

ECMAScript 6

Rafael Escalfoni

*adaptado de ECMAScript6 - Entre de cabeça
no futuro do JavaScript*



Sets e Weaksets

- Quando necessitarmos ter coleções com itens que não se repetem.
- Os itens podem ser qualquer coisa: números, strings, objetos ou mesmo funções





Loteria online

- Os usuários podem fazer o jogo eletronicamente
- Como o jogador não pode repetir números no seu jogo, usaremos um Set.
- Programando um Set:

```
function Set() {  
    var array = [];  
    this.add = function(valor) {  
        if(array.indexOf(valor) === -1) {  
            array.push(valor);  
        }  
    }  
    this.print = function() {  
        console.log(array);  
    }  
}
```





Métodos Set

- add(value) - inserir valor no Set
- delete(value) - remover um valor no Set
- clear() - remover todos os itens do Set
- has(value) - retorna **true** se o valor estiver presente no Set



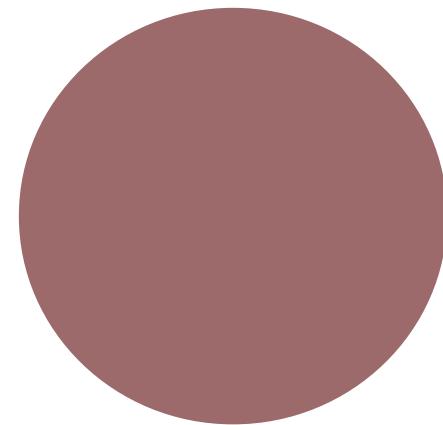
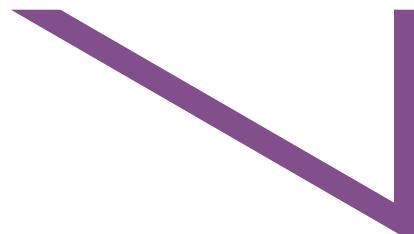


Weakset

- Muito similar ao Set, só que os valores precisam ser objetos.
- WeakSet não é iterável
- Não possui o método clear



Declaração de Variáveis em ECMAScript



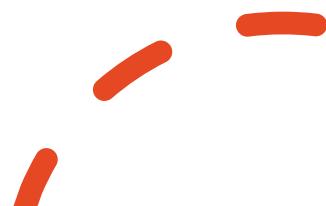


Linguagem fracamente tipada

- JavaScript é fracamente tipada, como Python, PHP, R

```
// Exemplos
var objeto = {};
var numero = 1;
var nome = "Alfredo";
var lista = [1,2,3];
```

- O interpretador JavaScript identifica os tipos pelos valores atribuídos e possibilita operá-los.





JavaScript tipado???

- TypeScript

[https://www.typescriptlang.org/.](https://www.typescriptlang.org/)

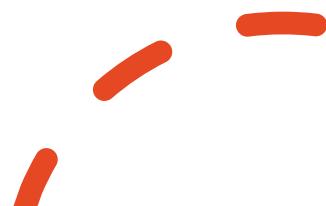




Constantes - **const**

- Quando quisermos que uma variável não altere o seu valor após ter sido criada, utilizaremos **constantes**
- **const** cria uma *referência constante* - o valor não é imutável
 - Se tentarmos atribuir **um novo valor** a variável, **dá erro**.

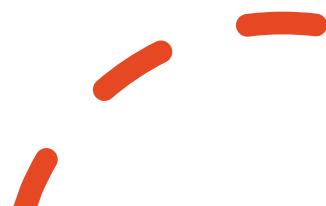
```
const dataNascimento = '21/07/1997';
dataNascimento = '25/08/2017';
// TypeError: Assignment to constant variable
```





let é o novo var

- A palavra reservada **let** foi criada para definir variáveis com um escopo diferente.
- Lembrando: **var** define variáveis em escopo global ou léxico (dentro de funções) mas não define escopo em bloco de comandos
- **let** define variáveis em escopo local e sinaliza erros em caso de redundância de variáveis, por exemplo.



var vs let

```
var mensagem = "Olá";
{
    var mensagem = "adeus";
}
console.log(mensagem);
```

```
const arrayVar = [];
for(var i = 1; i < 5; i++) {
    arrayVar.push(function () {
        console.log(i);
    });
}

const arrayVar = [];
for(let i = 1; i < 5; i++) {
    arrayVar.push(function () {
        console.log(i);
    });
}

arrayVar.forEach(function(funcao){
    funcao();
})

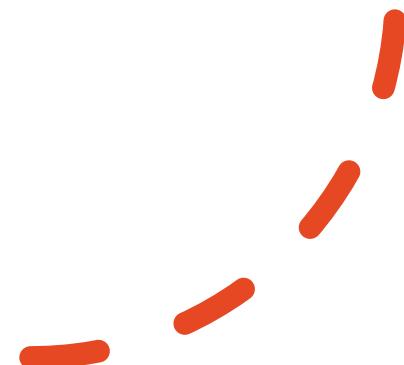
arrayLet.forEach(function(funcao){
    funcao();
})
```

Manipulando Textos com Template String



**Estamos
sempre às
voltas com:**

- Mensagens de texto exibidas para os usuários
- Construção de logs e auditoria
- Mecanismos de busca
- Tratamento de textos (leitura/escrita)
- Ordenação
- ...



