Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (front 1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

# Funções assíncronas em JavaScript

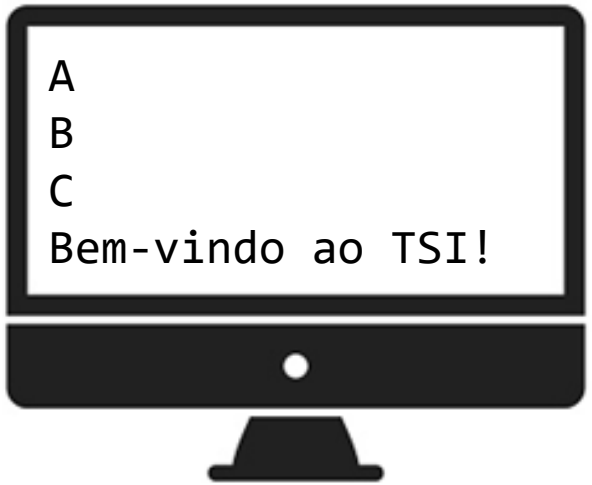
INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas

# Funções de `callback`

São funções executadas quando uma **determinada ação ocorre**.

Pensando em termos de web**: não é possível controlar em que tempo as coisas vão ocorrer.**

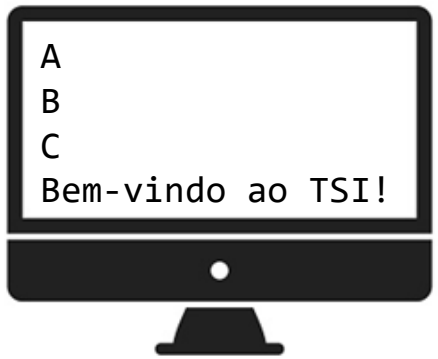
Acessos a BD, retornos de outras páginas ou API, exibição coisas na tela, etc... Não existe controle sobre isso.




```
function f1(){
    console.log('A');
}

function f2(){
    console.log('B');
}

function f3(){
    console.log('C');
}

f1();
f2();
f3();
console.log('Bem-vindo ao TSI!');
```

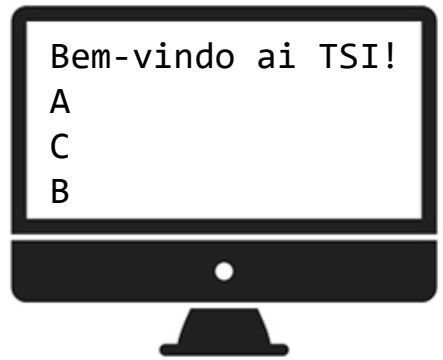
Exemplo_09

```
function f1(){
    console.log('A');
}

function f2(){
    console.log('B');
}

function f3(){
    console.log('C');
}


f1();
f2();
f3();
console.log('Bem-vindo ao TSI!');
```




```
function f1(){
    setTimeout(function(){
        console.log('A');
    }, 500);
}

function f2(){
    setTimeout(function(){
        console.log('B');
    }, 1000);

}

function f3(){
    setTimeout(function(){
        console.log('C');
    }, 800);
}

f1();
f2();
f3();
console.log('Bem-vindo ao TSI!');
```

Adicionamos um *timeout* a cada uma das funções para simular o **caos da web**.




Exemplo_09

# Aumentando o caos

```
// Versão original
function f1(){
    setTimeout(function(){
        console.log('A');
    }, 500);
}

function f2(){
    setTimeout(function(){
        console.log('B');
    }, 1000);

}
...
```

```
// 1) função que gera um número aleatório
// entre 1000 e 3000

function rand(min = 1000, max = 3000){
    const valor = Math.random() * (max-min) + min;

    return Math.floor(valor);
}
```

```
function f1(){
    setTimeout(function(){
        console.log('A');
    }, rand());
}

function f2(){
    setTimeout(function(){
        console.log('B');
    }, rand());

}

function f3(){
    setTimeout(function(){
        console.log('C');
    }, rand());
}

f1();
f2();
f3();
console.log('Bem-vindo ao TSI!');
```

2) Adicionamos um *timeout* a cada uma das funções **para simular o caos da web**.


Exemplo 09

# Utilizando funções de `callback`

- Funções que podem ou não ser que recebidas como parâmetro
- A ideia é garantir a ordem de execução.

```
function f1(){
    setTimeout(function(){
        console.log('A');
    }, rand());
}
```

Vamos modificar cada uma das funções para utilizar um `callback`

1) A função deve receber um parâmetro que será uma função de `callback`.

```
function f1(callback) {
  setTimeout(function() {
    console.log('f1');
    if (callback) callback();
  }, rand());
}
```

2) Executa o que a função se propõe.

3) Verifica se existe o callback. Em caso positivo, chama a função de `callback`.

**4) Alterar todas as funções que precisam estar encadeadas para executar as instruções em ordem.**

# Como organizar a ordem de execução por meio de *callbacks*?

```
function f1(){
    setTimeout(function(){
        console.log('A');
    }, rand());
}

function f2(){
    setTimeout(function(){
        console.log('B');
    }, rand());

}

function f3(){
    setTimeout(function(){
        console.log('C');
    }, rand());
}

f1();
f2();
f3();
console.log('Bem-vindo ao TSI!');
```

```
function f1(callback) {
  setTimeout(function() {
    console.log('A');
    if (callback) callback();
  }, rand());
}

function f2(callback) {
  setTimeout(function() {
    console.log('B');
    if (callback) callback();
  }, rand());
}

function f3(callback) {
  setTimeout(function() {
    console.log('C');
    if (callback) callback();
  }, rand());
}
```

f1

f2

f3

**5) Acertar as chamadas com os `callbacks`.**

## É preciso garantir a ordem de execução!!

```
f1();
f2();
f3();
console.log('Bem-vindo ao TSI!');
```

**Callback HELL**

```
f1(function(){
  f2(function(){
    f3(function(){
      console.log('Bem vindo TSI!');
    });
  });
});
```

**Garantimos a ordem de execução, aninhando `callbacks`.**

**`f1 -> f2 -> f3 -> Bem vindo TSI!`**

Exemplo

# Melhorando o código...


```
f1(function(){
  f2(function(){
    f3(function(){
      console.log('Bem-vindo ao TSI!');
    });
  });
});
```

```
f1(f1Callback);

function f1Callback() {
  f2(f2Callback);
}

function f2Callback() {
  f3(f3Callback);
}

function f3Callback() {
  console.log('Bem-vindo ao TSI!');
}
```

**Garantimos a ordem de execução, aninhando *callbacks*.**

**`f1 -> f2 -> f3 -> Bem-vindo ao TSI!`**

Existe um mecanismo mais moderno (e mais simples ) do que os *callbacks*, que são as ***promises***, que serão abordadas posteriormente.

Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (front 1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

# Funções assíncronas em JavaScript

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas