



# PRÁCTICA 6. CSS

---

Interfaces Persona Computador

Depto. Sistemas Informáticos y Computación

UPV

# Índice

- Hojas de estilo en cascada
- CSS en JavaFX
- Usando un fichero CSS externo
- Cambiando el tema
- CSS
- CSS en Scene Builder


# Hojas de Estilo en Cascada (I)

- CSS

- Cascading Style Sheets
- Extensión de HTML básico
- Permite dar estilo a las páginas web

- Ej

- Poner en **negrita** una palabra en HTML estándar, emplear etiqueta `<b>`

... poner `<b>en negrita</b>` un texto ...  ... poner **en negrita** un texto ...

- Si posteriormente se decide cambiar el texto en **negrita** a subrayado:

- Habrá que ir a cada lugar en la página y cambiar la etiqueta `<b>` por `<u>` explícitamente en todos y cada uno de los sitios en los que apareciera

... poner `<u>en negrita</u>` un texto ...  ... poner en negrita un texto ...

# Hojas de Estilo en Cascada (II)

- Si se desea cambiar el estilo de letra a Verdana y cambiar su color a rojo, se necesitaría una gran cantidad de código alrededor del texto:

```
<Font color = "#FF0000" face = "Verdana, Arial, Helvetica, sans-serif"><strong>  
Este es el texto</strong></font>
```

- Inconvenientes:
  - Gestión del estilo explícita
  - Código HTML farragoso
  - Sobrecarga de maquetación excesiva. Superior a la del contenido textual de la página

# Hojas de Estilo en Cascada (III)

- Ventajas CSS

- Crear un estilo personalizado una única vez
- Establecer todas sus propiedades
- Darle un nombre único
- Etiquetar el código HTML para aplicar estas propiedades específicas

```
<p class = "miEstilo"> Mi texto con estilos CSS </p>
```

- Entre las etiquetas en la parte superior de la página web se define código CSS a aplicar

```
<style type = "text/css">
  .miEstilo {
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-weight: bold;
    color: # FF0000;
  }
</style>
```

# Hojas de Estilo en Cascada (IV)

- Se puede incrustar código CSS directamente en la propia página cuando:
  - Los proyectos sean pequeños
  - Los estilos definidos se utilizan en una sola página
- Si el estilo definido va a afectar a muchas páginas, es ineficiente tener que copiar y pegar el código CSS en cada página: cambio de formato del estilo y transmisión por internet
- Mejor definir estilos CSS en un archivo separado y luego vincularlo a la página

```
<link rel="stylesheet" type="text/css" href="HojaDeEstiloGnal.css">
```

- La línea de código anterior vincula la hoja de estilos externa llamada 'HojaDeEstiloGnal.css' al documento HTML
- Colocar el código entre las etiquetas <head> </head> de cada página web a la que afecten los estilos especificados

# CSS en JavaFX (I)

- Los estilos por defecto de JavaFX 8 se encuentran en un archivo denominado **modena.css**
- Se encuentra dentro del archivo **/jdk1.8.x/jre/lib/ext/jfxrt.jar**
- El archivo **modena.css** aparece en la ruta **com/sun/javafx/scene/control/skin/modena/**
- Este estilo se aplica siempre a una aplicación JavaFX
- Añadiendo un estilo personal se puede reescribir los estilos por defecto definidos en *modena.css*
- *Tarea propuesta: Se puede abrir el archivo **jfxrt.jar** como si fuera un fichero zip. Extraer y consultar el archivo **modena.css** para ver qué estilos aparecen por defecto*

# Usando un fichero CSS externo

- Dos opciones:

Definir el estilo predeterminado o “tema” de la aplicación (user agent stylesheet), el cual es usado por defecto para renderizar cualquier nodo:

```
Application.setUserAgentStylesheet(  
    getClass().getResource("theme.css").toExternalForm());
```

- Añadiendo hojas de estilos a una escena:

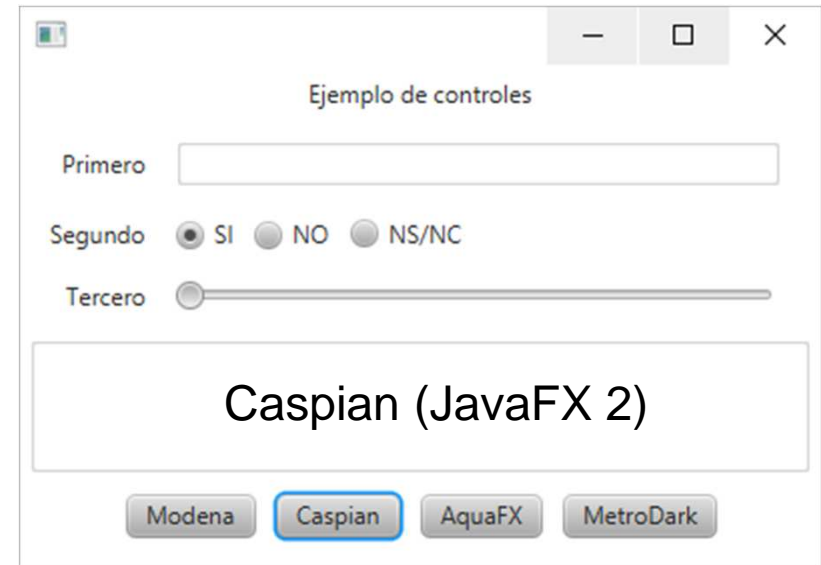
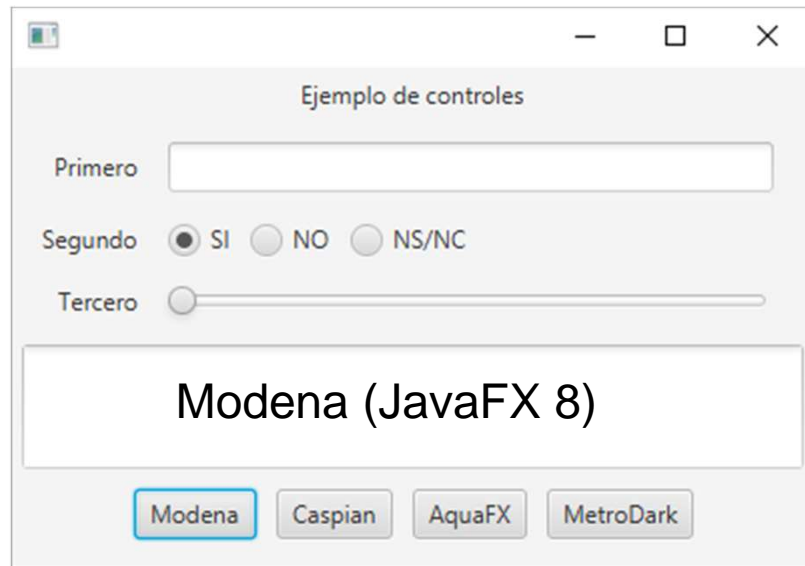
```
scene.getStylesheets().add(  
    getClass().getResource("skin.css").toExternalForm());
```

- Nota: También podemos reemplazar el estilo predeterminado de una escena mediante

```
scene.setUserAgentStylesheet(  
    getClass().getResource("theme.css").toExternalForm());
```



# Cambiando el tema...



# Cambiando el tema...

- Temas incluidos en JavaFX
  - Modena (JFX 8)
    - `Application.setUserAgentStylesheet(Application.STYLESHEET_MODENA);`
    - `Application.setUserAgentStylesheet(null);` // en JavaFX 8
  - Caspian (JFX 2)
    - `Application.setUserAgentStylesheet(Application.STYLESHEET_CASPIAN);`

# Estilos CSS

- Los estilos en un fichero CSS se especifican en bloques con la siguiente sintaxis:

```
<selector> {  
    <propiedad>: <valor>; // reglas  
    <propiedad>: <valor>;  
    ...  
}
```

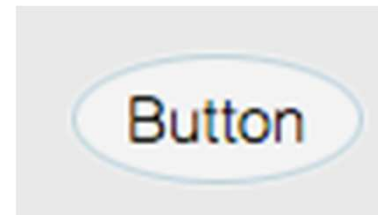
# Selectores

- Selectores
  - Sirven para seleccionar uno o varios nodos del grafo de escena de JavaFX para aplicarles un estilo
  - Tres tipos de identificadores que se pueden usar en los selectores:
    - `class`: una clase de estilo que se puede asignar a varios nodos. Un nodo también puede tener varias clases de estilo
    - `type selector`: coincide con el nombre sencillo de la clase (`Label`, `HBox`, `CheckBox`, etc.)
    - `id`: identificador de un nodo (no confundir con `fx:id`)

# Selección de nodos por clase

- Se puede añadir clases a un nodo con el método:  
`getStyleClass().add("num-button")`
- Para darle un estilo común a varios componentes, añadirles la misma clase
- Y luego, se puede introducir un bloque en el fichero CSS para establecer el diseño del nodo:

```
.num-button {  
-fx-background-color: white, rgb(189,218,230), white;  
-fx-background-radius: 50%;  
-fx-background-insets: 0, 1, 2;  
-fx-font-family: "Helvetica";  
-fx-text-fill: black;  
-fx-font-size: 20px;  
}
```



# Clases predefinidas

- Los controles de JavaFX definen sus propias clases:
  - button, check-box, combo-box, label, list-view...
  - Ver: <http://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html>
  - ¡Cuidado! Los contenedores no definen clase de estilo:

## HBox

*Style class: empty by default*

CSS Property	
-fx-spacing	<size>
-fx-alignment	[ top-left   top-center   top-right   bottom-left   bottom-center   bottom-right   baseline   center ]
-fx-max-width	<size>

- sin embargo definen un tipo de selector que coincide con su nombre de clase (HBox, VBox, GridPane...)

# Selección de nodos por id

- Se puede establecer el id de un nodo con el método:  
`void setId(String value)`

@FXML

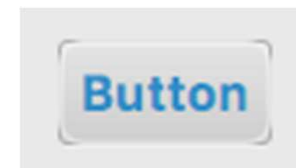
Button modena;

...

`modena.setId("boton-modena");`

- Y luego, se puede introducir un bloque en el fichero CSS para establecer el diseño del nodo:

```
#boton-modena {  
  -fx-text-fill: rgba(17, 145, 213);  
  -fx-border-color: rgba(255, 255, 255, .80);  
  -fx-border-radius: 8;  
  -fx-padding: 6 6 6 6;  
  -fx-font: bold italic 12pt "LucidaBrightDemiBold";  
}
```



# Otros métodos de selección

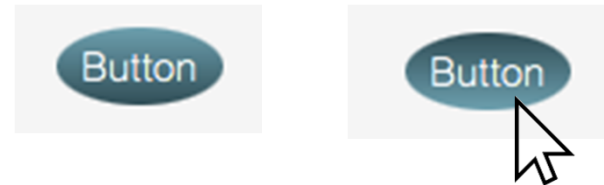
- Clase descendiente (por ejemplo, todas las etiquetas dentro de un checkbox):  
`.check-box .label { -fx-text-fill: black; }`
- Aplicar el estilo a todos los botones hijos directos de un HBox  
`HBox > .button { -fx-text-fill: black; }`
- Aplicar un mismo estilo a varias clases  
`.label, .text { -fx-font-size: 20px; }`



