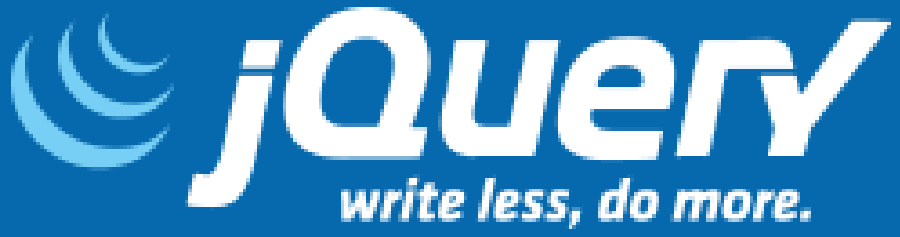


Andre Noel

Resumo

- Plugins jQuery
- Ajax
- JSON
- Sistema web

Plugins

- http://plugins.jquery.com/

inputmask

- No <input> do CPF, adicione o atributo:
  - data-mask="999.999.999-99"
- Nos campos do número do cartão com código de verificação, podemos usar:
  - data-mask="9999 9999 9999 9999 - 999"

inputmask

Include the js-files which you can find in the `dist` folder.

via Inputmask class

```
<script src="jquery.js"></script>
<script src="inputmask.js"></script>
<script src="inputmask.???.Extensions.js"></script>
```

```
var selector = document.getElementById("selector");

var im = new Inputmask("99-9999999");
im.mask(selector);

Inputmask({"mask": "(999) 999-9999", .... other options .....}).mask(selector);
Inputmask("9-a{1,3}9{1,3}").mask(selector);
Inputmask("9", { repeat: 10 }).mask(selector);
```

inputmask

via jquery plugin

```
<script src="jquery.js"></script>
<script src="inputmask.js"></script>
<script src="inputmask.???.Extensions.js"></script>
<script src="jquery.inputmask.js"></script>
```

or with the bundled version

```
<script src="jquery.js"></script>
<script src="jquery.inputmask.bundle.js"></script>
```

```
$(document).ready(function(){
  $(selector).inputmask("99-9999999");  //static mask
  $(selector).inputmask({"mask": "(999) 999-9999"}); //specifying options
  $(selector).inputmask("9-a{1,3}9{1,3}"); //mask with dynamic syntax
});
```

inputmask

via data-inputmask attribute

```
<input data-inputmask="'alias': 'date'" />
<input data-inputmask="'mask': '9', 'repeat': 10, 'greedy' : false" />
<input data-inputmask="'mask': '99-9999999'" />
```

```
$(document).ready(function(){
  $(":input").inputmask();
  or
  Inputmask().mask(document.querySelectorAll("input"));
});
```

Any option can also be passed through the use of a data attribute. Use data-inputmask-<***the name of the option***>="value"

```
<input id="example1" data-inputmask-clearmaskonlostfocus="false" />
<input id="example2" data-inputmask-regex="[a-za-zA-Z0-9!#$%&'*+/=?^_`{|}~-]+(?:\.[a-zA-Z0-9!#$%&'*+/=?^_`{|}~-]+)*@(
```

```
$(document).ready(function(){
  $("#example1").inputmask("99-9999999");
  $("#example2").inputmask("Regex");
});
```

inputmask

Optional masks

It is possible to define some parts in the mask as optional. This is done by using [ ].

Example:

```
$('#test').inputmask('(99) 9999[9]-9999');
```

This mask wil allow input like `(99) 99999-9999` or `(99) 9999-9999`.

Input => 12123451234 mask => (12) 12345-1234 (trigger complete)
Input => 121234-1234 mask => (12) 1234-1234 (trigger complete)
Input => 1212341234 mask => (12) 12341-234_ (trigger incomplete)

skipOptionalPartCharacter

inputmask

Dynamic masks

Dynamic masks can change during the input. To define a dynamic part use { }.

{n} => n repeats
{n,m} => from n to m repeats

Also {+} and {*} is allowed. + start from 1 and * start from 0.

