Programação Web: JavaScript – Criação de tabelas Dinâmicas, LocalStorage e JSON

Prof. Eduardo Lino

eduardo.lino@pucpr.br

Tabelas com valores Dinâmicos

- Para a construção de uma tabela com valores dinâmicos, basta ter por exemplo um array com dados, e modificar o HTML de uma tabela com o uso destes dados.

```
array = ["Eduardo", "Lino", "eduardo@pucpr.br"];

var conteudo = "";

conteudo += "<tr>";
conteudo += "<td>" + array[0] + "</td>";
conteudo += "<td>" + array[1] + "</td>";
conteudo += "<td>" + array[2] + "</td>";
conteudo += "</tr>";

$("#tabela").append(conteudo);

ou

$("#tabela").html(conteudo);
```

Tabelas com valores Dinâmicos

```html
<table id="tabela" border="1">
    <tr>
        <td>Nome</td>
        <td>Sobrenome</td>
        <td>E-mail</td>
    </tr>
</table>
```

```javascript
dados = [["Eduardo", "Lino", "eduardo@pucpr.br"], ["Alan", "Turing", "turing@pucpr.br"]];

var conteudo = "";

for(var i = 0; i < dados.length; i++){

    conteudo = "";
    conteudo += "<tr>";
    conteudo += "<td>" + dados[i][0] + "</td>";
    conteudo += "<td>" + dados[i][1] + "</td>";
    conteudo += "<td>" + dados[i][2] + "</td>";
    conteudo += "</tr>";

    $("#tabela").append(conteudo);
}
```

Tabelas com valores Dinâmicos

- Para adicionar array dentro de array, basta adicionar com o método push/unshift/splice um novo array para o array anterior, como no exemplo abaixo:

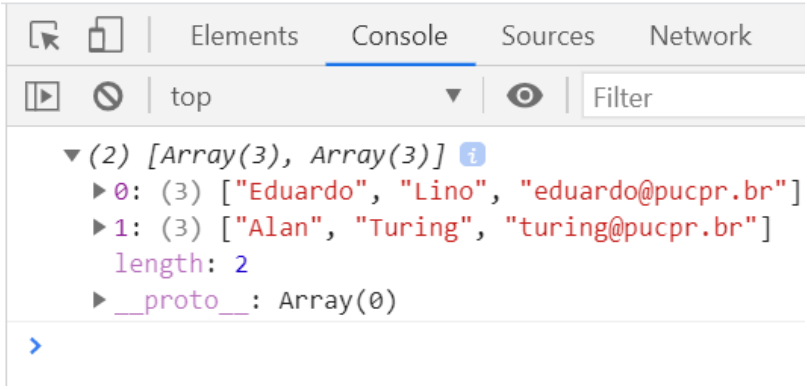
```
dados = [];
aux = [];
aux.push("Eduardo");
aux.push("Lino");
aux.push("eduardo@pucpr.br");

dados.push(aux);

aux = [];
aux.push("Alan");
aux.push("Turing");
aux.push("turing@pucpr.br");

dados.push(aux);
```

```
console.log(dados);
```



Exercício: Tabela com valores Dinâmicos

- Crie uma nova aplicação com HTML, CSS e JavaScript para o cadastro de usuários. Ao cadastrar um novo usuários com os dados abaixo, deverá automaticamente atualizar a tabela com um novo registro.

- Dados do usuário: **nome, sobrenome, idade e email**.

JavaScript: LocalStorage

- A propriedade **localStorage** permite acessar um objeto Storage local. A **localStorage** é similar ao **sessionStorage**. A única diferença é que enquanto os dados armazenados no **localStorage** não expiram, os dados no **sessionStorage** tem os seus dados limpos ao expirar a sessão da página — ou seja, quando a página (aba ou janela) é fechada.

- Os valores em um localStorage são sempre Strings, desta forma, caso queira armazenar um array/vetor, deve converter para um formato JSON, e ao obter novamente os dados, transformar para JSON novamente, já que na localStorage os dados serão um String em formato JSON.

