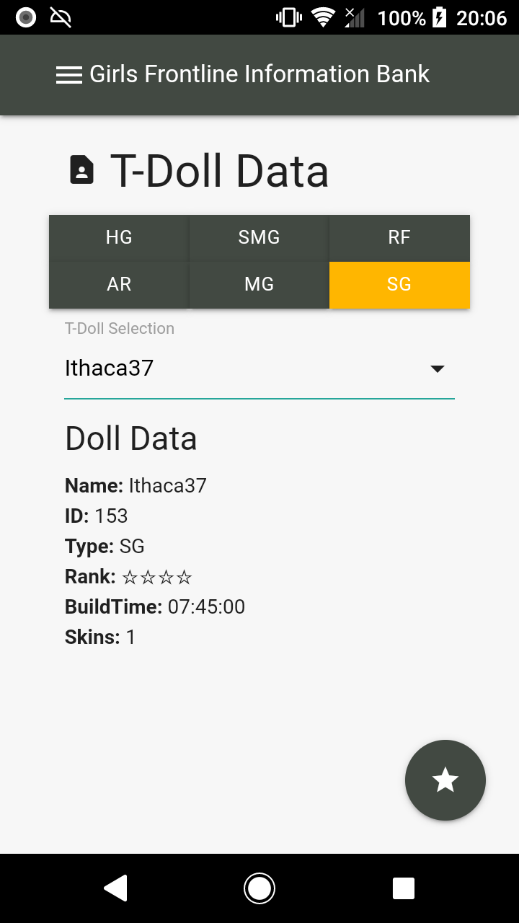
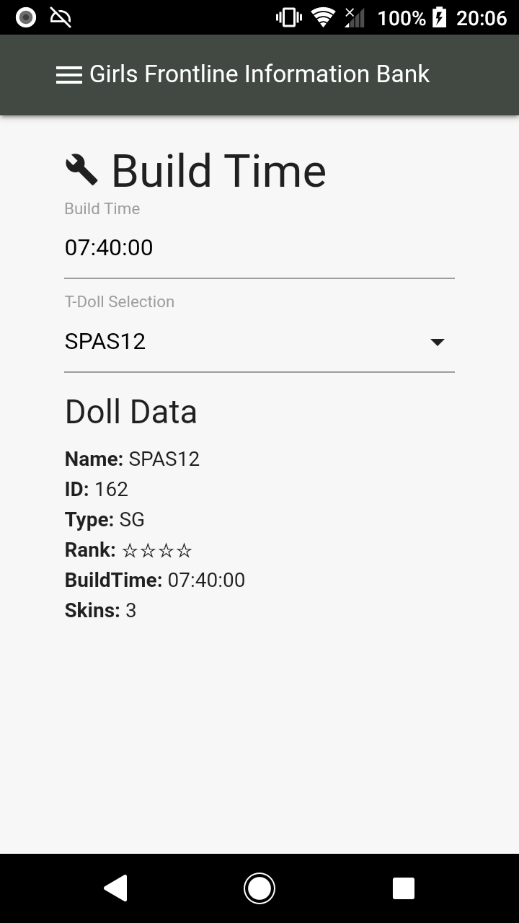
**Korte omschrijving van de app (vr 9 oktober)**

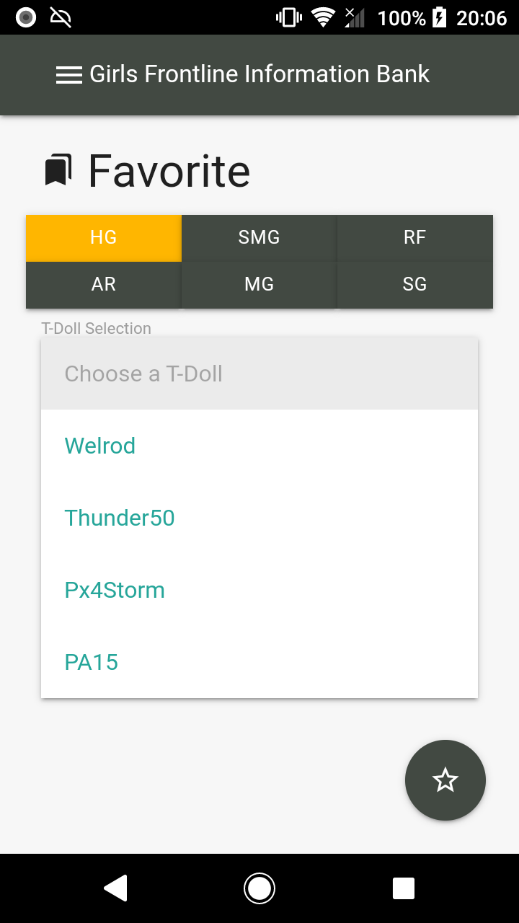
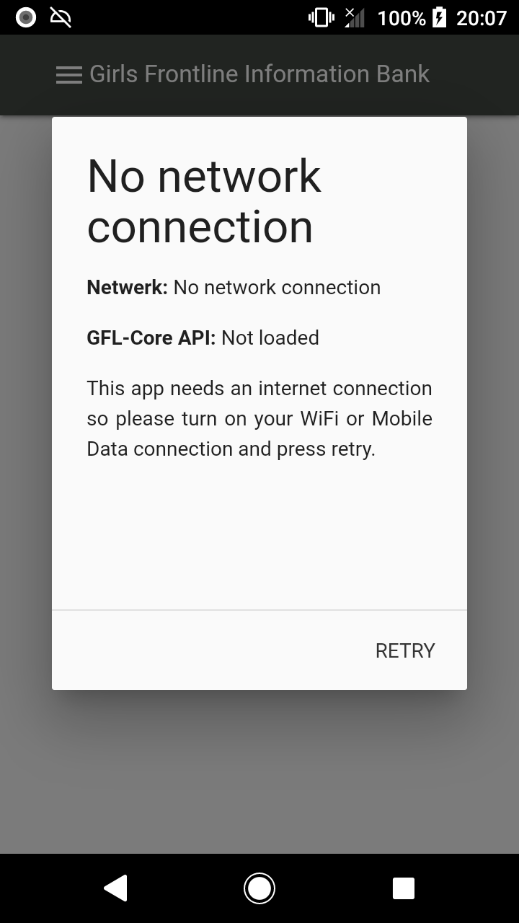
Mijn plan is om een app te ontwikkelen waarmee de gebruiker snel informatie (naam, stats, skills, skins,…) over een bepaald T-doll (character) van de game “Girls’ Frontline” kan opzoeken. Om dit efficiënt te laten verlopen kan ik gebruik maken van de “girlsfrontline-core API” die deze info beheert en up-to-date houd.

De gebruiker kan dan bepaalde T-doll’s opslaan in een lokale lijst, zodat hij snel toegang heeft tot hun informatie.

**Schermafbeeldingen (vr 29 oktober)**

Plaats hier een schermafbeelding van elk tabblad.





**Opsomming gebruikte Cordova plug-ins (vr 29 oktober)**

* **cordova-plugin-appinfo** (https://www.npmjs.com/package/cordova-plugin-appinfo)  
  Gebruikt voor de identifier, versie & build nummer van de applicatie op te halen, op deze manier gebeurt dit automatisch en hoef je geen html code te wijzigen bij een update of versie verandering.
* **cordova-plugin-network-information** (https://www.npmjs.com/package/@osvlabs/cordova-plugin-network-information)  
  Gebruikt voor het controleren of de app online kan tijdens het opstarten, zo niet dan krijgt de gebruiker een melding dat deze app internet toegang nodig heeft om correct te werken.
* **skwas-cordova-plugin-datetimepicker** (https://www.npmjs.com/package/skwas-cordova-plugin-datetimepicker)  
  Gebruikt voor de input van de Build Time, deze plugin geeft me meer vrijheid over hoe de time picker eruit ziet en functioneert tegenover de standaard MaterializeCSS time picker.
* **cordova-plugin-splashscreen** (https://www.npmjs.com/package/cordova-plugin-splashscreen)  
  Gebruikt voor de pop-in te verbergen, terwijl de app alles aan het inladen is bij startup.

