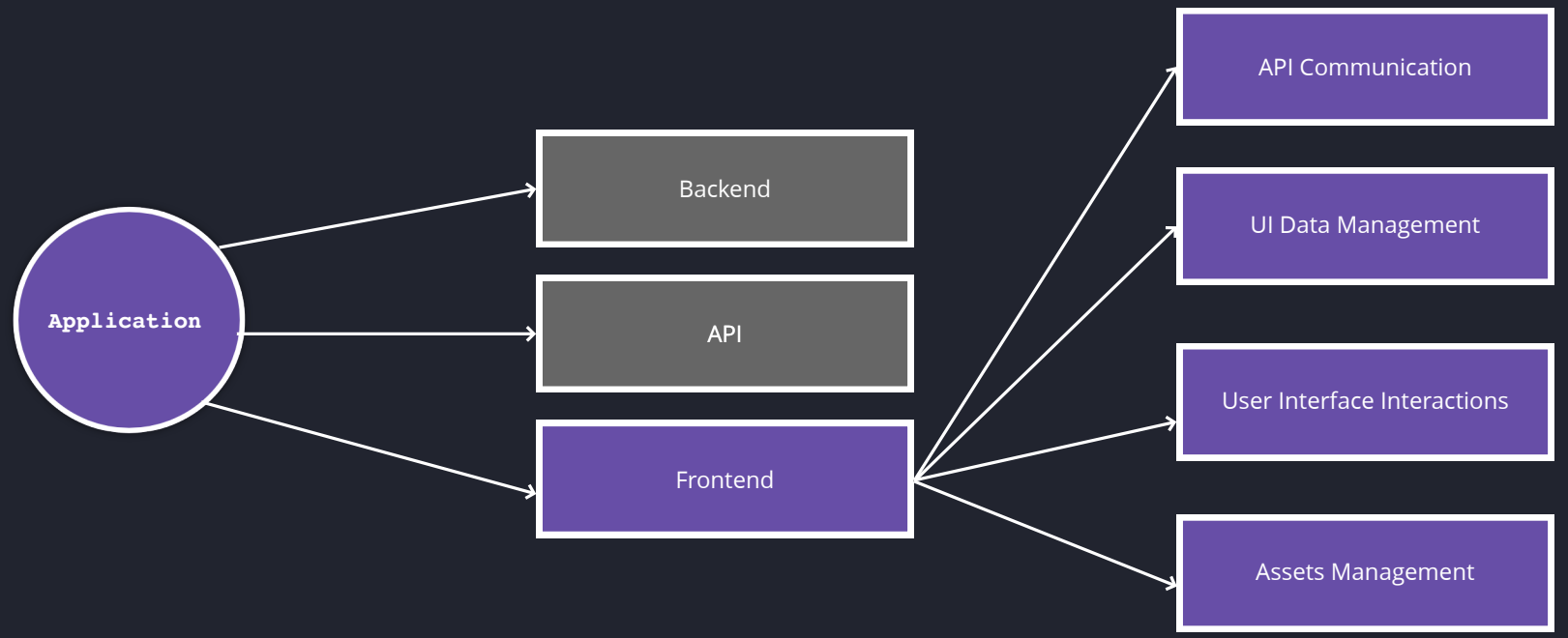
< Frontend System Design />
Fundamentals

Evgenii Ray

Introduction

- Who am I
- Goal of this course

What's Frontend System Design



Course Plan

1. Core Fundamentals

- Box Model
- Positioning System
- Formatting Context
- Stacking Context
- Browser Rendering Cycle and Reflow
- Composition Layers
- GPU Acceleration

2. DOM API

- API Refresher
- Querying methods comparison
- Optimizing query performance
- Coding Assignment #1

3. Web APIs for Complex UI Patterns

- Observer API Overview
- Intersection Observer
- Coding assignment #2
- Mutation Observer
- Coding assignment #3
- Resize Observer
- Coding Assignment #4

4. Virtualisation

- Pattern overview
- Live-coding

5. Application State design

- Search / Access Optimization
- Browser Storage API Overview
- Memory Offloading

6. Network

- Introduction to Browser Networking
- Protocols Overview
- Talking to server
  - Long-polling
  - Web-sockets
  - SSE
- REST / GraphQL