JavaScript: LocalStorage - sintaxe

```
window.localStorage.setItem("chave", "valor");

window.localStorage.getItem("chave");

window.localStorage.removeItem("chave");

window.localStorage.clear();
```

JavaScript: LocalStorage - sintaxe

```
dados = ["Eduardo", "Lino", "eduardo@pucpr.br"];
storage = window.localStorage;

storage.setItem("dados", JSON.stringify(dados));

var dadosDoStorage = JSON.parse(storage.getItem('dados'));
```

JavaScript: JSON

- O método **JSON.stringify()** converte valores em javascript para uma String JSON.

```
JSON.stringify(valor[, replacer[, espaço]])
```

- O método **JSON.parse()** analisa uma string JSON, construindo o valor ou um objeto JavaScript descrito pela string.

```
JSON.parse(text[, reviver])
```

O que é o JSON?

- JSON (**J**avaScript **O**bject **N**otation) ou Notação de Objetos JavaScript é um formato de dados baseado em texto leve completamente independente de linguagem.

- É baseada em um subconjunto da linguagem de programação JavaScript, sendo fácil de entender, gerar e manipular.

O que JSON **não** é?

- ✘ Excessivamente complexa.
- ✘ Um "formato" de documento.
- ✘ Uma linguagem de marcação.
- ✘ Uma linguagem de programação.

Porque utilizar JSON?

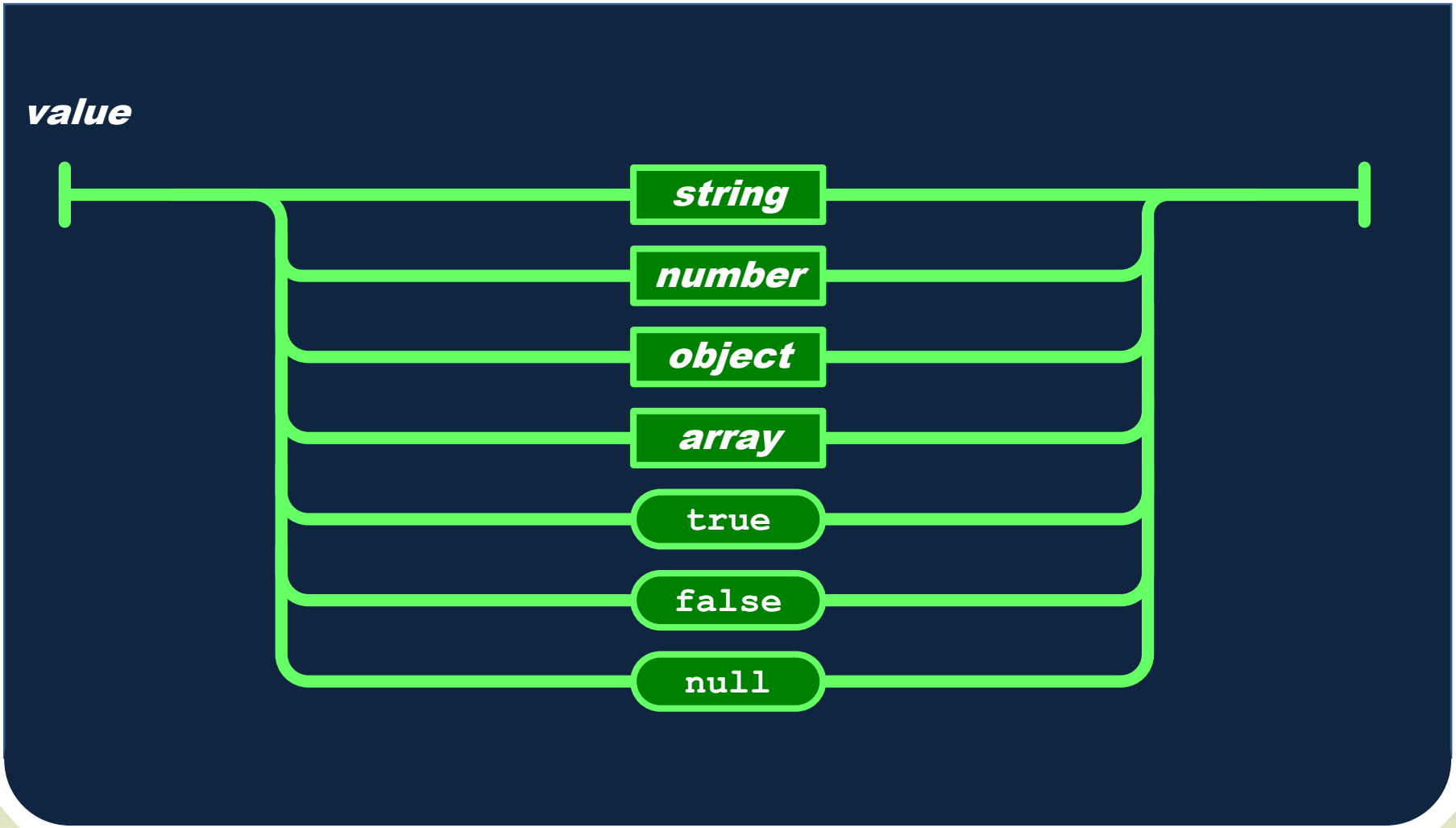
- É uma sintaxe direta.
- Pode ser analisada por diversas linguagens, entre elas, JavaScript, PHP, Java, C#, Python, Delphi, Lisp, Ruby, Objective-C, C/C++, entre outras.
- Fácil de entender, gerar e manipular.

