In [13]:
```
# ── Standard Library ──────────────────────────────────────────────
import os
import warnings
warnings.filterwarnings('ignore')

# ── ArcPy ─────────────────────────────────────────────────────────
import arcpy
from arcpy import env as arcpy_env
from arcpy.sa import Raster, Con, SetNull   # Spatial Analyst

# ── ArcGIS API for Python ─────────────────────────────────────────
from arcgis.gis import GIS
from arcgis.mapping import WebMap
from arcgis.features import FeatureLayer
import arcgis.geometry as geom

# ── Data Science ──────────────────────────────────────────────────
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np

print(f"arcpy version    : {arcpy.GetInstallInfo()['Version']}")
print(f"arcgis version   : {arcgis.__version__}")
print(f"pandas version   : {pd.__version__}")
print("✅ All libraries imported successfully.")
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[13], line 13
     11 # ── ArcGIS API for Python
_____________________________________________________
     12 from arcgis.gis import GIS
---> 13 from arcgis.mapping import WebMap
     14 from arcgis.features import FeatureLayer
     15 import arcgis.geometry as geom

ModuleNotFoundError: No module named 'arcgis.mapping'
```

In [14]:
```
# ── Set your project folder path here ─────────────────────────────
PROJECT_FOLDER = r"C:\Final_Projects\EastTroublesomeFire"

# ── Derived paths ─────────────────────────────────────────────────
GDB_NAME       = "EastTroublesome.gdb"
GDB_PATH       = os.path.join(PROJECT_FOLDER, GDB_NAME)
SCRATCH_FOLDER = os.path.join(PROJECT_FOLDER, "scratch")
DATA_FOLDER    = os.path.join(PROJECT_FOLDER, "data")
OUTPUT_FOLDER  = os.path.join(PROJECT_FOLDER, "outputs")

# ── Create directories if they don't exist ────────────────────────
for folder in [PROJECT_FOLDER, SCRATCH_FOLDER, DATA_FOLDER, OUTPUT_FOLDER]:
    os.makedirs(folder, exist_ok=True)

# ── Create File Geodatabase ───────────────────────────────────────
```

```
if not arcpy.Exists(GDB_PATH):
    arcpy.management.CreateFileGDB(PROJECT_FOLDER, GDB_NAME)
    print(f"✅ Geodatabase created: {GDB_PATH}")
else:
    print(f"✅ Geodatabase already exists: {GDB_PATH}")

# ── ArcPy Environment Settings ──────────────────────────────────
arcpy_env.workspace        = GDB_PATH
arcpy_env.scratchWorkspace = SCRATCH_FOLDER
arcpy_env.overwriteOutput  = True

# ── Coordinate Reference System ─────────────────────────────────
# NAD 1983 UTM Zone 13N — standard for Colorado GIS analysis
SR_UTM13N = arcpy.SpatialReference(26913)

print(f"\nProject folder  : {PROJECT_FOLDER}")
print(f"Geodatabase     : {GDB_PATH}")
print(f"Spatial Ref     : {SR_UTM13N.name}")
```

```
✅ Geodatabase already exists: C:\Final_Projects\EastTroublesomeFire\EastTroublesom
e.gdb

Project folder  : C:\Final_Projects\EastTroublesomeFire
Geodatabase     : C:\Final_Projects\EastTroublesomeFire\EastTroublesome.gdb
Spatial Ref     : NAD_1983_UTM_Zone_13N
```

In [15]:
```
from arcgis.gis import GIS
gis = GIS("https://www.arcgis.com", "Rflands92", "Crumble@jeremy1992")
gis
```

Out[15]: GIS @ https://FordLands92GIS.maps.arcgis.com

In [16]:
```
gis
```

Out[16]: GIS @ https://FordLands92GIS.maps.arcgis.com

In [18]:
```
from arcgis.gis import GIS
from arcgis.features import FeatureLayer

gis = GIS("home")   # or GIS()
```

