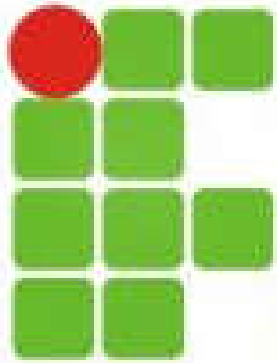
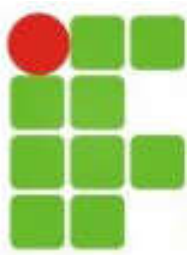
Fundamentos de Web Design 2

Professor Eng. Dr. Will Roger Pereira

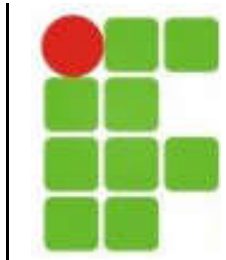


Objetivos desta aula



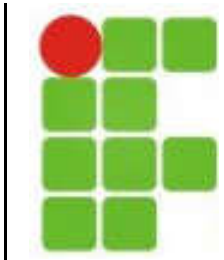
- Campos de texto com máscara.

Introdução



- A maneira principal do usuário inserir informações no sistema é através de campos de texto;
- Até o momento, sabemos validar os campos já preenchidos pelo usuário;
- Porém, de modo a melhorar ainda mais a experiência do usuário, é interessante tornar o preenchimento dos campos mais guiado;
- Para isto utilizaremos máscaras.

jQuery Mask

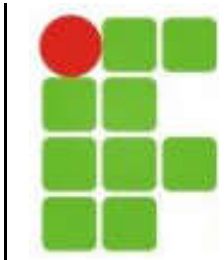


- Uma das APIs mais populares e eficientes para colocar máscaras em campos é a jQuery Mask;

- Desenvolvida por Igor Escobar, um programador brasileiro, possui código-fonte disponível:
  - https://github.com/igorescobar/jQuery-Mask-Plugin

- Para utilizá-la, utilize o arquivo abaixo, já disponível nesta aula, e coloque na mesma pasta do código:

Iniciando o jQuery Mask



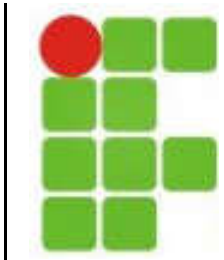
- Com o referido arquivo em mãos, abaixo da inserção do jQuery, insira o seguinte script em seu código:

```
<script src="jquery.mask.js"></script>
```

- Seu código utilizando jQuery Mask deve vir apenas após esta linha;

- Ela utiliza a sintaxe do próprio jQuery, apenas adicionando algumas ações e opções.

Sintaxe Inicial

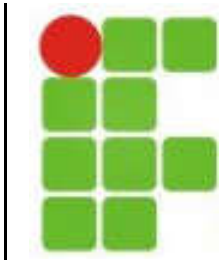


- Para adicionar uma máscara em um campo, siga a seguinte sintaxe:

```
CAMPO.mask(MÁSCARA)
```

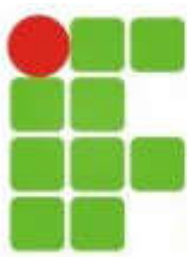
- Onde:
  - CAMPO → Seletor do campo onde deseja-se aplicar a máscara;
  - MÁSCARA → String representando a máscara a ser aplicada.

Máscara



- A máscara, argumento da ação **mask**, deve ser uma string;
- Todos os caracteres dela serão literais, exceto quando possuírem algum significado especial, já conhecido ou desenvolvido por você;
- Quando o usuário digitar os caracteres especiais corretamente, os demais caracteres literais serão preenchidos literalmente;
- Já se o usuário digitar um caractere que não condiz com o sentido do caractere especial, o mesmo não irá aparecer.

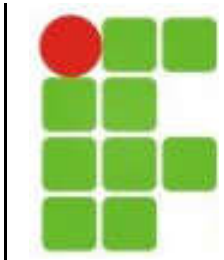
Sintaxe da Máscara



- Os caracteres especiais iniciais são os seguintes:
  - **0** → Representa um número obrigatório;
  - **9** → Representa um número opcional;
  - **S** → Representa uma letra;
  - **A** → Representa um caractere alfanumérico.

- Exemplos:
  - “00” → Um número com dois dígitos;
  - “00:00” → Dois números de dois algarismos separados por “:”;
  - “00/00/0000” → Uma data;
  - “AAA-0000” → Uma placa de carro;
  - “000.000.000-00” → Um número de CPF.

Opções na Máscara



- Na ação **mask**, como segundo argumento, é possível adicionar opções.

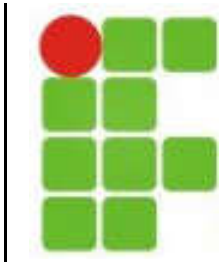
```
CAMPO.mask(MÁSCARA, [OPÇÕES])
```

- Sua sintaxe é a seguinte.

```
{opção1: valor, opção2: valor, ..., opçãoN: valor}
```

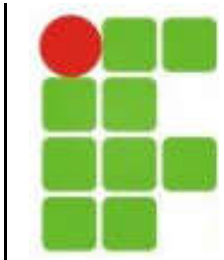
- São muito importantes para diversos outros aspectos, desde funcionalidades já prontas, até aplicação de expressões regulares e funções de callback.