Algumas empresas que utilizam JSON

Quais valores JSON aceita?

- Strings.
- Numbers.
- Booleans.
- Objetos.
- Arrays.
- **`Null`.**

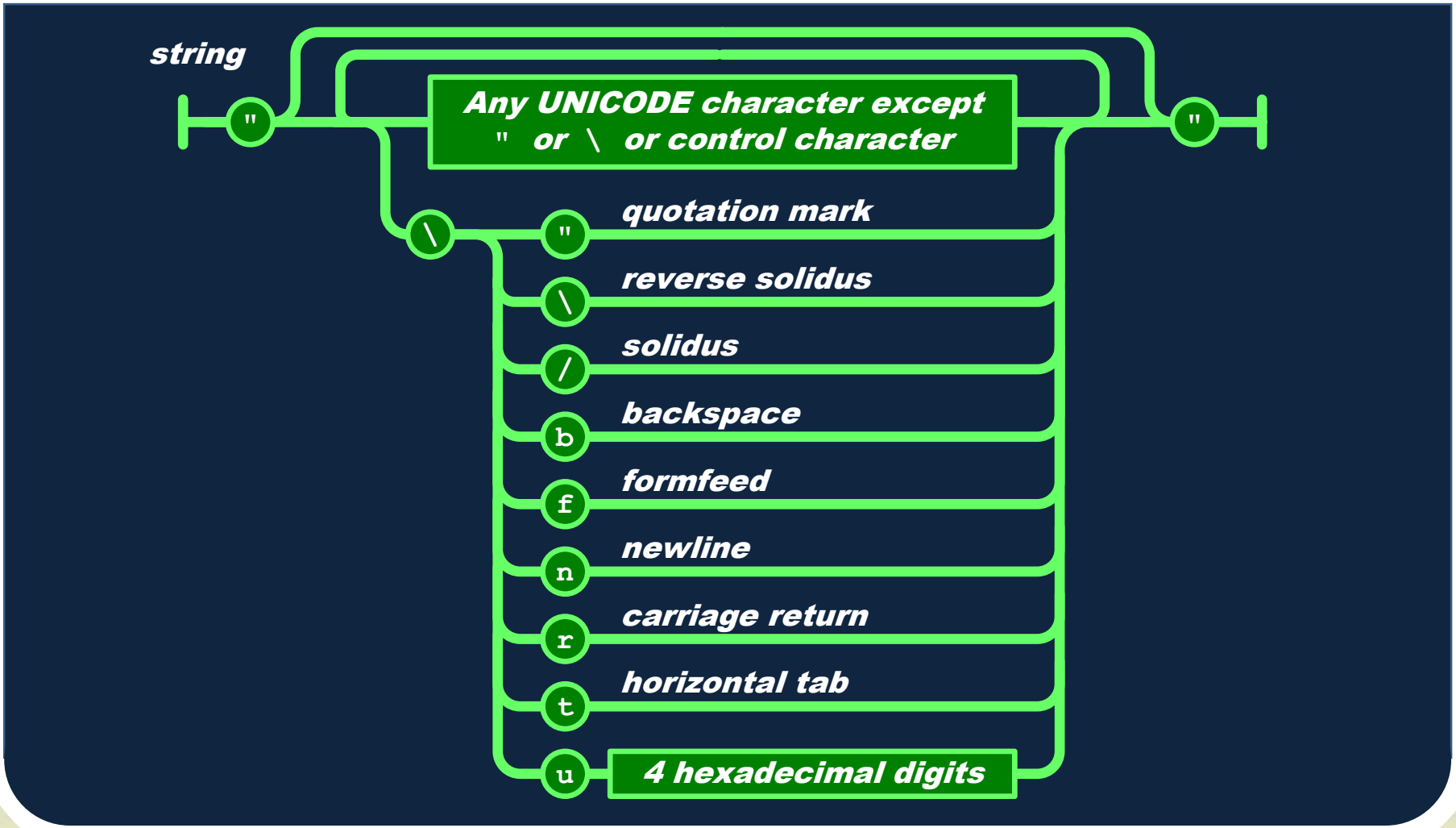
Quais valores JSON aceita?