**Infopagina over de app (zo 22 november)**

<https://vangestelaxel.sinners.be/2APPAI1/cordova/cordova_project>

**Bijlage / broncode (zo 22 novemner)**

Voeg de broncode van elk document (HTML, CSS, eigen JS, … ) toe als bijlage.

**Index.html**

<!doctype *html*>  
<html *lang*="nl">  
<head>  
 <meta *charset*="UTF-8">  
 <meta *name*="viewport"  
 *content*="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <link *rel*="stylesheet" *href*="https://fonts.googleapis.com/icon?family=Material+Icons">  
 <link *rel*="stylesheet" *href*="css/materialize.1.0.0.min.css">  
 <link *rel*="stylesheet" *href*="css/app.css">  
 <title>Girls' Frontline Information Bank</title>  
</head>  
<body *class*="white\_glf">  
<header *class*="navbar-fixed">  
 <nav *class*="grey\_gfl">  
 <div *class*="nav-wrapper container">  
 <a *href*="#!" *class*="brand-logo center">Girls'&nbsp;Frontline&nbsp;Information&nbsp;Bank</a>  
 <a *href*="#!" *data-target*="nav-mobile" *class*="sidenav-trigger show-on-large">  
 <i *class*="material-icons">menu</i>  
 </a>  
 </div>  
 </nav>  
</header>  
<main *class*="container">  
  
 *<!-- Fixed sideNav -->* <ul *id*="nav-mobile" *class*="sidenav">  
 *<!-- sideNav title -->* <li><div *class*="user-view grey\_gfl white\_glf-text">  
 <p *class*="name">Girls' Frontline Information Bank</p>  
 <p *class*="email">GFIB ver. <span *id*="APP\_info\_version"></span></p>  
 </div></li>  
 *<!-- sideNav Tab 1 -->* <li><a *href*="#!" *data-show*="tabTDollData" *class*="waves-effect"><i *class*="material-icons">contact\_page</i>T-Doll Data</a></li>  
 <li><div *class*="divider"></div></li>  
 *<!-- sideNav Tab 2 -->* <li><a *href*="#!" *data-show*="tabBuildTime" *class*="waves-effect"><i *class*="material-icons">build</i>Build Time</a></li>  
 <li><div *class*="divider"></div></li>  
 *<!-- sideNav Tab 3 -->* <li><a *href*="#!" *data-show*="tabFavorite" *class*="waves-effect"><i *class*="material-icons">bookmarks</i>Favorite</a></li>  
 <li><div *class*="divider"></div></li>  
 *<!-- sideNav Tab 4 -->* <li><a *href*="#!" *data-show*="tabSettings" *class*="waves-effect"><i *class*="material-icons">settings</i>Settings</a></li>  
 *<!-- sideNav Tab 5 -->* <li><a *href*="#!" *data-show*="tabAbout" *class*="waves-effect"><i *class*="material-icons">perm\_device\_information</i>About</a></li>  
 </ul>  
  
  
 *<!-- Grid -->* <div *class*="container">  
 *<!-- Main Tab 1 - T-Doll Data -->* <div *class*="row spa" *id*="tabTDollData">  
 <div *class*="col s12">  
 <h4><i *class*="material-icons">contact\_page</i> T-Doll Data</h4>  
 <div *id*="btns\_doll\_type" *class*="row">  
 <a *data-type*="HG" *class*="waves-effect waves-light btn-small grey\_gfl col s4 active">HG</a>  
 <a *data-type*="SMG" *class*="waves-effect waves-light btn-small grey\_gfl col s4">SMG</a>  
 <a *data-type*="RF" *class*="waves-effect waves-light btn-small grey\_gfl col s4">RF</a>  
 <a *data-type*="AR" *class*="waves-effect waves-light btn-small grey\_gfl col s4">AR</a>  
 <a *data-type*="MG" *class*="waves-effect waves-light btn-small grey\_gfl col s4">MG</a>  
 <a *data-type*="SG" *class*="waves-effect waves-light btn-small grey\_gfl col s4">SG</a>  
 </div>  
  
 <div *class*="input-field">  
 <select *id*="tdoll\_selection">  
 <option *value*="" *disabled selected*>Choose a Type</option>  
 </select>  
 <label *for*="tdoll\_selection">T-Doll Selection</label>  
 </div>  
 <h5>Doll Data</h5>  
 <div *id*="Doll\_Data"></div>  
 </div>  
  
  
 *<!-- Floating Favorite Button -->* <div *id*="btnFloat\_favorite" *class*="fixed-action-btn">  
 <a *class*="btn-floating btn-large grey\_gfl">  
 <i *class*="large material-icons">star</i>  
 </a>  
 </div>  
 </div>  
  
  
 *<!-- Main Tab 2 - Build Time -->* <div *class*="row spa" *id*="tabBuildTime">  
 <div *class*="col s12">  
 <h4><i *class*="material-icons">build</i> Build Time</h4>  
  
 <div *class*="input-field">  
 <input *id*="tdoll\_BuildTime" *type*="text" *class*="" *step*="1" *pattern*="[0-9]{2}:[0-9]{2}:[0-9]{2}" *value*="00:20:00" *placeholder*="HH:MM:SS" *maxlength*="8" *minlength*="8" *required*>  
 <label *for*="tdoll\_BuildTime">Build Time</label>  
 </div>  
 <div *class*="input-field">  
 <select *id*="tdoll\_selection\_BuildTime">  
 <option *value*="" *disabled selected*>Choose a Type</option>  
 </select>  
 <label *for*="tdoll\_selection\_BuildTime">T-Doll Selection</label>  
 </div>  
 <h5>Doll Data</h5>  
 <div *id*="Doll\_Data\_BuildTime"></div>  
 </div>  
  
  
 *<!-- Floating Favorite Button -->* <div *id*="btnFloat\_favorite\_buildTime" *class*="fixed-action-btn">  
 <a *class*="btn-floating btn-large grey\_gfl">  
 <i *class*="large material-icons">star</i>  
 </a>  
 </div>  
 </div>  
  
  
 *<!-- Main Tab 3 - Favorite -->* <div *class*="row spa" *id*="tabFavorite">  
 <div *class*="col s12">  
 <h4><i *class*="material-icons">bookmarks</i> Favorite</h4>  
 <div *id*="btns\_favorite" *class*="row">  
 <a *data-type*="HG" *class*="waves-effect waves-light btn-small grey\_gfl col s4 active">HG</a>  
 <a *data-type*="SMG" *class*="waves-effect waves-light btn-small grey\_gfl col s4">SMG</a>  
 <a *data-type*="RF" *class*="waves-effect waves-light btn-small grey\_gfl col s4">RF</a>  
 <a *data-type*="AR" *class*="waves-effect waves-light btn-small grey\_gfl col s4">AR</a>  
 <a *data-type*="MG" *class*="waves-effect waves-light btn-small grey\_gfl col s4">MG</a>  
 <a *data-type*="SG" *class*="waves-effect waves-light btn-small grey\_gfl col s4">SG</a>  
 </div>  
  
 <div *class*="input-field">  
 <select *id*="tdoll\_selection\_favorite">  
 <option *value*="" *disabled selected*>Choose a Type</option>  
 </select>  
 <label *for*="tdoll\_selection\_favorite">T-Doll Selection</label>  
 </div>  
 <h5>Doll Data</h5>  
 <div *id*="Doll\_Data\_favorite"></div>  
 </div>  
 *<!-- Floating Favorite Button -->* <div *id*="btnFloat\_Unfavorite" *class*="fixed-action-btn">  
 <a *class*="btn-floating btn-large grey\_gfl">  
 <i *class*="large material-icons">star\_outline</i>  
 </a>  
 </div>  
 </div>  
  
