PRÁCTICA 5. CSS

Interfaces Persona Computador

Depto. Sistemas Informáticos y Computación

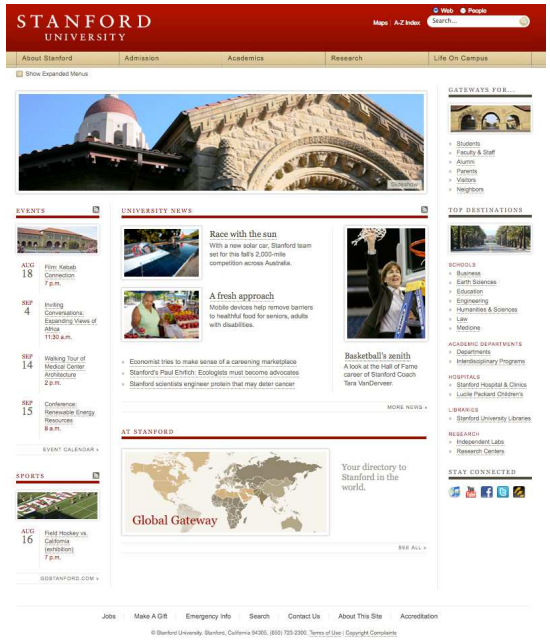
UPV

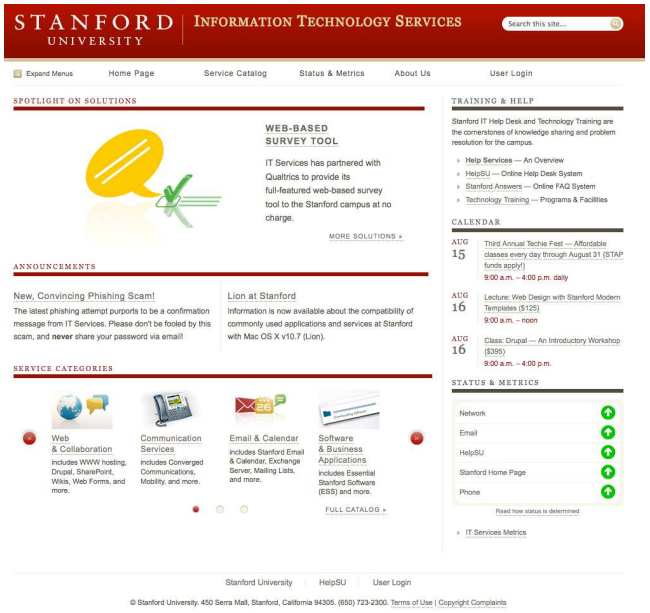
Índice

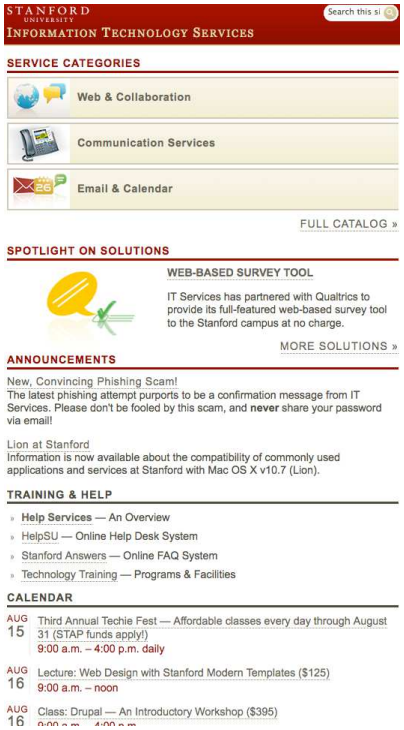
- Hojas de estilo en cascada
- CSS en JavaFX
- Formas de definir el estilo en JavaFX
- Usando un fichero CSS externo
- Propiedades en CSS
- Prioridades de los estilos
- CSS en Scene Builder

Hojas de estilo en cascada

- La hojas de estilo CSS nacen en el contexto de las aplicaciones Web.
- El contenido se define en HTML y la apariencia en CSS

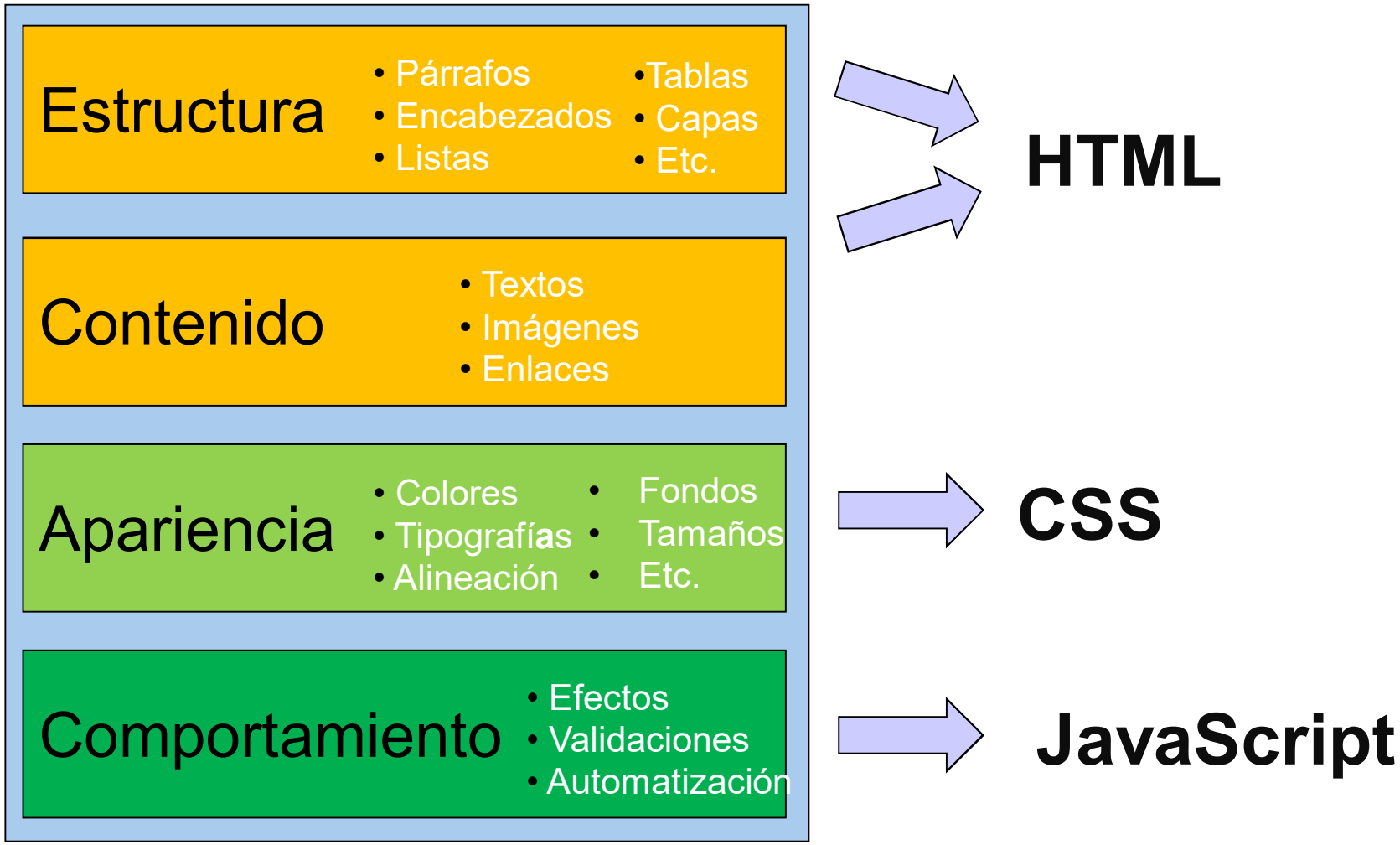






Hojas de estilo en cascada

- Estructura de una página Web



Hojas de estilo en cascada

- Los estilos CSS deben darse de alta en un fichero, no obstante:
  - Pueden declararse dentro de un HTML mediante la etiqueta <style>
  - Pueden aplicarse directamente sobre un elemento concreto en la propiedad "style".
- Lo correcto es llevar los estilos a un o unos ficheros css, pero se permite añadir pequeños retoques directamente sobre el HTML.

CSS en JavaFX

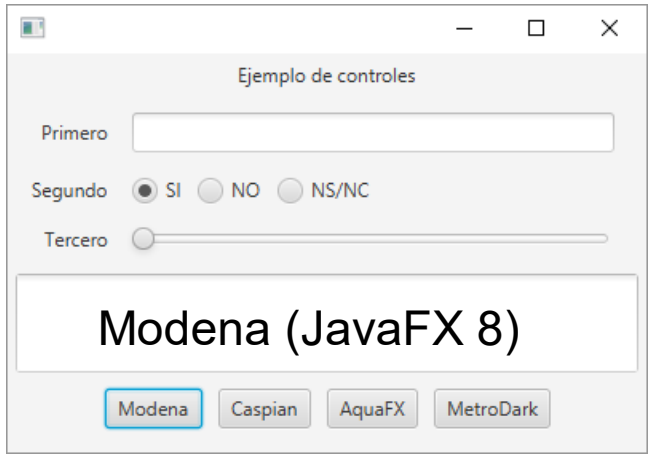
- Mediante CSS en Java se puede dar estilo a un control, a una escena, o a una aplicación entera.
- Está basado en el estándar Web W3C CSS version 2.1.
- Algunas diferencias menores pueden consultarse en:
- http://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html

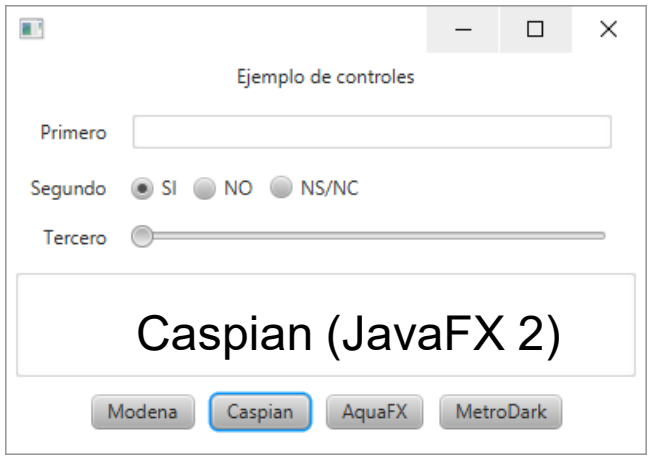
Definir el estilo en JavaFX

- Al igual que en una aplicación html existen tres formas para definir el estilo:
- Estilo por defecto JavaFX contiene dos estilos: Modena y Caspian.
- Para cambiarlo usar en el método `start()`

```
Application.setUserAgentStylesheet(STYLESHEET_CASPIAN);
```

- Si no se indica nada se usa Modena.css



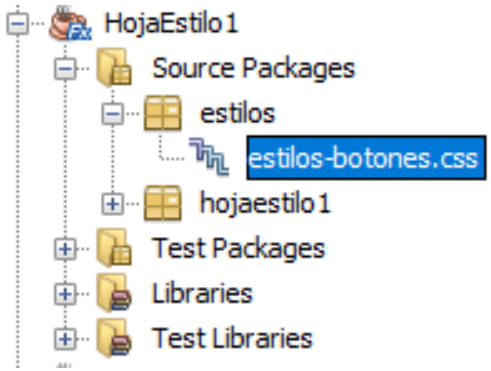


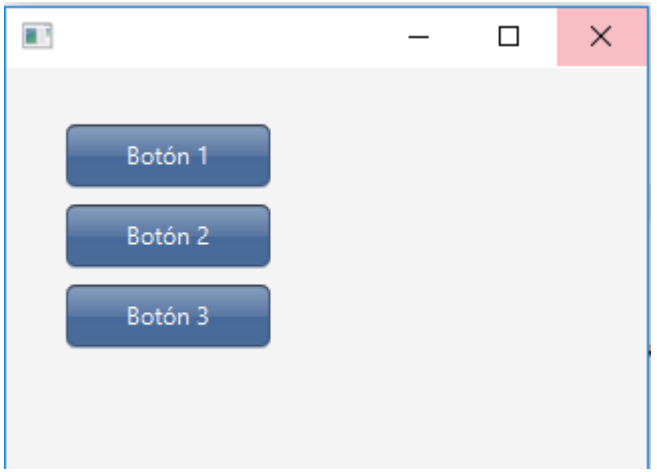
Métodos para definir CSS en JavaFX

- Se puede aplicar a una escena una hoja de estilo, definida en un archivo externo

```
Scene scene = new Scene(root);
String css =  this.getClass().getResource("/estilos/estilos-botones.css") .toExternalForm();
scene.getStylesheets().add(css);
```

Archivo externo, explicado después





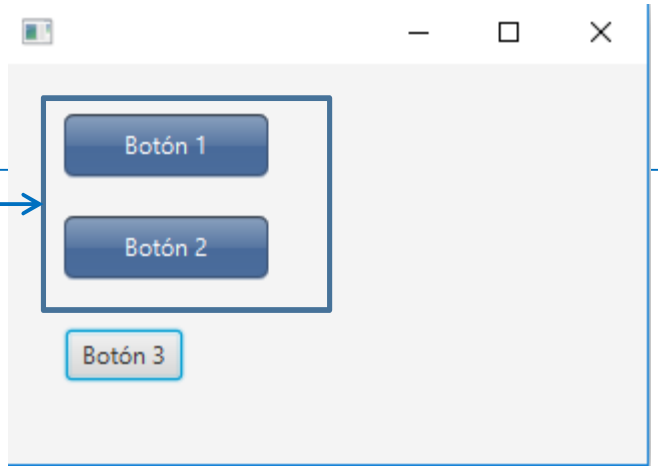
Métodos para definir CSS en JavaFX

- Hoja de estilo específica para un control descendiente de Parent, VBox en el ejemplo

```
public class FXMLDocumentController implements Initializable {
@FXML
private VBox vBox;

@Override
 public void initialize(URL url, ResourceBundle rb) {
 // TODO
 String css = this.getClass().getResource("/estilos/estilos-botones.css")
                   .toExternalForm();
 vBox.getStylesheets().add(css);
 }
}
```

Se aplica a los controles dentro de VBox

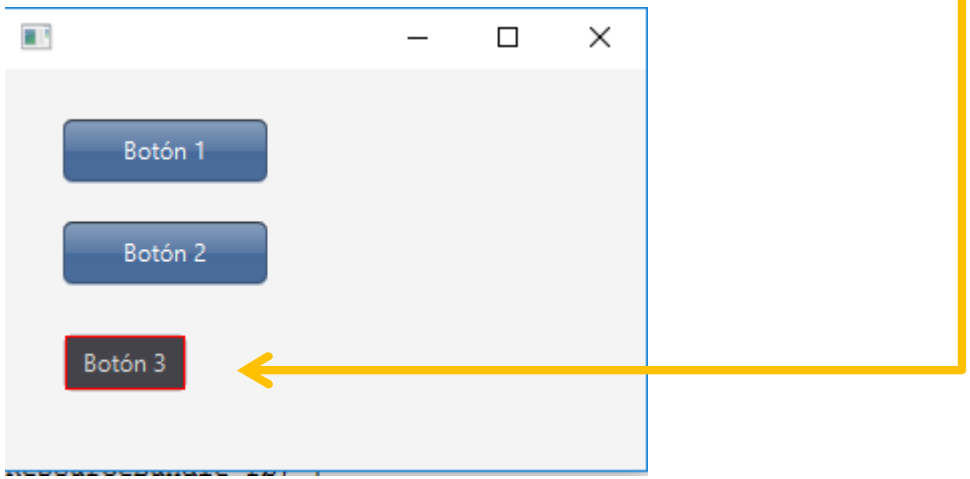


No asignamos estilo a la escena

Métodos para definir CSS en JavaFX

- Estilo para un componente mediante código.

```
String styles =
        "-fx-background-color: #444349;" +
        "-fx-border-color: #ff0000;"  +
        "-fx-text-fill: #dddddd;";
button3.setStyle(styles);
```



Resumen CSS externo en JavaFX

- 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());
```

Estilos CSS

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

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

- El selector indica a qué elemento se aplicará el estilo
- Cada regla contiene una propiedad y un valor

Estilos CSS

- Ejemplo de fichero

Definición de estilo

```
.button { /* Set style for all buttons */
-fx-text-fill: #dddddd; /* Sets font colour (hex) */
-fx-background-color: #444349; /* Sets background colour */
-fx-padding: 5px; /* Inner spacing on all sides */
}

#ok { /* Set style for just OK button */
-fx-font-weight: bold; /* Set font to bold */
}
```

Selector con id ok, las reglas se aplicarían al botón con ese id

Selector button, las reglas se aplicarían a todos los botones

Estilos CSS: selectores de tipo

- Los selectores en CSS pueden tomar alguna de las siguientes formas:
  - Selectores de tipo: el nombre se corresponde con los nodos de una interfaz de JavaFX. Empiezan con un punto seguido del nombre de la clase de estilo

JavaFX Class	*CSS Style Class*
Button	button
CheckBox	check-box
ChoiceBox	choice-box
ComboBox	combo-box
Label	label
ListCell	list-cell
ListView	list-view
Menu	menu
MenuBar	menu-bar
MenuButton	menu-button
MenuItem	menu-item
RadioButton	radio-button
Separator	separator
TableView	table-view
TextField	text-field
ToggleButton	toggle-button
Tooltip	tooltip
TreeCell	tree-cell
TreeView	tree-view

Botón con apariencia iphone

```
.button {
    -fx-background-color:
        #a6b5c9,
        linear-gradient(#303842 0%, #3e5577 20%, #375074 100%),
        linear-gradient(#768aa5 0%, #849cbb 5%, #5877a2 50%, #486a9a 51%, #4a6c9b 100%);
    -fx-background-insets: 0 0 -1 0,0,1;
    -fx-background-radius: 5,5,4;
    -fx-padding: 7 30 7 30;
    -fx-text-fill: #242d35;
    -fx-font-family: "Helvetica";
    -fx-font-size: 12px;
    -fx-text-fill: white;
}
```

Estilos CSS: selectores de clase

- Selectores de clase: son nombres definidos por el usuario, también empiezan por .
- Una clase de estilo se puede asignar a varios nodos. Un nodo también puede tener varias clases de estilo.

```
.num-button {
-fx-background-color: white, gray,lightblue;
-fx-background-radius: 50%;
-fx-background-insets: 0, 1, 2;
-fx-font-family: "Helvetica";
-fx-text-fill: black;
}
```

- Para añadir la clase de estilo a un nodo usamos el método:

```
Button btn1 = new Button("Button");
btn1.getStyleClass().add("num-button"); //observe que no se pone el .
```

- Para darle un estilo común a varios componentes, añadirles la misma clase de estilo.

Estilos CSS: selectores por id

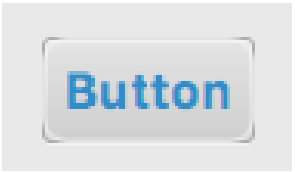
- Se puede establecer el id de un nodo en Scene Builder (no confundir id con fx id) o bien con el método

```
void setId(String value)

@FXML
Button modena;
…
modena.setId("boton-modena"); // no lleva #
```

- 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";
}
```



Estilos CSS: propiedades

- Todas las propiedades de estilo empiezan con –fx-
- Los nodos que contienen texto pueden utilizar los siguientes estilos:

Property	Value
`-fx-font-family`	The actual name of the font, or one of the following generic font types: `serif`, `sans-serif`, `cursive`, `fantasy`, or `monospace`.
`-fx-font-size`	A number followed by the unit of measure, which is usually `pt` (points) or `px` (pixels).
`-fx-font-style`	`normal`, `italic`, or `oblique`.
`-fx-font-weight`	`normal`, `bold`, `bolder`, `lighter`, `100`, `200`, `300`, `400`, `500`, `600`, `700`, `800`, or `900`.
`-fx-font`	A shorthand property that combines all other properties mentioned here into a single value that lists the style, weight, size, and family. Separate the values with spaces. If you want, you can omit the style and weight.

Estilos CSS propiedades

- Para un botón con texto.

```
.button
{
-fx-font-family: sans-serif;
-fx-font-size: 10pt;
-fx-font-style: normal;
-fx-font-weight: normal
}
```

- La forma abreviada:

```
.button
{
-fx-font: 10pt sans-serif;
}
```

Estilos CSS propiedades: color fondo

- La clase `Region` tiene una propiedad `-fx-background-color` que permite definir el color de fondo, la heredan las clases `Layout` y `Control`
- Para aplicar un color de fondo a un panel de Layout debemos dar al panel un id o una clase de estilo.
- Los colores de fondo pueden ser definidos de la siguiente forma:
- Nombre del color. JavaFX proporciona 148 nombres de colores

```
-fx-background-color: red
-fx-background-color: papayawhip
```

- Lista completa de colores:

http://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html#typecolor

Estilos CSS propiedades: color fondo

- Color RGB: El número del color se expresa en hexadecimal como rojo-verde-azul, con dos dígitos para - cada componente del color y todo con el prefijo #

```
-fx-background-color: #f5f5f5  // white smoke
// o de manera equivalente
-fx-background-color: rgb(245,245,245)
fx-background-color: rgba(245,245,245,0)
```

- Definido en un componente y referenciado después

```
.root {
my-color: black;
}
.my-style {
fx-background-color: my-color;
}
```

Estilos CSS propiedades: borde

- La clase Region define varias propiedades que permiten aplicar estilo a los bordes de un componente

Property	Value
`-fx-border-width`	A number followed by the unit of measure, usually expressed in pixels (`px`)
`-fx-border-style`	`none`, `solid`, `dotted`, or `dashed`
`-fx-border-color`	A color

```
.bordered
{
-fx-border-width: 4px;
-fx-border-color: black;
-fx-border-style: dashed;
}
```

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 center | bottom-right | basel

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

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; }
```

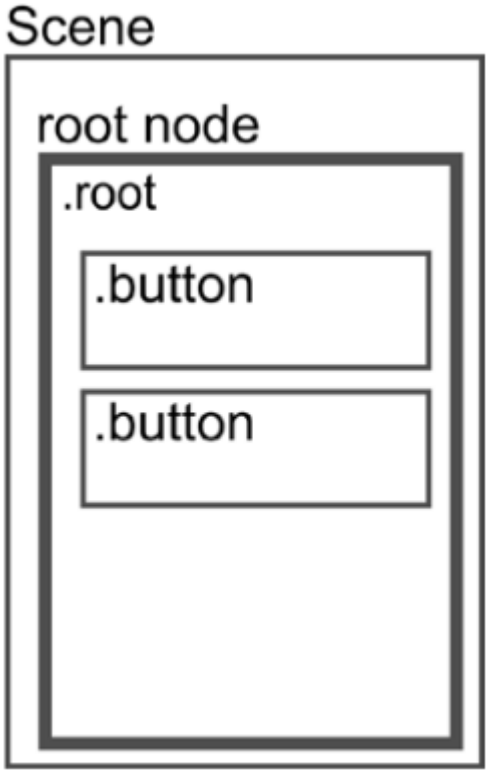
- Aplicar un mismo estilo a varias clases

```
.label, .text { -fx-font-size: 20px; }
```

Herencia de estilos

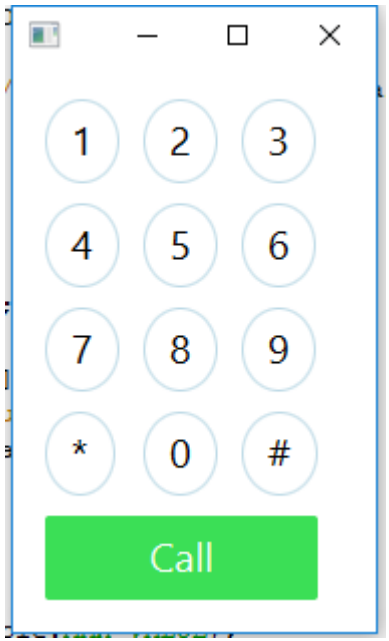
- La clase .root representa la raíz de todas las clases
  - las propiedades definidas en .root se aplicarán a todos los elementos de la escena, salvo a los que redefinan dicha propiedad

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



Pseudoclases

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



Ratón sobre el botón con el 3, hover

Pseudoclases

- En el archivo de estilo

```
.root {
-fx-background-color: white;
-fx-font-size: 20px;
bright-green: rgb(59,223, 86);
bluish-gray: rgb(189,218,230);
}
.num-pad {
-fx-padding: 15px, 15px, 15px, 15px;
-fx-hgap: 10px;
-fx-vgap: 8px;
}
.num-button {
-fx-background-color: white, bluish-gray, white;
-fx-background-radius: 50%;
-fx-background-insets: 0, 1, 2;
-fx-font-family: "Helvetica";
-fx-text-fill: black;
}
```

```
.num-button:hover {
-fx-background-color: black,
 white, black;
-fx-text-fill: white;
}
```

Estableciendo el estilo de un nodo por código

- El método `setStyle` como hemos visto permite establecer estilos por código:

```
@FXML
Button win8;

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

Prioridades de los estilos

- El estilo puede ser definido de varias formas o en varios sitios.

Código de la aplicación

```
Button yesBtn = new Button("Yes");
yesBtn.setStyle("-fx-font-size: 16px");
yesBtn.setFont(new Font(10));
Scene scene = new Scene(yesBtn);
scene.getStylesheets().addAll("resources/css/stylespriorities.css");
```

Estilo inline con setStyle

Estilo por código setFont

Estilo de la escena

Archivo stylepriorities.css

```
.button {
-fx-font-size: 24px;
-fx-font-weight: bold;
}
```

Prioridades de estilos

Inline style (prioridad más alta)
Hoja de estilo del padre (Parent style sheets)
Hoja de estilo de la escena (Scene style sheets)
Valores fijados en el código usando JavaFX API
User agent style sheets (prioridad más baja)

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
cancel	applies if this Button receives VK_ESC if the event is not otherwise consumed
default	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
armed	applies when the **armed** variable is true

Also has all pseudo-classes of Labeled

Labeled

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

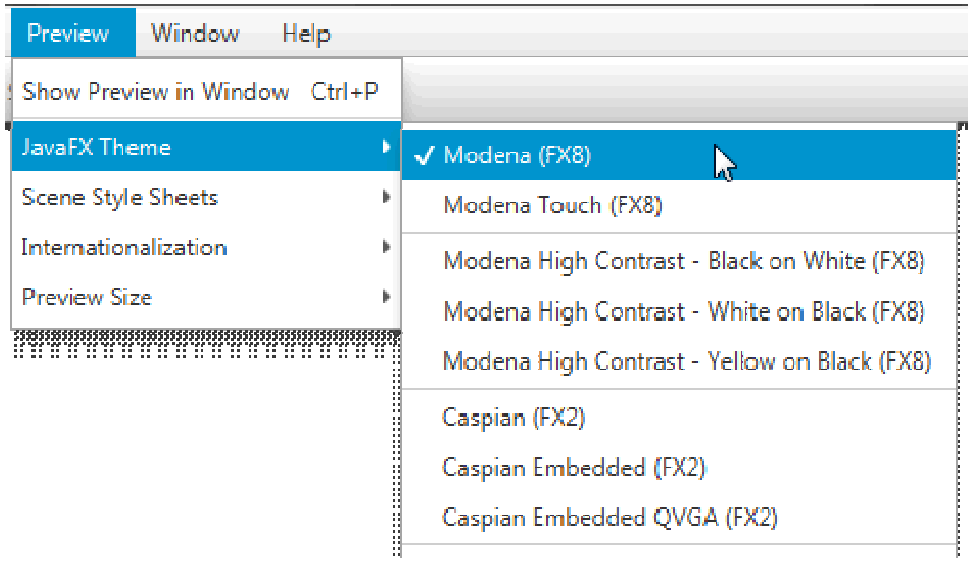
Also has properties of Control

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)

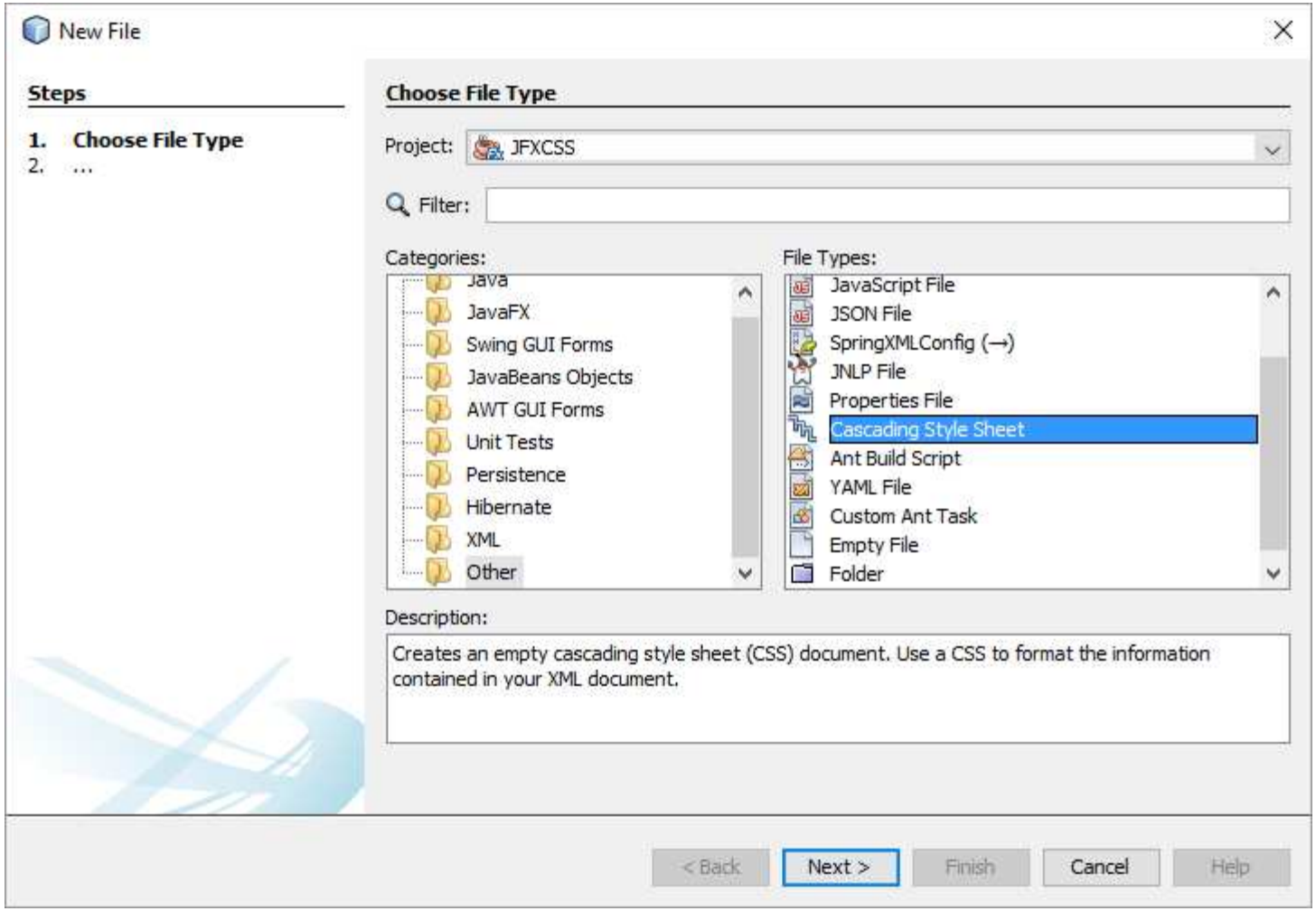
- Se puede cambiar el tema utilizado seleccionando la opción *Preview* de la barra de menú y seleccionar uno de los temas JavaFX
- En la lista desplegable se puede seleccionar un tema específico basado en *Módena*
- Se puede seleccionar el tema basado en *Caspian*



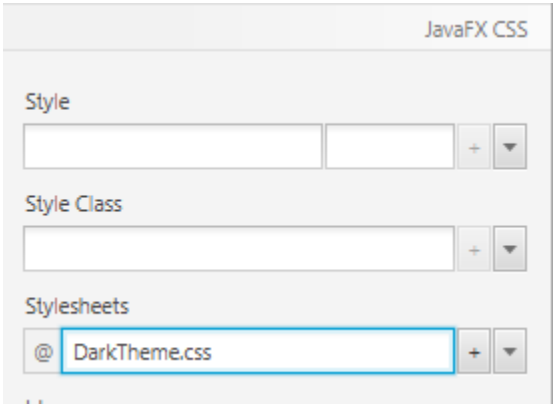
CSS en SceneBuilder (III)

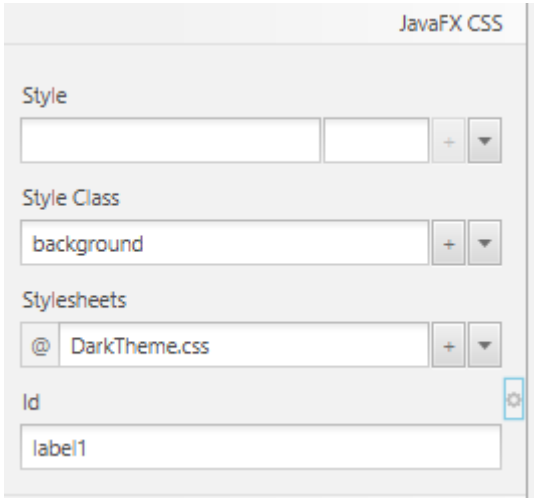
- Se puede agregar las reglas CSS a toda la escena, dentro de un contenedor determinado o a cualquier nodo de la escena
- Se puede personalizar el estilo cambiando las propiedades de un componente dado a través de la sección *Properties* del panel *Inspector* o definiendo reglas de estilo en un archivo 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*)

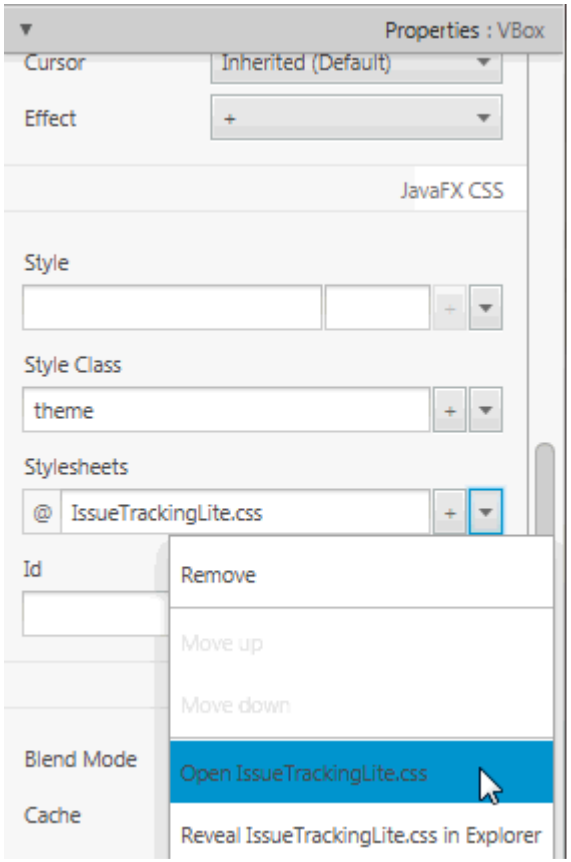
Añadir clases o establecer el identificador a cualquier nodo del grafo incluso aunque no existan en los ficheros CSS cargados

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

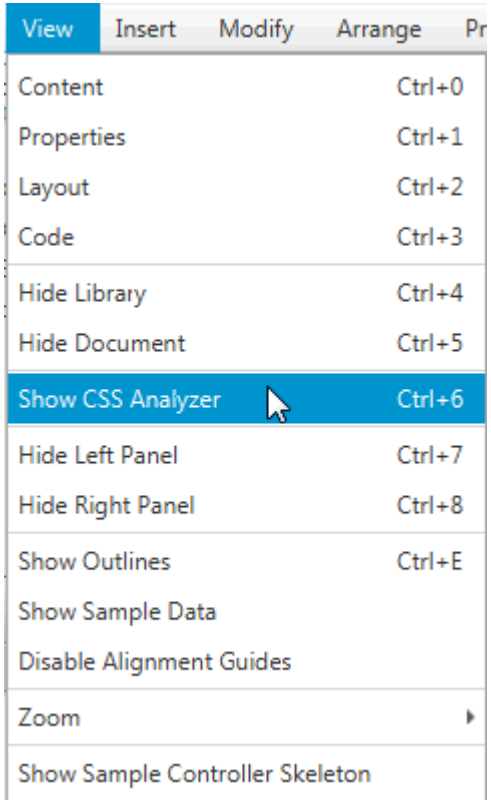


Revelar la ubicación del archivo CSS abre una ventana del navegador de ficheros situado en la carpeta donde aparece el fichero CSS seleccionado

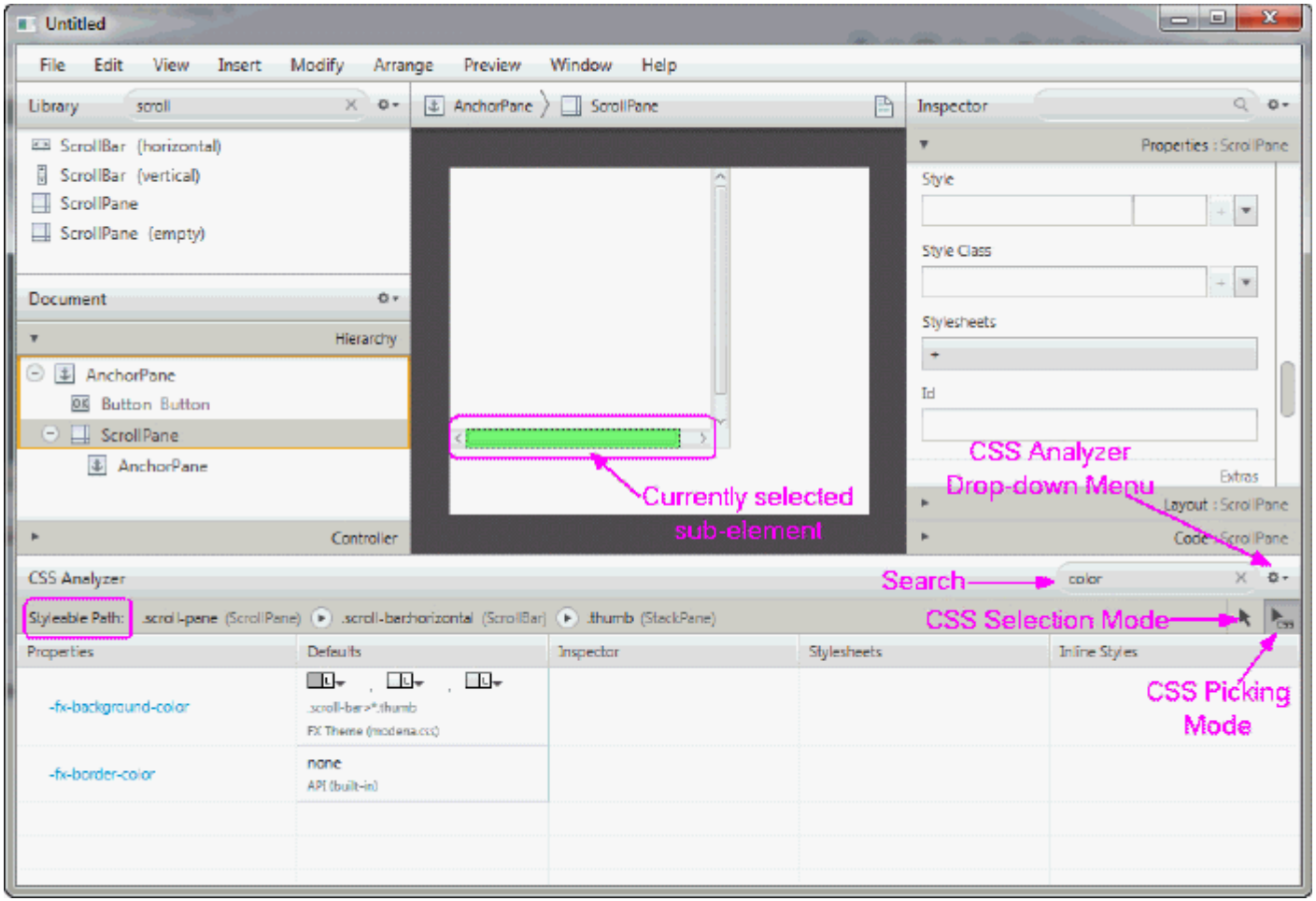
También se puede navegar hasta el archivo CSS a través del panel analizador de CSS

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*

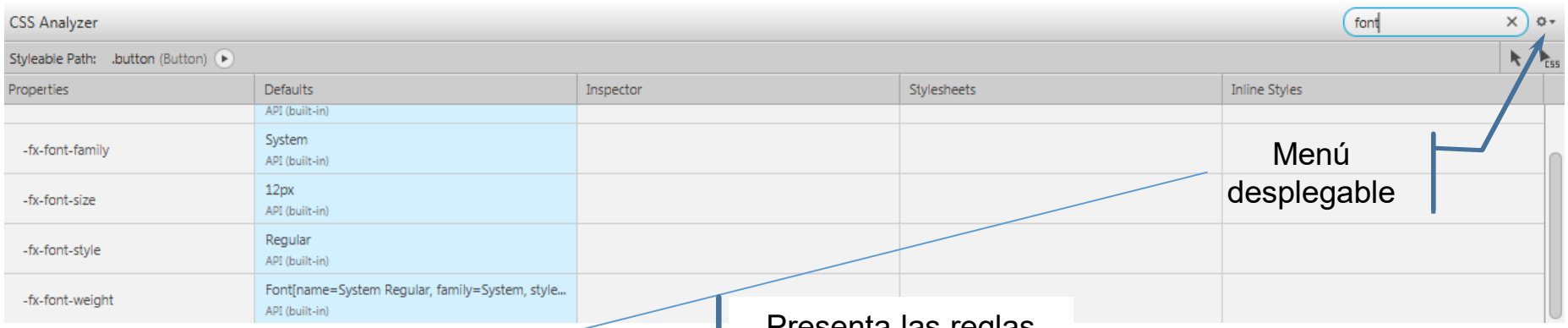


Panel analizador de CSS (II)



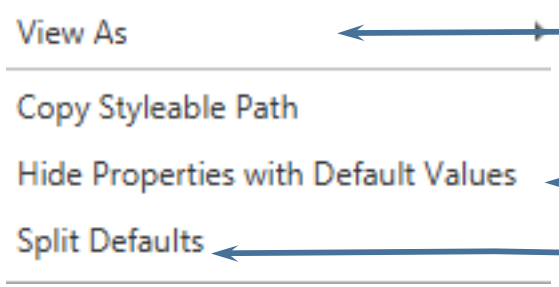
Panel analizador de CSS (III)

Filtrado de propiedades por nombre



Menú desplegable

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



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

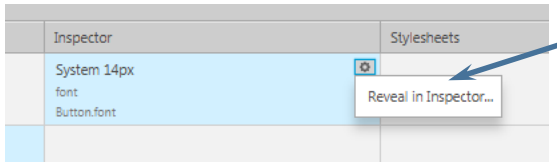
Separa en dos columnas los valores predeterminados en *API* y *temas FX*
Join Defaults vuelve a agrupar las columnas

Panel analizador de CSS (IV)

Valores por defecto, (API y tema JavaFX) que posee la propiedad

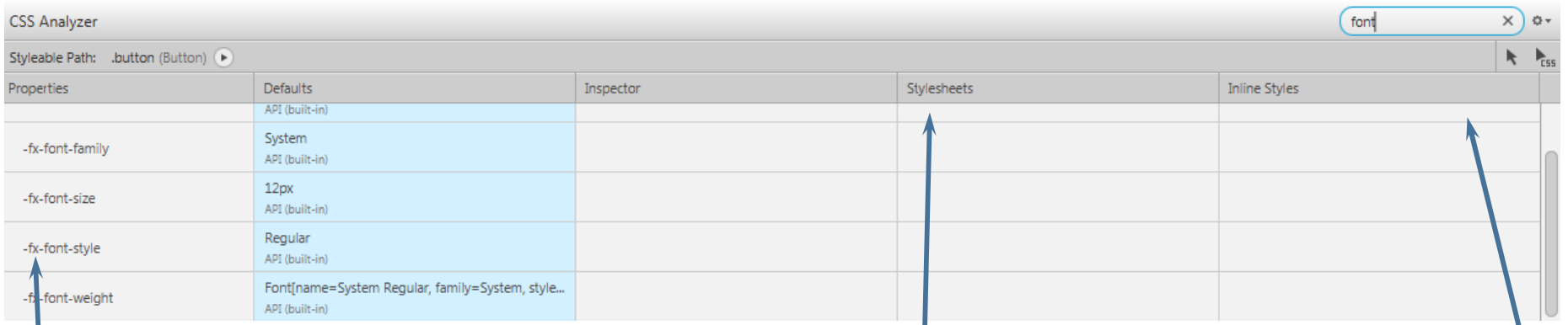
Propiedades de estilo disponibles para el elemento seleccionado en ese momento.

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



Resalta en azul la propiedad en el panel *Inspector*

Panel analizador de CSS (IV)

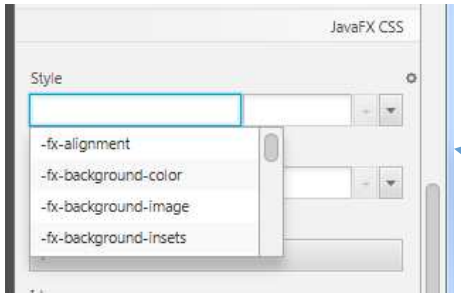


Propiedad establecida por un *stylesheet* del nodo o de algún ascendiente

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

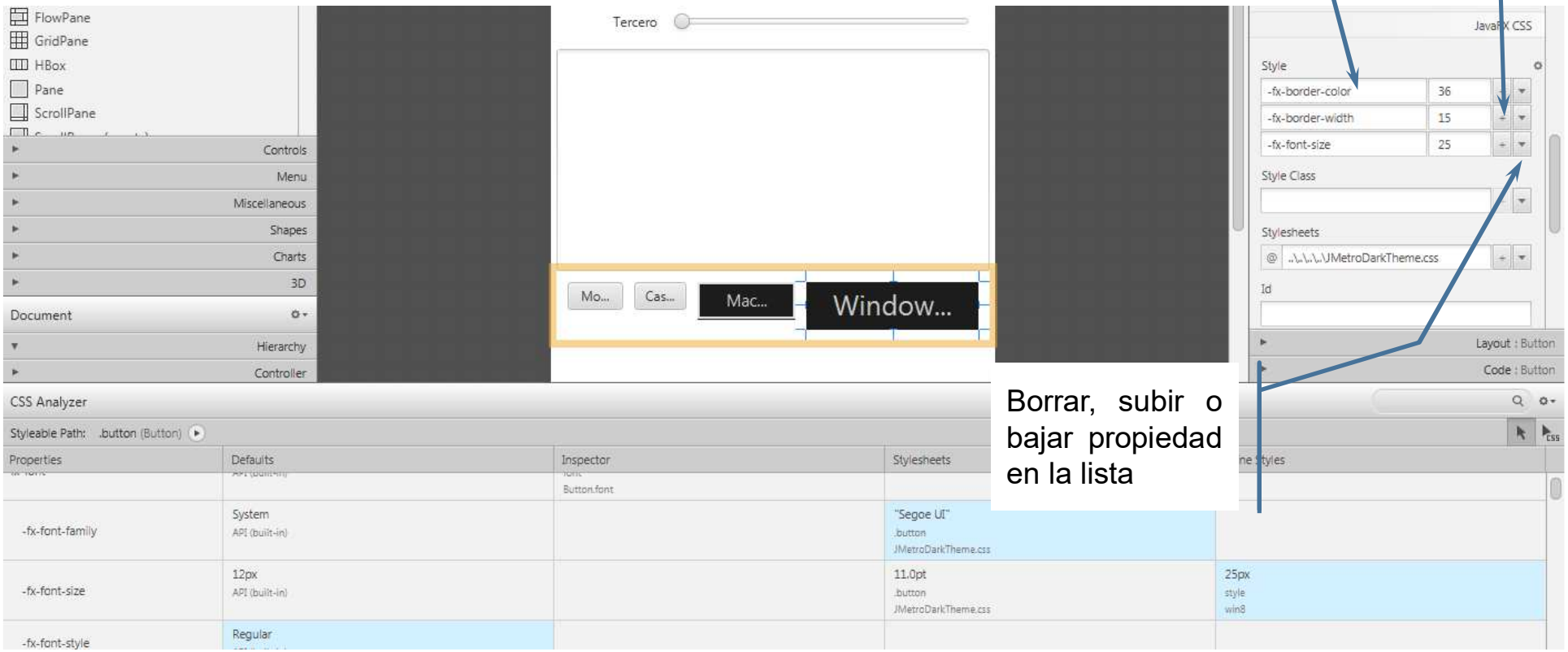
Propiedad establecida *en línea*, es decir como *Style* dentro de la sección *JavaFX CSS* del apartado *Properties* del *Inspector*

Panel analizador de CSS (V)



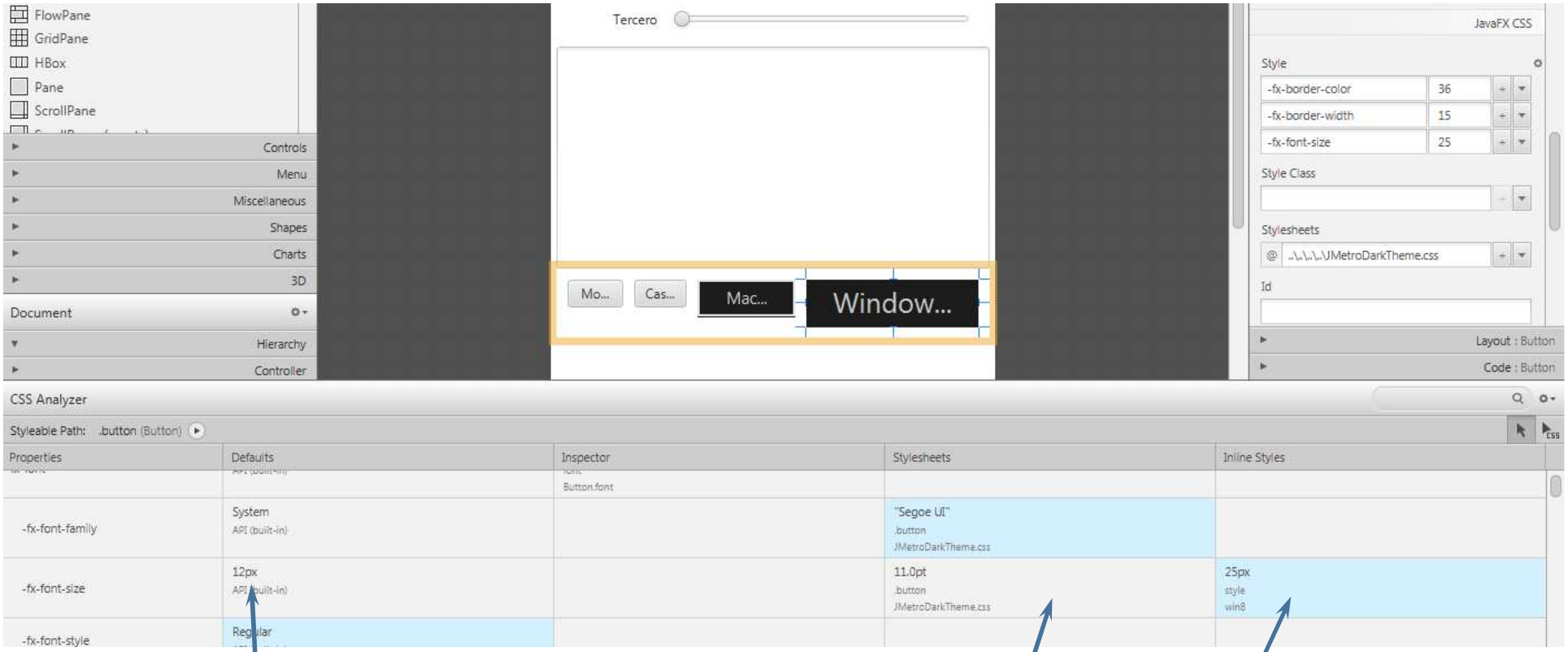
Botón + añade nuevas propiedades a modificar

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



Borrar, subir o bajar propiedad en la lista

Panel analizador de CSS (V)

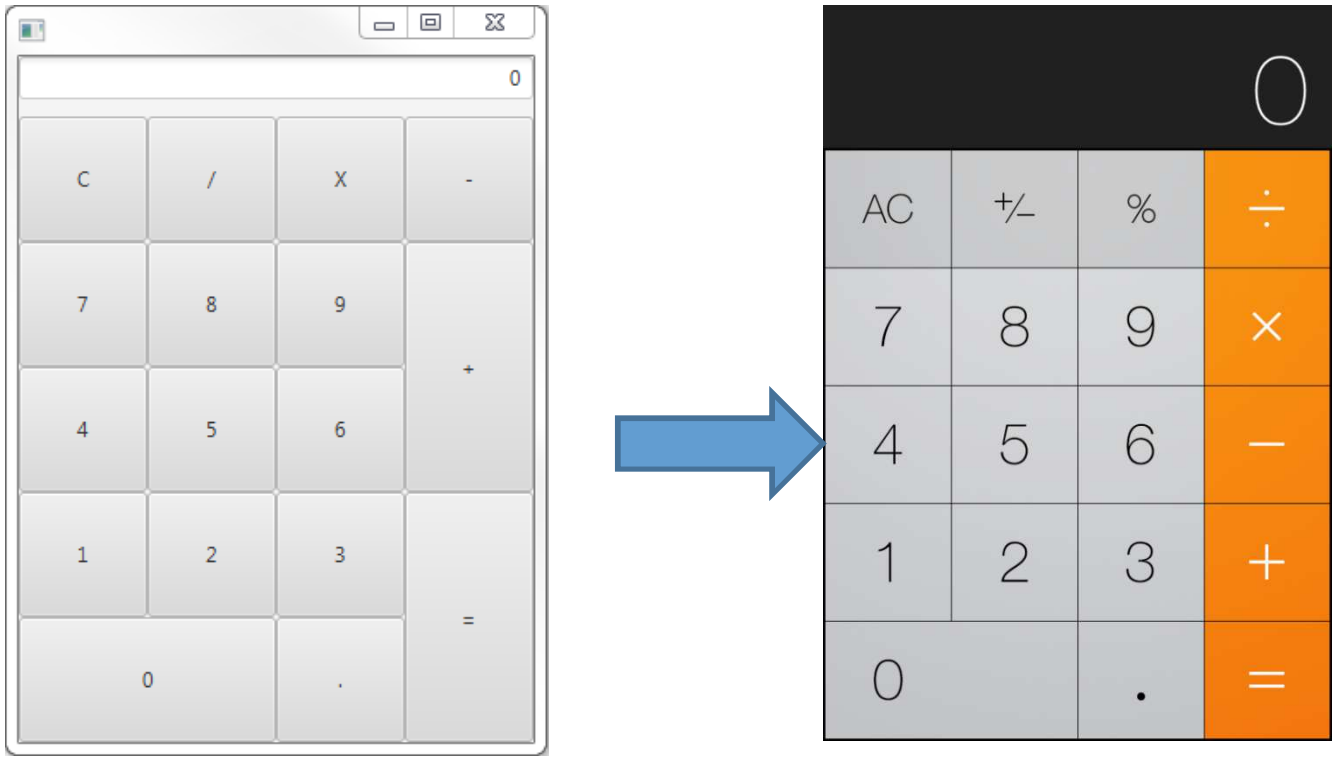


Si no se indica nada explícitamente, se asume el estilo por defecto

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