JSON: Strings

- Sequencia de zero ou mais caracteres.
- Envolta de "aspas duplas”.
- Não aceita dentro das apas apenas uma barra invertida “\” separadamente.

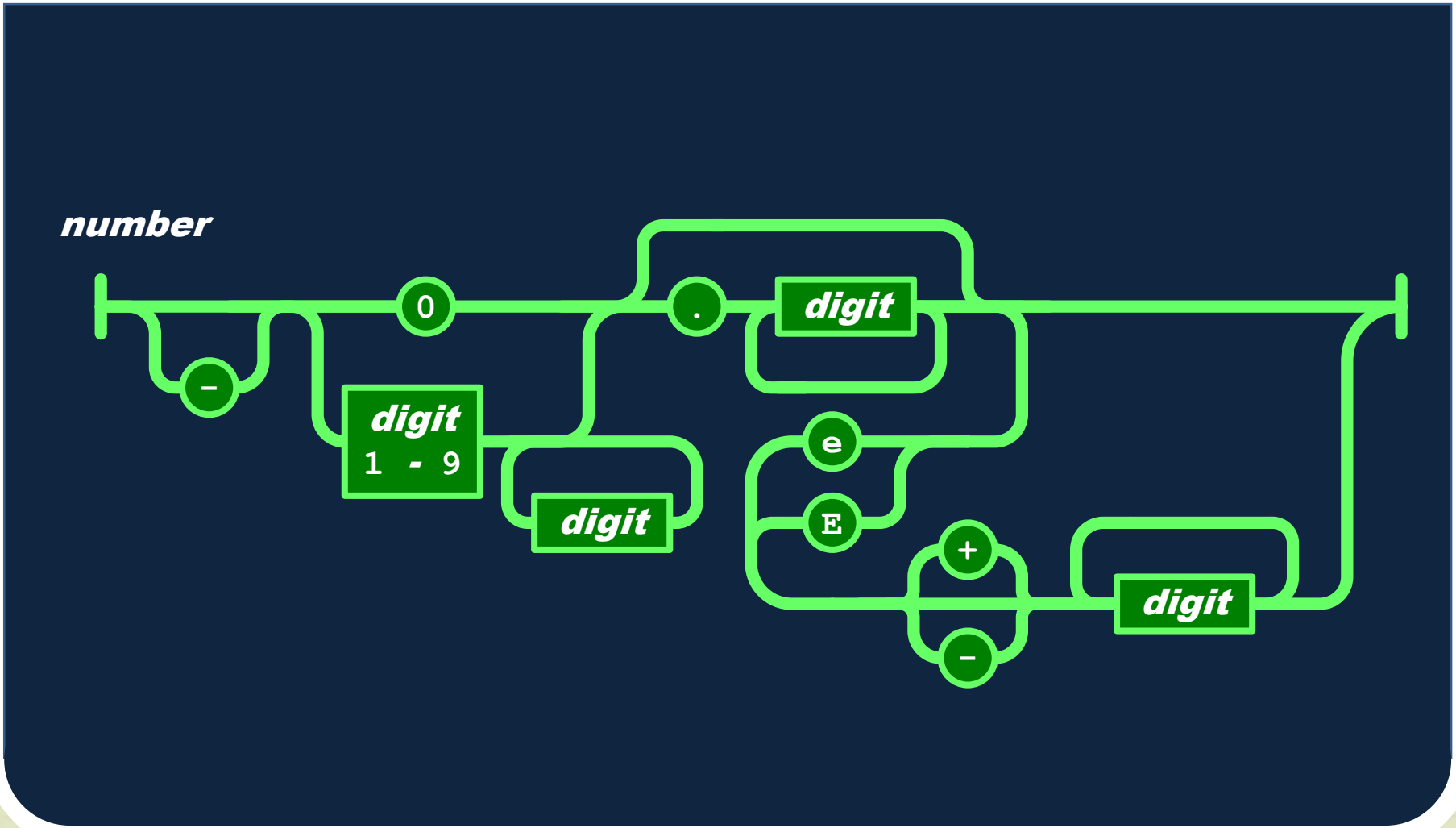
JSON: Strings



JSON: Number

- Inteiro.
- Real.
- Não aceita Octa ou Hexadecimal.
- Não aceita **`NaN`** ou **`Infinity`**.
  - Usa apenas **`null`**.

JSON: Number



JSON: Booleans

- True.
- False.

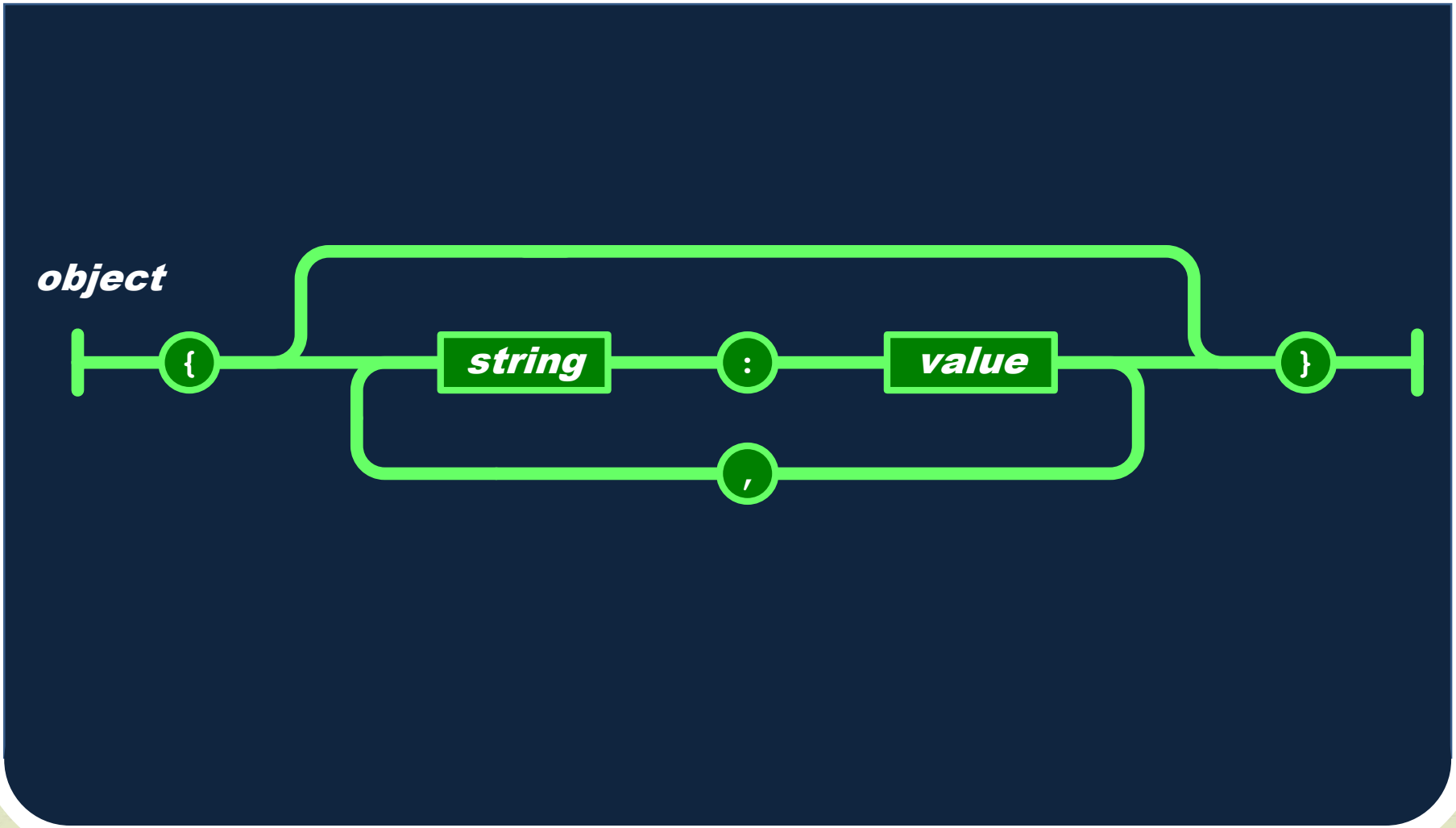
JSON: Null

- Null é um valor que não é nada, ou seja, sem valor.

JSON: Objeto

- Objetos são espaços que contém valores não ordenados de pares de chave/valor.
- Objetos são envolvidos em { }.
- ”,” separa os pares chave/valor.
- ”:” separa chaves e valores.
- Chaves são strings.

JSON: Objeto



JSON: Objeto

```
{"nome":"Alan Turing","casado":
true,"nacionalidade":"Britânico",
"nascimento":1912,
"falecimento":{"local":"Reino
Unido","ano":1954}}
```