7. Web Application Perfomrance

- Javascript
- CSS
- Images
- Other Assets

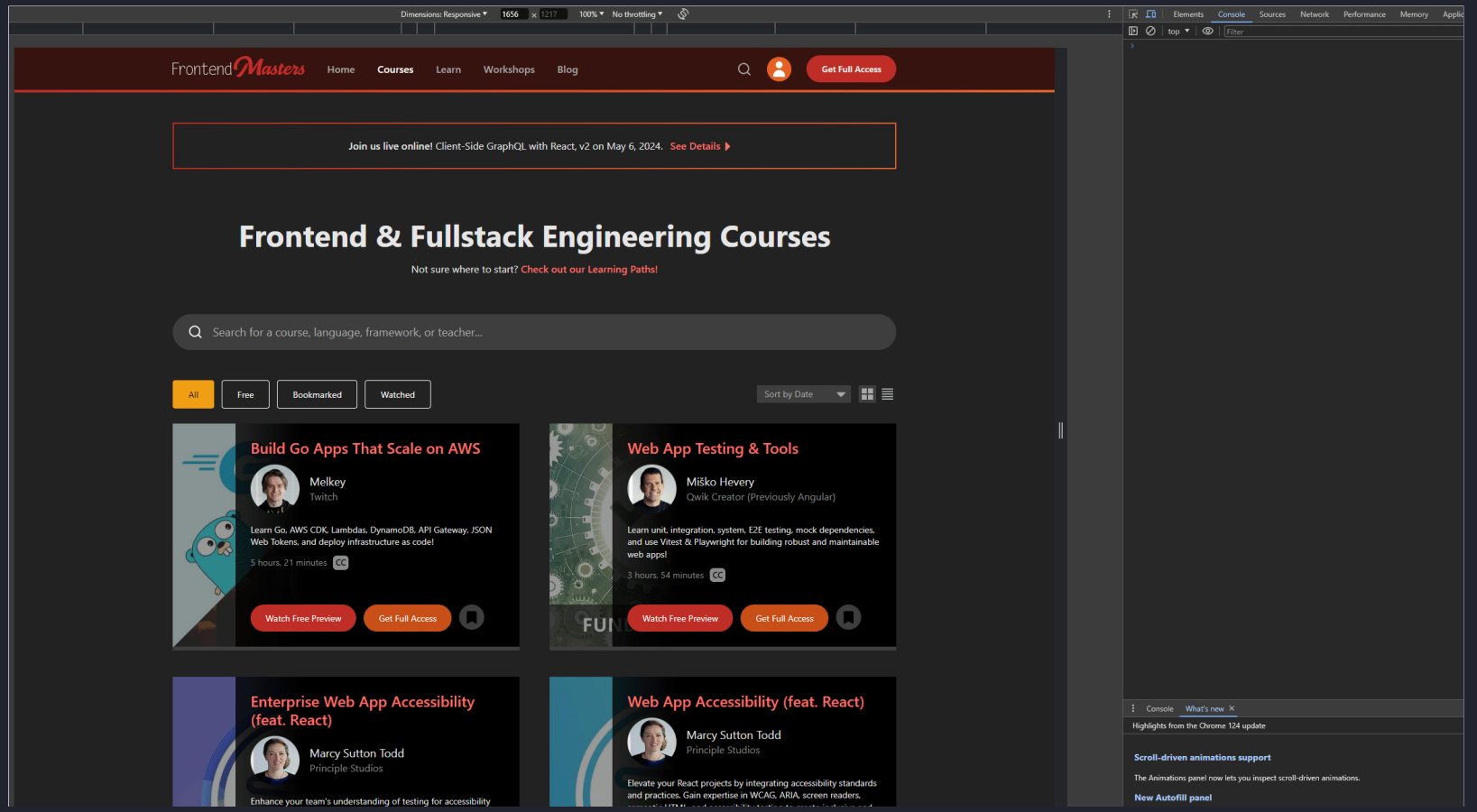
8. Bonus Section

Part 1

< Core Fundamentals />

Box Model

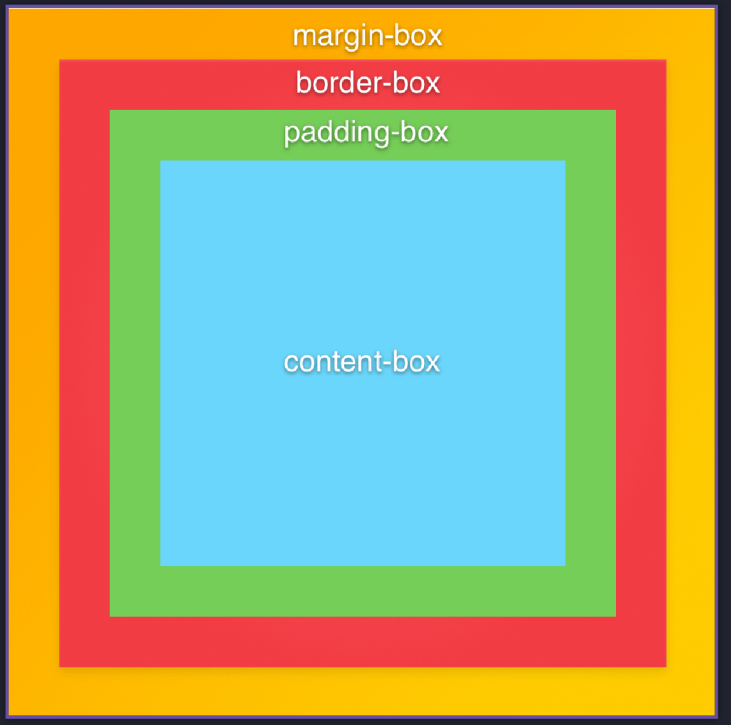
```
const allElements = document.querySelectorAll('*');
allElements.forEach(el => el.style.border = "2px solid green");
```



Box Anatomy

The box model comprises several layers, each serving a distinct purpose in the layout of an element:

- **Content Box:** This is the area that contains the content of the block.
- **Padding Box:** This layer surrounds the content box, providing space aroundthe content.
- **Border Box:** This includes the space of the border that encircles both the padding box and the content box.
- **Margin Box:** This represents the external space outside the element's border.



https://codepen.io/RayEuji/embed/XWGGpJL

Box Properties

Box Size

- Intrinsic
- Restricted

Box Type

- Inline
- Block
- Anonymous

Box Size

The size can be:

1. **Intrinsic:** This means the box uses its content to determine the space it occupies.
2. **Restricted:** This indicates that the box's size is governed by a set of rules applied to it. It can be:
    1. Explicit **`width`** and **`height`** set via CSS.
    2. Constrained by parent elements or other boxes through mechanisms like:
        1. **`flex`** or **`grid`** layout systems.
        2. `Percentage (%)` of parent size.
        3. The **`aspect-ratio`** property of images, etc.
        4. The presence of other children in the DOM tree

https://codepen.io/RayEuji/embed/NWJZLzO

Box type

There are several types of boxes

- `block` level (including, but not restricted by **display: block**)
- `inline` level
- `Anonymous` box

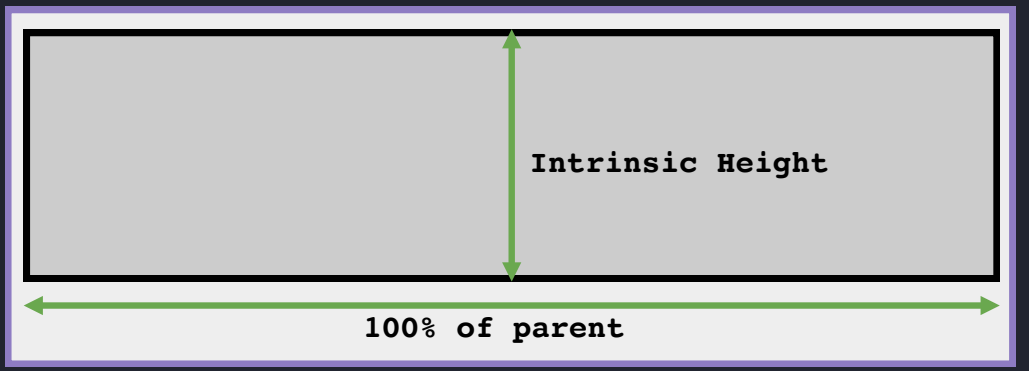
https://codepen.io/RayEuji/embed/yLwdQRj

Box type: Block

1. The element is rendered like a **block**.
    1. Block level element takes **100%** of the **parent** container width
    2. The **height** of the content is equal to the intrinsic size.
2. The element is rendered from top to bottom.
3. Participate in **Block Context Formatting (BCF)**

<address>	<article>	<aside>	<blockquote>	<canvas>
<dd>		<dl>	<dt>	<fieldset>
<figcaption>	<figure>	<footer>	<form>	<h1>-<h6>
<header>	<hr>		<main>	<nav>
<noscript>		<p>	<pre>	<section>
<table>	<tfoot>		<video>	

Block Element

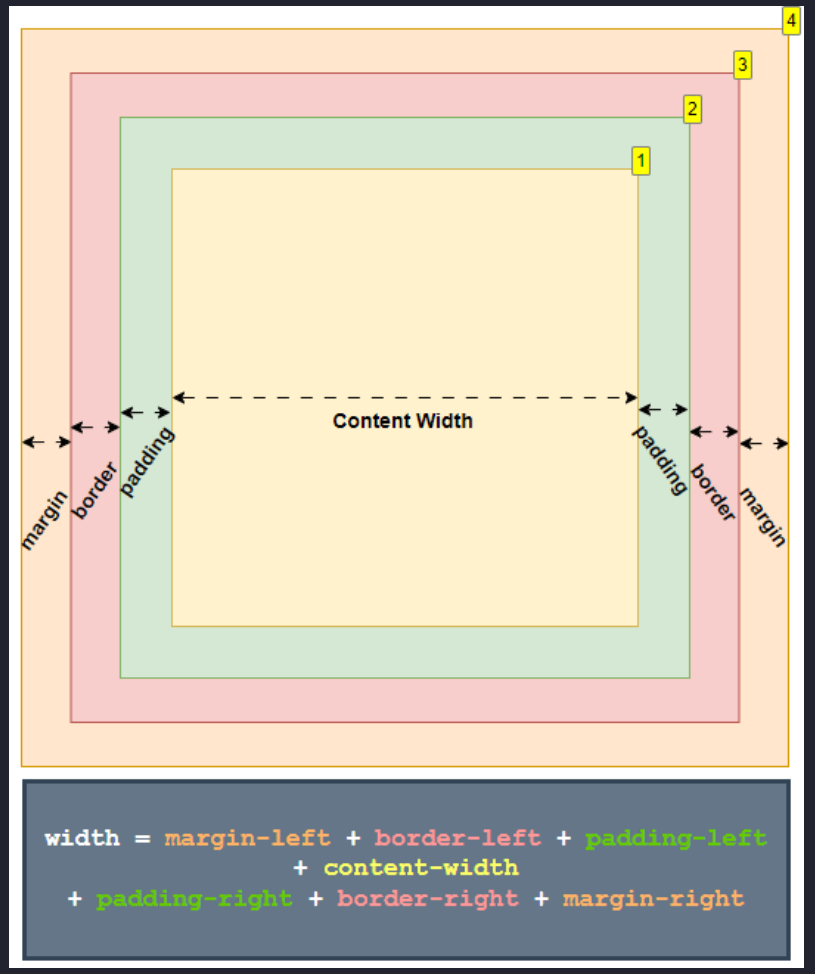


Anonymous box

```
1 <div>
2  <p>
3     This will be wrapped into TextNode
4  </p>
5 </div>
6 ->> This will be an anonymous box -<<
```