  
 *<!-- Main Tab 4 - Settings -->* <div *class*="row spa" *id*="tabSettings">  
 <div *class*="col s12">  
 <h4><i *class*="material-icons">settings</i> Settings</h4>  
 <div *class*="input-field">  
 <select *id*="settings\_sorting\_method">  
 <option *value*="" *disabled*>Choose a Sorting Method</option>  
 </select>  
 <label *for*="settings\_sorting\_method">Sorting Method</label>  
 </div>  
 </div>  
 </div>  
  
  
 *<!-- Main Tab 5 - About -->* <div *class*="row spa" *id*="tabAbout">  
 <div *class*="col s12">  
 <div *class*="center-align">  
 <h4><i *class*="material-icons">perm\_device\_information</i>About</h4>  
 <img *class*="responsive-img" *src*="img/icon.png" *alt*="APP Icon" *style*="max-width: 100px">  
 <h6 *style*="margin-bottom: 0; margin-top: 3px">Girls' Frontline Information Bank</h6>  
 <div *id*="APP\_info\_all">  
 <p *class*="center-align" *style*="margin-top:0; margin-bottom: 0">Identifier Not Found</p>  
 <p *class*="center-align" *style*="margin-top: 0">Ver. x.x.x (Build xxxx)</p>  
 </div>  
 <a *href*="https://vangestelaxel.sinners.be/2APPAI1/cordova/cordova\_project/#download" *target*="\_blank" *class*="waves-effect waves-light btn-small grey\_gfl">Check for updates</a>  
  
 <div *style*="margin-top: 25px">  
 <h6 *class*="center-align" *style*="margin-bottom: 3px">Created by</h6>  
 <p *class*="center-align" *style*="margin-bottom: 0; margin-top: 0">Axel Van Gestel</p>  
 <p *class*="center-align" *style*="margin-bottom: 0; margin-top: 0">Mail: <a *href*="mailto:r0784084@student.thomasmore.be">r0784084@student.thomasmore.be</a></p>  
 <p *class*="center-align" *style*="margin-bottom: 0; margin-top: 0">SMS: <a *href*="sms:+32483060747">+32483060747</a></p>  
 <p *class*="center-align" *style*="margin-top: 0">TEL: <a *href*="tel:+32483060747">+32483060747</a></p>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
  
  
  
  
 *<!-- Network Modal -->* <div *id*="NetworkModal" *class*="modal modal-fixed-footer">  
 <div *class*="modal-content">  
 <h4>No network connection</h4>  
 <p *id*="networkState" *class*="email"><b>Netwerk: </b>No Data</p>  
 <p *id*="GFLC\_API\_Loaded" *class*="email"><b>GFL-Core API: </b>Not loaded</p>  
 <p>This app needs an internet connection to work so make sure your Wi-Fi or Mobile Data is on and press retry.</p>  
 </div>  
 <div *class*="modal-footer">  
 <a *id*="NetworkModal\_Retry" *class*="waves-effect waves-green btn-flat">Retry</a>  
 </div>  
 </div>  
</main>  
  
  
  
  
<script *src*="js/jquery.3.5.1min.js"></script>  
<script *src*="js/materialize.1.0.0.min.js"></script>  
<script *crossorigin src*="https://unpkg.com/girlsfrontline-core/umd/gfcore.min.js"></script> *<!-- Needed for API to work -->*<script *src*="https://cdnjs.cloudflare.com/ajax/libs/i18next/19.8.4/i18next.min.js" *crossorigin*="anonymous"></script> *<!-- Needed for API to work -->*<script *src*="https://cdnjs.cloudflare.com/ajax/libs/i18next-xhr-backend/3.2.2/i18nextXHRBackend.min.js" *crossorigin*="anonymous"></script> *<!-- Needed for API to work -->*<script *src*="cordova.js"></script>  
<script *defer src*="js/app.js"></script>  
*<!-- --- Custom Scripts --- -->*<script *defer src*="js/sorting\_methods.js"></script>  
<script *defer src*="js/app\_info.js"></script>  
<script *defer src*="js/Network\_State.js"></script>  
<script *defer src*="js/GirlsFrontlineCoreAPI.js"></script>  
<script *defer src*="js/MaterialDateTimePicker\_SKWAS.js"></script>  
<script *defer src*="js/Settings.js"></script>  
</body>  
</html>  
  
  
*<!-- TODO: (optional) Add Welcome page -->  
<!-- TODO: (optional) Add option to purge favorites -->  
<!-- TODO: (optional) Add T-Doll Skill display -->*

**app.css**

*/\* ------ Variables: colors,... ------ \*/*:*root* {  
 --button\_padding: 0px;  
 --orange\_gfl-color: #ffb600;  
 --white\_glf-color: #f7f7f7;  
 --light\_grey\_gfl-color: #cecfd6;  
 --grey\_gfl-color: #424942;  
}  
  
  
*/\* --- Navbar CCS --- \*/*.*spa* { display: none; }  
*#tabTDollData* { display: block; }  
  
  
a.*brand-logo* {  
 font-size: 1.2rem;  
}  
  
p {  
 text-align: justify;  
}  
  
.*user-view* p {  
 margin-bottom: 0;  
 margin-top: 0;  
}  
  
.*user-view* img {  
 max-width: 64px;  
 max-height: 64px;  
}  
  
*#tabTDollData*, *#tabBuildTime*, *#tabFavorite* {  
 padding-bottom: var(--button\_padding);  
}  
  
  
*/\* --- Tab CSS --- \*/*.*btn-small* {  
 border-radius: 0;  
}  
  
*#tabTDollData* .*active*, *#tabFavorite* .*active* {  
 *!important*;background-color: var(--orange\_gfl-color);  
}  
  
.*btn-small*:*focus*, .*btn-small*:*hover*, .*btn-large*:*focus*, .*btn-large*:*hover* {  
 background-color: var(--orange\_gfl-color);  
}  
  
  
.*orange\_gfl* {  
 background-color: var(--orange\_gfl-color);  
}  
.*orange\_gfl-text* {  
 color: var(--orange\_gfl-color);  
}  
  
.*white\_glf* {  
 background-color: var(--white\_glf-color);  
}  
.*white\_glf-text* {  
 color: var(--white\_glf-color);  
}  
  
.*light\_grey\_glf* {  
 background-color: var(--light\_grey\_gfl-color);  
}  
.*light\_grey\_glf-text* {  
 color: var(--light\_grey\_gfl-color);  
}  
  
.*grey\_gfl* {  
 background-color: var(--grey\_gfl-color);  
}  
.*grey\_gfl-text* {  
 color: var(--grey\_gfl-color);  
}  
  
*/\* ----- Override Materialize CSS ----- \*/*.*toast* {  
 justify-content: center;  
}  
  
table {  
 border-collapse: separate;  
}  
td, th {  
 padding: 0;  
}  
  
  
  
*/\* T-Doll tile buff Table \*/*.*tile\_grid\_table* {  
 background: var(--grey\_gfl-color);  
 padding: 5px;  
 border-collapse: separate;  
 border-spacing: 2px;  
}  
.*tile\_grid\_table* td {  
 background: #6b696b;  
 width: 30%;  
 padding-bottom : 30%; */\* = width for a 1:1 aspect ratio \*/* border-radius: 0;  
}  
.*tile\_grid\_table* td.*buff* {  
 background: #00ffde;  
}  
.*tile\_grid\_table* td.*standing* {  
 background: #ffffff;  
}

**app.js**

$(*function* () {  
 document.addEventListener("deviceready", *onDeviceReady*, *false*);  
  
  
 *// ---------- Button Stuff ----------  
 // SideNav initialisation* $('.*sidenav*').sidenav(); */\* https://materializecss.com/sidenav.html \*/* $('.*sidenav* a').click(*function* () {  
 $('.*spa*').hide(); *// Alles met de tag "spa" wordt gehide* $('#' + $(*this*).data('show')).show(); *// De huidige link dat je klikt wordt geshowed* $('.*sidenav*').sidenav('close'); *// De navbar sluit zichzelf* });  
  
  
 *// Form Selection Initialization (dropdowns)* $('select').formSelect();  
});  
  