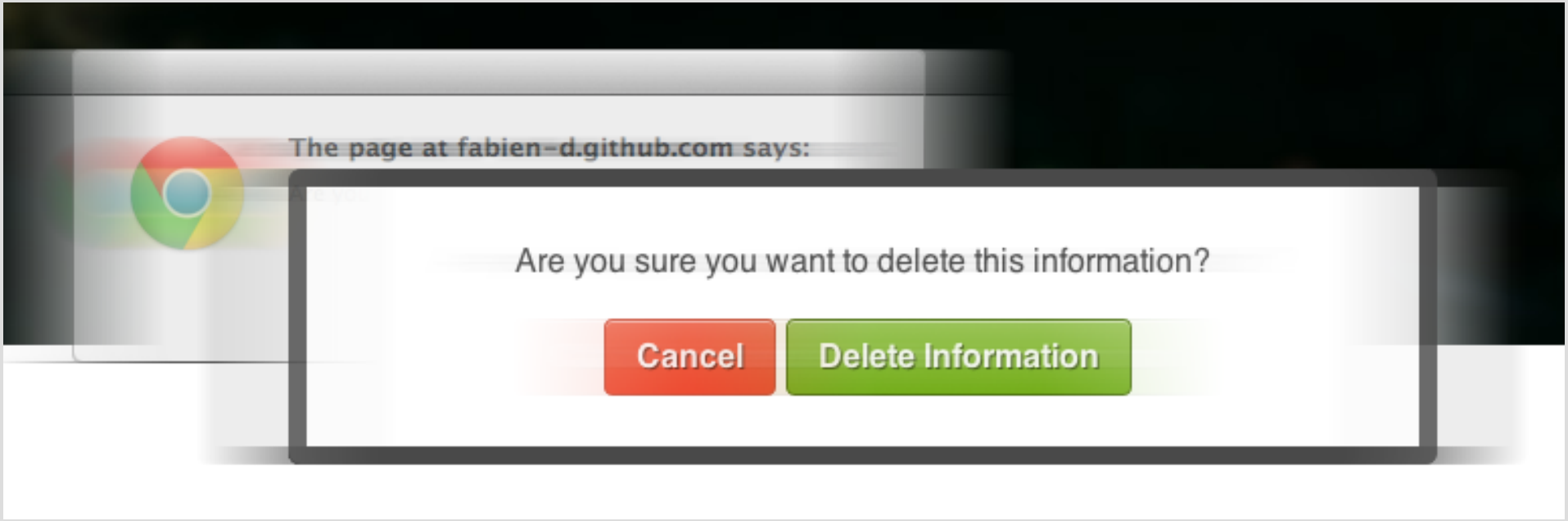
```
$(document).ready(function(){
  $(selector).inputmask("aa-9{4}");  //static mask with dynamic syntax
  $(selector).inputmask("aa-9{1,4}");  //dynamic mask ~ the 9 def can be occur 1 to 4 times

  //email mask
  $(selector).inputmask({
    mask: "*{1,20}[.*{1,20}][.*{1,20}][.*{1,20}]@*{1,20}[.*{2,6}][.*{1,2}]",
    greedy: false,
    onBeforePaste: function (pastedValue, opts) {
      pastedValue = pastedValue.toLowerCase();
      return pastedValue.replace("mailto:", "");
    },
    definitions: {
      '*': {
        validator: "[0-9A-Za-z!#$%&'*+/=?^_`{|}~\-]",
        cardinality: 1,
        casing: "lower"
      }
    }
  });
});
```