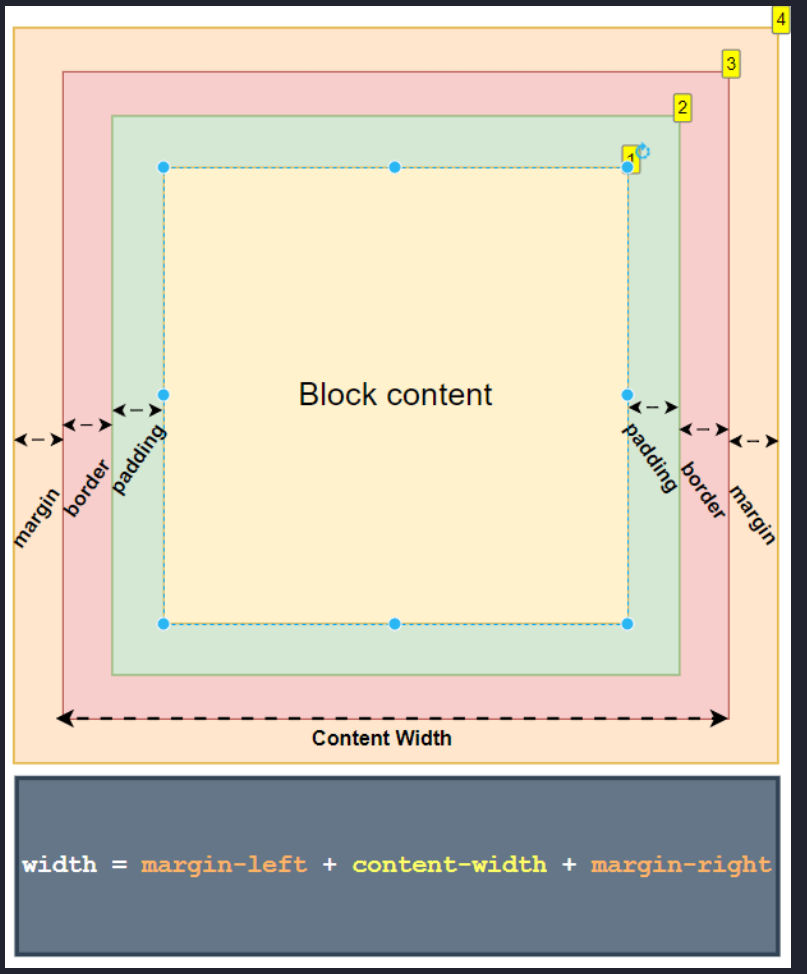
Mathematics of Block Elements

box-sizing: content-box




```
width = margin-left + border-left + padding-left
        + content-width
  + padding-right + border-right + margin-right
```

box-sizing: border-box




```
width = margin-left + content-width + margin-right
```

Inline Elements

The key characteristics of inline elements are:

1. They render as a **string**, flowing from **left to right** and from **top to bottom**.
2. They participate in an **Inline Formatting Context (IFC)**
3. They generate **inline-level** boxes.

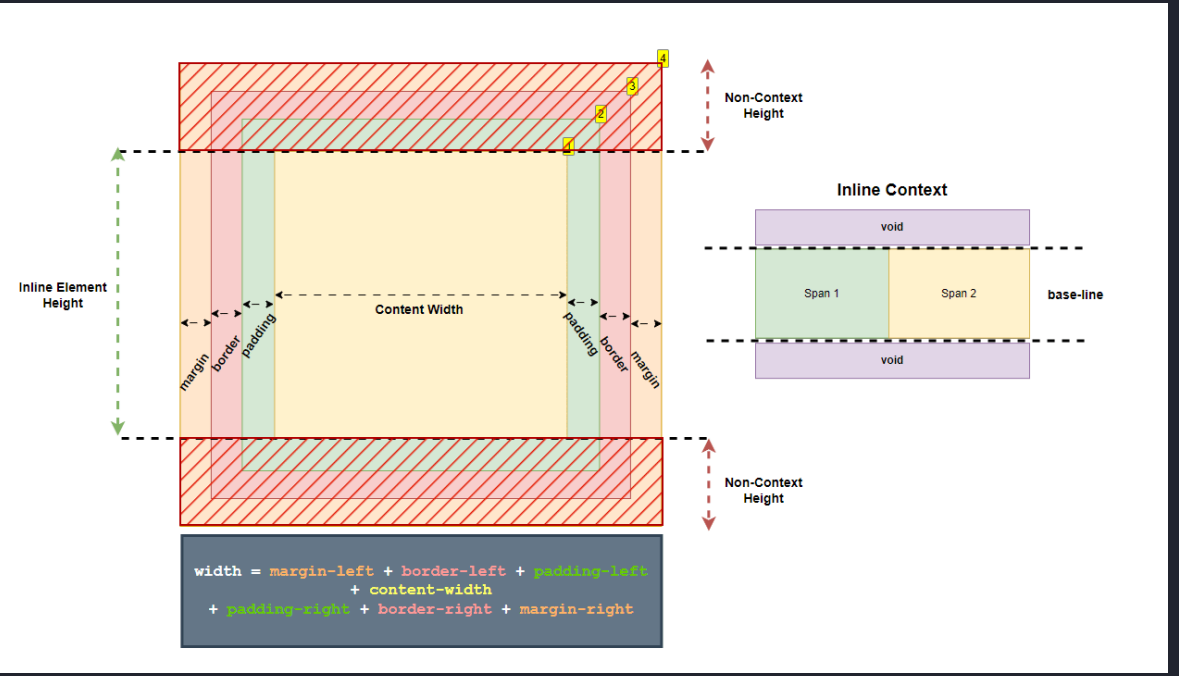
```
<div>
  Some text inside the  block element
  <span>
     This is inside inline context
  </span>
</div>
```

<a>	<abbr>	<acronym>
 	<button>	<cite>
<i>	<img>	<input>
<object>	<output>	<q>
<small>		
<time>	<tt>	<var>