In [19]:
```
# ──────────────────────────────────────────────────────────────────
MTBS_SHAPEFILE = r"C:\Data_Folder\Fire_data_bundles_78wAioKSiyc3L2nn3JGV\mtbs\2020\

if arcpy.Exists(MTBS_SHAPEFILE):
    print("✅ MTBS shapefile found — using local file.")
    FIRE_PERIMETER_SOURCE = "local"
else:
    print("⚠️  MTBS shapefile not found locally.")
    print("   Falling back to NIFC Feature Service (Option B)...")
    FIRE_PERIMETER_SOURCE = "service"
```

```
✅ MTBS shapefile found — using local file.
```

In [20]:
```
# ── Copy fire perimeter to geodatabase and reproject ──────────────
FIRE_PERIMETER_FC = os.path.join(GDB_PATH, "fire_perimeter_utm")
```

```
if FIRE_PERIMETER_SOURCE == "local":
    arcpy.management.Project(
        in_dataset       = MTBS_SHAPEFILE,
        out_dataset      = FIRE_PERIMETER_FC,
        out_coor_system  = SR_UTM13N
    )
else:
    # Reproject the service-pulled feature class
    arcpy.management.Project(
        in_dataset       = os.path.join(GDB_PATH, "fire_perimeter_raw"),
        out_dataset      = FIRE_PERIMETER_FC,
        out_coor_system  = SR_UTM13N
    )

# ── Print basic info ──────────────────────────────────────────────
count = int(arcpy.management.GetCount(FIRE_PERIMETER_FC).getOutput(0))
desc  = arcpy.Describe(FIRE_PERIMETER_FC)
print(f"✅ Fire perimeter loaded")
print(f"   Feature count    : {count}")
print(f"   Spatial reference: {desc.spatialReference.name}")
print(f"   Extent           : {desc.extent}")
```

```
✅ Fire perimeter loaded
   Feature count    : 1
   Spatial reference: NAD_1983_UTM_Zone_13N
   Extent           : 392138.7306 4439520.498 438457.3574 4468309.7398 NaN NaN NaN N
aN
```

