

Usage

Configuração

Para usar o funifier em produção, insira a seguinte linha na tag head de todas as paginas que deseja habilitar:

```
<script src="//client2.funifier.com/2.0.0/funifier.min.js" type="text/javascript"></script>
```

Iniciação Async mode

Para iniciar o funifier de maneira assíncrona, insira uma função antes do script do funifier:

```
window.funifierAsyncInit = function(){  
    Funifier.init({  
        apiKey : 'SUA API KEY'  
    });  
};
```

Dessa maneira, assim que o funifier tiver sido processado, irá chamar a função 'funifierAsyncInit' caso exista.

Iniciação Sync mode

Você pode inicializar também o funifier a qualquer momento após o carregamento do script do funifier chamando seu método de início:

```
Funifier.init({  
    apiKey : 'SUA API KEY'  
});
```

Documentação

Componentes

O funifier tem algumas bibliotecas integradas, que pode ser usado em seus widgets/integrações, note no entanto, que as que tem o prefixo '_' são de uso do core do Funifier e por isso não tem garantia de estar presente numa versão futura.

Funifier contém os seguintes componentes:

- [jquery](#): Funifier._\$ 1.11.1
- [async](#): Funifier._async
- [lodash](#): Funifier._
- [validator](#): Funifier._validator
- [mustache](#): Funifier._mustache
- [headjs](#): Funifier.headjs
- [magnificPopup](#): Funifier._\$.magnificPopup
- [qs](#): Funifier._qs

Para a documentação dos componentes acima, verifique no site do respectivo componente.

Métodos

- [init](#)
- [listen](#)
- [unlisten](#)
- [api](#)
- [track](#)
- [auth](#)
 - [authenticate](#)
 - [isAuthenticate](#)
 - [logout](#)

init(config, callback)

Utilizado para iniciar a api do funifier. Recebe como parâmetro um objeto e um callback.

config

- apiKey : String - Apikey configurada no studio.
- hideInline : Boolean - Define a exibição do modo inline para admin. **default** : false
- showNotification: Boolean - Exibir/ocultar as notificações. **default** : true
- disableCustomCss : Boolean - Define se o funifier ira carregar o css dos widgets. **default** : custom-{api_key}-v{version}.css.
- disableCustomJs: Boolean - Define se o funifier ira carregar o js com as funções de widgets. **default** : custom-{api_key}-v{version}.js.
- url
 - service : String - Url do service. **default** : '//service2.funifier.com/2.0.0/'
 - client : String - Url do client. **default** : '//client2.funifier.com/2.0.0'
 - studio : String - Url do studio. **default** : '//studio2.funifier.com'
 - css : String - Caminho de css para sobrescrever o padrão. **default** : null
 - js : String - Caminho do js para sobrescrever o padrão. **default** : null

callback

Recebe uma função que é executada assim que o funifier terminou de inicializar os objetos para a apiKey informada, sendo, que ja carregou todas as dependencias, scripts e styles.

Parâmetros:

- err Recebe um erro caso tenha um

Exemplo

```
Funifier.init({
  apiKey : 'SUA API KEY',
  hideInline: false,
  url : {
    css : 'http://www.meudominio.com/custom-css-replace.css'
  }
},function(err){
  if(err){
    console.log('Ocorreu um erro');
  }
});
```

listen(name, callback)

Monitora um evento do funifier

- name : String - Nome do evento a ser monitorado
 - callback : Function - Função a ser executada.
-

unlisten(name)

Para remover eventos do funifier monitorados pelo listen.

- name : String - Nome do evento a excluir o listen.
 -
-

api(endpoint,params,callback)

Método para comunicação com o service.

- endpoint : String - Metodo a ser requisitado no service.
- params : Object - Parâmetros a ser enviado para o service.
- callback : Function - retorno

Exemplos

```
Funifier.api('get_leaderboard',
  {sort: 1, limit: 10, view: 0, group : 1},
  function(err,data){
    if(err==null){
      alert('Lista de leaderboards recebida')
      console.log(data);
    }else{
      alert('Ocorreu um erro :(')
    }
  }
);
```

```
Funifier.api('get_widget_data',
  null,
  function(err,data){
    if(err==null){
      alert('Retorno recebido')
      console.log(data);
    }else{
      alert('Ocorreu um erro :(')
    }
  }
);
```

api(endpoint,params,options,callback)

Método para comunicação com o service.