Exercícios

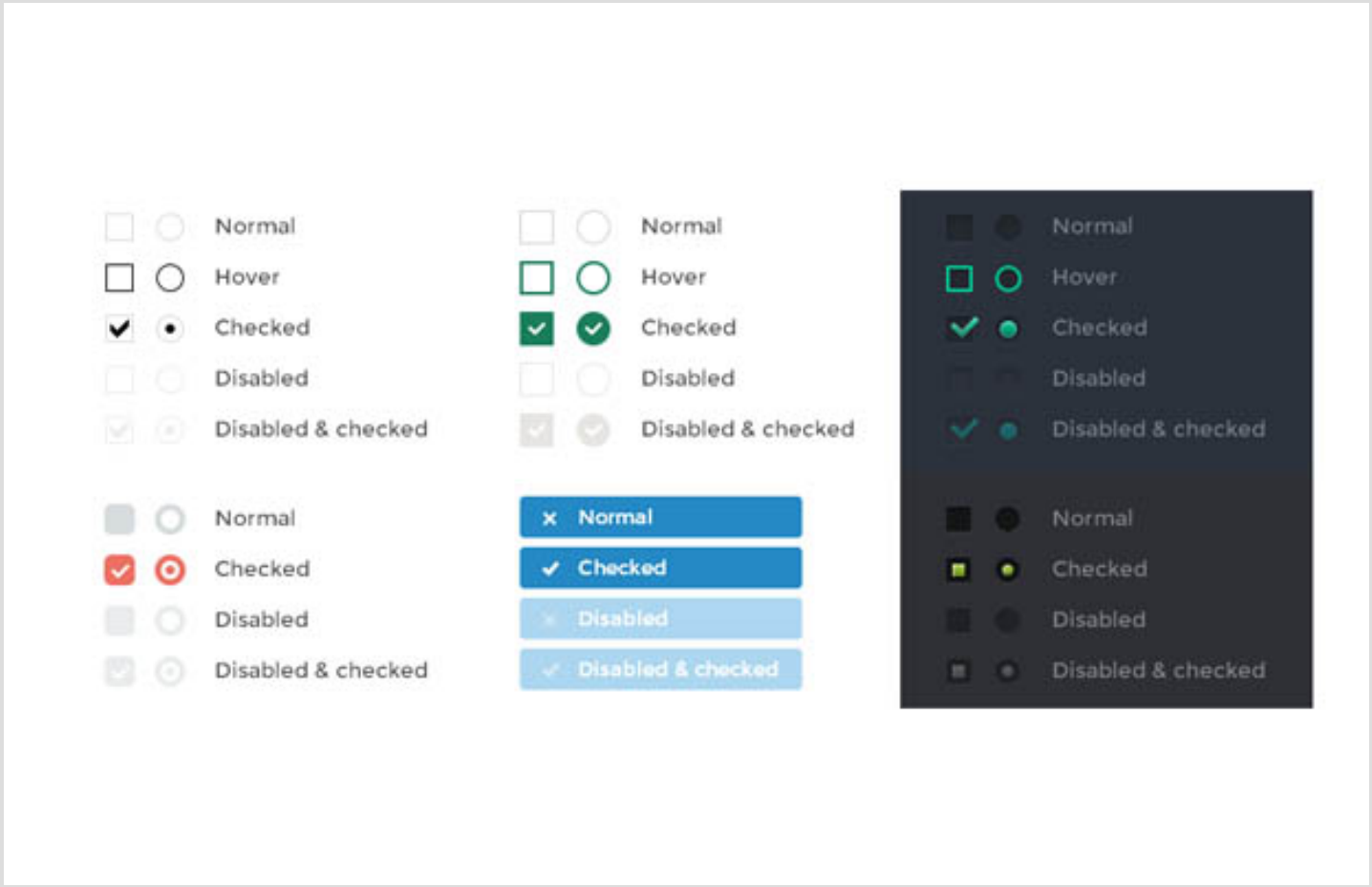
- Crie um formulário com campos com validação para:
  - Data de nascimento
  - RG
  - CPF
  - Telefone DDD + 8 ou 9 dígitos (com hífen)

- http://tutorialzine.com/2013/04/50-amazing-jquery-plugins/

alertify

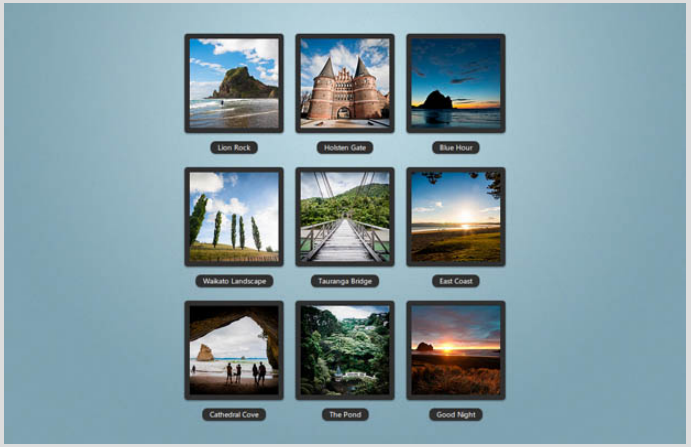

The page at fabien-d.github.com says:
Are you sure you want to delete this information?
Cancel
Delete Information

icheck


Normal
Hover
Checked
Disabled
Disabled & checked
Normal
Hover
Checked
Disabled
Disabled & checked
Normal
Hover
Checked
Disabled
Disabled & checked
Normal
Checked
Disabled
Disabled & checked
Normal
Checked
Disabled
Disabled & checked
Normal
Hover
Checked
Disabled
Disabled & checked
Normal
Checked
Disabled
Disabled & checked

TouchTouch

- http://demo.tutorialzine.com/2012/04/mobile-touch-gallery/



Jquery Mobile

Como criar um plugin

- Primeiro, objetos!
  - $("a").css("color", "red");
- Sempre que você usa a função *$* para selecionar elementos, um objeto é retornado
- O objeto pega os métodos do objeto *$.fn*
- *$.fn* contém todos os métodos de objetos jQuery

Criando um plugin básico

- Vamos criar um plugin que torne verde o texto dos elementos retornados
- Tudo o que temos que fazer é incluir uma função greenify ao $.fn e ela estará disponível junto com os outros métodos

```
1 | $.fn.greenify = function() {
2 |     this.css( "color", "green" );
3 | };
4 |
5 | $( "a" ).greenify(); // Makes all the links green.
```

Notice that to use `.css()`, another method, we use `this`, not `$( this )`. This is because our `greenify` function is a part of the same object as `.css()`.

