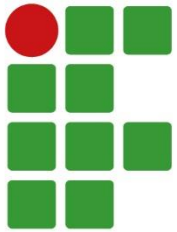


INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

Aula 07

JSON e AJAX

Prof. Tiago Henrique Trojahn

tiagotrojahn@ifsp.edu.br

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Introdução

- Uma das necessidades básicas da web é a troca de mensagens entre computadores diferentes
  - Diferentes programas, sistemas operacionais, hardware...

- Como fazer a comunicação ser efetiva, tanto pro computador como para nós, seres humanos?

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Que tal XML?

```
<biblioteca>
        <livro>
            <autores>
                <autor>Ingrid</autor>
                <autor>Paulo</autor>
            </autores>
            <titulo>O livro</titulo>
            <ano>1998</ano>
        </livro>
        <livro>
            <autores>
                <autor>Paulo</autor>
                <autor>Antônia</autor>
            </autores>
            <titulo>A história</titulo>
            <ano>2016</ano>
        </livro>
</biblioteca>
```

- É de fácil leitura por seres humanos!
- Um computador consegue compreender facilmente também!
- Porém: desperdiça muito espaço para representar os dados.

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Que tal então arquivos binários?

```
10101000101001010111011100100011100
01101110001111001101010101100110101 0
10011010101010101010101010100000010
10101111100000110110101010101010101 0
11111110010101010100100010101010010 1
01010101010101010101011101010101 01
01010101010101111110000000
```

- É rápido e econômico
- Computadores consegue lidar com ele

- Porém
  - Podem ser criados sem qualquer padrão, deixando de funcionar em alguns casos
  - Nenhum ser humano consegue entender isso!

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Solução?

- JSON!
  - Inspirado em XML, é padronizado o suficiente para ser compreendido pelos computadores e por seres humanos!
  - É enxuto o suficiente, não precisando transmitir dados desnecessários.
  - Baseado na representação de objetos de JavaScript, mas que pode ser usado em qualquer outra linguagem!
  - *It just works!*

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

JSON

- Imagine a seguinte informação:

nome	Sherlock
sobrenome	Holmes
profissao	Detetive
telefone	5555-1234

Usando XML...

```
<nome>Sherlock</nome>
<sobrenome>Holmes</sobrenome>
<profissao>Detetive</profissao>
<telefone>5555-1234</telefone>
```

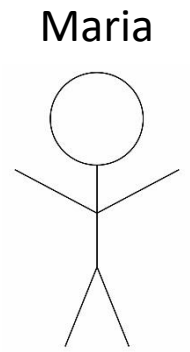
Usando JSON...

```
{
    "nome": "Sherlock",
    "sobrenome": "Holmes",
    "profissao": "Detetive",
    "telefone": "5555-1234"
}
```

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Formato de arquivo

- Os dados são organizados em pares, chamado de *chave* e *valor*, separados por :




```
{"nome" : "maria"}
```

