

CLASSES

Classes

Constructor Function

Funções responsáveis pela criação de objetos. O conceito de uma função construtora de objetos é implementado em outras linguagens como Classes.

```
function Button(text, background) {  
  this.text = text;  
  this.background = background;  
}
```

```
Button.prototype.element = function() {  
  const buttonElement = document.createElement('button');  
  buttonElement.innerText = this.text;  
  buttonElement.style.background = this.background;  
  return buttonElement;  
}
```

```
const blueButton = new Button('Comprar', 'blue');
```

Class

O ES6 trouxe uma nova sintaxe para implementarmos funções construtoras. Agora podemos utilizar a palavra chave `class`. É considerada `syntactical sugar`, pois por baixo dos panos continua utilizado o sistema de protótipos de uma função construtora para criar a classe.

```
class Button {  
  constructor(text, background) {  
    this.text = text;  
    this.background = background;  
  }  
  element() {  
    const buttonElement = document.createElement('button');  
    buttonElement.innerText = this.text;  
    buttonElement.style.background = this.background;  
    return buttonElement;  
  }  
}  
  
const blueButton = new Button('Comprar', 'blue');
```

Class vs Constructor Function

```
class Button {  
  constructor(propriedade) {  
    this.propriedade = propriedade;  
  }  
  metodo1() {}  
  metodo2() {}  
}
```

```
function Button(propriedade) {  
  this.propriedade = propriedade;  
}  
Button.prototype.metodo1 = function() {}  
Button.prototype.metodo1 = function() {}
```

Constructor

O método `constructor(args) {}` é um método especial de uma classe. Nele você irá definir todas as propriedades do objeto que será criado. Os argumentos passados em new, vão direto para o constructor.

```
class Button {  
  constructor(text, background, color) {  
    this.text = text;  
    this.background = background;  
    this.color = color;  
  }  
}  
  
const blueButton = new Button('Clique', 'blue', 'white');  
// Button {text: 'Clique', background: 'blue', color: 'white'}
```

Constructor Return

Por padrão a classe retorna this. Ou seja, this é o objeto criado com o `new Class`. Podemos modificar isso alterando o return do constructor, o problema é que perderá toda a referência do objeto.

```
class Button {  
  constructor(text) {  
    this.text = text;  
    return this.element(); // não fazer  
  }  
  element() {  
    document.createElement('button').innerText = this.text;  
  }  
}  
  
const btn = new Button('Clique');  
// <button>Clique</button>
```

This

Assim como em uma função construtora, `this` faz referência ao objeto criado com `new`. Você pode acessar as propriedades e métodos da classe utilizando o `this`.

```
class Button {  
  constructor(text) {  
    this.text = text;  
  }  
  element() {  
    const buttonElement = document.createElement('button')  
    buttonElement.innerText = this.text;  
    return buttonElement;  
  }  
  appendElementTo(target) {  
    const targetElement = document.querySelector(target);  
    targetElement.appendChild(this.element());  
  }  
}  
  
const blueButton = new Button('Clique');  
blueButton.appendElementTo('body');
```

Propriedades

Podemos passar qualquer valor dentro de uma propriedade.

```
class Button {  
  constructor(options) {  
    this.options = options;  
  }  
}  
  
const blueOptions = {  
  backgroundColor: 'blue',  
  color: 'white',  
  text: 'Clique',  
  borderRadius: '4px',  
}  
  
const blueButton = new Button(blueOptions);  
blueButton.options;
```


Static vs Prototype

Por padrão todos os métodos criados dentro da classe irão para o protótipo da mesma. Porém podemos criar métodos diretamente na classe utilizando a palavra chave `static`. Assim como `[] .map()` é um método de uma array e `Array.from()` é um método do construtor Array.

```
class Button {  
  constructor(text) {  
    this.text = text;  
  }  
  static create(background) {  
    const elementButton = document.createElement('button');  
    elementButton.style.background = background;  
    elementButton.innerText = 'Clique';  
    return elementButton;  
  }  
}  
  
const blueButton = Button.create('blue');
```

Static

Você pode utilizar um método `static` para retornar a própria classe com propriedades já pré definidas.

```
class Button {  
  constructor(text, background) {  
    this.text = text;  
    this.background = background;  
  }  
  element() {  
    const elementButton = document.createElement('button');  
    elementButton.innerText = this.text;  
    elementButton.style.background = this.background;  
    return elementButton  
  }  
  static createBlue(text) {  
    return new Button(text, 'blue');  
  }  
}  
  
const blueButton = Button.createBlue('Comprar');
```