Encadeamento

- O exemplo anterior funciona, mas...

```
1 $.fn.greenify = function() {
2     this.css( "color", "green" );
3     return this;
4 }
5
6 $( "a" ).greenify().addClass( "greenified" );
```

Protegendo a $

- Outras bibliotecas podem usar $
  - Jquery.noConflict()
- Para evitar conflitos, mas ainda funcionar com $

```
1 (function ( $ ) {
2
3     $.fn.greenify = function() {
4         this.css( "color", "green" );
5         return this;
6     };
7
8 }( jQuery ));
```

Criando escopo

- A função invocada permite a criação de variáveis privadas

```
1  (function ( $ ) {
2
3      var shade = "#556b2f";
4
5      $.fn.greenify = function() {
6          this.css( "color", shade );
7          return this;
8      };
9
10 }( jQuery ));
```

Minimizando os rastros

- Criar muitas funções pode aumentar as chances de seus métodos serem sobrescritos por outro plugin ou de seu plugin sobrescrever outros métodos
- Ao invés disto:

```
1  (function( $ ) {
2
3      $.fn.openPopup = function() {
4          // Open popup code.
5      };
6
7      $.fn.closePopup = function() {
8          // Close popup code.
9      };
10
11 }( jQuery ));
```

Minimizando os rastros

- Prefira isto:

```
1  (function( $ ) {
2
3      $.fn.popup = function( action ) {
4
5          if ( action === "open") {
6              // Open popup code.
7          }
8
9          if ( action === "close" ) {
10             // Close popup code.
11         }
12
13     };
14
15 }( jQuery ));
```

Use parâmetros para controlar a ação

O método .each()

- Seu objeto pode conter um número de elementos do DOM, dependendo do seletor
- Se for manipular elementos específicos, use o método .each() para realizar o looping

```
1 | $.fn.myNewPlugin = function() {
2 |
3 |     return this.each(function() {
4 |         // Do something to each element here.
5 |     });
6 |
7 | };
```

Retornamos os resultados do *.each()*, ao invés de *this*. *.each()* também é encadeável

Aceitando opções

```
(function ( $ ) {

    $.fn.greenify = function( options ) {

        // This is the easiest way to have default options.
        var settings = $.extend({
            // These are the defaults.
            color: "#556b2f",
            backgroundColor: "white"
        }, options );

        // Greenify the collection based on the settings variable.
        return this.css({
            color: settings.color,
            backgroundColor: settings.backgroundColor
        });

    };

}( jQuery ));
```

Example usage:

```
$( "div" ).greenify({
    color: "orange"
});
```

Juntando tudo

```
(function( $ ) {

    $.fn.showLinkLocation = function() {

        this.filter( "a" ).each(function() {
            var link = $( this );
            link.append( " (" + link.attr( "href" ) + ")" );
        });

        return this;

    };

}( jQuery ));

// Usage example:
$( "a" ).showLinkLocation();
```

```
<!-- Before plugin is called: -->
<a href="page.html">Foo</a>

<!-- After plugin is called: -->
<a href="page.html">Foo (page.html)</a>
```

Otimizando

```
1  (function( $ ) {
2
3      $.fn.showLinkLocation = function() {
4
5          this.filter( "a" ).append(function() {
6              return " (" + this.href + ")";
7          });
8
9          return this;
10
11     };
12
13 }( jQuery ));
```

Exercícios

- Crie um plugin que coloque o conteúdo do atributo alt da imagem como legenda abaixo dela
- Crie um plugin codifique os textos, fazendo as seguintes substituições:

Trocar:	Por:
e	enter
i	inis
o	omber
u	ufter
a	ais

AJAX

AJAX

- XMLHttpRequest
  - 2003
- Asynchronous JavaScript and XML
- Callback

$.ajax()

- GET vs POST
- Tipos de dados
  - text
  - html
  - script
  - json
  - jsonp
  - xml

Assíncrono

- A resposta não é imediata
- O código abaixo não funciona

```
1 | var response;
2 |
3 | $.get( "foo.php", function( r ) {
4 |     response = r;
5 | });
6 |
7 | console.log( response ); // undefined
```

Assíncrono

- A resposta não é imediata
- O código abaixo não funciona
- Quando passado como função callback funciona (quando concluir):

```
1 | $.get( "foo.php", function( response ) {
2 |     console.log( response ); // server res
3 | });
```

Exemplo

```
<script type="text/javascript">
// Using the core $.ajax() method
$.ajax({

    // The URL for the request
    url: "plugin1.html",

    // The data to send (will be converted to a query string)
    data: {
        id: 123
    },

    // Whether this is a POST or GET request
    type: "GET",

    // The type of data we expect back
    dataType : "html",
})
  // Code to run if the request succeeds (is done);
  // The response is passed to the function
  .done(function( html ) {
     $( "<div class=\"content\">").html( html ).appendTo( "body" );
  })
  // Code to run if the request fails; the raw request and
  // status codes are passed to the function
  .fail(function( xhr, status, errorThrown ) {
    alert( "Sorry, there was a problem!" );
    console.log( "Error: " + errorThrown );
    console.log( "Status: " + status );
    console.dir( xhr );
  })
  // Code to run regardless of success or failure;
  .always(function( xhr, status ) {
    alert( "The request is complete!" );
  });
</script>
```

```
<script type="text/javascript">
// Using the core $.ajax() method
$.ajax({

    // The URL for the request
    url: "plugin1.html",

    // The data to send (will be converted to a query string)
    data: {
        id: 123
    },

    // Whether this is a POST or GET request
    type: "GET",

    // The type of data we expect back
    dataType : "html",
})
  // Code to run if the request succeeds (is done);
  // The response is passed to the function
  .done(function( html ) {
     $( "<div class=\"content\">").html( html ).appendTo( "body" );
  })
  // Code to run if the request fails; the raw request and
  // status codes are passed to the function
  .fail(function( xhr, status, errorThrown ) {
    alert( "Sorry, there was a problem!" );
```

```
    },

    // Whether this is a POST or GET request
    type: "GET",

    // The type of data we expect back
    dataType : "html",
})
  // Code to run if the request succeeds (is done);
  // The response is passed to the function
  .done(function( html ) {
     $( "<div class=\"content\">").html( html ).appendTo( "body" );
  })
  // Code to run if the request fails; the raw request and
  // status codes are passed to the function
  .fail(function( xhr, status, errorThrown ) {
    alert( "Sorry, there was a problem!" );
    console.log( "Error: " + errorThrown );
    console.log( "Status: " + status );
    console.dir( xhr );
  })
  // Code to run regardless of success or failure;
  .always(function( xhr, status ) {
    alert( "The request is complete!" );
  });
</script>
```

Ajax e formulários

- .serialize() e .serializeArray()
- Validação (client-side)

```
1  // Using validation to check for the presence of an input
2  $( "#form" ).submit(function( event ) {
3
4      // If .required's value's length is zero
5      if ( $( ".required" ).val().length === 0 ) {
6
7          // Usually show some kind of error message here
8
9          // Prevent the form from submitting
10         event.preventDefault();
11     } else {
12
13         // Run $.ajax() here
14     }
15 });
```

Ajax e formulários

```
1  // Validate a phone number field
2  $( "#form" ).submit(function( event ) {
3      var inputtedPhoneNumber = $( "#phone" ).val();
4
5      // Match only numbers
6      var phoneNumberRegex = /^\d*$/;
7
8      // If the phone number doesn't match the regex
9      if ( !phoneNumberRegex.test( inputtedPhoneNumber ) ) {
10
11         // Usually show some kind of error message here
12
13         // Prevent the form from submitting
14         event.preventDefault();
15     } else {
16
17         // Run $.ajax() here
18     }
19 });
```

JSON

- **J**ava**S**cript **O**bject **N**otation
- {"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
  ]}

JSON

- Em XML:

XML Example

```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

.ajaxStart() / .ajaxStop()

```
1  // Setting up a loading indicator using Ajax Events
2  $( "#loading_indicator" )
3      .ajaxStart(function() {
4          $( this ).show();
5      })
6      .ajaxStop(function() {
7          $( this ).hide();
8      });
```

Exercícios

- Crie um formulário com campos estado e cidade. Preencha as opções a partir dos dados do arquivo JSON