# Pseudoclasses

- Se usan para establecer el estilo de nodos que tienen distintos estados
  - Por ejemplo, un botón tiene los estados: hover, selected, focused, etc.

```
.num-button {  
-fx-background-color: linear-gradient(#61a2b1, #2A5058);  
-fx-background-radius: 50%;  
-fx-background-insets: 0, 1, 2;  
-fx-font-family: "Helvetica";  
-fx-text-fill: white;  
}  
  
.num-button:hover {  
-fx-background-color: linear-gradient(#2A5058, #61a2b1);  
}
```



# Estableciendo el estilo de un nodo por código

- El método `setStyle` permite establecer estilos por código:

```
@FXML
```

```
Button win8;
```

```
win8.setOnMouseEntered(ae -> win8.setStyle("-fx-font-size: 15px"));
```

```
win8.setOnMouseExited(ae -> win8.setStyle(""));
```

# Herencia de estilos

- La clase `.root` representa la raíz de todas las clases
  - las propiedades definidas en `.root` se aplicarán a todos los elementos de la escena, salvo a los que redefinan dicha propiedad

```
.root {  
    -fx-font-size: 12px;  
}  
.button {  
    -fx-font-size: 20px;  
}
```

# Propiedades

- ¿Qué propiedades puedo cambiar en cada nodo?
  - Busca en la CSS Reference Guide
    - <http://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html>

## Button

Style class: *button*

The Button control has all the properties of [ButtonBase](#)

### Pseudo-classes

CSS Pseudo-class	Comments
<a href="#">cancel</a>	applies if this Button receives VK_ESC if the event is not otherwise consumed
<a href="#">default</a>	applies if this Button receives VK_ENTER if the event is not otherwise consumed

Also has all pseudo-classes of [ButtonBase](#)

## ButtonBase

The ButtonBase control has all the properties of [Labeled](#)

### Pseudo-classes

CSS Pseudo-class	Comments
<a href="#">armed</a>	applies when the <code>armed</code> variable is true

Also has all pseudo-classes of [Labeled](#)

## Labeled

CSS Property	Values	Default	Comments
<a href="#">-fx-alignment</a>	[ top-left   top-center   top-right   center-left   center   center-right bottom-left   bottom-center   bottom-right   baseline-left   baseline-center   baseline-right ]	center-left	
<a href="#">-fx-text-alignment</a>	[ left   center   right   justify ]	left	text-align from CSS spec maps to textAlignment in JavaFX
<a href="#">-fx-text-overflow</a>	[ center-ellipsis   center-word-ellipsis   clip   ellipsis   leading-ellipsis   leading-word-ellipsis   word-ellipsis ]	ellipsis	
<a href="#">-fx-wrap-text</a>	<boolean>	false	
<a href="#">-fx-font</a>	<font>	platform dependent	inherits The initial value is that of Font.getDefault()
<a href="#">-fx-underline</a>	<boolean>	false	
<a href="#">-fx-graphic</a>	<uri>	null	
<a href="#">-fx-content-display</a>	[ top   right   bottom   left   center   right   graphic-only   text-only ]	left	
<a href="#">-fx-graphic-text-gap</a>	<size>	4	
<a href="#">-fx-label-padding</a>	<size>   <size> <size> <size> <size>	[0,0,0,0]	
<a href="#">-fx-text-fill</a>	<paint>	black	
<a href="#">-fx-ellipsis-string</a>	<string>	...	

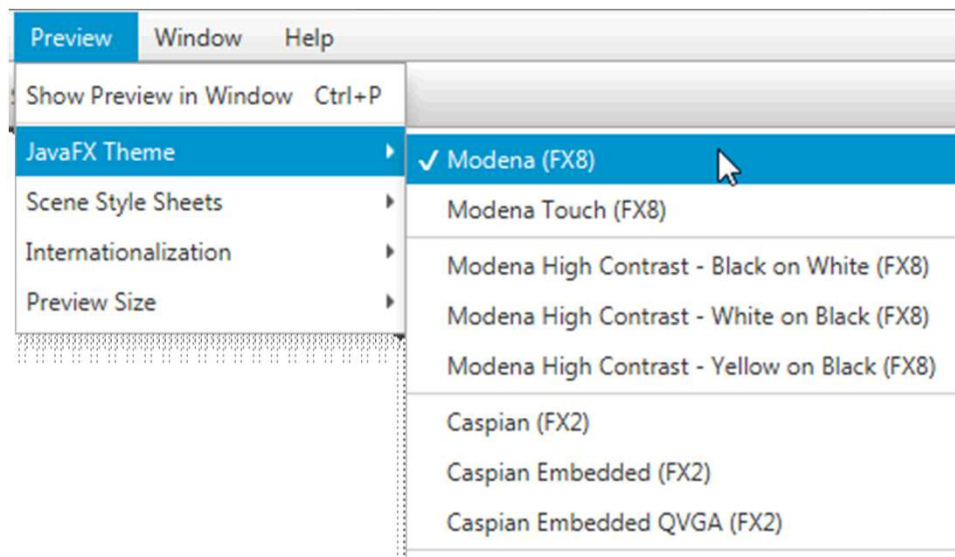
Also has properties of [Control](#)

# CSS en SceneBuilder (I)

- Scene Builder permite a cualquier nodo del grafo seleccionar el estilo predefinido que le afecte
- Obliga a determinar la hoja de estilos CSS en la que reside dicho estilo y a especificar explícitamente el nombre del estilo seleccionado
- Por defecto, cada proyecto JavaFX 8 emplea una hoja de estilo que se encuentran en el archivo **modena.css**
- Scene Builder emplea este estilo predefinido cada vez que se arrastra un control desde el panel izquierdo *Library* hasta los paneles *Content* o *Hierarchy*

# CSS en SceneBuilder (II)

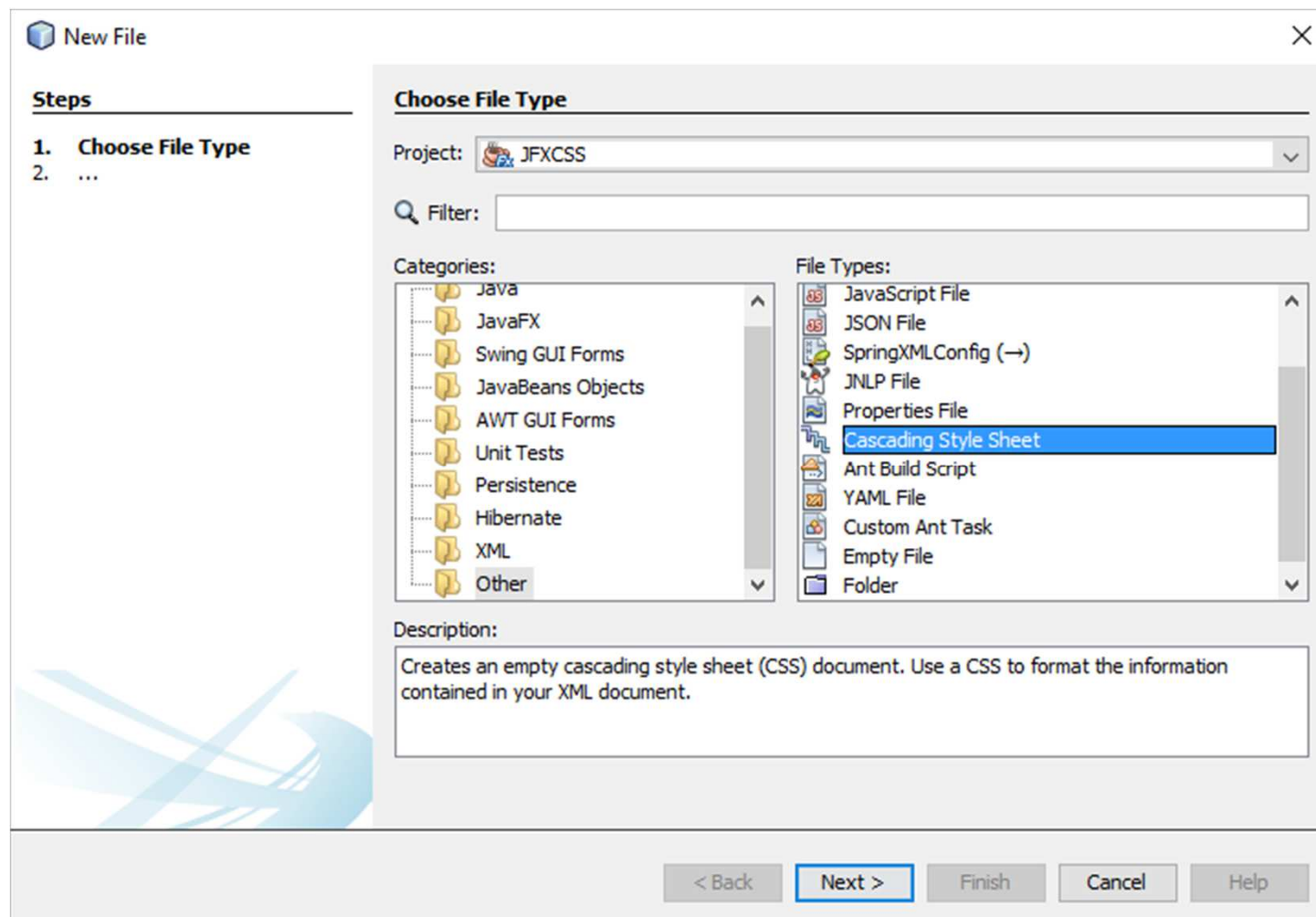
- Se puede cambiar el tema utilizado seleccionando la opción *Preview* de la barra de menú y seleccionar uno de los temas JavaFX
- En la lista desplegable se puede seleccionar un tema específico basado en *Módena*
- Se puede seleccionar el tema basado en *Caspian*



## CSS en SceneBuilder (III)

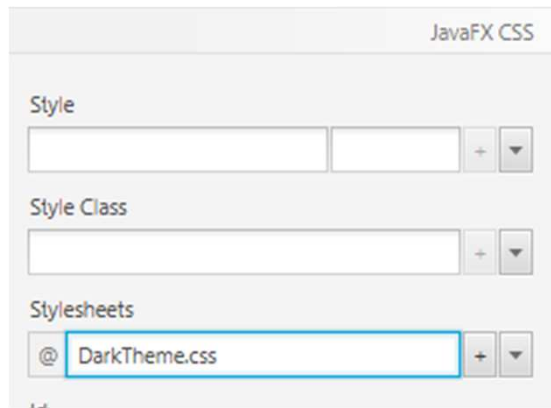
- Se puede agregar las reglas CSS a toda la escena, dentro de un contenedor determinado o a cualquier nodo de la escena
- Se puede personalizar el estilo cambiando las propiedades de un componente dado a través de la sección *Properties* del panel *Inspector* o definiendo reglas de estilo en un archivo CSS propio de la aplicación
- Scene Builder no genera ficheros CSS. Hay que crear el fichero y rellenarlo manualmente.
- Scene Builder actualizará la vista tan pronto se realice una modificación del fichero CSS

# CSS en SceneBuilder (IV)





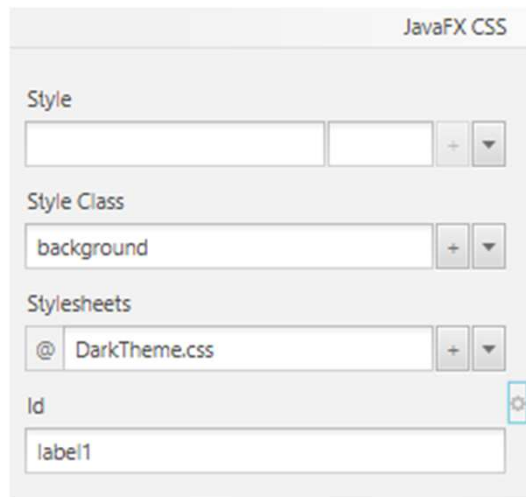
# CSS en SceneBuilder (V)