Chave

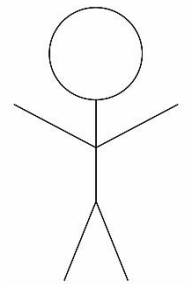
Valor

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Formato de arquivo

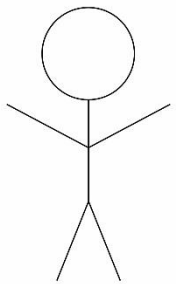
- {} → Representa um objeto

Maria



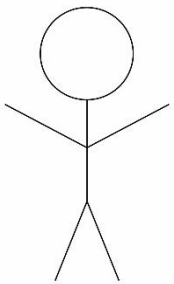
```
{"nome" : "maria"}
```

José



```
{"nome" : "josé"}
```

João

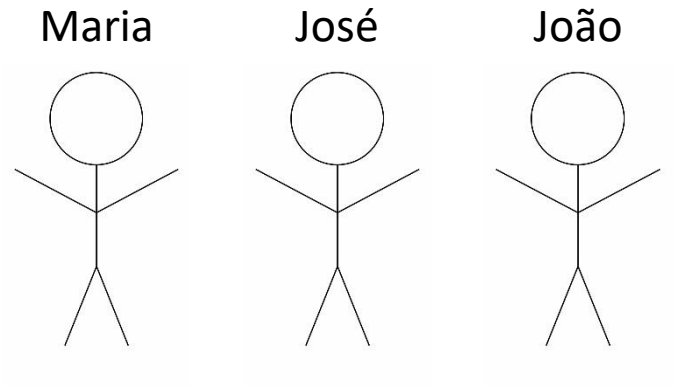


```
{"nome" : “joão"}
```

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Formato de arquivo

- [] → Representa um vetor ou conjunto de dados




```
[
    {
    "nome": "maria"
    },
    {
    "nome": "joão"
    },
    {
    "nome": "josé"
    }
]
```

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Valores possíveis

- Um valor pode ser:
  - String
  - Número
  - true ou false
  - null
  - Objeto
  - Array

- Apenas strings devem estar entre aspas duplas “como essa aqui”.

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Exemplo

nome	Moriarty
idade	34
profissao	Gênio do crime
vivo	true
extra	1234.65
	false
	vivo
	{“ajudantes” : 0}

```
{
    "nome": "Moriarty",
    "idade": 34,
    "profissao": "Gênio do crime",
    "vivo": true,
    "extra": [
        1234.65,
        false,
        "vivo",
        {
            "ajudantes": 0
        }
    ]
}
```

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Exercício

- Crie um JSON para o XML ao lado

```
<biblioteca>
        <livro>
            <autores>
                <autor>Ingrid</autor>
                <autor>Paulo</autor>
            </autores>
            <titulo>O livro</titulo>
            <ano>1998</ano>
        </livro>
        <livro>
            <autores>
                <autor>Paulo</autor>
                <autor>Antônia</autor
                >
            </autores>
            <titulo>A
            história</titulo>
            <ano>2016</ano>
        </livro>
</biblioteca>
```

INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos

Exercício - Gabarito

```xml
<biblioteca>
        <livro>
            <autores>
                <autor>Ingrid</autor>
                <autor>Paulo</autor>
            </autores>
            <titulo>O livro</titulo>
            <ano>1998</ano>
        </livro>
        <livro>
            <autores>
                <autor>Paulo</autor>
                <autor>Antônia</autor>
            </autores>
            <titulo>A história</titulo>
            <ano>2016</ano>
        </livro>
</biblioteca>
```

```json
{
    "livros": [
        {
            "autores": [
                "Ingrid",
                "Paulo"
            ],
            "titulo": "O livro",
            "ano": 1998
        },
        {
            "autores": [
                "Paulo",
                "Antônia"
            ],
            "titulo": "A história",
            "ano": 2016
        }
    ]
}
```

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

JSON com JavaScript

```
var pessoa = '{"nome" : "Moriarty", "idade" : 34}';
var json_convertido = JSON.parse(pessoa);

console.log(pessoa);
console.log(json_convertido);
```

`JSON.parse(string);`

- Converte uma *string* no formato JSON para um objeto em JavaScript

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

JSON com JavaScript

- Depois de ser convertido em objeto, os dados podem ser acessados facilmente pelo JavaScript

```
var pessoa = '{"nome": "Moriarty", "idade": 34,
              "profissao": "Gênio do crime"}';
var json_convertido = JSON.parse(pessoa);

console.log(json_convertido.nome);
if(json_convertido.idade > 18) {
      console.log("Maior de idade!");
} else {
      console.log("Menor de idade!");
}
console.log(json_convertido.profissao);
```

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

JSON com JavaScript

- Um objeto também pode ser transformado em JSON...

```
var objeto_pessoa = {
      "nome" : "Moriarty",
      "idade" : 34,
      "profissao" : "Gênio do crime"
   }

var json_string = JSON.stringify(objeto_pessoa);
console.log(json_string);
```

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

JSON com jQuery

- Podemos obter um JSON da internet usando jQuery

```
$.getJSON(url, dados, funcao_para_tratamento);
```

- Os dados obtidos já são automaticamente “convertidos” de JSON para objetos JavaScript

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

JSON com jQuery

```
$.getJSON(url, dados, funcao_para_tratamento);
```

- url:

  “site.php”, “pagina_externa.html”, “http://webservice.com/acessar.php” ...