In [27]:

```
# ── USFS Feature Service (no download needed) ─────────────────────
# Alternatively download from: https://data.fs.usda.gov/geodata/edw/datasets.php

USFS_FOREST_URL = "https://apps.fs.usda.gov/arcx/rest/services/EDW/EDW_ForestSystem

FOREST_SHAPEFILE = os.path.join(DATA_FOLDER, "S_USA.AdministrativeForest.shp")  # l
FOREST_FC_UTM    = os.path.join(GDB_PATH, "national_forests_co_utm")

if arcpy.Exists(FOREST_SHAPEFILE):
    # Project local shapefile to UTM 13N
    arcpy.management.Project(
        in_dataset      = FOREST_SHAPEFILE,
        out_dataset     = FOREST_FC_UTM,
        out_coor_system = SR_UTM13N
    )
    print("✅ National Forest boundaries loaded from local shapefile.")
else:
    # Fallback: download/query ALL forests and then clip to Colorado locally
    forest_layer = FeatureLayer(USFS_FOREST_URL)  # no gis needed

    # 1) Download all forests (no state filter)
    all_forests = forest_layer.query(
        where="1=1",
        out_fields="*",
        return_geometry=True
    )
```

```
# 2) Save to geodatabase
all_forests.save(GDB_PATH, "national_forests_raw")

# 3) Clip to Colorado boundary (YOU must provide a Colorado boundary dataset)
CO_BOUNDARY = os.path.join(GDB_PATH, "colorado_boundary")  # <-- you must creat

arcpy.analysis.Clip(
    in_features=os.path.join(GDB_PATH, "national_forests_raw"),
    clip_features=CO_BOUNDARY,
    out_feature_class=os.path.join(GDB_PATH, "national_forests_co_raw")
)

# 4) Project clipped result to UTM 13N
arcpy.management.Project(
    in_dataset=os.path.join(GDB_PATH, "national_forests_co_raw"),
    out_dataset=FOREST_FC_UTM,
    out_coor_system=SR_UTM13N
)

print("✅ National Forests clipped to Colorado and projected to UTM 13N.")
```

```
---------------------------------------------------------------------------
ExecuteError                              Traceback (most recent call last)
Cell In[27], line 34
     31 # 3) Clip to Colorado boundary (YOU must provide a Colorado boundary datase
t)
     32 CO_BOUNDARY = os.path.join(GDB_PATH, "colorado_boundary")  # <-- you must cr
eate this
---> 34 arcpy.analysis.Clip(
     35     in_features=os.path.join(GDB_PATH, "national_forests_raw"),
     36     clip_features=CO_BOUNDARY,
     37     out_feature_class=os.path.join(GDB_PATH, "national_forests_co_raw")
     38 )
     40 # 4) Project clipped result to UTM 13N
     41 arcpy.management.Project(
     42     in_dataset=os.path.join(GDB_PATH, "national_forests_co_raw"),
     43     out_dataset=FOREST_FC_UTM,
     44     out_coor_system=SR_UTM13N
     45 )

File C:\Program Files\ArcGIS\Pro\Resources\ArcPy\arcpy\analysis.py:200, in Clip(in_f
eatures, clip_features, out_feature_class, cluster_tolerance)
    198     return retval
    199 except Exception as e:
--> 200     raise e

File C:\Program Files\ArcGIS\Pro\Resources\ArcPy\arcpy\analysis.py:196, in Clip(in_f
eatures, clip_features, out_feature_class, cluster_tolerance)
    192 from arcpy.arcobjects.arcobjectconversion import convertArcObjectToPythonObj
ect
    194 try:
    195     retval = convertArcObjectToPythonObject(
--> 196         gp.Clip_analysis(*gp_fixargs((in_features, clip_features, out_featur
e_class, cluster_tolerance), True))
    197     )
    198     return retval
    199 except Exception as e:

File C:\Program Files\ArcGIS\Pro\Resources\ArcPy\arcpy\geoprocessing\_base.py:533, i
n Geoprocessor.__getattr__.<locals>.<lambda>(*args)
    531 val = getattr(self._gp, attr)
    532 if callable(val):
--> 533     return lambda *args: val(*gp_fixargs(args, True))
    534 else:
    535     return convertArcObjectToPythonObject(val)

ExecuteError: Failed to execute. Parameters are not valid.
ERROR 000732: Clip Features: Dataset C:\Final_Projects\EastTroublesomeFire\EastTroub
lesome.gdb\colorado_boundary does not exist or is not supported
Failed to execute (Clip).
```

In [28]:
```
import glob, os
glob.glob(os.path.join(DATA_FOLDER, "*.shp"))
```

Out[28]:
```
[]
```

In [29]:
```
print("DATA_FOLDER =", DATA_FOLDER)
print("GDB_PATH =", GDB_PATH)
print("Exists DATA_FOLDER?", os.path.exists(DATA_FOLDER))
print("Exists GDB_PATH?", os.path.exists(GDB_PATH))
```

```
DATA_FOLDER = C:\Final_Projects\EastTroublesomeFire\data
GDB_PATH = C:\Final_Projects\EastTroublesomeFire\EastTroublesome.gdb
Exists DATA_FOLDER? True
Exists GDB_PATH? True
```

In [30]:
```
import os
print(os.listdir(DATA_FOLDER))
```

```
[]
```

In [31]:
```
import arcpy
arcpy.env.workspace = GDB_PATH
print("Feature classes in GDB:")
print(arcpy.ListFeatureClasses())
```

```
Feature classes in GDB:
['fire_perimeter_utm', 'national_forests_raw']
```