  
*// ---------- Function Stuff ----------  
function onDeviceReady*() {  
 console.log('Device is ready');  
 Settings.init(); *// Set Settings for the APP* AppInfo.init(); *// Get & Set the APP Info* NetworkState.init(); *// Get the Network State on Launch* GirlsFrontlineCoreAPI.init();  
 MaterialDateTimePicker.init();  
}

**app\_info.js**

*let* AppInfo = *function* () {  
 *// ---------- Global Variables & Stuff ----------  
 // Cache DOM for performance  
 let* $tabAbout = $('#tabAbout');  
 *let* $APP\_info\_all = $tabAbout.find('#APP\_info\_all');  
 *let* $APP\_info\_identifier = $('#APP\_info\_identifier');  
 *let* $APP\_info\_version = $('#APP\_info\_version');  
 *let* $APP\_info\_build = $('#APP\_info\_build');  
  
  
  
  
  
 *// ---------- Function Stuff ----------  
 let* init = *function* () {  
 *// Get APP data  
 let* data = `  
 <p class="center-align" style="margin-top:0; margin-bottom: 0">${navigator.appInfo.identifier}</p>  
 <p class="center-align" style="margin-top: 0">Ver. ${navigator.appInfo.version} (Build ${navigator.appInfo.build})</p>  
 `;  
  
 *// Set APP data as HTML* $APP\_info\_all.html(data);  
 $APP\_info\_identifier.text(navigator.appInfo.identifier);  
 $APP\_info\_version.text(navigator.appInfo.version);  
 $APP\_info\_build.text(navigator.appInfo.build);  
 };  
  
  
  
 *// ---------- Global Function returns (outside name : inside name) ----------  
 return* {  
 init: init,  
 };  
}();

**GirlsFrontlineCoreAPI.js**

*let* GirlsFrontlineCoreAPI = *function* () {  
 *// ---------- Global Variables & Stuff ----------  
 // TODO: dynamically populate type buttons  
 const* doll\_types = ['hg', 'smg', 'rf', 'ar', 'mg', 'sg']  
 *const* doll\_stat\_types = {  
 armor : 'ARMOR',  
 armorPiercing : 'AP',  
 criticalPercent : 'CRIT',  
 dodge : 'EVA',  
 hit : 'ACC',  
 hp : 'HP',  
 pow : 'DMG',  
 rate : 'ROF',  
 speed : 'MOBILITY',  
 cooldown : 'COOLDOWN',  
 }  
  
 *// Settings  
 let* setting\_sorting\_method;  
  
 *// Local Storage: Favorite dolls list  
 let* favorite\_doll\_ids = [];  
  
 *// Cache last selected Type  
 let* selected\_type = undefined;  
 *let* selected\_type\_favorited = undefined;  
  
 *// Cache DOM for performance  
 let* $tabTDollData = $('#tabTDollData');  
 *let* $tabBuildTime = $('#tabBuildTime');  
 *let* $tabFavorite = $('#tabFavorite');  
 *let* $tdoll\_selection = $tabTDollData.find('#tdoll\_selection');  
 *let* $btns\_doll\_type = $tabTDollData.find('#btns\_doll\_type');  
 *let* $btnFloat\_favorite = $tabTDollData.find('#btnFloat\_favorite');  
 *let* $tdoll\_selection\_favorite = $tabFavorite.find('#tdoll\_selection\_favorite');  
 *let* $btns\_favorite = $tabFavorite.find('#btns\_favorite');  
 *let* $btnFloat\_Unfavorite = $tabFavorite.find('#btnFloat\_Unfavorite');  
 *let* $tdoll\_BuildTime = $tabBuildTime.find('#tdoll\_BuildTime');  
 *let* $tdoll\_selection\_BuildTime = $tabBuildTime.find('#tdoll\_selection\_BuildTime');  
 *let* $btnFloat\_favorite\_buildTime = $tabBuildTime.find('#btnFloat\_favorite\_buildTime');  
 *let* $Doll\_Data = $tabTDollData.find('#Doll\_Data');  
 *let* $Doll\_Data\_favorite = $tabFavorite.find('#Doll\_Data\_favorite');  
 *let* $Doll\_Data\_BuildTime = $tabBuildTime.find('#Doll\_Data\_BuildTime');  
  
  
 *// Setup i18next* i18next.use(i18nextXHRBackend).init({  
 fallbackLng: 'ko-KR',  
 lng: 'en-US',  
 load: 'currentOnly',  
 ns: ['gfcore'],  
 whitelist: ['ko-KR', 'ja-JP', 'en-US', 'zh-CN'],  
 backend: {  
 loadPath: 'https://unpkg.com/girlsfrontline-core@2.1.5-beta9/build/i18n/{{lng}}/{{ns}}.json',  
 crossDomain: *true*,  
 },  
 });  
  
  
  
  
  
 *// ---------- Button Stuff ----------  
 // --- T-Doll Data tab ---  
 // Get selected T-Doll data from dropdown* $tdoll\_selection.on('change', *function*() {  
 *let* id = *this*.value;  
 *// console.log("Dropdown: Value = ", this.value);  
 if* (id !== "") {  
 \_render\_html\_doll\_data(*parseInt*(id))  
 }  
 })  
  
  
 *// T-Doll type button array selection (for dropdown)* $btns\_doll\_type.children().on('click', *function*() {  
 *let* $btn\_pressed = $(*this*); *// Cache pressed button  
 let* doll\_type = $btn\_pressed.data('type');  
 *// console.log('Button: type = ', doll\_type);* $btn\_pressed.siblings().addBack().removeClass("active");  
 $btn\_pressed.addClass("active");  
  
 get\_dolls\_by\_type(doll\_type.toLowerCase());  
 reset\_html\_doll\_data();  
 })  
  
  
 *// Get selected T-Doll data from dropdown and add to Favorites* $btnFloat\_favorite.on('click', *function*() {  
 *let* id = $tdoll\_selection.val();  
 *// console.log("Favorite Button: adding ID = ", id)  
 if* (id !== *null*) {  
 \_addFavoriteDoll(*parseInt*(id));  
 }  
 *else* {  
 console.error("Favorite Button: " + id + " is NaN!");  
 M.toast({html: 'No T-Doll selected to Favorite!', displayLength: 2000, classes: 'red accent-4'})  
 }  
 })  
  
  
 *// Get selected T-Doll data from dropdown and add to Favorites* $btnFloat\_favorite\_buildTime.on('click', *function*() {  
 *let* id = $tdoll\_selection\_BuildTime.val();  
 *// console.log("Favorite Button: adding ID = ", id)  
 if* (id !== *null*) {  
 \_addFavoriteDoll(*parseInt*(id));  
 }  
 *else* {  
 console.error("Favorite Button: " + id + " is NaN!");  
 M.toast({html: 'No T-Doll selected to Favorite!', displayLength: 2000, classes: 'red accent-4'})  
 }  
 })  
  
  
 *// --- Favorite Tab ---  
 // Get selected T-Doll data from favorites dropdown* $tdoll\_selection\_favorite.on('change', *function*() {  
 *let* id = *this*.value;  
 *// console.log("Dropdown: Value = ", this.value);  
 if* (id !== "") {  
 \_render\_html\_doll\_data(*parseInt*(id),*true*)  
 }  
 })  
  
  
 *// T-Doll type button array favorites selection (for dropdown)* $btns\_favorite.children().on('click', *function*() {  
 *let* $btn\_pressed = $(*this*); *// Cache pressed button  
 let* doll\_type = $btn\_pressed.data('type');  
 *// console.log('Button: type = ', doll\_type);* $btn\_pressed.siblings().addBack().removeClass("active");  
 $btn\_pressed.addClass("active");  
  
 get\_dolls\_by\_type(doll\_type.toLowerCase(), *true*);  
 reset\_html\_doll\_data(*true*);  
 })  
  