- dados (opcional):

  “idade=10”, “nome=ifsp”, “op=cadastro&nome=web&q=10+90”, ...

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

JSON com jQuery

```
function funcao_para_tratamento(dados) {
   console.log(dados.nome);
   console.log(dados.idade);
   console.log(dados.profissao);
}

$.getJSON("pagina_de_exemplo.php", funcao_para_tratamento);
```

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

JSON com jQuery

- Os dados retornados pelo endereço podem ter vários objetos. Nesse caso, é necessário percorrer os objetos como no exemplo abaixo:

```
function funcao_para_tratamento(dados) {
    for (var i = 0; i < dados.length; i++) {
        console.log(dados[i].algo);
        console.log(dados[i].outro_algo);
    }
}
```

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

JSON com jQuery

- Os dados podem ser percorridos também usando o método *each* do jQuery (para objetos)

```
function funcao_para_tratamento(dados) {
    $.each(dados, function(chave, valor) {
        console.log("O valor de " + chave + " é " + valor);
    });
}
```

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Exercício

- A URL https://ifspsaocarlos.000webhostapp.com/ifsp/exemplo_carros.php retorna um JSON contendo um array de objetos representando diferentes carros.
- Desenvolva um website que exiba as particularidades de cada modelo de carro retornado usando uma tabela, com as seguintes propriedades:
  - Nome
  - Fabricante
  - Ano
  - Potência
  - Preço
  - Acessórios (um array, representado na tabela por um <ul>)
- Use o jQuery.

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Exercício

- Existe um webservice que oferece os dados de uma localidade no Brasil dado seu CEP.
- Usando o código fonte presente no moodle, desenvolva um sistema que, dado um CEP digitado pelo usuário (somente números, oito dígitos), preencha os campos com os respectivos valores obtidos via JSON.
- Trate os casos de CEP inválido com um alerta de “CEP não encontrado”.
- O JSON pode ser obtido usando o seguinte endereço de consulta:

```
"https://viacep.com.br/ws/CEP/json/unicode/"
```

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

AJAX

- AJAX é um conjunto de tecnologias utilizada por sites modernos, permitindo iterações rápidas e fluídas ao usuário.

- AJAX – **A**synchronous **J**avaScript **a**nd XML

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

AJAX

- Síncrono: o próximo comando/instrução só é executado quando o comando/instrução atual for finalizada
  - Exemplo: o site carrega apenas quando todos os dados estão carregados

- Assíncrono: permite que a próxima instrução/comando execute mesmo enquanto o comando atual está sendo executado
  - Exemplo: exibir a lista de amigos, mesmo se a lista ainda não está totalmente carregada.

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

AJAX

- O jQuery oferece um método .ajax bastante flexível, capaz de suportar diversas configurações diferentes.
  - Todos as opções são opcionais

```
$.ajax({
    url: 'http://www.algumsite.com', // Destino. Pode ser externo (site) ou local (arquivo)
    type: 'POST', // O método de envio. GET ou POST
    data: {// As informações que deseja-se enviar. Objeto transforma-se em JSON
        "nome": "Mickey Mouse",
        "idade": 18
    },
    async: true, // Se a requisição é assíncrona (true) ou não (false)
    success: function(msg) {
        // Faz algo quando a resposta foi recebida. msg é entendido como json.
        processaResposta(msg);
    },
    error: function(request, status, error) {
        // Faz algo quando houve algum erro.
        alert(error);
    }
});
```

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Type?