Abrir un archivo *fxml* en el Scene Builder

Seleccionar el nodo raíz en la sección *Hierarchy*

En la vista *Properties* añadir el archivo *DarkTheme.css* como hoja de estilo (campo denominado *Stylesheets* dentro de la sección *JavaFX CSS*)



Añadir clases o establecer el identificador a cualquier nodo del grafo incluso aunque no existan en los ficheros CSS cargados

Los estilos CSS que se definen en un elemento padre afectan a la forma en la que se muestra dicho objeto y también a todos sus elementos hijos

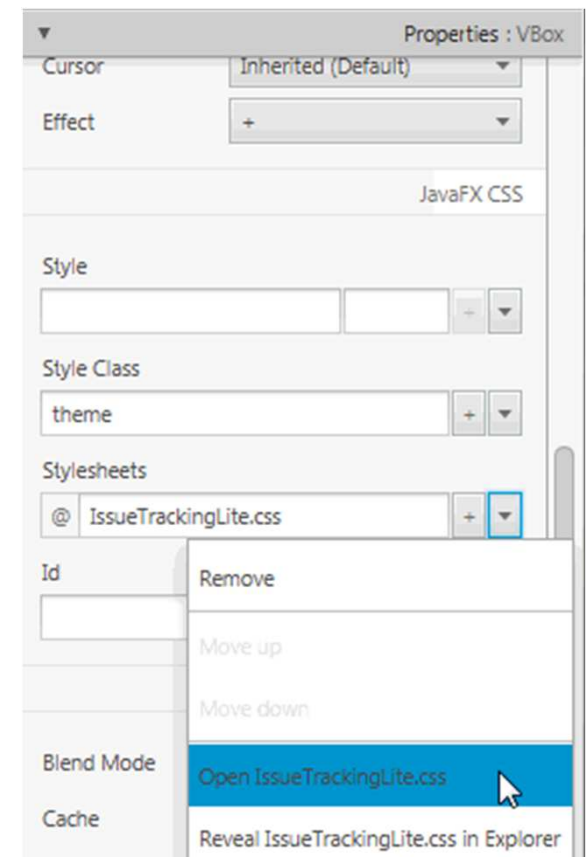
# CSS en SceneBuilder (VI)

Se puede editar un archivo CSS con el bloc de notas:

1. En la sección *Properties* del panel *Inspector*, hacer clic en la flecha desplegable en la parte derecha de la hoja de estilo
2. Seleccionar el comando *Abrir archivo.CSS* a editar

*Revelar la ubicación del archivo CSS* abre una ventana del navegador de ficheros situado en la carpeta donde aparece el fichero CSS seleccionado

También se puede navegar hasta el archivo CSS a través del panel analizador de CSS