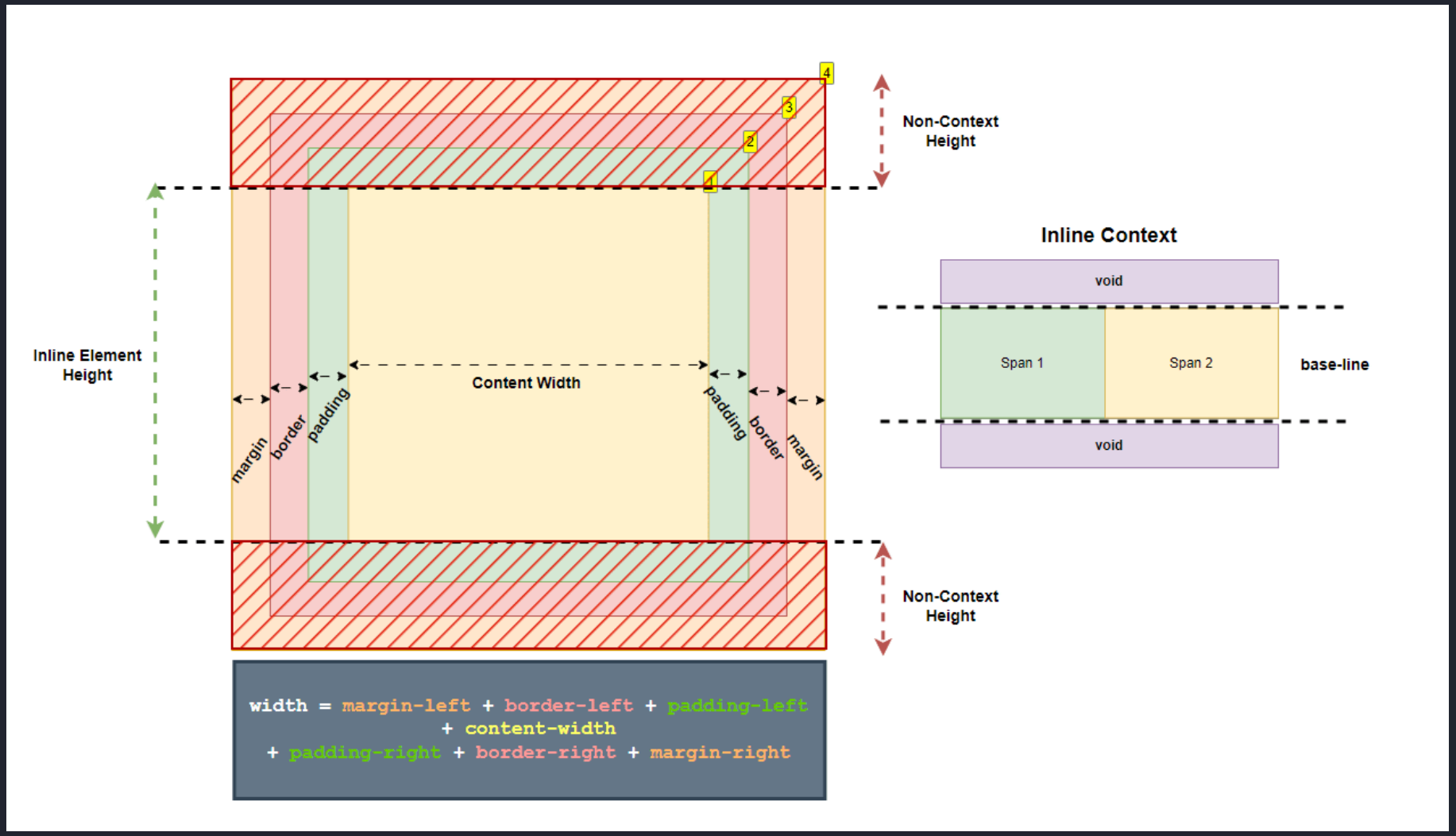
<b>	<bdo>	<big>
<code>	<dfn>	
<kbd>	<label>	<map>
<samp>	<script>	<select>
<sub>	<sup>	<textarea>

Mathematics of Inline Elements

1. Does not respond to **`width`** and **`height`** properties.
    1. Completely ignores them.
2. Does not react to **`vertical margins`**.
    1. Ignores them.
3. Inline **`padding`** does not alter the **`height`** of the inline element.