In [32]:
```
import os, arcpy

arcpy.env.workspace = GDB_PATH

in_fc   = os.path.join(GDB_PATH, "national_forests_raw")
clip_fc = os.path.join(GDB_PATH, "fire_perimeter_utm")
out_fc  = os.path.join(GDB_PATH, "national_forests_in_fire")

# (Optional) delete if it already exists so reruns don't fail
if arcpy.Exists(out_fc):
    arcpy.management.Delete(out_fc)

arcpy.analysis.Clip(in_fc, clip_fc, out_fc)

print("✅ Clipped forests saved to:", out_fc)
print("Count:", int(arcpy.management.GetCount(out_fc)[0]))
```

```
✅ Clipped forests saved to: C:\Final_Projects\EastTroublesomeFire\EastTroublesom
e.gdb\national_forests_in_fire
Count: 2
```

In [33]:
```
with arcpy.da.SearchCursor(out_fc, ["FORESTNAME"]) as cur:
    names = sorted({row[0] for row in cur if row[0]})
for n in names:
    print(n)
```

```
Arapaho and Roosevelt National Forests
Medicine Bow-Routt National Forest
```

In [34]:
```
import arcpy, os

fire_fc   = os.path.join(GDB_PATH, "fire_perimeter_utm")
forest_fc = os.path.join(GDB_PATH, "national_forests_in_fire")

print("Fire SR:", arcpy.Describe(fire_fc).spatialReference.name)
```

```
print("Forest SR:", arcpy.Describe(forest_fc).spatialReference.name)
```

```
Fire SR: NAD_1983_UTM_Zone_13N
Forest SR: WGS_1984_Web_Mercator_Auxiliary_Sphere
```

In [35]:
```
FIRE_FOREST_INTERSECT = os.path.join(GDB_PATH, "fire_forest_intersect")

if arcpy.Exists(FIRE_FOREST_INTERSECT):
    arcpy.management.Delete(FIRE_FOREST_INTERSECT)

arcpy.analysis.Intersect(
    in_features=[fire_fc, forest_fc],
    out_feature_class=FIRE_FOREST_INTERSECT,
    join_attributes="ALL",
    output_type="INPUT"
)

print("✅ Intersect complete. Count:", int(arcpy.management.GetCount(FIRE_FOREST_IN
```

```
✅ Intersect complete. Count: 4
```

In [36]:
```
for fc in [fire_fc, FIRE_FOREST_INTERSECT]:
    existing = [f.name for f in arcpy.ListFields(fc)]
    if "CALC_ACRES" not in existing:
        arcpy.management.AddField(fc, "CALC_ACRES", "DOUBLE")

    arcpy.management.CalculateGeometryAttributes(
        in_features=fc,
        geometry_property=[["CALC_ACRES", "AREA"]],
        area_unit="ACRES"
    )

print("✅ Acreage fields calculated.")
```

```
✅ Acreage fields calculated.
```

In [37]:
```
total_fire_acres = sum(row[0] for row in arcpy.da.SearchCursor(fire_fc, ["CALC_ACRE

forest_burn_rows = [(row[0], row[1]) for row in arcpy.da.SearchCursor(
    FIRE_FOREST_INTERSECT, ["CALC_ACRES", "FORESTNAME"]
)]

forest_burn_acres = sum(r[0] for r in forest_burn_rows)
outside_forest_acres = total_fire_acres - forest_burn_acres

import pandas as pd
forest_df = pd.DataFrame(forest_burn_rows, columns=["Acres", "ForestName"])
by_forest = forest_df.groupby("ForestName")["Acres"].sum().reset_index()
by_forest.columns = ["National Forest", "Acres Burned"]
by_forest["% of Forest Burn"] = (by_forest["Acres Burned"] / forest_burn_acres * 10
by_forest = by_forest.sort_values("Acres Burned", ascending=False)

print("=" * 55)
print("EAST TROUBLESOME FIRE - ACREAGE SUMMARY")
print("=" * 55)
print(f"Total Fire Perimeter            : {total_fire_acres:12,.1f} acres")
print(f"Burned in National Forest       : {forest_burn_acres:12,.1f} acres")
```

```
print(f"Burned Outside National Forest  : {outside_forest_acres:12,.1f} acres")
print(f"% within National Forest        : {forest_burn_acres/total_fire_acres*100:1
print("=" * 55)
print("\nBreakdown by National Forest:")
print(by_forest.to_string(index=False))
```

```
============================================================
EAST TROUBLESOME FIRE - ACREAGE SUMMARY
============================================================
Total Fire Perimeter            :    188,812.8 acres
Burned in National Forest       :    134,735.0 acres
Burned Outside National Forest  :     54,077.8 acres
% within National Forest        :         71.4%
============================================================

Breakdown by National Forest:
                      National Forest  Acres Burned  % of Forest Burn
Arapaho and Roosevelt National Forests  90709.102963              67.3
    Medicine Bow-Routt National Forest  44025.932211              32.7
```

In [39]:
```
# ── Path to MTBS Thematic Burn Severity Raster ──────────────────────
# The MTBS package contains a classified severity raster (values 1-5)
SEVERITY_RASTER = r"C:\ArcGIS\BootcampGIS\Final Project wildfire\mtbs\2020\co402031

if not arcpy.Exists(SEVERITY_RASTER):
    print("⚠️  Severity raster not found at specified path.")
    print(f"   Expected: {SEVERITY_RASTER}")
    print("   Download from https://www.mtbs.gov/direct-download")
    print("   Event ID: CO4023510601020201021")
    SEVERITY_AVAILABLE = False
else:
    SEVERITY_AVAILABLE = True
    print(f"✅ Severity raster found: {SEVERITY_RASTER}")

    # Describe the raster
    r_desc = arcpy.Describe(SEVERITY_RASTER)
    print(f"   Raster size   : {r_desc.width} × {r_desc.height} pixels")
    print(f"   Cell size     : {r_desc.meanCellWidth:.1f} meters")
    print(f"   Spatial ref   : {r_desc.spatialReference.name}")
```

```
✅ Severity raster found: C:\ArcGIS\BootcampGIS\Final Project wildfire\mtbs\2020\co
402031062392020101 4\co402031062392020101 4_20200904_20210907_dnbr6.tif
   Raster size   : 1742 × 1162 pixels
   Cell size     : 30.0 meters
   Spatial ref   : USA_Contiguous_Albers_Equal_Area_Conic_USGS_version
```

