mat|r

Bienvenidos a la Clase N°06

En unos minutos comenzamos...

Clase 06

Introducción a mat|r script

mat|r project

Ecosistema 360 para la creación de experiencias digitales

2020

Expositor

Agustina Dinamarca

Capacitadora mat|r

Se dedica a representar el producto en actividades que tienen como objetivo educar a usuarios sobre qué es mat|r, para qué sirve y cómo usarlo a través del dictado de webinars y capacitaciones, y el desarrollo de contenido educativo.

OBJETIVOS DE LA CLASE

- Estudiar qué son las funciones en mat|r script, dónde se las definen y cómo usarlas en una aplicación.
- Aprender a asociar funciones a los distintos eventos que ocurren en los componentes visuales de un layout.

HOY APRENDERÁS

- Módulos y funciones. Definiciones. Tipos de funciones: con y sin retorno, y con y sin argumentos.
- Cómo acceder y usar funciones en cualquier lugar del código de una aplicación.
- Cómo vincular funciones a eventos en los componentes visuales de un layout.

MÓDULOS Y FUNCIONES

¿QUÉ ES UN MÓDULO?

UN MÓDULO PROVEE UN MECANISMO PARA AGRUPAR VARIABLES Y FUNCIONES.

- Generan un alcance global.
- Un módulo se define con la palabra reservada: `Module`

Ej.:

```
Module moduleName {


}
```

¿QUÉ ES UNA FUNCIÓN?

UNA FUNCIÓN ES UN BLOQUE DE CÓDIGO AUTÓNOMO QUE PERMITE EJECUTAR SENTENCIAS DE CÓDIGO.

- Cuando se ejecuta una función el control pasa al punto donde esta inicia. Una vez finalizada su ejecución, se devuelve el control al punto desde el que se hizo la llamada a la función.
- No se permite sobrecarga de funciones **en un mismo módulo**. Ej.:

```
Module miModulo {
    Integer funcion1 (Integer num) {
        return num
    }
    Double funcion1 (Double otroNum) {
        return otroNum
    }
}
```

X

VARIABLES DE MÓDULO (REPASO)

- Tienen alcance global.
- Pueden modificar su valor.
- De definen en los módulos y pueden ser de cualquier tipo de dato. Ej.:

```
Module Salario {
  Double  porcentVacaciones
  Bool    esContratoMensual
  Array<Integer>    mesesContrato
  Map<Persona> empleados
}
```

- Pueden ser accedidas y asignadas de forma estática: se especifica el nombre del módulo seguido por un punto y el nombre de la variable. Ej.:

```
Salario.porcentVacaciones = 0.2
```

TIPOS DE FUNCIÓN

1. **Con Parámetro/s y Retorno**
2. **Sin Parámetro/s y con Retorno**
3. **Con Parámetro/s y sin Retorno**
4. **Sin Parámetro/s y sin Retorno**

PARÁMETROS DE FUNCIÓN

LOS PARÁMETROS DE FUNCIÓN SON UN CONJUNTO DE ARGUMENTOS, SEPARADOS POR COMAS Y ENCERRADOS ENTRE PARÉNTESIS.

- Para cada parámetro, se debe definir su tipo y un nombre. Ej.:

```
funcionSalarioBase(Integer aniosConsolidados){
    //Bloque de código
}

funcionSalarioEmpleados(Integer aniosConsolidados, Array<Persona>
empleados){
    //Bloque de código
}
```

RETORNO DE FUNCIONES

FUNCIONES CON RETORNO

1. Una función con retorno debe especificar:
    - el tipo de retorno.
    - nombre de función.
    - paréntesis con sus parámetros.
    - el cuerpo de la función que se define con llaves.

2. El cuerpo de la función termina con la sentencia `return` que marca el final de la función.

EJEMPLO

FUNCIONES CON RETORNO

```
Integer cuadradoDeUnNumero( Integer num) {
        return num*num
}


Persona crearPersona( String name, Integer dni ) {
     Persona p = Persona()
     p.nombre = name
     p.dni = dni
     return p
}
```

Donde,

```
Model Persona {
    String nombre
    Integer dni

}
```

RETORNO DE FUNCIONES

FUNCIONES SIN RETORNO

Para las funciones SIN retorno se debe especificar: la sentencia `void` previo a la definición del nombre de función.