- No protocolo HTTP, são previstas diversas formas de comunicação entre um cliente e o servidor. Esses métodos são chamado como verbos HTTP (*HTTP Verbs*). Entre eles:
  - GET
  - HEAD
  - POST
  - PUT
  - DELETE
- Cada verbo tem uma finalidade própria, embora possam ser usadas de maneira intercambiável.
- Mais informações:
  - https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Type?

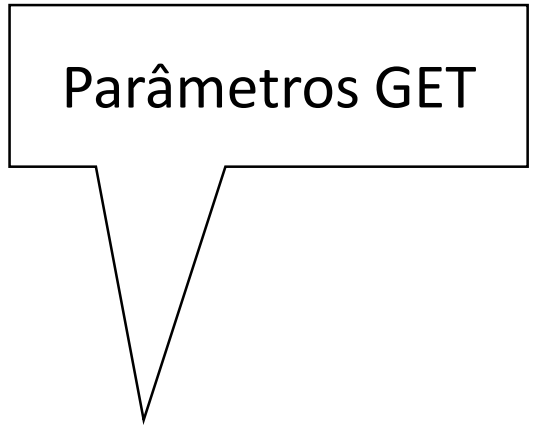
- Na prática, usamos normalmente dois métodos, o **GET** e o **POST.**

GET	POST
Passagem de valores pelo **endereço** da mensagem	Passagem de valores pelo **corpo** da mensagem
Possui um tamanho máximo fixo	Não possui um tamanho máximo
É de fácil manipulação, inclusive por um usuário comum	Mais difícil de ser manipulado
Inadequado para diversos tipos de informações sensíveis, como senhas, endereços de email e outros	Inadequado para identificar o estado ou página particular de um website.

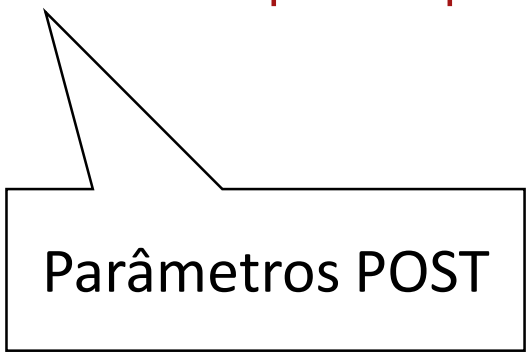
- É importante frisar que uma mesma requisição pode conter tanto parâmetros GET e POST

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

Type?




```
$.ajax({
    url: 'http://www.algumsite.com?id=10&op=cadastrar',
    type: 'POST',
    data: "local=sao+carlos&estado=sao+paulo&pais=brasil"
});
```



INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

getJSON vs ajax?

- Qual a diferença entre os métodos *getJSON* e *ajax*?
  - O *ajax* suporta qualquer tipo de resposta (HTML, XML, JSON). O *getJSON* espera apenas um JSON.
  - *getJSON* não possui diversos campos (como o error do *ajax*)
  - *getJSON* utiliza apenas o método GET. Já o *ajax* suporta os demais métodos ou verbos HTTP.

- Na dúvida, use o método *ajax*

INSTITUTO FEDERAL
São Paulo
Câmpus
São Carlos

AJAX

- Para facilitar, além do método ajax, existem dois métodos auxiliares para os métodos GET e POST.

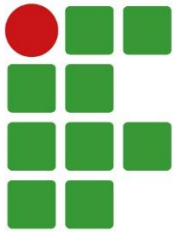
```
$.get("http://site_externo.com/cadastrar.php","pm1=val1&pm2=val2", function(data) {
    // Código a ser executado quando o resultado for obtido
});

$.post("outro_site.php", function (data) {
    // Código a ser executado quando o resultado for obtido
});
```

- Ambos métodos são similares ao getJSON, mas suportam qualquer tipo de retorno (e não apenas JSON).

Obrigado!

Prof. Tiago Henrique Trojahn

tiagotrojahn@ifsp.edu.br



INSTITUTO FEDERAL
São Paulo
Câmpus São Carlos