In [41]:
```
SQM_TO_ACRES = 1 / 4046.8564224
```

In [42]:
```
if SEVERITY_AVAILABLE:
    # ── Check out Spatial Analyst license ──────────────────────────
    arcpy.CheckOutExtension("Spatial")

    # ── Severity class definitions ─────────────────────────────────
    severity_classes = {
        1: "Unburned / Very Low",
        2: "Low Severity",
        3: "Moderate Severity",
```

```
    4: "High Severity",
    5: "Increased Greenness"
}
severity_colors = {
    1: "#78c679",   # green
    2: "#fed976",   # yellow
    3: "#fd8d3c",   # orange
    4: "#e31a1c",   # red
    5: "#2c7bb6"    # blue (regrowth)
}

# ── Set analysis mask to National Forest intersection ──────────────────
arcpy_env.mask = FIRE_FOREST_INTERSECT
arcpy_env.snapRaster = SEVERITY_RASTER

severity_ras = Raster(SEVERITY_RASTER)

# ── Tabulate Area by severity class ────────────────────────────────────
SEVERITY_TABLE = os.path.join(GDB_PATH, "severity_by_class")
arcpy.sa.TabulateArea(
    in_zone_data        = FIRE_FOREST_INTERSECT,
    zone_field          = "FORESTNAME",
    in_class_data       = SEVERITY_RASTER,
    class_field         = "Value",
    out_table           = SEVERITY_TABLE,
    processing_cell_size= 30
)

# ── Read results to pandas ─────────────────────────────────────────────
sev_rows = []
sev_fields = [f.name for f in arcpy.ListFields(SEVERITY_TABLE)]
with arcpy.da.SearchCursor(SEVERITY_TABLE, sev_fields) as cur:
    for row in cur:
        sev_rows.append(dict(zip(sev_fields, row)))

sev_df = pd.DataFrame(sev_rows)

# ── Reshape and convert m² → acres ─────────────────────────────────────
class_cols = [c for c in sev_df.columns if c.startswith("VALUE_")]
sev_long   = sev_df.melt(id_vars=["FORESTNAME"],
                         value_vars=class_cols,
                         var_name="Class", value_name="Area_m2")
sev_long["ClassValue"] = sev_long["Class"].str.replace("VALUE_", "").astype(int
sev_long["SeverityLabel"] = sev_long["ClassValue"].map(severity_classes)
sev_long["Acres"] = sev_long["Area_m2"] / 4046.8564224

# Overall severity summary
severity_summary = sev_long.groupby(["ClassValue","SeverityLabel"])["Acres"].su
severity_summary["% of Burn Area"] = (severity_summary["Acres"] / severity_summ
severity_summary = severity_summary.sort_values("ClassValue")

print("=" * 60)
print("  EAST TROUBLESOME — BURN SEVERITY (within National Forest)")
print("=" * 60)
print(severity_summary[["SeverityLabel","Acres","% of Burn Area"]].to_string(in
print("=" * 60)
```

```
    arcpy.CheckInExtension("Spatial")
```

```
============================================================
  EAST TROUBLESOME — BURN SEVERITY (within National Forest)
============================================================
      SeverityLabel        Acres  % of Burn Area
Unburned / Very Low 17736.878334            13.2
       Low Severity 28924.450927            21.5
  Moderate Severity 48819.226426            36.3
      High Severity 38615.973410            28.7
Increased Greenness   314.021519             0.2
============================================================
```

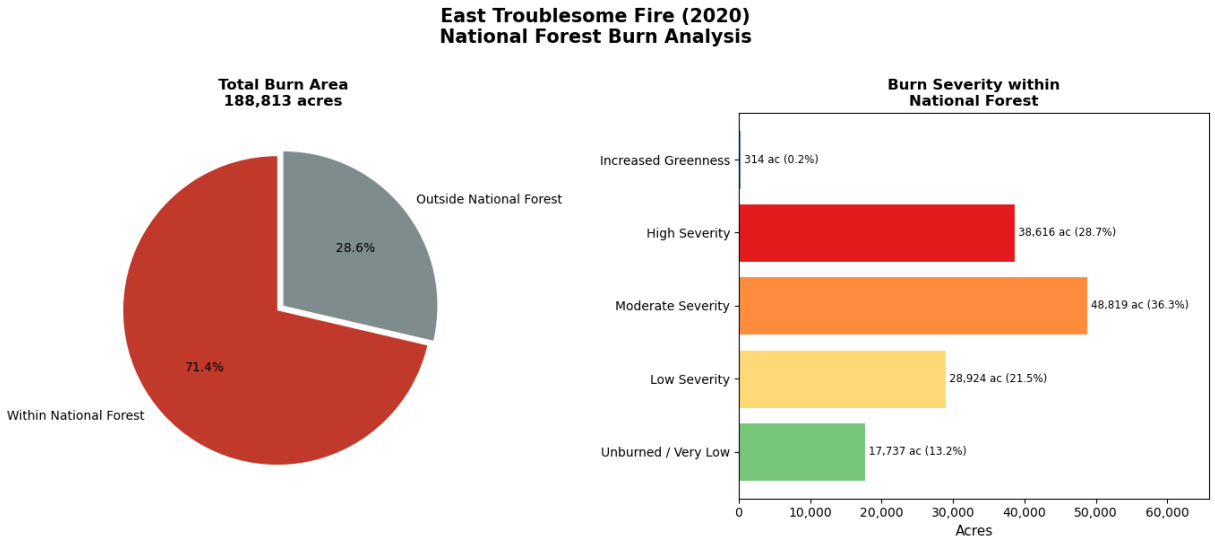
In [44]:
```
import matplotlib.pyplot as plt
```