Opções Simples



- Limpar campo caso o conteúdo não se adeque à mascara:
  - **clearIfNotMatch** → Se for **true**, habilita esta funcionalidade.
- Selecionar o conteúdo do campo ao ganhar foco:
  - **selectOnFocus** → Se for **true**, habilita esta funcionalidade.
- Preenchimento reverso → Máscara começa da direita:
  - **reverse** → Se for **true**, habilita esta funcionalidade.
- Placeholder → Texto informativo
  - **placeholder** → A string será colocada como placeholder do campo.

Recursividade com Dígitos

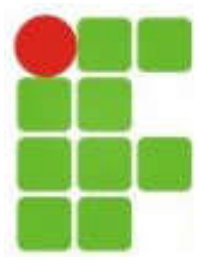


- Permite que um padrão envolvendo números possa se repetir dentro da máscara;

- Uso do caractere especial **#**;

- Exemplo: “#.##0,00”
  - Obrigatório: Um dígito na parte real e dois na parte fracionária;
  - Recursividade: A cada três dígitos presentes, eles estarão separados por ponto, sendo 1 à esquerda e dois à direita;
  - 9,99; 99,99; 999,99; 9.999,99; 99.999,99, etc.

Definindo novos símbolos especiais



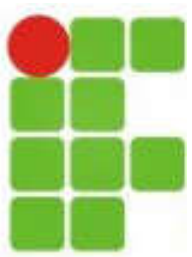
- É possível definir, utilizando expressões regulares, qual o sentido de um símbolo arbitrariamente;

- Isto é feito através da opção **translation**;

{SÍMBOLO1: {DEFINIÇÕES},
SÍMBOLO2: {DEFINIÇÕES}...}

- Onde:
  - SÍMBOLO → Letra a ter um significado especial;
  - DEFINIÇÕES → Definição do significado do SÍMBOLO.

Definição de símbolo especial com expressão regular



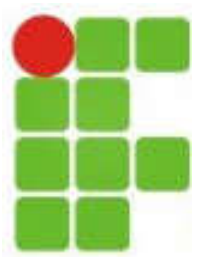
- Dentro das definições de um símbolo especial, é possível definir quais caracteres são válidos via expressão regular;

- Para isto, utilize o parâmetro pattern:

```
pattern: /expressão regular/
```

- **Como um símbolo especial representa apenas um caractere, não devem ser utilizados quantificadores e nem agrupadores!**

Definição de símbolo especial opcional



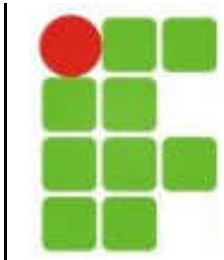
- Dentro das definições de um símbolo especial, é possível fazer com que ele seja opcional, assim como o caractere especial **9**;

- Para isto, utilize o parâmetro **optional**:

```
optional: true
```

- Deste modo, ele não será mais obrigatório na máscara, mas poderá estar presente.

Definição de símbolo especial com recursividade



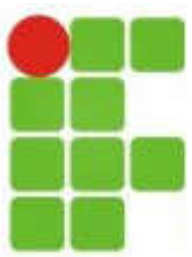
- Dentro das definições de um símbolo especial, é possível prever sua repetição

- Para isto, utilize o parâmetro **recursive**:

```
recursive: true
```

- Deste modo, ele poderá aparecer diversas vezes no padrão sequencialmente.

Definição de símbolo especial com recursividade



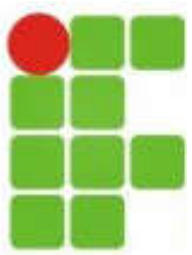
- Dentro das definições de um símbolo especial, é possível que, ao invés de suprimir um caractere inválido, o mesmo seja automaticamente substituído;

- Para isto, utilize o parâmetro **fallback**:

fallback: “caractere”

- Deve ser utilizado com o parâmetro **pattern**.

Removendo a máscara de um campo

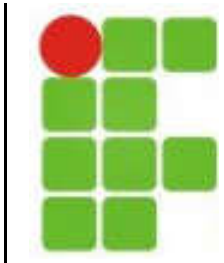


- Para remover a máscara de um campo, siga a seguinte sintaxe:

```
CAMPO.unmask()
```

- Onde:
  - CAMPO → Seletor do campo onde deseja-se aplicar a máscara;
  - MÁSCARA → String representando a máscara a ser aplicada.

Eventos com callback nas opções



- Quando o usuário interage com o campo, é possível tratar estas ações, tratando eventos;

- Os eventos são parâmetros definidos nas opções da máscara, e seus valores devem ser funções de callback;

- **onComplete** → Quando o campo é preenchido completamente de acordo com a máscara;
- **onChange** → Quando o valor do campo é modificado;
- **onInvalid** → Quando um caractere inválido é digitado.