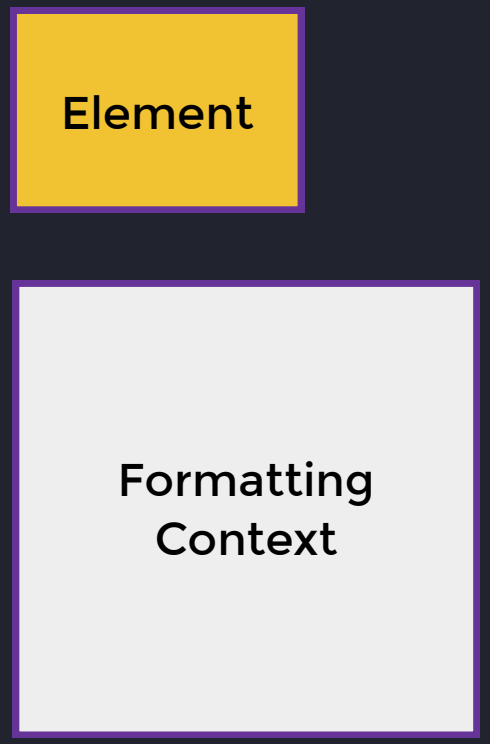


Mathematics of Inline Elements



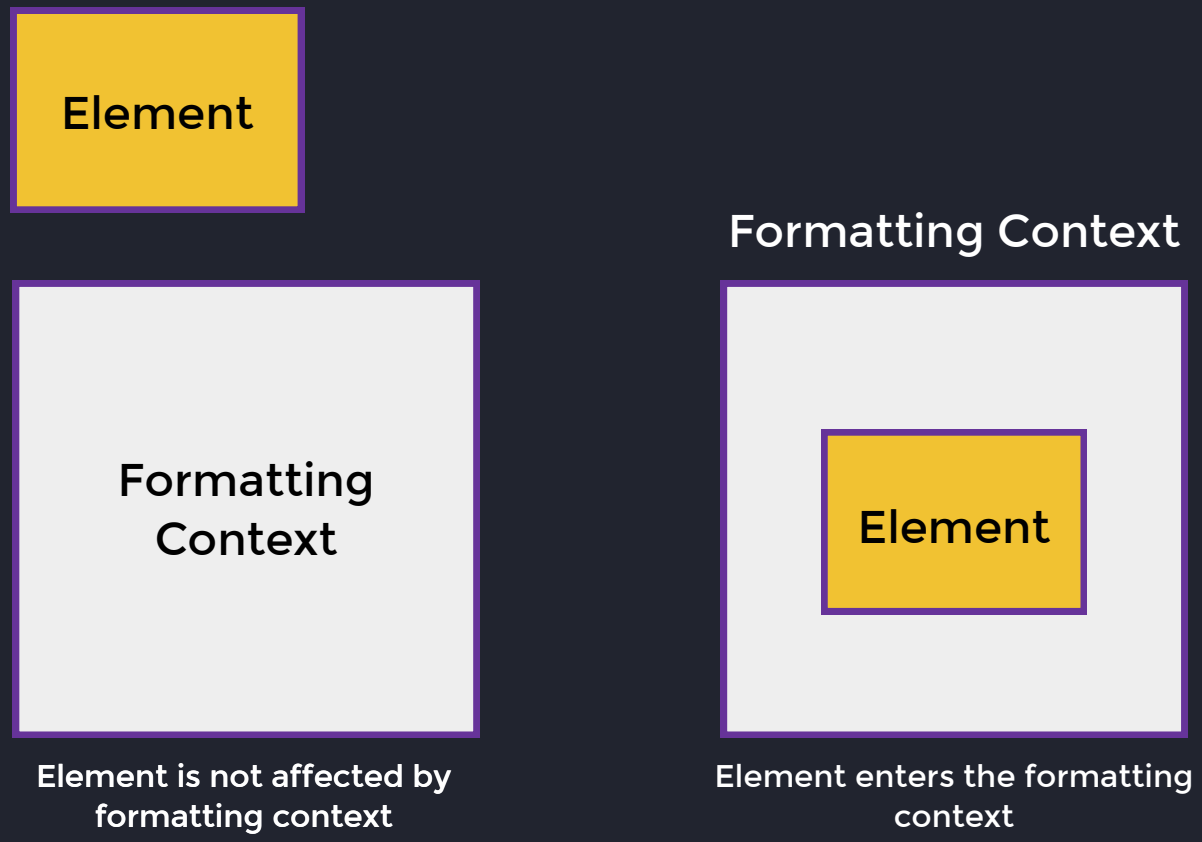
Browser Formatting Context

Browser Formatting Context - 1



Element is not affected by formatting context

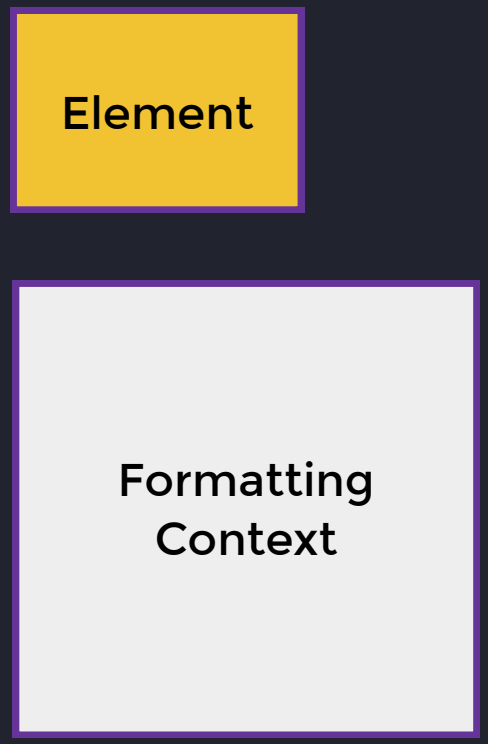
Browser Formatting Context - 1



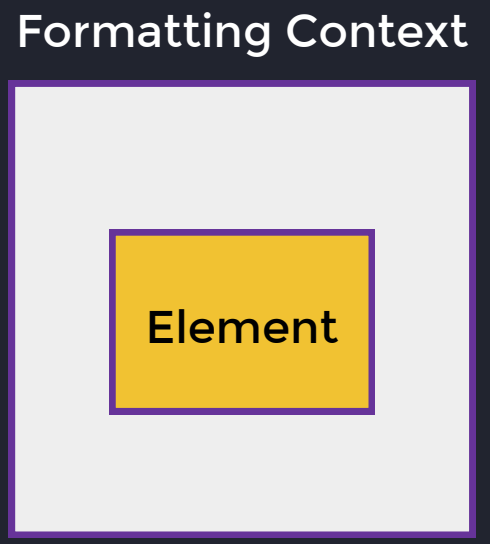
Element is not affected by formatting context

Element enters the formatting context

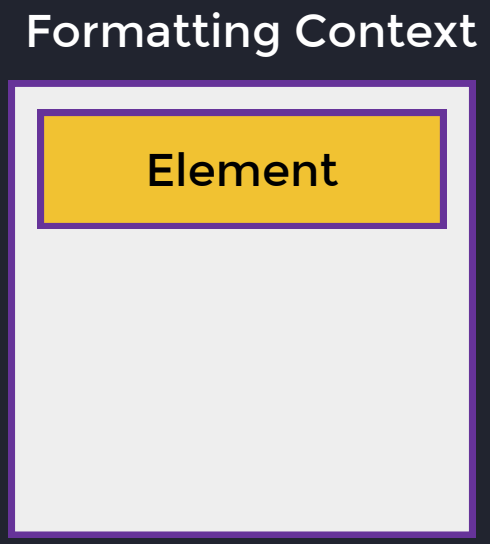
Browser Formatting Context - 1



Element is not affected by formatting context

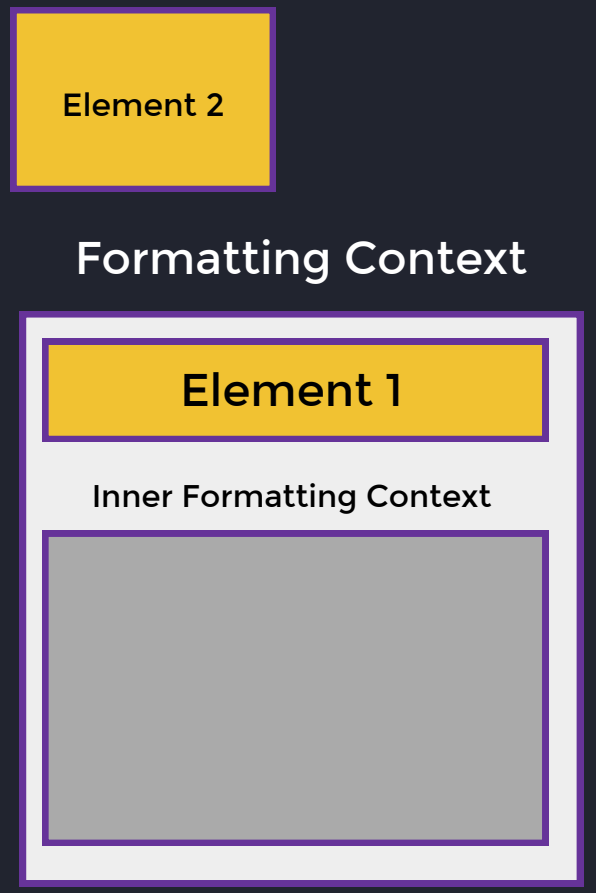


Element enters the formatting context



Now, element has to span across the whole container since it's a rule of formatting context