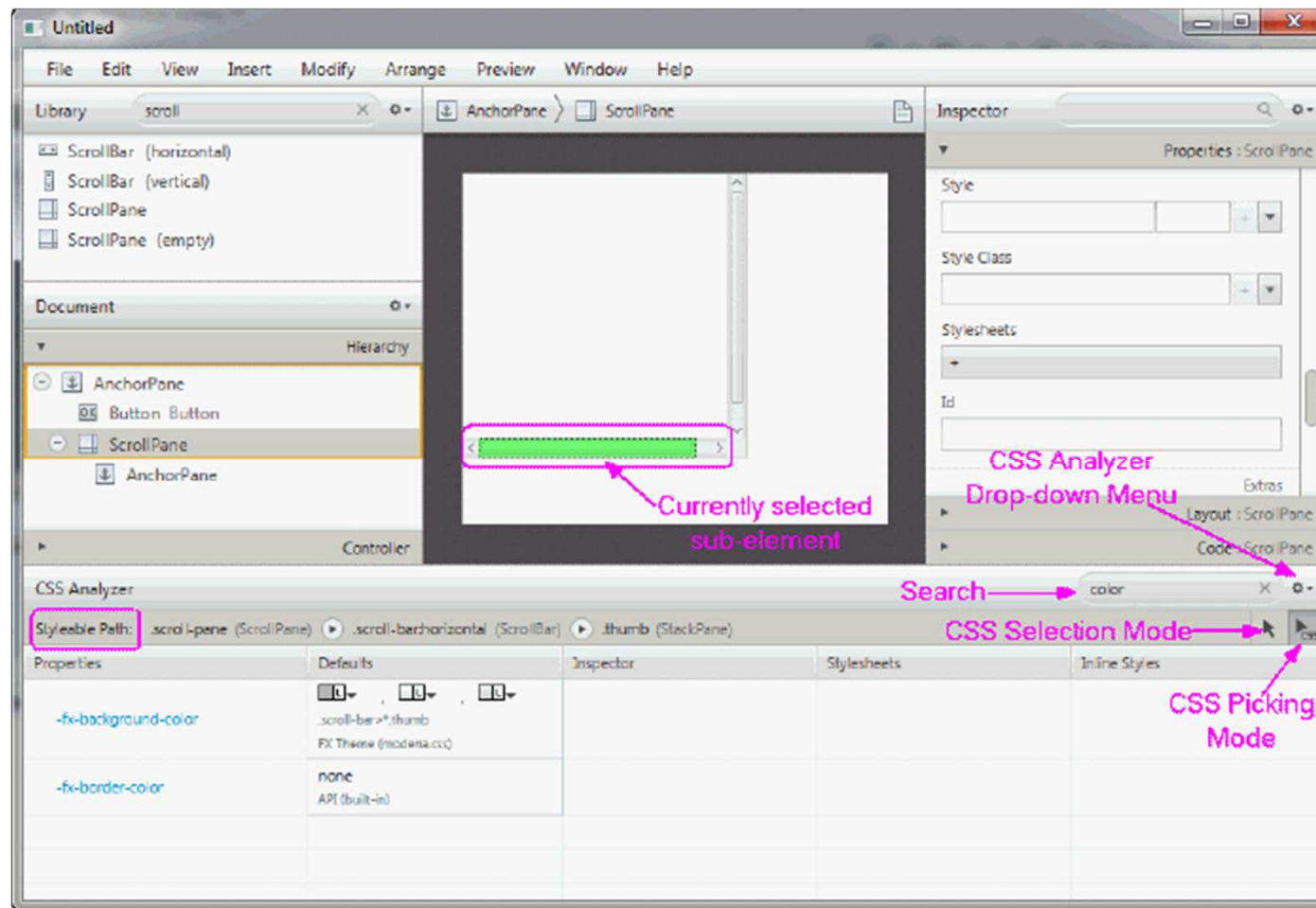


# Panel analizador de CSS (I)

- Permite comprender cómo diversas posibles reglas CSS pueden afectar visualmente a un elemento de interfaz gráfica de usuario
- Presenta una visión de todas las posibles fuentes de los valores de las propiedades
- Cada aspecto de un elemento de la interfaz gráfica puede provenir de cualquiera de las reglas CSS predefinidas
- Las fuentes se enumeran en orden de prioridad, lo que le permite entender por qué una determinada fuente tiene prioridad sobre otra
- Permite navegar a la fuente de valor de la propiedad CSS para solucionar problemas de hojas de estilo
- Para mostrar el panel, seleccionar la opción *View* en el menú principal y luego *Mostrar Analizador CSS*

View	Insert	Modify	Arrange	Pr
Content				Ctrl+0
Properties				Ctrl+1
Layout				Ctrl+2
Code				Ctrl+3
Hide Library				Ctrl+4
Hide Document				Ctrl+5
Show CSS Analyzer				Ctrl+6
Hide Left Panel				Ctrl+7
Hide Right Panel				Ctrl+8
Show Outlines				Ctrl+E
Show Sample Data				
Disable Alignment Guides				
Zoom				
Show Sample Controller Skeleton				

# Panel analizador de CSS (II)



# Panel analizador de CSS (III)

Filtrado de propiedades  
por nombre

The screenshot shows the CSS Analyzer window. At the top, there's a search bar with 'font' entered. Below it, a table displays CSS properties for the selector '.button (Button)'. The table has five columns: Properties, Defaults, Inspector, Stylesheets, and Inline Styles. The 'Defaults' column is highlighted in light blue. Below the table, there are four buttons: 'View As', 'Copy Styleable Path', 'Hide Properties with Default Values', and 'Split Defaults'. Arrows point from the text labels to these buttons and the search bar.

Properties	Defaults	Inspector	Stylesheets	Inline Styles
	API (built-in)			
-fx-font-family	System API (built-in)			
-fx-font-size	12px API (built-in)			
-fx-font-style	Regular API (built-in)			
-fx-font-weight	Font{name=System Regular, family=System, style... API (built-in)			

Menú  
desplegable

Presenta las reglas  
de estilo en diferentes  
formatos predefinidos:  
**Tables, Rules y Text**

View As

Copy Styleable Path

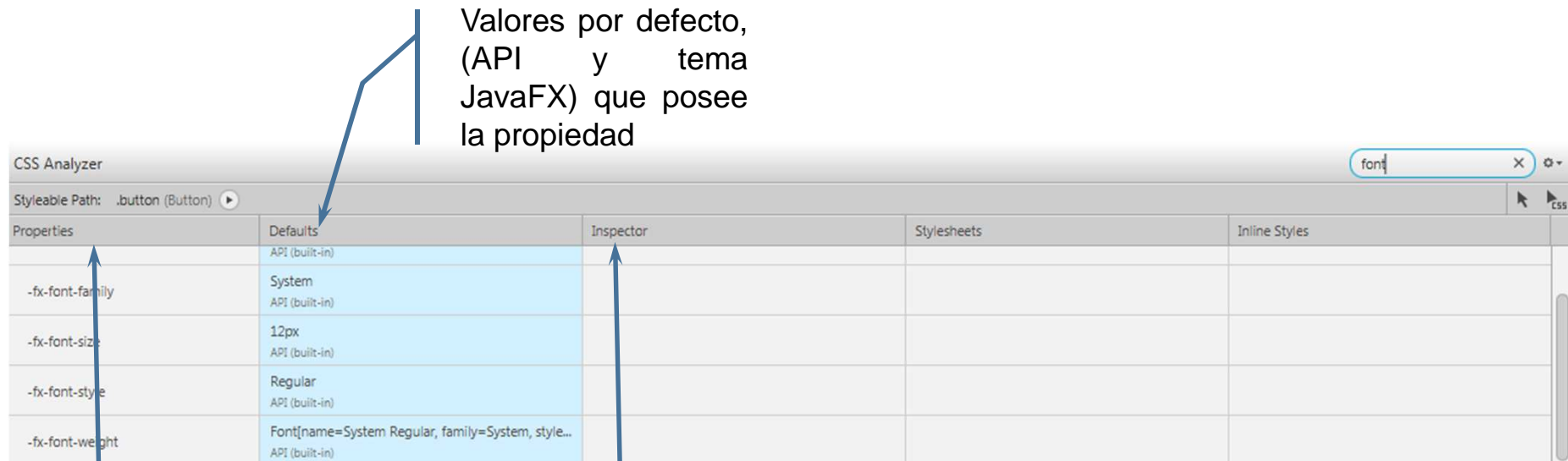
Hide Properties with Default Values

Split Defaults

Separa en dos columnas los  
valores predeterminados en  
*API* y temas *FX*  
*Join Defaults* vuelve a  
agrupar las columnas

Permite limpiar los  
contenidos por defecto y  
centrarse en sólo en los  
modificados

# Panel analizador de CSS (IV)



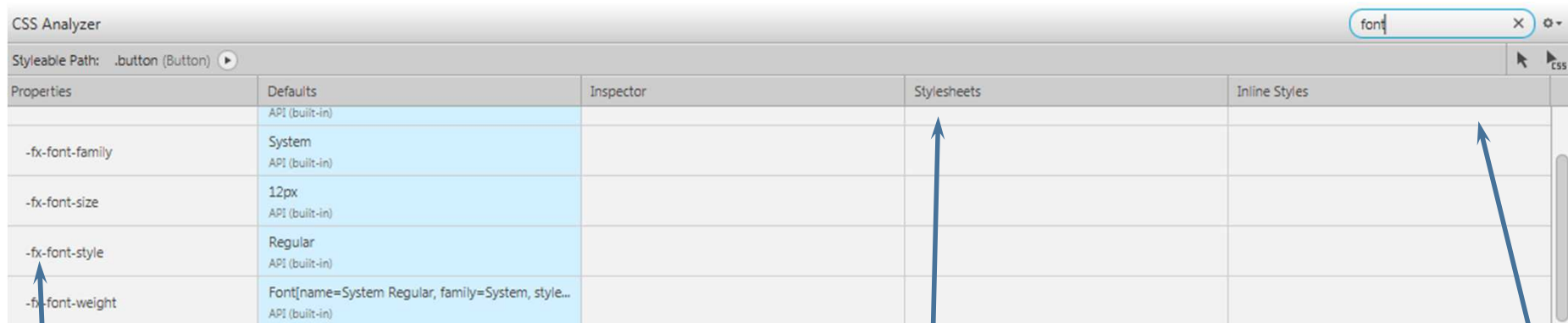
Propiedades de estilo disponibles para el elemento seleccionado en ese momento.

Valor de la propiedad establecido mediante el panel *Inspector*.



Resalta en azul la propiedad en el panel *Inspector*

# Panel analizador de CSS (IV)

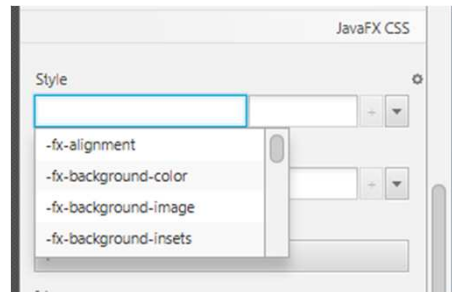


Propiedad establecida por un *stylesheet* del nodo o de algún ascendiente

Pulsar sobre el nombre de la propiedad lleva a la documentación en línea de la JavaFX CSS Reference Guide

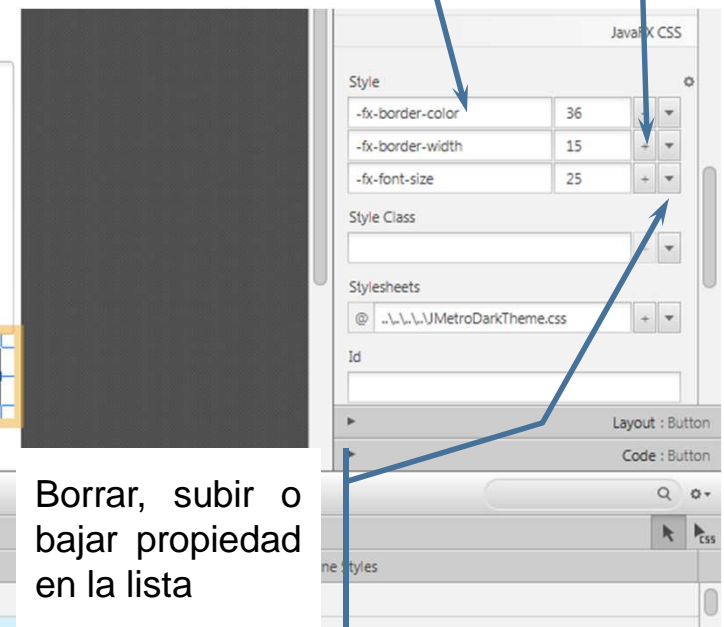
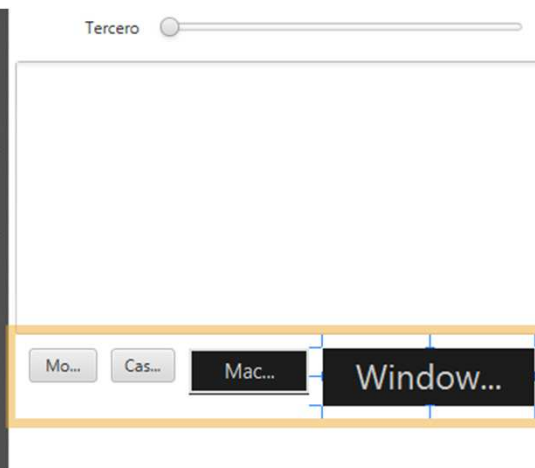
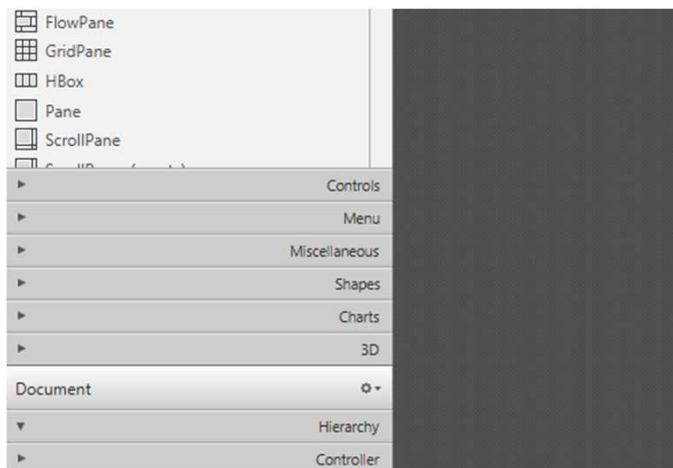
Propiedad establecida *en línea*, es decir como *Style* dentro de la sección *JavaFX CSS* del apartado *Properties* del *Inspector*

