# Shadow DOM Fundamentals

Cory House
bitnative.com
Twitter: @housecor

hardcore developer training

## Agenda

- The scoping problem
- Today’s Hacks
- Shadow DOM vs Light DOM
- Terminology
    - Shadow Root
    - Shadow Boundary
    - Shadow Host
    - DOM Subtrees

## We Need Scoping

How do I avoid accidentally styling other parts of the page?

How do I avoid other people accidentally styling my component?

How do I hide away my markup from accidental manipulation?

# A Tale of Two DOM's

**Light DOM**

The DOM you know today

**Shadow DOM**

The DOM that hides away complexity

Logical DOM

## Shadow DOM is Already Used Today

```
<input type="range">

<video controls width="250"></video>

<input type="date" />
```

Shadow DOM in Native HTML Elements

## Demo

## Shadow DOM Hacks

Shadow DOM encapsulates DOM Subtrees and styles

Today’s ways to get similar behavior:

`<iframe>`

- Clunky to read
- Undescriptive
- Excessive encapsulation
- No clean API

`<canvas>`

- Accessibility issues
- SEO issues
- Can’t easily compose
- Can’t extend existing elements

## Create Shadow DOM

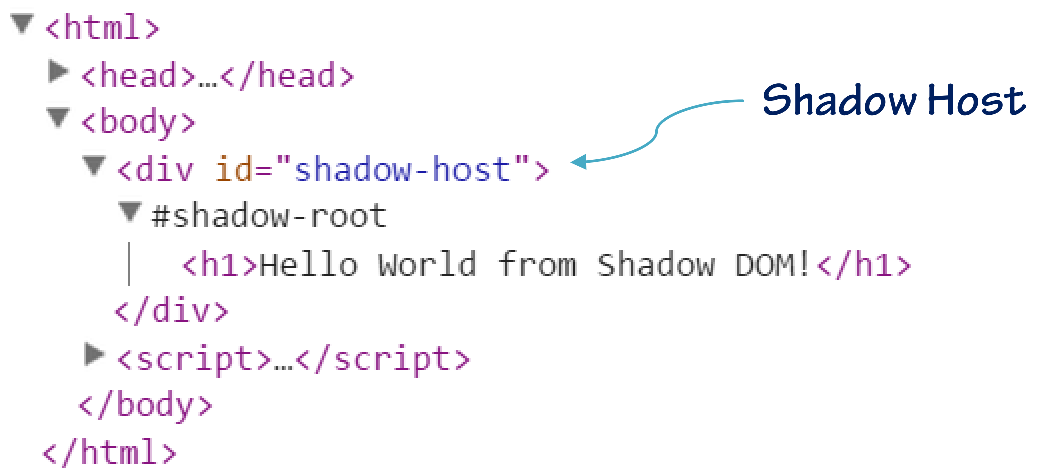
1. Select shadow host
2. Create a shadow root
3. Add elements
   - innerHTML
   - appendChild

Creating Shadow DOM

## Demo

## Shadow Host

**The element in the Light DOM that is hosting the Shadow DOM.**

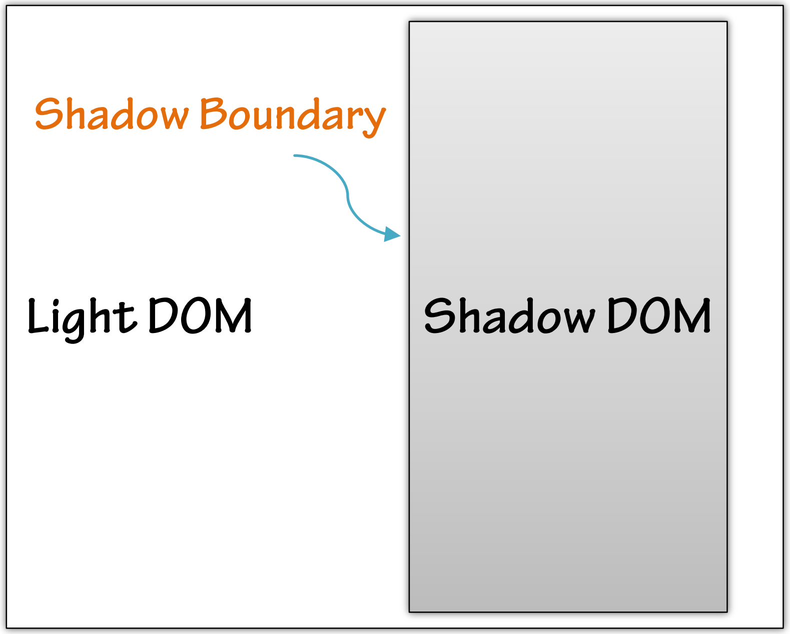


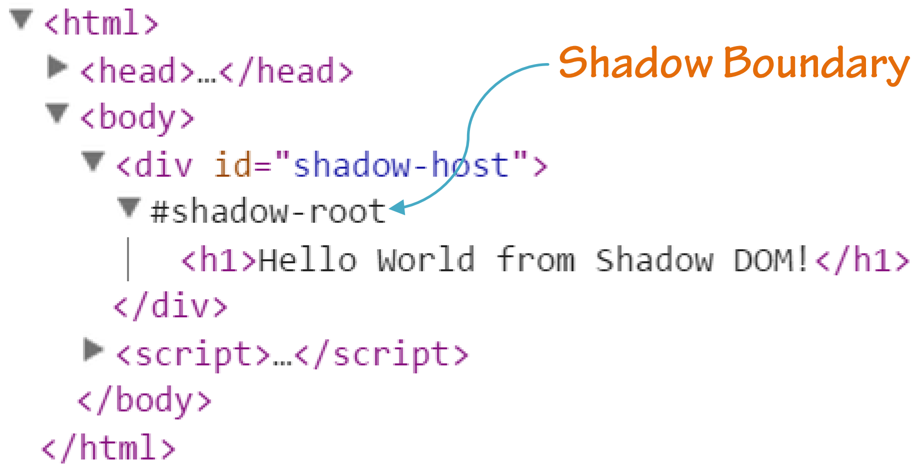
```
▼ <html>
  ► <head>…</head>
  ▼ <body>
    ▼ <div id="shadow-host">
      ▼ #shadow-root
      │   <h1>Hello World from Shadow DOM!</h1>
      </div>
    ► <script>…</script>
    </body>
  </html>
```

## Shadow Boundary

Ordinary CSS selectors inside shadow root only match/style the Shadow DOM.

Why? The Shadow Boundary.





```
▼ <html>
  ▶ <head>…</head>
  ▼ <body>
    ▼ <div id="shadow-host">
      ▼ #shadow-root
        <h1>Hello World from Shadow DOM!</h1>
      </div>
    ▶ <script>…</script>
    </body>
</html>
```

## Shadow Boundary

Barrier that separates the light DOM from the shadow DOM

Encapsulates DOM Subtree

Keeps styles in main doc from styling the shadow DOM and vice versa*

*There's some ways around it

## Shadow Boundary Benefits

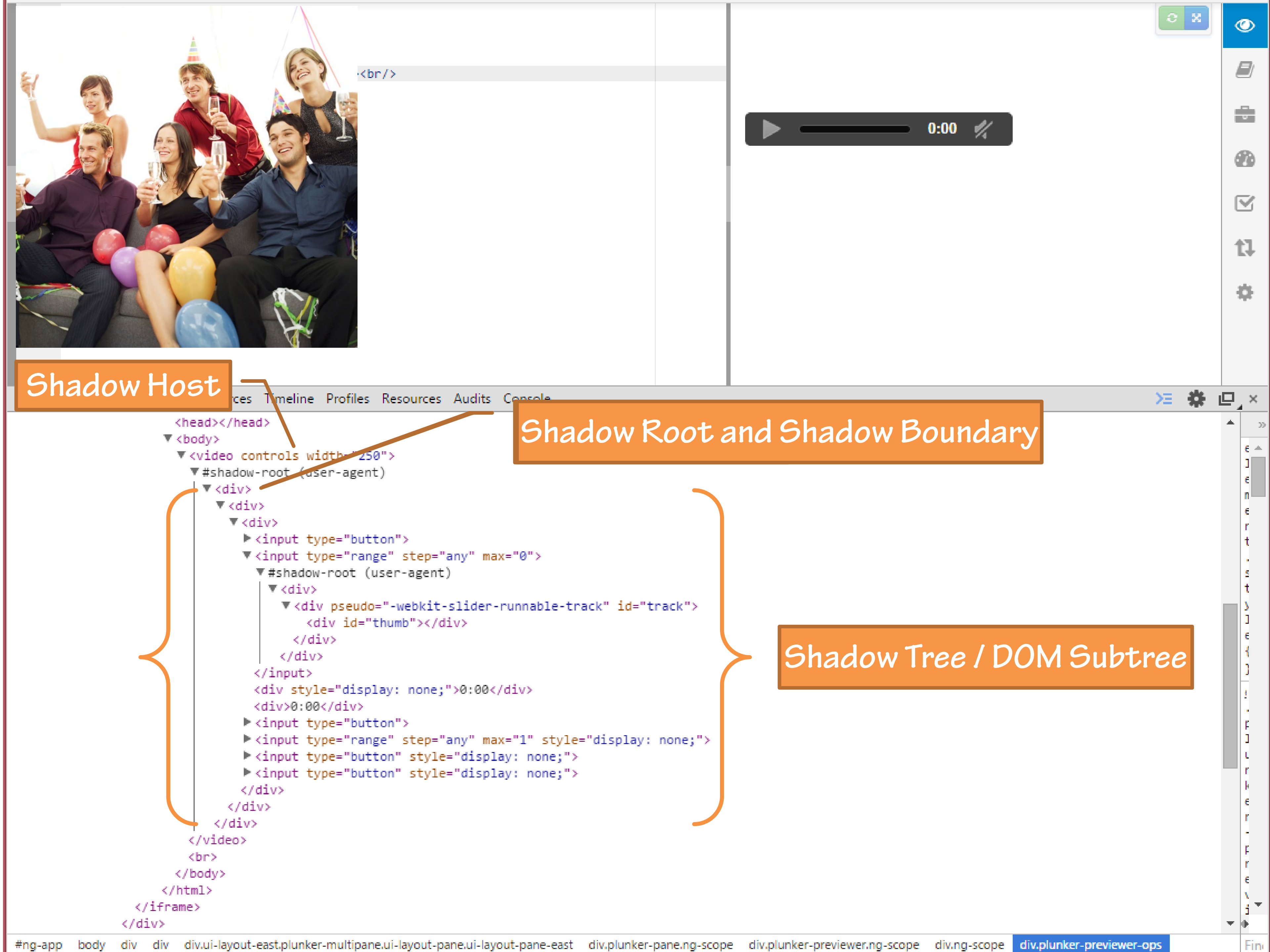
Simpler Selectors

Simpler Markup

Enhanced Readability

Avoids Accidental Styling

```
<style>
  body { color: red }
</style>
```


Shadow Host
Shadow Root and Shadow Boundary
Shadow Tree / DOM Subtree
0:00
ces Timeline Profiles Resources Audits Console
#ng-app body div div div.ui-layout-east.plunker-multipane.ui-layout-pane.ui-layout-pane-east div.plunker-pane.ng-scope div.plunker-previewer.ng-scope div.ng-scope div.plunker-previewer-ops

```
<br/>
```

```
    <head></head>
  ▼<body>
    ▼<video controls width="250">
      ▼#shadow-root (user-agent)
        ▼<div>
          ▼<div>
            ▼<div>
              ▶<input type="button">
              ▼<input type="range" step="any" max="0">
                ▼#shadow-root (user-agent)
                  ▼<div>
                    ▼<div pseudo="-webkit-slider-runnable-track" id="track">
                        <div id="thumb"></div>
                      </div>
                    </div>
                </input>
                <div style="display: none;">0:00</div>
                <div>0:00</div>
              ▶<input type="button">
              ▶<input type="range" step="any" max="1" style="display: none;">
              ▶<input type="button" style="display: none;">
              ▶<input type="button" style="display: none;">
              </div>
            </div>
          </div>
      </video>
      <br>
    </body>
  </html>
</iframe>
</div>
```

## ShadowRoot DOM Methods

getElementById()

getElementsByClassName()

getElementsByTagName()

getElementsByTagNameNS()

querySelector()

querySelectorAll()

ShadowRoot DOM Methods

## Demo

## Is JavaScript Encapsulated in the Shadow DOM?

**Nope.**

**Shadow DOM JavaScript is not encapsulated**

## Demo

## Summary

Light DOM vs Shadow DOM

Shadow DOM provides encapsulation for DOM and styles

Many new terms!

| | |
|---|---|
| Shadow Root | Shadow Host |
| Shadow Boundary | DOM Subtrees |

JavaScript is *not* encapsulated