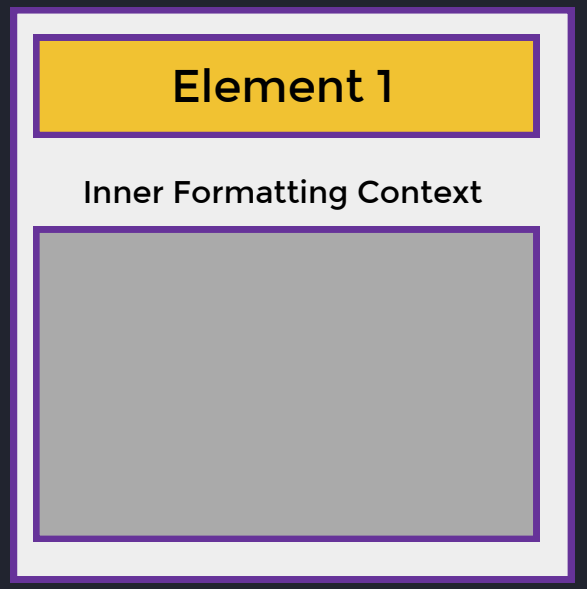
Browser Formatting Context - 2



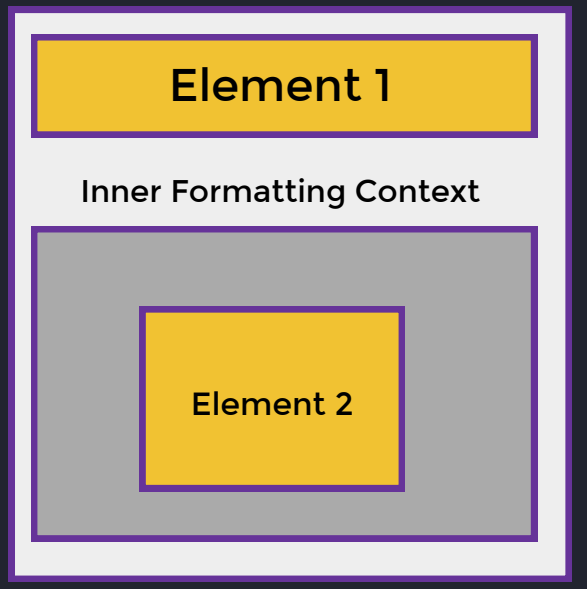
Browser Formatting Context - 2

Element 2

Formatting Context



Formatting Context



Browser Formatting Context - 2

Element 2

Formatting Context

Element 1

Inner Formatting Context

Formatting Context

Element 1

Inner Formatting Context

Element 2

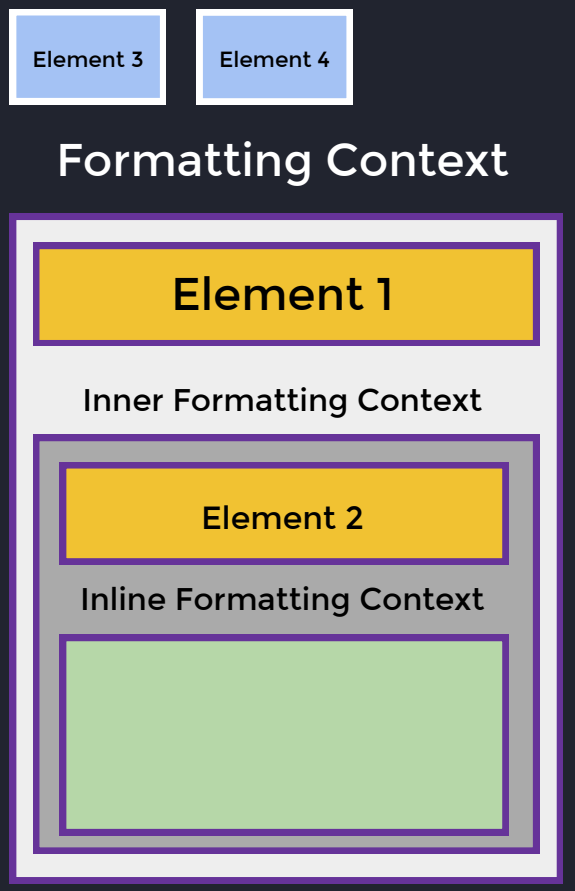
Formatting Context

Element 1

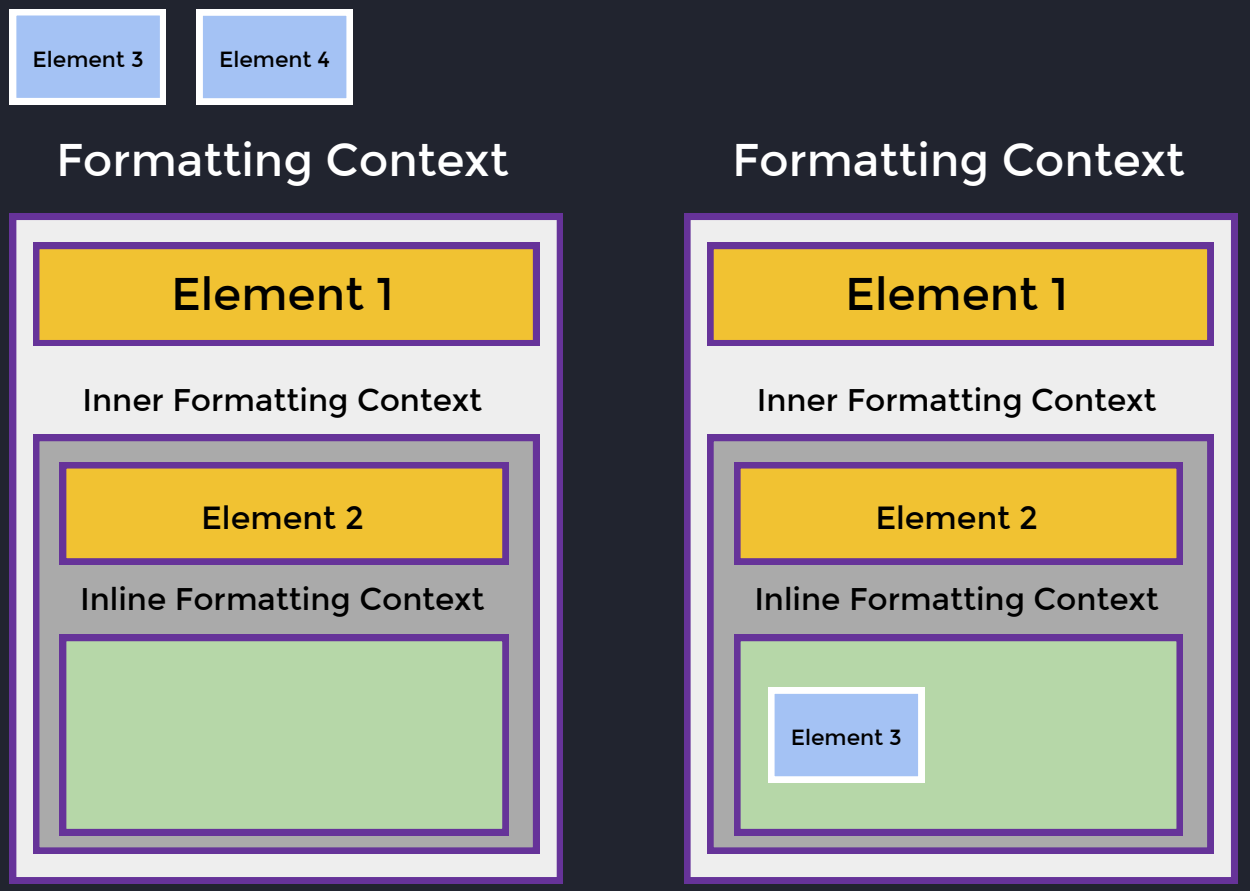
Inner Formatting Context

Element 2

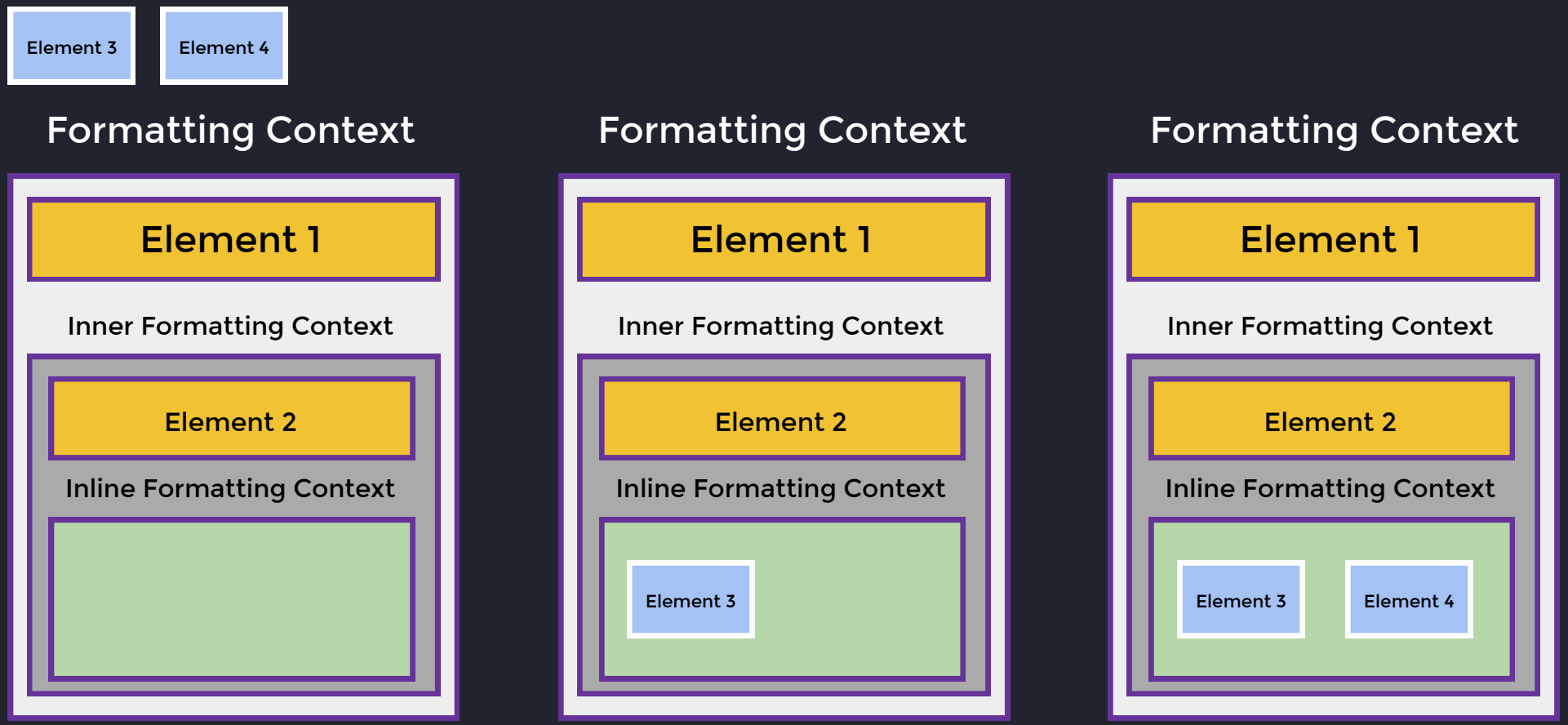
Browser Formatting Context - 3



Browser Formatting Context - 3



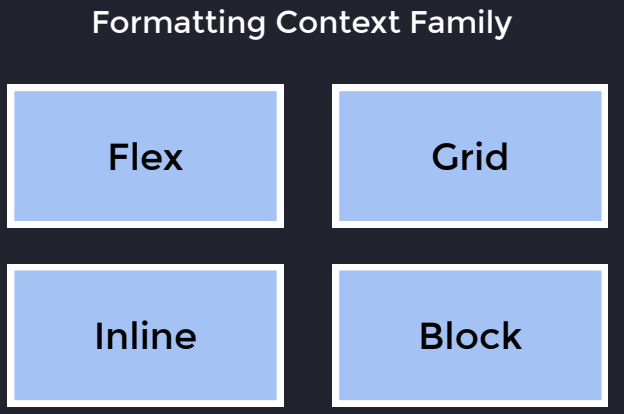
Browser Formatting Context - 3



Formatting Context - Key Ideas

The Key Ideas of Formatting Contexts:

- **Isolation:** Elements within a context are shielded from the rules of external contexts.
- **Scalability:** Introducing a new ruleset for elements is as simple as creating a new **Context** (examples: flex-box, grid).