# Panel analizador de CSS (V)



Al clicar campo de texto, aparece lista desplegable con propiedades modificables CSS

Botón + añade nuevas propiedades a modificar



Borrar, subir o bajar propiedad en la lista

Properties	Defaults	Inspector	Stylesheets
-fx-font-family	System API (built-in)	Button.font	"Segoe UI" .button JMetroDarkTheme.css
-fx-font-size	12px API (built-in)		11.0pt .button JMetroDarkTheme.css
-fx-font-style	Regular		25px style win8



# Panel analizador de CSS (V)

The screenshot shows the CSS Analyzer tool interface. At the top, there's a visual representation of a UI element (a button labeled 'Window...') with a yellow selection box. Below this is the 'CSS Analyzer' panel, which contains a table of CSS properties for the selected element.

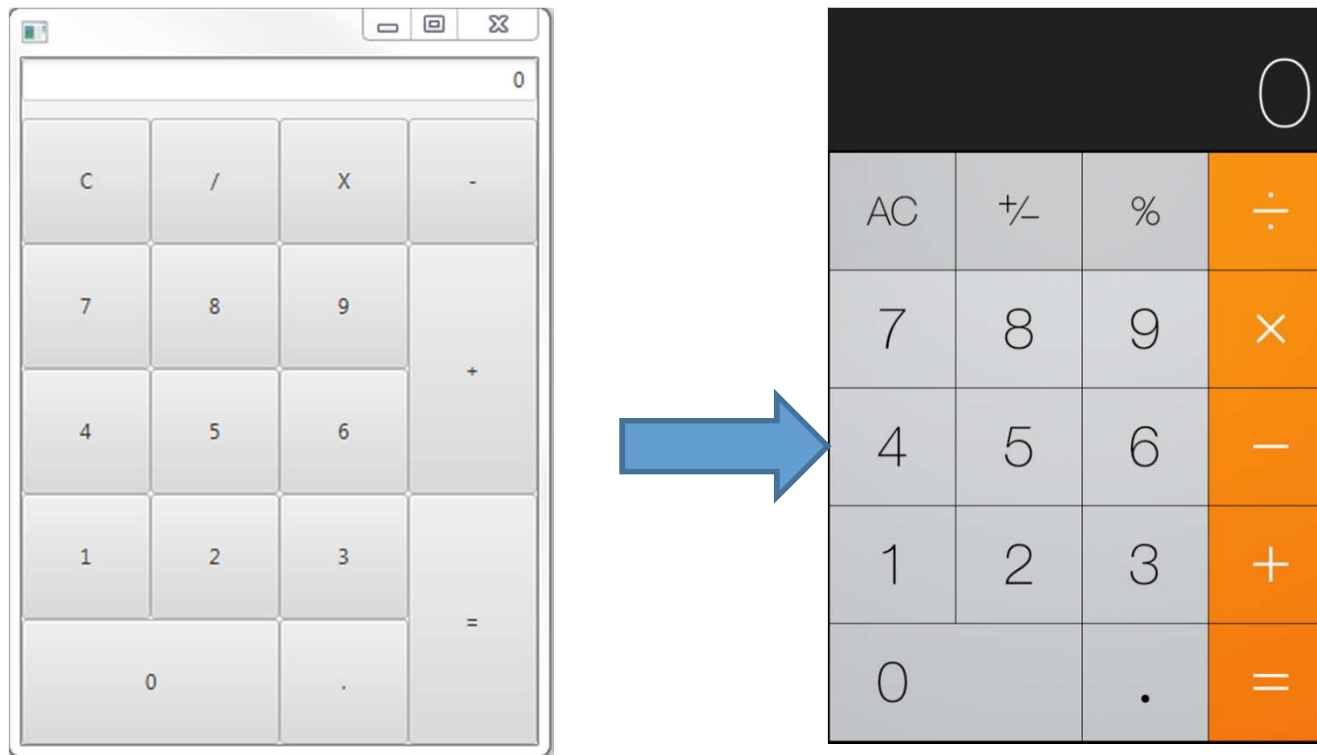
Properties	Defaults	Inspector	Stylesheets	Inline Styles
-fx-font-family	System API (built-in)	Button.font	"Segoe UI" .button JMetroDarkTheme.css	
-fx-font-size	12px API (built-in)		11.0pt .button JMetroDarkTheme.css	25px style win8
-fx-font-style	Regular			

Si no se indica nada explícitamente, se asume el estilo por defecto

Después lo definido por el CSS asignado

Lo definido explícitamente tiene máxima prioridad de formato

# Actividad propuesta



- Modifica la interfaz creada en la práctica 2 para darle una apariencia similar a la calculadora de iOS usando CSS
- Haz que los botones cambien de color cuando están pulsados

# Bibliografía

- C. Dea y otros. JavaFX 8. Introduction by Example. Apress. 2014
  - Capítulo 6
- Oracle.
  - Skin Applications with CSS
    - <http://www.oracle.com/pls/topic/lookup?ctx=javase80&id=JFXUI733>
  - CSS Reference Guide
    - <http://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html>
  - JavaFX Scene Builder: User Guide
    - <http://docs.oracle.com/javase/8/scene-builder-2/user-guide/index.html>