In [45]:
```
import pandas as pd
import numpy as np
```

In [46]:
```
fig, axes = plt.subplots(1, 2, figsize=(14, 6))
fig.suptitle("East Troublesome Fire (2020)\nNational Forest Burn Analysis",
             fontsize=15, fontweight='bold', y=1.01)

# ── Chart 1: Overall Acreage Breakdown (Pie) ──────────────────────
ax1 = axes[0]
labels  = ["Within National Forest", "Outside National Forest"]
sizes   = [forest_burn_acres, outside_forest_acres]
colors  = ["#c0392b", "#7f8c8d"]
explode = (0.05, 0)

wedges, texts, autotexts = ax1.pie(
    sizes, labels=labels, colors=colors, autopct='%1.1f%%',
    explode=explode, startangle=90, textprops={'fontsize': 10}
)
ax1.set_title(f"Total Burn Area\n{total_fire_acres:,.0f} acres", fontsize=12, fontw

# ── Chart 2: Severity Breakdown (Horizontal Bar) ──────────────────
ax2 = axes[1]

if SEVERITY_AVAILABLE:
    sev_plot = severity_summary.sort_values("ClassValue", ascending=True)
    bar_colors = [severity_colors.get(int(v), '#aaaaaa') for v in sev_plot["ClassVa
    bars = ax2.barh(sev_plot["SeverityLabel"], sev_plot["Acres"],
                    color=bar_colors, edgecolor='white', linewidth=0.8)
    ax2.set_xlabel("Acres", fontsize=11)
    ax2.set_title("Burn Severity within\nNational Forest", fontsize=12, fontweight=
    ax2.xaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: f'{x:,.0f}'))

    # Add value labels
    for bar, val, pct in zip(bars, sev_plot["Acres"], sev_plot["% of Burn Area"]):
        ax2.text(bar.get_width() + 500, bar.get_y() + bar.get_height()/2,
                 f"{val:,.0f} ac ({pct}%)", va='center', fontsize=8.5)
    ax2.set_xlim(0, sev_plot["Acres"].max() * 1.35)
else:
    ax2.text(0.5, 0.5, "Severity raster not loaded\n(see Section 6 instructions)",
             ha='center', va='center', transform=ax2.transAxes, fontsize=11, color=
```

```
    ax2.axis('off')

plt.tight_layout()

# Save figure
chart_path = os.path.join(OUTPUT_FOLDER, "burn_analysis_charts.png")
plt.savefig(chart_path, dpi=150, bbox_inches='tight')
print(f"✅ Chart saved to: {chart_path}")
plt.show()
```

```
✅ Chart saved to: C:\Final_Projects\EastTroublesomeFire\outputs\burn_analysis_charts.png
```



In [49]:
```
# ── Create interactive map centered on Grand County, Colorado ──────────────
fire_map = gis.map("Grand County, Colorado")
fire_map.zoom = 9
fire_map = gis.map("Grand County, Colorado", {"basemap": "satellite"})
fire_map
```

Out[49]: Map()

In [56]:
```
# Set the layers you want
fire_map._layer_list = [FIRE_PERIMETER_FC, FIRE_FOREST_INTERSECT]

# Turn ON the layer list display (no parentheses!)
fire_map._show_layer_list = True

# Show the map
fire_map
```

Out[56]: Map()

In [ ]: