

```
//@version=5
```

```
indicator("Momentum Flow Build w/ FVG v2", overlay=true)
```

```
// =====
```

```
// 🚩 Sessions (unchanged)
```

```
// =====
```

```
// Session 1 Inputs
```

```
session1_enabled = input.bool(true, title="Enable Session 1", group="🚩 Session 1")
```

```
session1_time = input.session("1800-0000", title="Session 1 Time", group="🚩 Session 1")
```

```
session1_border_color = input.color(#ffed4f, title="Session 1 Border Color", group="🚩 Session 1")
```

```
session1_bg_color = input.color(color.new(#ffed4f, 90), title="Session 1 Background Color", group="🚩 Session 1")
```

```
// Session 2 Inputs
```

```
session2_enabled = input.bool(true, title="Enable Session 2", group="🚩 Session 2")
```

```
session2_time = input.session("0000-0600", title="Session 2 Time", group="🚩 Session 2")
```

```
session2_border_color = input.color(color.blue, title="Session 2 Border Color", group="🚩 Session 2")
```

```
session2_bg_color = input.color(color.new(color.rgb(255, 229, 79), 90), title="Session 2 Background Color", group="🚩 Session 2")
```

```
// Session 3 Inputs
```

```
session3_enabled = input.bool(true, title="Enable Session 3", group="🚩 Session 3")
```

```
session3_time = input.session("0930-0935", title="Session 3 Time", group="🚩 Session 3")
```

```
session3_border_color = input.color(color.new(color.yellow,0), title="Session 3 Border Color", group="🚫 Session 3")
```

```
session3_bg_color = input.color(color.new(color.yellow, 90), title="Session 3 Background Color", group="🚫 Session 3")
```

```
// Helper Functions
```

```
is_new_bar(sess) =>
```

```
    t = time("D", sess, "America/New_York")
```

```
    na(t[1]) and not na(t) or t[1] < t
```

```
is_in_session(sess) =>
```

```
    not na(time("D", sess, "America/New_York"))
```

```
get_session_end(sess, durationHours, durationMinutes) =>
```

```
    t = time("D", sess, "America/New_York")
```

```
    if na(t)
```

```
        na
```

```
    else
```

```
        t + (durationHours * 60 * 60 * 1000) + (durationMinutes * 60 * 1000)
```

```
plot_session_boxes(session_enabled, session_time, border_color, bg_color, durationHours, durationMinutes) =>
```

```
    var box[] session_boxes = array.new_box()
```

```
    if session_enabled and is_in_session(session_time)
```

```
        new_bar = is_new_bar(session_time)
```

```
        var float session_low = na
```

```
        var float session_high = na
```

```

session_low := new_bar ? low : math.min(session_low, low)
session_high := new_bar ? high : math.max(session_high, high)

var int session_start = na
session_start := new_bar ? time : session_start

session_end = get_session_end(session_time, durationHours, durationMinutes)

if new_bar
    session_box = box.new(left=session_start, bottom=session_low, right=session_end,
top=session_high,
        border_width=1, xloc=xloc.bar_time, border_style=line.style_solid,
        border_color=border_color, bgcolor=bg_color)
    array.push(session_boxes, session_box)
else
    if array.size(session_boxes) > 0
        box.set_right(array.get(session_boxes, array.size(session_boxes) - 1), session_end)
        box.set_top(array.get(session_boxes, array.size(session_boxes) - 1), session_high)
        box.set_bottom(array.get(session_boxes, array.size(session_boxes) - 1),
session_low)

// Plot Sessions
plot_session_boxes(session1_enabled, session1_time, session1_border_color,
session1_bg_color, 6, 0)
plot_session_boxes(session2_enabled, session2_time, session2_border_color,
session2_bg_color, 6, 0)