Ej.:

```
void procesarPagosSalarios( Array<Persona> empleados ) {
    service.procesarPagos.call(empleados)
}
```

ACCESO A VARIABLES DENTRO DE UNA FUNCIÓN

Dentro del bloque de una función se pueden acceder a:

- variables locales,
- a las variables definidas en el módulo que contiene a la función o de otros módulos.
- a las globales.

```
Module Salario {
  Double                 porcentVacaciones
  Integer                salarioBase
  Bool                   esContratoMensual
  Map<Persona>     empleados

  Double obtenerSalarioBase(Integer aniosConsolidados) {
    return (Salario.salarioBase + (aniosConsolidados * Salario.porcentVacaciones))
  }
}
```

LLAMADA A FUNCIONES

PUEDEN SER INVOCADAS DE MANERA ESTÁTICA DESDE CUALQUIER BLOQUE DE EJECUCIÓN.

```
nombreModulo.nombreFuncion([parametros])
```

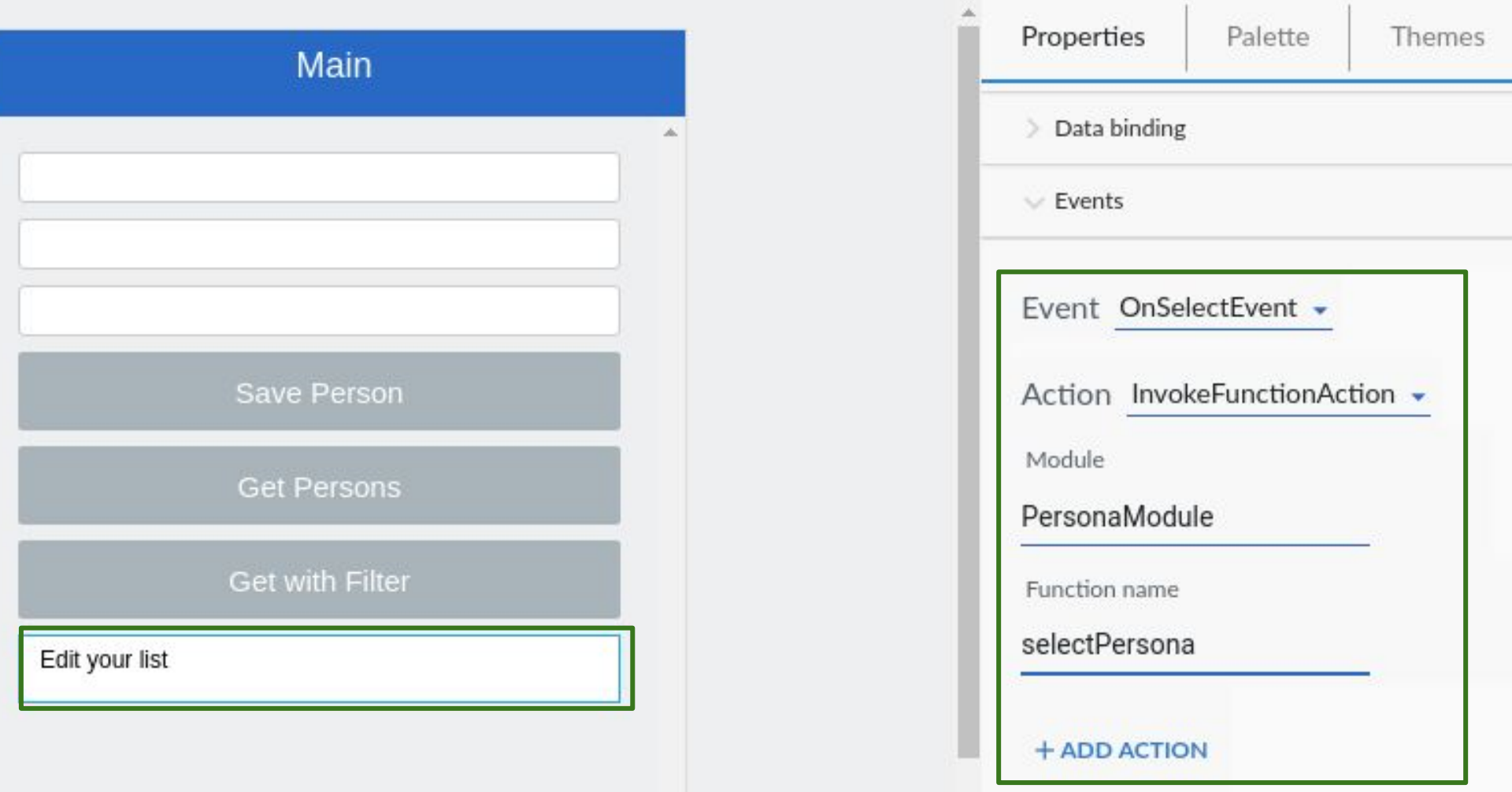
VINCULAR FUNCIONES A EVENTOS EN LA UI

¿CÓMO ASOCIARLOS?