- **Predictability:** With a strict set of rules, the placement of elements is predictable.



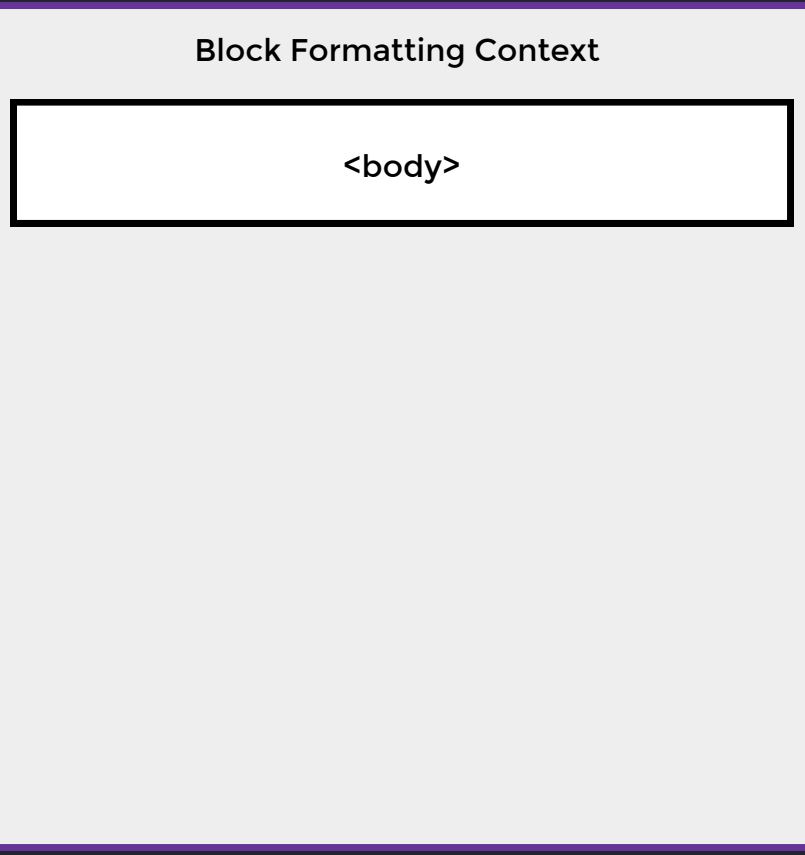
Time to become a browser

```
<html>
<body>
<h1>This is a Heading</h1>
<ul>
 <li style="display:inline-block">
    <p>Paragraph 1</p>
  <p>Paragraph 2</p>
 </li>
 <li style="display:inline-block">
    <p>Paragraph 3</p>
    <p>Paragraph 4</p>
 </li>
 <section style="display: flex; flex-direction: row">
   <p>Paragraph 5</p>
   <p>
     <span>Paragraph 6</span>
   </p>
 </section>
</ul>
</body>
</html>
```

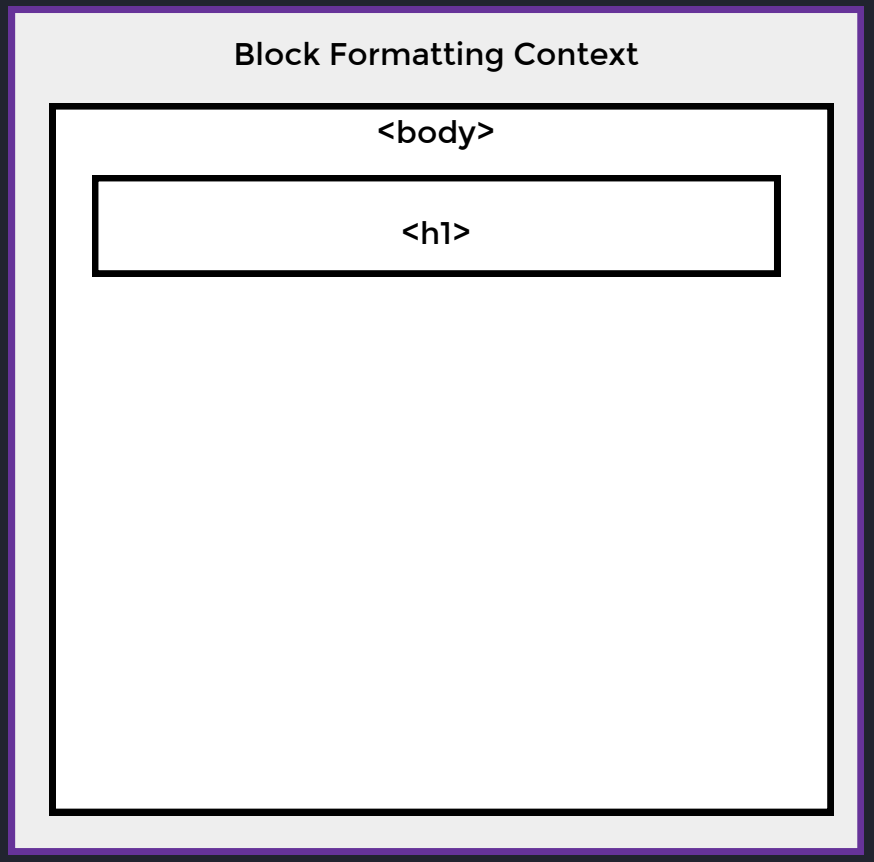
Block Formatting Context

```
<html>
<body>
<h1>This is a Heading</h1>
<ul>
 <li style="display:inline-block">
    <p>Paragraph 1</p>
  <p>Paragraph 2</p>
 </li>
 <li style="display:inline-block">
    <p>Paragraph 3</p>
    <p>Paragraph 4</p>
 </li>
 <section style="display: flex; flex-direction: row">
   <p>Paragraph 5</p>
   <p>
     <span>Paragraph 6</span>
   </p>
 </section>
</ul>
</body>
</html>
```