- endpoint : String - Metodo a ser requisitado no service.
- params : Object - Parâmetros a ser enviado para o service.
- options : Object - Parâmetros adicionais.
- callback : Function - retorno

Exemplos

```
Funifier.api('get_leaderboard',
  {sort: 1, limit: 10, view: 0, group : 1},
  {type: 'GET',contentType:'application/json'},
  function(err,data){
    if(err==null){
      alert('Lista de leaderboards recebida')
      console.log(data);
    }else{
      alert('Ocorreu um erro :(')
    }
  }
);
```

track(options,callback)

Traqueia uma ação

options

- `action` : String - Id ou nome da action.
- `attributes` : Array - lista de atributos, opcional.

callback

Função executada apos o track.

- `err` - Recebe um erro ou nulo.
- `data` - Response do servidor.

Exemplo

```
Funifier.track({
  action: 'visitarPagina'
},function(err,data){
  if(err==null){
    alert('Action traqueada')
  }
});
```

auth

authenticate(options,callback)

Função usada para autenticar jogadores.

options

- `auth_mode` : String - Tipo de autenticação, valores disponiveis.
 - facebook
 - google
 - IMPLICIT
 - PASSWORD
- `player` : String - Id do jogador, requerido para `auth_mode` 'IMPLICIT' e 'PASSWORD'. (String)
- `password` : String - Senha do jogador, requerido para `auth_mode` 'PASSWORD'
- `modal` : Boolean - Define se a autenticação via google ou facebook sera feita atraves de um popup, (opcional usado apenas para `auth_mode` 'google' e 'facebook')

callback

Função que de retorno. (Não é executada caso o auth_mode for facebook ou google e modal for igual a false)

- err - Recebe um erro ou nulo.
- data - Response do servidor:

Exemplo Facebook

```
Funifier.auth.authenticate({
  auth_mode: 'facebook',
  modal: true
},function(err,data){
  if(err==null){
    alert('Seja bem vindo')
  }
});
```

Exemplo via Senha

```
Funifier.auth.authenticate({
  auth_mode: 'PASSWORD',
  player: 'jogador1'
  password: 'pass@123'
},function(err,data){
  if(err==null){
    alert('Seja bem vindo')
  }
});
```

isAuthenticate(callback)

Verifica se o jogador está autenticado.

callback

- isAuthenticate :Boolean

Exemplo

```
Funifier.auth.isAuthenticate(function(_isAuthenticate){
  if(isAuthenticate){
    alert('Seja bem vindo, você está autenticado');
  }else{
    alert('Desculpe, faça login, você não está autenticado');
  }
});
```

logout(callback)

Efetua logout do jogador.

callback

- err
- data

Exemplo

```
Funifier.auth.logout(function(err){
```

```

    if(err==null){
        alert('Jogador deslogado com sucesso');
    }else{
        alert('Ocorreu um erro');
    }
});

```

Widget

A estrutura basica de um widget:

```

var ct = '<div><a href="#">Hello World!</a></div>'
render(ct,function(config){
    var elemento = config.element;
    elemento.find('a').click(function(){
        alert('Você clicou no link');
    });
});

```

O método render recebe uma string ou um objeto jquery e um metodo de retorno onde pode-se incluir eventos.

Dentro do script você tem acesso a variável 'json', 'theme_html', '\$' e 'jQuery'. O jquery disponível no widget é um jquery do objeto Funifier, caso queira acessar o jquery da pagina, use window.\$ ou window.jQuery.

A variável json está disponivel em alguns tipos de widgets.

Você pode usar o mustache para renderizar seu html.

```

Funifier.auth.isAuthenticated(function(isAuthenticate) {
    json.isAuthenticated = isAuthenticated;
    var str = Mustache.render(theme_html,json);
    render(str,function(config){
        var el = config.element;
        //your code
    });
});

```

Você pode requisitar um recurso no servidor antes de renderizar.

```

Funifier.auth.isAuthenticated(function(isAuthenticate) {
    Funifier.api('get_widget_data', {include_challenges: true}, function(err,json){
        json.isAuthenticated = isAuthenticated;
        var str = Mustache.render(theme_html,json);
        render(str,function(config){
            var el = config.element;
            //your code
        });
    });
});

```

Debug

Por padrão o Funifier Client não exibe qualquer informação de error/info no console sendo necessário habilitar. Para habilitar todas as mensagens do funifier, abra o console javascript e digite:

```

Funifier.debug.enable('*');

```

Você pode habilitar o log para mensagens específicas:

```
Funifier.debug.enable('Funifier:social:');
```

```
Funifier.debug.enable('Funifier:core:');
```

Caso queira desabilitar as mensagens:

```
Funifier.debug.disable();
```