Facilidades de String

- Ler e escrever palavras, frases e textos
- Descobrir seu tamanho
- Criar substrings
- Fazer diferenciação e ordenação
- Buscar elementos específicos (com expressões regulares)
- Eliminar e transformar
- ...



Exemplo

```
const login = "ecmascript";
const dia = "7 de abril";
const ano = "2021";

const mensagem = "Olá, " + login + "\nHoje é: " + dia + " de "
+ ano;

console.log(mensagem);

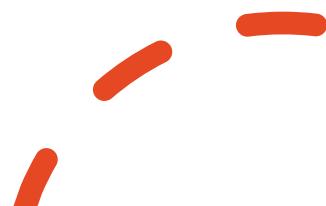
// saída:
// Olá, ecmascript!
// Hoje é: 7 de abril de 2021
```





Problemas

- Concatenação de expressões enormes - dificulta a legibilidade
- Precisamos lidar com quebras de linhas com caracteres especiais (\n, \r)
- Template Strings!
 - Template strings simples
 - Template strings marcados (tags)





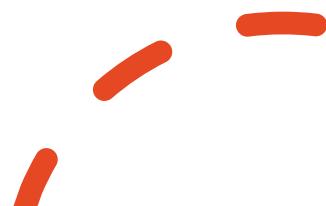
Template String Simples

- Em vez de:

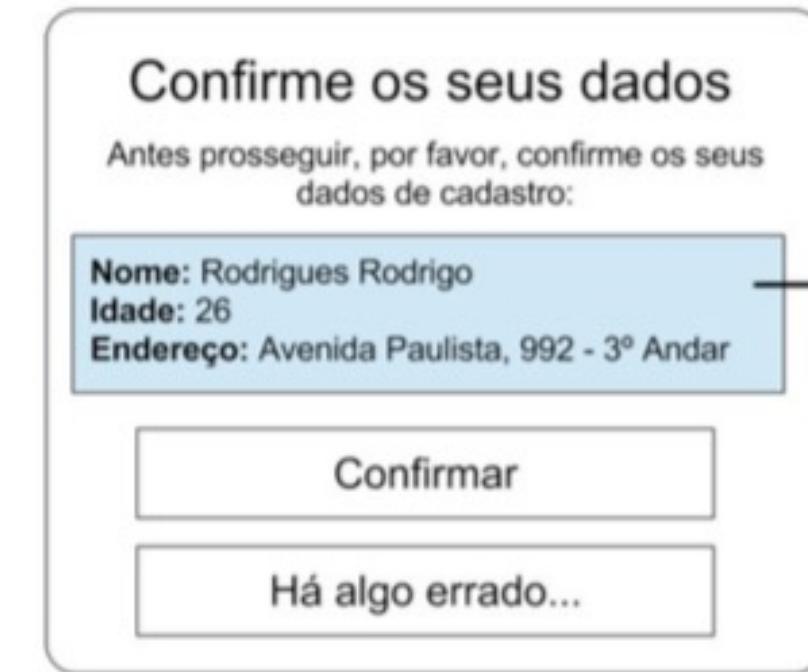
```
let nome = "Roberto";  
console.log("Bem-vindo, " + nome);  
// saída: Bem-vindo, Roberto
```

- Que tal:

```
let nome = "Roberto";  
console.log(`Bem-vindo, ${nome}`);
```



```
const div =  
`<div>  
  <p><b>Nome:</b> ${nome}</p>  
  <p><b>Idade:</b> ${idade}</p>  
  <p><b>Endereço:</b> ${endereco}</p>  
</div>`;
```





Template Strings Marcado

- Considere um sistema com saudações específicas
 - Bom dia, boa tarde ou boa noite, dependendo do horário

```
const horas = new Date().getHours();
const mensagem = `Bom dia, são ${horas} horas`;
console.log(mensagem); // ex: Bom dia, são 12 horas
```

Horários	Saudação esperada
6h - 12h	Bom dia
12h - 18h	Boa tarde
18h - 6h	Boa noite



Template Strings Marcado

```
const mensagem = defineMensagem`Bom dia, são ${horas} horas`;
```

- Podemos usar a tag para definir uma função que recebe dois parâmetros:
 - Uma lista de strings
 - Cada uma das expressões do template

Horários	Saudação esperada
6h - 12h	Bom dia
12h - 18h	Boa tarde
18h - 6h	Boa noite

Template Strings Marcado

```
const mensagem = defineMensagem`Bom dia, são ${horas} horas`;

function defineMensagem(strings, ... values) {
  console.log(strings);
  console.log(values);
}
```



Template Strings Marcado

```
const mensagem = defineMensagem`"${}" ${horas} horas`;

function defineMensagem(strings, ...values) {
  const hora = values[1];
  if(hora >= 6 && hora <= 12) {
    values[0] = "Bom dia";
  } else if (hora > 12 && hora < 18) {
    values[0] = "Boa tarde";
  } else {
    values[0] = "Boa noite";
  }
  values[0] = `${values[0]}, são `;
  return `${values[0]}${strings[0]}${hora}${strings[2]}`;
}
```