  
 *// Get selected T-Doll data from dropdown and add to Favorites* $btnFloat\_Unfavorite.on('click', *function*() {  
 *let* id = $tdoll\_selection\_favorite.val();  
 *// console.log("UnFavorite Button: removing ID = ", id);  
 if* (id !== *null*) {  
 \_deleteFavoriteDoll(*parseInt*(id));  
 }  
 *else* {  
 console.error("UnFavorite Button: " + id + " is NaN!");  
 M.toast({html: 'No T-Doll selected to Unfavorite!', displayLength: 2000, classes: 'red accent-4'})  
 }  
 })  
  
  
 *// --- Build Time tab ---  
 // Open TimePicker on clicking the Build Time text field* $tdoll\_BuildTime.on('click', *function*() {  
 MaterialDateTimePicker.showTimePicker()  
 })  
  
  
 *// Get selected T-Doll data from dropdown* $tdoll\_selection\_BuildTime.on('change', *function*() {  
 *let* id = *this*.value;  
 *// console.log("Dropdown: Value = ", this.value);  
 if* (id !== "") {  
 \_render\_html\_doll\_data(*parseInt*(id), undefined, *true*)  
 }  
 })  
  
  
  
 *// ---------- Function Stuff ----------  
 // initialise the systems using the API  
 let* init = *function* () {  
 *// Read Favorite dolls from local storage  
 let* favorite\_doll\_ids\_str = localStorage.getItem('favorite\_doll\_ids');  
 *if* (favorite\_doll\_ids\_str !== *null*) {  
 favorite\_doll\_ids = []; *// Empty array* favorite\_doll\_ids = JSON.parse(favorite\_doll\_ids\_str);  
 }  
  
  
 get\_dolls\_by\_type(); *// Get T-Doll by Type & Set the T-Doll Dropdown* get\_dolls\_by\_type(undefined,*true*); *// Get T-Doll by Type and Favorite & Set the T-Doll Favorites Dropdown* reset\_html\_doll\_data();  
 reset\_html\_doll\_data(*true*,undefined);  
 reset\_html\_doll\_data(undefined,*true*);  
 };  
  
  
 *// Get a list of all T-Dolls of a certain Type  
 let* get\_dolls\_by\_type = *function* (*input\_type* = 'hg', *favorite* = *false*) {  
 *// console.log("Input: Type = " + input\_type + ",", "Favorite = " + favorite);  
 try* {  
 *if* (doll\_types.includes(*input\_type*)) {  
 *let* dolls\_by\_type = []  
 gfcore.dolls.forEach(*function* (*tdoll*) {  
 *switch* (*favorite*) {  
 *case true*:  
 selected\_type\_favorited = *input\_type*;  
 *if* (*tdoll*.type === *input\_type* && favorite\_doll\_ids.includes(*tdoll*.id)) {  
 *// console.log(tdoll.type + " - " + tdoll.codename)* dolls\_by\_type.push([*tdoll*.id, *tdoll*.codename, *tdoll*.rank, *tdoll*.buildTime]);  
 }  
 *break*;  
 *default*:  
 selected\_type = *input\_type*;  
 *if* (*tdoll*.type === *input\_type*) {  
 *// console.log(tdoll.type + " - " + tdoll.codename, tdoll)* dolls\_by\_type.push([*tdoll*.id, *tdoll*.codename, *tdoll*.rank, *tdoll*.buildTime]);  
 }  
 }  
 });  
  
 *// console.log("T-Doll of type = " + input\_type, dolls\_by\_type);* dolls\_by\_type.sort(setting\_sorting\_method);  
 \_set\_doll\_selection\_dropdown(dolls\_by\_type, *favorite*);  
 }  
 }  
 *catch* (*err*) {  
 console.error("get\_dolls\_by\_type: Failed:", *err*)  
 }  
 }  
  
  
 *// Get a List of all T-Dolls with a certain Build Time  
 let* get\_dolls\_by\_buildTime = *function* (*input\_buildTime* = 1200) {  
 *// console.log("Build Time = " + input\_buildTime);  
 try* {  
 *let* dolls\_by\_buildTime = []  
 gfcore.dolls.forEach(*function* (*tdoll*) {  
 *if* (*tdoll*.buildTime === *input\_buildTime* && *tdoll*.id < 20000 && *tdoll*.rank !== 7) {  
 *// console.log(tdoll.buildTime + " - " + tdoll.codename)* dolls\_by\_buildTime.push([*tdoll*.id, *tdoll*.codename, *tdoll*.rank, *tdoll*.buildTime]);  
 }  
 }  
 );  
  
 *// console.log("dolls\_by\_buildTime = ", dolls\_by\_buildTime)  
 if* (dolls\_by\_buildTime.length === 0) {  
 M.toast({html: 'No T-Dolls found with selected Build Time.', displayLength: 2000, classes: 'grey\_gfl'});  
 }  
 dolls\_by\_buildTime.sort(setting\_sorting\_method);  
 \_set\_doll\_selection\_dropdown(dolls\_by\_buildTime, undefined, *true*);  
 }  
 *catch* (*err*) {  
 console.error("get\_dolls\_by\_buildTime: Failed:", *err*)  
 }  
 }  
  
  
 *// TODO: favorite & buildTime booleans to Selector  
 // Sets the T-Doll Dropdown content  
  
 let* \_set\_doll\_selection\_dropdown = *function* (*input\_doll\_list*, *favorite* = *false*, *buildTime* = *false*) {  
 *let* selector = undefined;  
  
 *if* (*favorite* === *true*) {  
 selector = $tdoll\_selection\_favorite;  
 }  
 *else if* (*buildTime* === *true*) {  
 selector = $tdoll\_selection\_BuildTime;  
 }  
 *else* {  
 selector = $tdoll\_selection;  
 }  
  
 selector.empty() *// Empty current dropdown list* .append("<option value='' disabled selected>Choose a T-Doll</option>");  
 *input\_doll\_list*.forEach(*function* (*doll*) { *// Dynamically add Dolls to the list* selector.append("<option value='" + *doll*[0] + "'>" + *doll*[1] + "</option>");  
 })  
 *// Form Selection ReInitialization* $('select').formSelect();  
 }  
  
  
 *// Function to hold all the parsing functions  
 let* \_parsers = *function* () {  
 *// Rank conversion (nr --> stars)  
 let* parse\_rank = *function* (*rank*) {  
 *if* (*rank* === 7) {  
 *return* "&#10029;" *// Special* } *else* {  
 *return* "&#9733;".repeat(*rank*);  
 }  
 }  
  
 *// Digimind conversion to table  
 let* parse\_digimind = *function* (*mindupdate*) {  
 *// Check if mindupdate is 'undefined', if so set to No  
 if* (*mindupdate* === undefined) {  
 *return* "No";  
 } *else* {  
 *// console.log("Digimind = ", digimind\_upgrade)  
 return* `  
 <table>  
 <tr>  
 <td><b>Mod 1:</b></td>  
 <td><b>Mod 2:</b></td>  
 <td><b>Mod 3:</b></td>  
 </tr>  
 <tr>  
 <td>Cores: ${*mindupdate*[0].core}</td>  
 <td>Cores: ${*mindupdate*[1].core}</td>  
 <td>Cores: ${*mindupdate*[2].core}</td>  
 </tr>  
 <tr>  
 <td>Fragments: ${*mindupdate*[0].mempiece}</td>  
 <td>Fragments: ${*mindupdate*[1].mempiece}</td>  
 <td>Fragments: ${*mindupdate*[2].mempiece}</td>  
 </tr>  
 </table>  
 `;  
 }  
 }  
  
  
 *// Convert buildTime (seconds) to String (HH:MM:SS)  
 let* parse\_buildtime = *function* (*buildTime*) {  
 *// Convert seconds to Date  
 let* BuildTimeOBJ = *new* Date((*buildTime* - 3600) \* 1000); *// -3600 seconds (1 hour) to count for timezone differences in calculations  
 return* MaterialDateTimePicker.dateTimetoString(BuildTimeOBJ);  
 }  
  
  
 *let* parse\_armor = *function* (*armor*) {  
 *// Check if Armor is 'undefined', if so set to 0  
 if* (*armor* === undefined) {  
 *return* 0;  
 } *else* {  
 *return armor*;  
 }  
 }  
  
  
 *// Convert to an indexed array containing the correct tags for each tile  
 let* parse\_formation\_buff\_tiles = *function* (*effect*) {  
 *let* tiles\_table = ["", "", "", "", "", "", "", "", ""];  
 *let* tile\_doll\_center = *effect*.effectCenter;  
 *let* tiles\_doll\_buffs = *effect*.effectPos;  
  
 tiles\_doll\_buffs.forEach(*function* (*tile*) {  
 tiles\_table[*tile* - 1] = "buff" *// -1 so array starts at 0* })  
 tiles\_table[tile\_doll\_center -1] = "standing"  
 *// console.log("tiles\_table", tiles\_table);  
 return* tiles\_table;  
 }  
  
  
 *// Convert Buffs Type to <p> tags  
 let* parse\_formation\_buffs\_type = *function* (*effectType*) {  
 *let* tiles\_effect\_type = ''  
 *let* tiles\_doll\_effect\_type = *effectType*;  
  
 *if* (*typeof* tiles\_doll\_effect\_type === "string") {  
 tiles\_effect\_type += `<p style="margin: 0"><b>Buffs: </b>${tiles\_doll\_effect\_type.toUpperCase()}</p>`  
 }  
 *else* {  
 tiles\_doll\_effect\_type.forEach(*function* (*type*) {  
 tiles\_effect\_type += `<p style="margin: 0"><b>Buffs: </b>${*type*.toUpperCase()}</p>`  
 })  
 }  
  
 *return* tiles\_effect\_type;  
 }  
  
  
 *// Convert Buffs to <p> tags  
 let* parse\_formation\_buffs = *function* (*gridEffect*) {  
 *let* tiles\_effect\_table = ''  
 *let* tiles\_doll\_effect = *gridEffect*;  
 *for* (*let* key *in* tiles\_doll\_effect) {  
 tiles\_effect\_table += `<p style="margin: 0"><b>${doll\_stat\_types[key]}: </b>+${tiles\_doll\_effect[key]}%</p>`  
 }  
 *return* tiles\_effect\_table;  
 }  
  
  
 *// ---------- Global Function returns (outside name : inside name) ----------  
 return* {  
 parse\_rank: parse\_rank,  
 parse\_digimind: parse\_digimind,  
 parse\_buildtime: parse\_buildtime,  
 parse\_armor: parse\_armor,  
 parse\_formation\_buff\_tiles: parse\_formation\_buff\_tiles,  
 parse\_formation\_buffs\_type: parse\_formation\_buffs\_type,  
 parse\_formation\_buffs: parse\_formation\_buffs,  
 };  
 }();  
  
  
 *// Sets the T-Doll HTML Data on screen  
 let* \_render\_html\_doll\_data = *function* (*input\_id*, *favorite* = *false*, *buildTime* = *false*) {  
 *let* doll = gfcore.dolls.find(({*id*}) => *id* === *input\_id*);  
 *// console.log(doll.codename + "\_Data = ", doll);  
  
 // Parse tiles for css classes  
 let* tiles\_table = \_parsers.parse\_formation\_buff\_tiles(doll.effect);  
  
 *// Data to HTML  
 let* doll\_data = `  
 <b>Name: </b>${doll.codename}<br>  
 <b>ID: </b>${doll.id}<br>  
 <b>Type: </b>${doll.type.toUpperCase()}<br>  
 <b>Rank: </b>${\_parsers.parse\_rank(doll.rank)}<br>  
 <b>BuildTime: </b>${\_parsers.parse\_buildtime(doll.buildTime)}<br>  
 <b>Skins: </b>${doll.skins.length}<br>  
 <b>Digimind: </b>${\_parsers.parse\_digimind(doll.mindupdate)}  
  
 <h5>Stats</h5>  
 <table>  
 <tr>  
 <td><b>${doll\_stat\_types['hp']}: </b>${doll.stats.hp}</td>  
 <td><b>${doll\_stat\_types['pow']}: </b>${doll.stats.pow}</td>  
 <td><b>${doll\_stat\_types['hit']}: </b>${doll.stats.hit}</td>  
 </tr>  
 <tr>  
 <td><b>${doll\_stat\_types['dodge']}: </b>${doll.stats.dodge}</td>  
 <td><b>${doll\_stat\_types['speed']}: </b>${doll.stats.speed}</td>  
 <td><b>${doll\_stat\_types['rate']}: </b>${doll.stats.rate}</td>  
 </tr>  
 <tr>  
 <td><b>${doll\_stat\_types['armorPiercing']}: </b>${doll.stats.armorPiercing}</td>  
 <td><b>${doll\_stat\_types['criticalPercent']}: </b>${doll.stats.criticalPercent}</td>  
 <td><b>${doll\_stat\_types['armor']}: </b>${\_parsers.parse\_armor(doll.stats.armor)}</td>  
 </tr>  
 </table>  
  
 <h5>Formation Buff</h5>  
 <div class="row">  
 <div class="col s6">  
 <table class="tile\_grid\_table">  
 <tbody>  
 <tr>  
 <td class="${tiles\_table[6]}"></td> <!-- Tile 7 -->  
 <td class="${tiles\_table[7]}"></td> <!-- Tile 8-->  
 <td class="${tiles\_table[8]}"></td> <!-- Tile 9 -->  
 </tr>  
 <tr>  
 <td class="${tiles\_table[3]}"></td> <!-- Tile 4 -->  
 <td class="${tiles\_table[4]}"></td> <!-- Tile 5 -->  
 <td class="${tiles\_table[5]}"></td> <!-- Tile 6 -->  
 </tr>  
 <tr>  
 <td class="${tiles\_table[0]}"></td> <!-- Tile 1 -->  
 <td class="${tiles\_table[1]}"></td> <!-- Tile 2 -->  
 <td class="${tiles\_table[2]}"></td> <!-- Tile 3 -->  
 </tr>  
 </tbody>  
 </table>  
 </div>  
  
 <div class="col s6">  
 ${\_parsers.parse\_formation\_buffs\_type(doll.effect.effectType)}  
 ${\_parsers.parse\_formation\_buffs(doll.effect.gridEffect)}  
 </div>  
 </div>  
 `;  
  
 *if* (*favorite* === *true*) {  
 $Doll\_Data\_favorite.html(doll\_data);  
 }  
 *else if* (*buildTime* === *true*) {  
 $Doll\_Data\_BuildTime.html(doll\_data);  
 }  
 *else* {  
 $Doll\_Data.html(doll\_data);  
 }  
 }  
 *// TODO: favorite & buildTime booleans to Selector  
 let* reset\_html\_doll\_data = *function* (*favorite* = *false*, *buildTime* = *false*) {  
 *// Data to HTML  
 let* doll\_data = `  
 <b>Name: </b>No Data<br>  
 <b>ID: </b>No Data<br>  
 <b>Type: </b>No Data<br>  
 <b>Rank: </b>No Data<br>  
 <b>BuildTime: </b>No Data<br>  
 <b>Skins: </b>No Data<br>  
 <b>Digimind: </b>No Data  
   
 <h5>Stats</h5>  
 <table style="width:100%">  
 <tr>  
 <td><b>${doll\_stat\_types['hp']}: </b>0</td>  
 <td><b>${doll\_stat\_types['pow']}: </b>0</td>  
 <td><b>${doll\_stat\_types['hit']}: </b>0</td>  
 </tr>  
 <tr>  
 <td><b>${doll\_stat\_types['dodge']}: </b>0</td>  
 <td><b>${doll\_stat\_types['speed']}: </b>0</td>  
 <td><b>${doll\_stat\_types['rate']}: </b>0</td>  
 </tr>  
 <tr>  
 <td><b>${doll\_stat\_types['armorPiercing']}: </b>0</td>  
 <td><b>${doll\_stat\_types['criticalPercent']}: </b>0</td>  
 <td><b>${doll\_stat\_types['armor']}: </b>0</td>  
 </tr>  
 </table>  
   
 <h5>Formation Buff</h5>  
 <div class="row">  
 <div class="col s6">  
 <table class="tile\_grid\_table">  
 <tbody>  
 <tr>  
 <td></td>  
 <td></td>  
 <td></td>  
 </tr>  
 <tr>  
 <td></td>  
 <td></td>  
 <td></td>  
 </tr>  
 <tr>  
 <td></td>  
 <td></td>  
 <td></td>  
 </tr>  
 </tbody>  
 </table>  
 </div>  
  
 <div class="col s6">  
 <p style="margin: 0"><b>Buffs: </b>No Data</p>  
 <p style="margin: 0"><b>Effect: </b>No Data</p>  
 </div>  
 </div>  
 `;  
  
 *if* (*favorite* === *true*) {  
 $Doll\_Data\_favorite.html(doll\_data);  
 }  
 *else if* (*buildTime* === *true*) {  
 $Doll\_Data\_BuildTime.html(doll\_data);  
 }  
 *else* {  
 $Doll\_Data.html(doll\_data);  
 }  
 }  
  
  
  
  
  
 *// ---------- Local Storage stuff ----------  
 // Write the Favorite T-Dolls Array to Local Storage  
 let* \_setLocalStorage = *function*() {  
 *let* favorited\_dolls = *function* () {  
 console.log("Save Favorited T-Dolls to Local Storage");  
 *// console.log('favorite\_doll\_ids[]', favorite\_doll\_ids);* localStorage.setItem('favorite\_doll\_ids', JSON.stringify(favorite\_doll\_ids)); *// localStorage.setItem('key', 'value')* }  
  
  
 *// ---------- Global Function returns (outside name : inside name) ----------  
 return* {  
 favorited\_dolls: favorited\_dolls,  
 };  
 }();  
  
  
 *let* \_addFavoriteDoll = *function*(*id*){  
 *// console.log('Added favorite T-Doll with ID = ' + id);  
 if* (!favorite\_doll\_ids.includes(*id*)) {  
 favorite\_doll\_ids.push(*id*); *// Add the ID to the end of Array* \_setLocalStorage.favorited\_dolls();  
 get\_dolls\_by\_type(selected\_type\_favorited, *true*)  
 M.toast({html: 'T-Doll Favorited', displayLength: 2000, classes: 'grey\_gfl'})  
 }  
 *else* {  
 *// console.error("Not Added favorite T-Doll with ID = " + id + " because of duplicate.")* M.toast({html: 'T-Doll already Favorited', displayLength: 2000, classes: 'grey\_gfl'})  
 }  
 };  
  
  
 *let* \_deleteFavoriteDoll = *function*(*id*){  
 *// console.log("Remove favorite T-Doll with ID = ", id);  
 if*(*confirm*('Remove this T-Doll?')) {  
 *for*(*let* i = 0; i < favorite\_doll\_ids.length; i++){  
 *if* ( favorite\_doll\_ids[i] === *id*) {  
 favorite\_doll\_ids.splice(i, 1); *// Delete the element with Index 'I" from the Array* }  
 }  
 \_setLocalStorage.favorited\_dolls();  
 get\_dolls\_by\_type(selected\_type\_favorited, *true*)  
 reset\_html\_doll\_data(*true*)  
 M.toast({html: 'T-Doll removed from Favorites', displayLength: 2000, classes: 'grey\_gfl'})  
 }  
 };  
  
  
  
  
  
 *// ---------- Settings stuff ----------  
 let* set\_settings = *function* () {  
 *let* sorting\_mode = *function* (*sorting\_mode*) {  
 setting\_sorting\_method = *sorting\_mode*;  
 *// console.log("GFCoreAPI: set\_settings: setting\_sorting\_method = ", setting\_sorting\_method);  
 // Form Selection ReInitialization* get\_dolls\_by\_type(selected\_type);  
 get\_dolls\_by\_type(selected\_type\_favorited, *true*);  
 get\_dolls\_by\_buildTime();  
 }  
  
  
 *// ---------- Global Function returns (outside name : inside name) ----------  
 return* {  
 sorting\_mode: sorting\_mode,  
 };  
 }()  
  
  
  
 *// ---------- Global Function returns (outside name : inside name) ----------  
 return* {  
 init: init,  
 get\_dolls\_by\_type: get\_dolls\_by\_type,  
 get\_dolls\_by\_buildTime: get\_dolls\_by\_buildTime,  
 reset\_html\_doll\_data: reset\_html\_doll\_data,  
 set\_settings: set\_settings,  
 };  
}();

**MaterialDateTimePicker\_SKWAS.js**

*let* MaterialDateTimePicker = *function* () {  
 *// ---------- Global Variables & Stuff ----------  
 let* myDate = *new* Date(1970,1,1,0,0,0,0); *// our baseline Date (00:00:00)  
  
  
 // Cache DOM for performance  
 let* $tdoll\_BuildTime = $('#tdoll\_BuildTime');  
  
  
  
  
  
 *// ---------- Function Stuff ----------  
 let* init = *function* () {  
 *// console.log("baseline Date = ", \_dateTimetoSeconds(myDate))  
 // console.log("baseline Date: HH:MM:SS", dateTimetoString(myDate))* \_getInputDate()  
 GirlsFrontlineCoreAPI.get\_dolls\_by\_buildTime(\_dateTimetoSeconds(myDate));  
 }  
  
  
 *let* showTimePicker = *function* () {  
 cordova.plugins.DateTimePicker.show({  
 mode: "time",  
 titleText: "Build Time",  
 date: myDate,  
 success: *function* (*newDate*) {  
 *// Handle new time.* \_setInputDate(*newDate*);  
 GirlsFrontlineCoreAPI.get\_dolls\_by\_buildTime(\_dateTimetoSeconds(*newDate*));  
 GirlsFrontlineCoreAPI.reset\_html\_doll\_data(undefined,*true*);  
 }  
 })  
 }  
  
  
 *let* \_getInputDate = *function* () {  
 *let* buildTime\_input = $tdoll\_BuildTime.val()  
 *// console.log("buildTime\_input", buildTime\_input)  
 let* [hours, minutes, seconds] = buildTime\_input.split(':');  
 myDate.setHours(*parseInt*(hours))  
 myDate.setMinutes(*parseInt*(minutes))  
 myDate.setSeconds(*parseInt*(seconds))  
 *// console.log("myDate: HH:MM:SS", dateTimetoString(myDate))  
 // console.log("myDate.getTime()", \_dateTimetoSeconds(myDate))* }  
  
  
 *let* \_setInputDate = *function* (*date\_input*) {  
 myDate = *date\_input* $tdoll\_BuildTime.val(dateTimetoString(myDate))  
 }  
  
  
 *let* dateTimetoString = *function* (*date\_input*) {  
 *return date\_input*.getHours().toString().padStart(2, '0') + ":" + *date\_input*.getMinutes().toString().padStart(2, '0') + ":" + *date\_input*.getSeconds().toString().padStart(2, '0');  
 }  
  
  
 *let* \_dateTimetoSeconds = *function* (*date\_input*) {  
 *return* (*date\_input*.getTime() - 2674800000) / 1000;  
 }  
  
  
  
  
  
 *// ---------- Global Function returns (outside name : inside name) ----------  
 return* {  
 init: init,  
 showTimePicker: showTimePicker,  
 dateTimetoString: dateTimetoString,  
 };  
}()

**Network\_State.js**

*let* NetworkState = *function* () {  
 *// ---------- Global Variables & Stuff ----------  
 let* states = {  
 '2g' : 'Cell 2G connection',  
 '3g' : 'Cell 3G connection',  
 '4g' : 'Cell 4G connection',  
 'cellular' : 'Cell generic connection',  
 'ethernet' : 'Ethernet connection',  
 'none' : 'No network connection',  
 'unknown' : 'Unknown connection',  
 'wifi' : 'WiFi connection'  
 };  
  
  
 *// Cache DOM for performance  
 let* $NetworkModal\_Retry = $('#NetworkModal\_Retry');  
 *let* $GFLC\_API\_Loaded = $('#GFLC\_API\_Loaded');  
 *let* $networkState = $('#networkState');  
 *let* $NetworkModal = $('#NetworkModal');  
  
  
  
 *// ---------- Button Stuff ----------  
 // Checks the network state and if not offline reloads the app (Index.html)* $NetworkModal\_Retry.on('click', *function*() {  
 *// console.log("Button: NetworkModal\_Retry");  
 let* NetworkState = \_getState();  
  
 *if* (NetworkState !== 'none') {  
 $GFLC\_API\_Loaded.html(`<b>GFL-Core API: </b>Loading`);  
 location.reload();  
 }  
 })  
  
  
  
 *// ---------- Function Stuff ----------  
 // Initialise the Network State Check  
 let* init = *function*(){  
 *// Get current Network State  
 let* networkState = \_getState()  
  
 *// If offline --> Show "No network connection" Modal  
 if* (networkState === 'none') {  
 \_show\_modal()  
 }  
 };  
  
  
 *// Get current Network State  
 let* \_getState = *function*(){  
 *let* networkState = navigator.connection.type;  
 console.log('Connection type: ', networkState);  
 $networkState.html(`<b>Netwerk: </b>${states[networkState]}`);  
 *return* networkState;  
 };  
  
  
 *// Show "No network connection" Modal  
 let* \_show\_modal = *function* () {  
 $NetworkModal.modal().modal('open');  
 }  
  
  
 *// Close "No network connection" Modal  
 let* close\_modal = *function* () {  
 $NetworkModal.modal().modal('close');  
 }  
  
  
  
 *// ---------- Global Function returns (outside name : inside name) ----------  
 return* {  
 init: init,  
 close\_modal: close\_modal,  
 };  
}();

**Settings.js**

*let* Settings = *function* () {  
 *// TODO: Setting for default tab  
 // TODO: Option to clear Favorites  
 // ---------- Global Variables & Stuff ----------  
 // key : [function, "Sorting Name"]  
 const* sorting\_methods = {  
 id : [Sorting\_methods.array\_index\_0\_number, "By ID"],  
 name : [Sorting\_methods.array\_index\_1\_string, "By name"],  
 rank : [Sorting\_methods.array\_index\_2\_number, "By rank (low -> high)"],  
 rank\_reversed : [Sorting\_methods.array\_index\_2\_number\_reversed, "By rank (high -> low)"],  
 buildTime : [Sorting\_methods.array\_index\_3\_number, "By build time (low -> high)"],  
 buildTime\_reversed : [Sorting\_methods.array\_index\_3\_number\_reversed, "By build time (high -> low)"],  
 }  
  
 *// Local Storage: Sorting Method  
 let* ls\_sorting\_method = sorting\_methods['name'][0];  
  
  
 *// Cache DOM for performance  
 let* $tabSettings = $('#tabSettings');  
 *let* $settings\_sorting\_method = $tabSettings.find('#settings\_sorting\_method');  
  
  
  
  
  
 *// ---------- Button Stuff ----------  
 // --- T-Doll Data tab ---  
 // Get selected T-Doll data from dropdown* $settings\_sorting\_method.on('change', *function*() {  
 *let* mode = *this*.value;  
 *// console.log("Dropdown Sorting: Value = ", this.value);  
 // console.log("Dropdown Sorting: Method = ", sorting\_methods[mode][0]);  
 if* (mode !== "") {  
 ls\_sorting\_method = sorting\_methods[mode][0];  
 \_setLocalStorage.sorting\_method();  
 GirlsFrontlineCoreAPI.set\_settings.sorting\_mode(ls\_sorting\_method);  
 }  
 });  
  
  
  
  
  
 *// ---------- Function Stuff ----------  
 let* init = *function* () {  
 *// Add all the options to there dropdowns* \_set\_dropdowns();  
  
 *// Read Settings from local storage* \_getLocalStorage();  
  
 *// Set settings in other scripts* \_set\_settings();  
 };  
  
  
 *let* \_set\_dropdowns = *function* () {  
 *// --- Sorting Method Dropdown ---* $settings\_sorting\_method.empty() *// Empty current dropdown list* .append("<option value='' disabled selected>Choose a Sorting Method</option>");  
  
 *for* (*let* key *in* sorting\_methods) { *// Dynamically add Sorting Methods to the list  
 let* name = sorting\_methods[key][1];  
 $settings\_sorting\_method.append("<option value='" + key + "'>" + name + "</option>");  
 }  
  
 *// Form Selection ReInitialization* $('select').formSelect();  
 }  
  
  
 *// Set settings in other scripts  
 let* \_set\_settings = *function* () {  
 console.log("Settings: Settings Set");  
 GirlsFrontlineCoreAPI.set\_settings.sorting\_mode(ls\_sorting\_method);  
 };  
  
  
  
 *// ---------- Local Storage stuff ----------  
 // Function to write the settings to Local Storage  
 let* \_setLocalStorage = *function*() {  
 *let* sorting\_method = *function* () {  
 console.log("Saved Sorting Method settings to Local Storage");  
 *for* (*let* key *in* sorting\_methods) {  
 *let* value = sorting\_methods[key][0]  
 *if* (value === ls\_sorting\_method) {  
 localStorage.setItem('setting\_sorting\_method', key); *// localStorage.setItem('key', 'value')* }  
 }  
 };  
  
  
 *// ---------- Global Function returns (outside name : inside name) ----------  
 return* {  
 sorting\_method: sorting\_method,  
 };  
 }();  
  
  
 *// Read Settings from local storage  
 let* \_getLocalStorage = *function* () {  
 *// --- Sorting Method ---  
 let* ls\_sorting\_method\_key = localStorage.getItem('setting\_sorting\_method');  
 *if* (ls\_sorting\_method\_key !== *null*) {  
 ls\_sorting\_method = sorting\_methods[ls\_sorting\_method\_key][0];  
 $settings\_sorting\_method.val(ls\_sorting\_method\_key);  
 *// console.log("Settings: Init: ls\_sorting\_method = ", ls\_sorting\_method);* } *else* {  
 \_setLocalStorage.sorting\_method();  
 $settings\_sorting\_method.val(localStorage.getItem('setting\_sorting\_method'));  
 }  
 }  
  
  
  
 *// ---------- Global Function returns (outside name : inside name) ----------  
 return* {  
 init: init,  
 };  
}();

**sorting\_methods.js**

*let* Sorting\_methods = *function* () {  
 *// TODO: Add setting to select default sort method  
 // ---------- Function Stuff ----------  
 // By ID  
 let* array\_index\_0\_number = *function*(*a*, *b*) {  
 *return a*[0] - *b*[0];  
 }  
  
 *// By name  
 let* array\_index\_1\_string = *function*(*a*, *b*) {  
 *return a*[1].localeCompare(*b*[1]);  
 }  
  
 *// By rank (1 -> 7)  
 let* array\_index\_2\_number = *function*(*a*, *b*) {  
 *return a*[2] - *b*[2];  
 }  
  
 *// By rank (7 -> 1)  
 let* array\_index\_2\_number\_reversed = *function*(*a*, *b*) {  
 *return b*[2] - *a*[2];  
 }  
  
 *// By buildTime (low -> high)  
 let* array\_index\_3\_number = *function*(*a*, *b*) {  
 *return a*[3] - *b*[3];  
 }  
  
 *// By buildTime (high -> low)  
 let* array\_index\_3\_number\_reversed = *function*(*a*, *b*) {  
 *return b*[3] - *a*[3];  
 }  
  
  
  
 *// ---------- Global Function returns (outside name : inside name) ----------  
 return* {  
 array\_index\_0\_number: array\_index\_0\_number,  
 array\_index\_1\_string: array\_index\_1\_string,  
 array\_index\_2\_number: array\_index\_2\_number,  
 array\_index\_2\_number\_reversed: array\_index\_2\_number\_reversed,  
 array\_index\_3\_number: array\_index\_3\_number,  
 array\_index\_3\_number\_reversed: array\_index\_3\_number\_reversed,  
 };  
}();