1. Ir al UI Build, al layout de interés, y agrega o selecciona un componente visual.
2. Al hacer click en el componente, ir a “Propiedades”, “Events”.

Según el componente se tendrán habilitados ciertos eventos, y según estos, el tipo de acción.

3. Seleciona un tipo de evento.
4. Selecciona un tipo de acción.
5. Completa los campos que aparezcan.
6. Click en “+ADD ACTION” y listo (:


Main
Save Person
Get Persons
Get with Filter
Edit your list
Properties
Palette
Themes
Data binding
Events
Event OnSelectEvent
Action InvokeFunctionAction
Module
PersonaModule
Function name
selectPersona
+ ADD ACTION

RESUMEN

RESUMEN

- Las funciones pueden o no recibir parámetros o retornar valores, ejecutan código, y pueden ser invocadas a través de eventos o en cualquier lugar de la app.
- Se pueden asociar la acción de funciones a los distintos eventos que ocurren en los componentes visuales de un layout.

EXTRA :D

TIPO DE DATO: TIMER

PROGRAMA UNA LLAMADA A UNA FUNCIÓN LUEGO DE QUE CIERTO INTERVALO DE TIEMPO HAYA TRANSCURRIDO

CONSTRUCTOR	DESCRIPCIÓN
`Timer(delay: double, function: FunctionReference, repeat: Bool)`	Crea una instancia de timer que programa la ejecución de la función argumento, luego de una cantidad de segundos igual al argumento delay. El argumento repeat indica si el timer es repetitivo, es decir, si debe repetirse o no la invocación de la función argumento luego de la primer llamada.
`Timer(delay: double, function: FunctionReference)`	Crea una instancia de timer que programa la ejecución de la función argumento, luego de una cantidad de segundos igual al argumento delay. Este constructor crea una instancia de timer no repetitivo, es decir, la invocación a la función argumento se produce una sola vez.

TIPO DE DATO: TIMER

Para referenciar la función debe utilizarse la sintaxis:

```
@function(nombreModulo.nombreFuncion)
```

y la misma debe cumplir 2 condiciones:

- **retornar void**
- recibir un solo **argumento de tipo Timer.**

MÉTODOS DE TIMER

MÉTODO	DESCRIPCIÓN
`void` **`fire`**`()`	Desencadena la invocación del argumento function con el que fue construida la instancia del timer. En caso de timers repetitivos, los mismos no serán desprogramados. Por el contrario los timers no-repetitivos son desprogramados por más que el tiempo especificado en el argumento delay no haya sido cumplido.
`void` **`start`**`()`	Programa la ejecución de la función argumento con el que fue construida la instancia, que será invocada cuando se complete el tiempo establecido por el argumento de construcción delay. Si repeat = false, la programación del timer es cancelada luego de la invocación a la función. La segunda llamada al método start puede reiniciar el timer nuevamente.

MÉTODOS DE TIMER

MÉTODO	DESCRIPCIÓN
`void stop()`	Detiene el timer en caso de que ya haya sido iniciado (programado), o no produce ningún efecto en caso contrario.

EJEMPLO

1. **Desarrollaremos una aplicación que me muestre longitud recorrida, velocidad promedio y tiempo a medida que me desplazo por la ciudad, usando el GPS del dispositivo. Los valores mencionados se actualizarán automáticamente a medida que detecte nuevas posiciones espaciales.**

   Se usarán *broker..location., Timer, regla con cláusula listen y when, y una función.**

IDEA

Crear una sola experiencia en la cual el usuario:

1. **Podrá visualizar mediante etiquetas la longitud recorrida, tiempo recorrido y rapidez promedio.**
2. **Podrá empezar y parar la adquisición y cálculo de datos con los botones “Empezar” y “Detener” respectivamente.**

¿CONSULTAS O DUDAS?

COMUNÍCATE NOSOTROS

support@matrproject.com

http://forum.matrproject.com

mat|r

matrproject.com

Contacto

Soledad Peñaloza
Community Manager
soledad.p@matrproject.com

Sergio Farias
Product Development
sergio.f@matrproject.com

Agustina Dinamarca
Capacitadora mat|r
agustina.d@matrproject.com

mat|r

¡Muchas Gracias!

mat|r project

Ecosistema 360 para la creación de experiencias digitales