```

```
plot_session_boxes(session3_enabled, session3_time, session3_border_color,
session3_bg_color, 1, 30)
```

```
// =====
```

```
// ✨ Fair Value Gaps (FVG)
```

```
// =====
```

```
grpFVG = " ✨ Fair Value Gaps"
```

```
fvg_enabled   = input.bool(true, "Enable FVG Detection", group=grpFVG)
```

```
fvg_extend_fill = input.bool(true, "Extend Boxes Until Filled", group=grpFVG)
```

```
fvg_min_pct    = input.float(0.00, "Min Gap Size (% of Price)", step=0.01, group=grpFVG)
```

```
fvg_bull_color = input.color(color.new(color.lime, 62), "Bullish FVG Color", group=grpFVG)
```

```
fvg_bear_color = input.color(color.new(color.red, 62), "Bearish FVG Color",
group=grpFVG)
```

```
// Internal state
```

```
var box[] fvg_boxes = array.new_box()
```

```
var float[] fvg_top   = array.new_float()
```

```
var float[] fvg_bottom = array.new_float()
```

```
var bool[] fvg_isBull = array.new_bool()
```

```
var bool[] fvg_active = array.new_bool()
```

```
var bool[] fvg_touched = array.new_bool()
```

```
var bool[] fvg_signaled = array.new_bool()
```

```
f_add_fvg(isBull, top, bottom) =>
```

```
    bcolor = isBull ? fvg_bull_color : fvg_bear_color
```

```

bx = box.new(left=time, right=time, top=top, bottom=bottom,
             xloc=xloc.bar_time, bgcolor=bgcolor, border_color=bgcolor)
array.push(fvg_boxes, bx)
array.push(fvg_top, top)
array.push(fvg_bottom, bottom)
array.push(fvg_isBull, isBull)
array.push(fvg_active, true)
array.push(fvg_touched, false)
array.push(fvg_signaled, false)

// Detect FVGs
if fvg_enabled and bar_index >= 2
    bull_cond = low > high[2]
    bear_cond = high < low[2]

    bull_size = bull_cond ? (low - high[2]) : 0.0
    bear_size = bear_cond ? (low[2] - high) : 0.0


    bull_ok = bull_cond and (fvg_min_pct <= 0 or (bull_size / close) * 100 >= fvg_min_pct)
    bear_ok = bear_cond and (fvg_min_pct <= 0 or (bear_size / close) * 100 >= fvg_min_pct)

    if bull_ok
        f_add_fvg(true, low, high[2])
    if bear_ok
        f_add_fvg(false, low[2], high)

```

```

// =====

//  FVG Management + Entries

// =====

var bool buySignal = false
var bool sellSignal = false

buySignal := false
sellSignal := false

if fvg_enabled and array.size(fvg_boxes) > 0
    for i = 0 to array.size(fvg_boxes) - 1
        if array.get(fvg_active, i)
            bx = array.get(fvg_boxes, i)
            top = array.get(fvg_top, i)
            bot = array.get(fvg_bottom, i)
            bull = array.get(fvg_isBull, i)
            touched = array.get(fvg_touched, i)
            signaled = array.get(fvg_signaled, i)

            box.set_right(bx, time)

            // Mark retrace INTO FVG
            in_gap = low <= top and high >= bot
            if in_gap
                array.set(fvg_touched, i, true)

```

```

// Entry logic AFTER retrace
if touched and not signaled
    if bull and close > top
        buySignal := true
        array.set(fvg_signaled, i, true)
    if not bull and close < bot
        sellSignal := true
        array.set(fvg_signaled, i, true)

// Stop extending after full fill
filled = bull ? (low <= bot) : (high >= top)
if filled and fvg_extend_fill
    array.set(fvg_active, i, false)

// =====

// 📌 Plots & Alerts

// =====

plotshape(buySignal, title="FVG Buy", style=shape.labelup, location=location.belowbar,
    color=color.lime, text="BUY", size=size.small)

plotshape(sellSignal, title="FVG Sell", style=shape.labeldown, location=location.abovebar,
    color=color.red, text="SELL", size=size.small)

```

```
alertcondition(buySignal, title="FVG Buy Signal", message="Bullish FVG retrace + close  
confirmation on {{ticker}}")
```

```
alertcondition(sellSignal, title="FVG Sell Signal", message="Bearish FVG retrace + close  
confirmation on {{ticker}}")
```