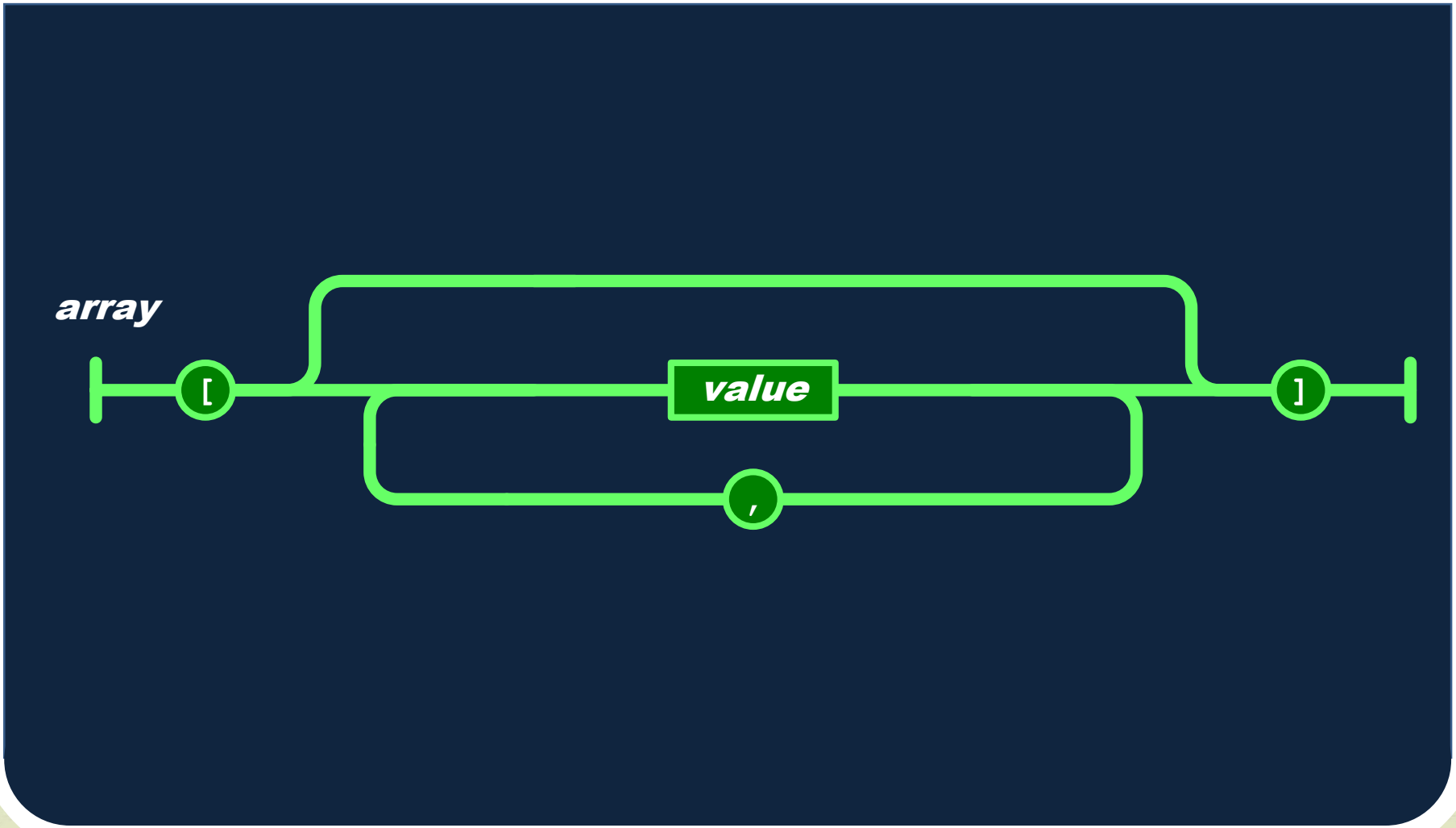
JSON: Objeto

```
{
    "nome":           "Alan Turing",
    "casado":         true,
    "nacionalidade": "Britânico",
    "falecimento": {
        "local":    "Reino Unido",
        "ano":      1954,
    }
}
```

JSON: Array

- Arrays são sequências ordenadas de valores.
- Arrays são envolvidos por [ ].
- ”,” separa os valores.
- Um array pode ser iniciado com o índice 0 ou 1.

JSON: Arrays



JSON: Arrays

```
["Domingo", "Segunda", "Terça",
"Quarta", "Quinta", "Sexta",
"Sabado"]

[
   [0, -1, 0],
   [1, 0, 0],
   [0, 0, 1]
]
```

Exercício: LocalStorage

- Crie uma nova aplicação com HTML, CSS e JavaScript para o cadastro de usuários. (pode utilizar o mesmo do exercício anterior). Ao cadastrar um novo usuários com os dados abaixo, deverá automaticamente atualizar um array e enviá-lo para o localStorage.
- Dados do usuário: **nome, sobrenome, idade e email**.

- Deverá ter um botão na página para redirecionar para outra página. Nesta outra página, deverá obter os dados do localStorage que foi populado na página anterior, e lista-los na tela dentro de uma <table>.

Contato

eduardo.lino@pucpr.br