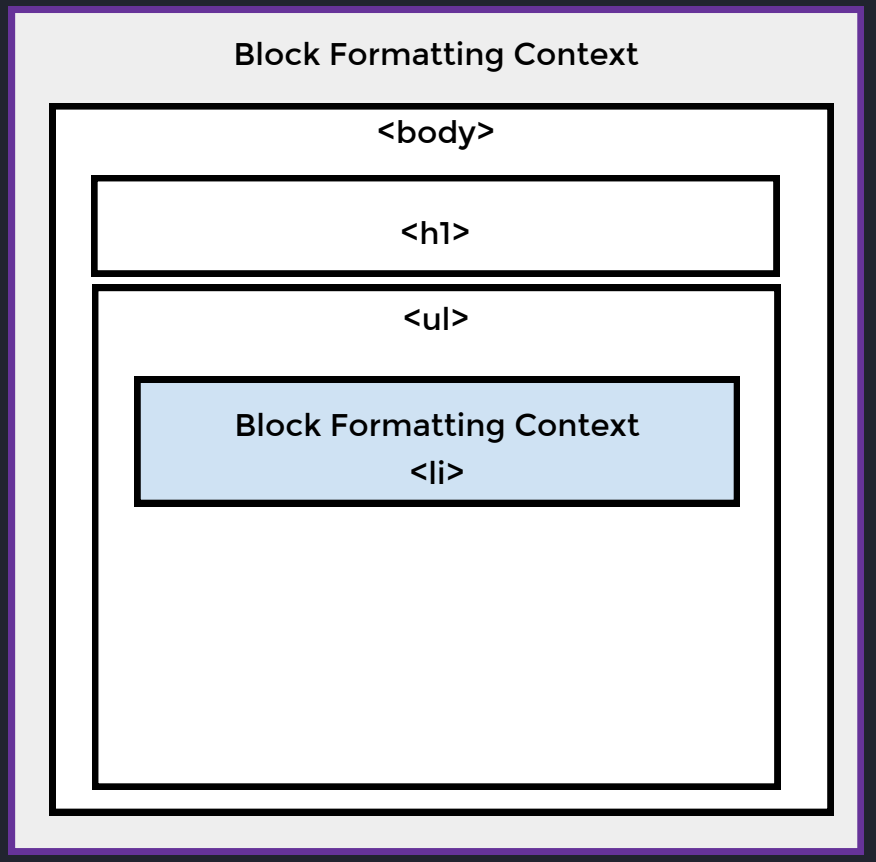
Block Formatting Context



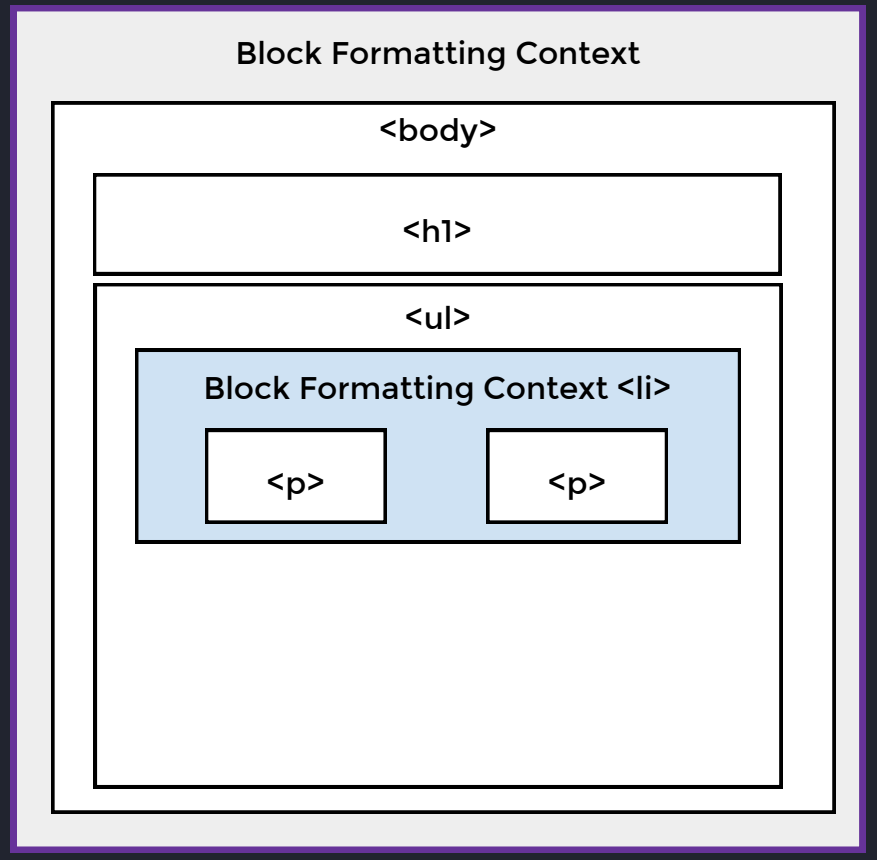

```
<html>
<body>
<h1>This is a Heading</h1>
<ul>
 <li style="display:inline-block">
    <p>Paragraph 1</p>
  <p>Paragraph 2</p>
 </li>
 <li style="display:inline-block">
    <p>Paragraph 3</p>
    <p>Paragraph 4</p>
 </li>
 <section style="display: flex; flex-direction: row">
   <p>Paragraph 5</p>
   <p>
     <span>Paragraph 6</span>
   </p>
 </section>
</ul>
</body>
</html>
```



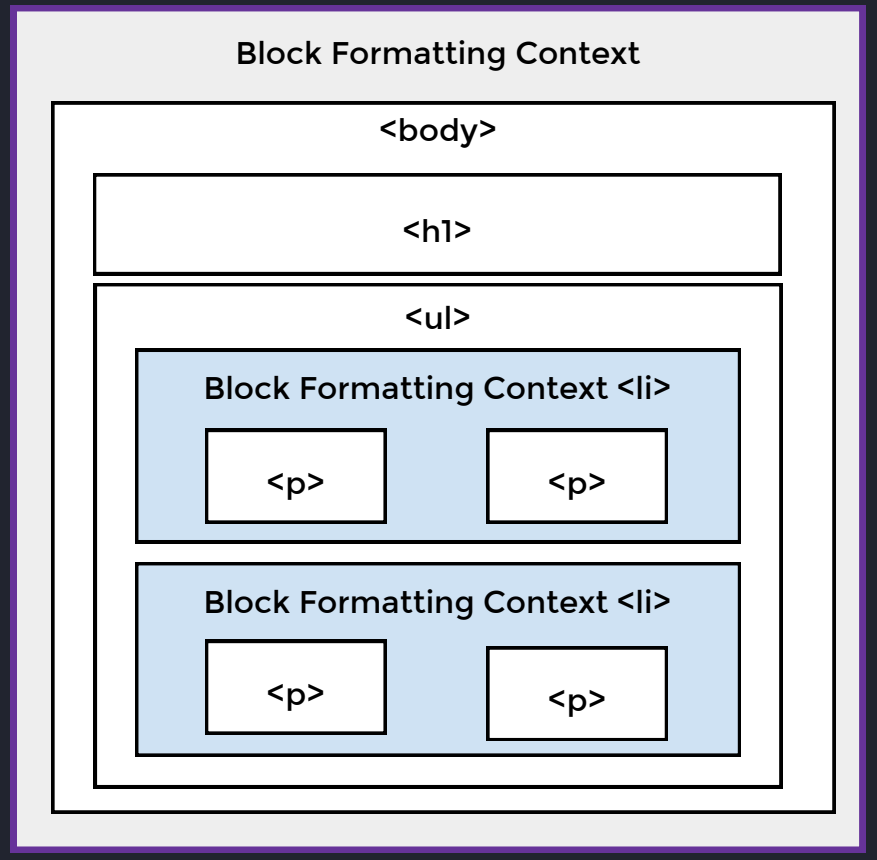

```
<html>
<body>
<h1>This is a Heading</h1>
<ul>
 <li style="display:inline-block">
    <p>Paragraph 1</p>
  <p>Paragraph 2</p>
 </li>
 <li style="display:inline-block">
    <p>Paragraph 3</p>
    <p>Paragraph 4</p>
 </li>
 <section style="display: flex; flex-direction: row">
   <p>Paragraph 5</p>
   <p>
     <span>Paragraph 6</span>
   </p>
 </section>
</ul>
</body>
</html>
```



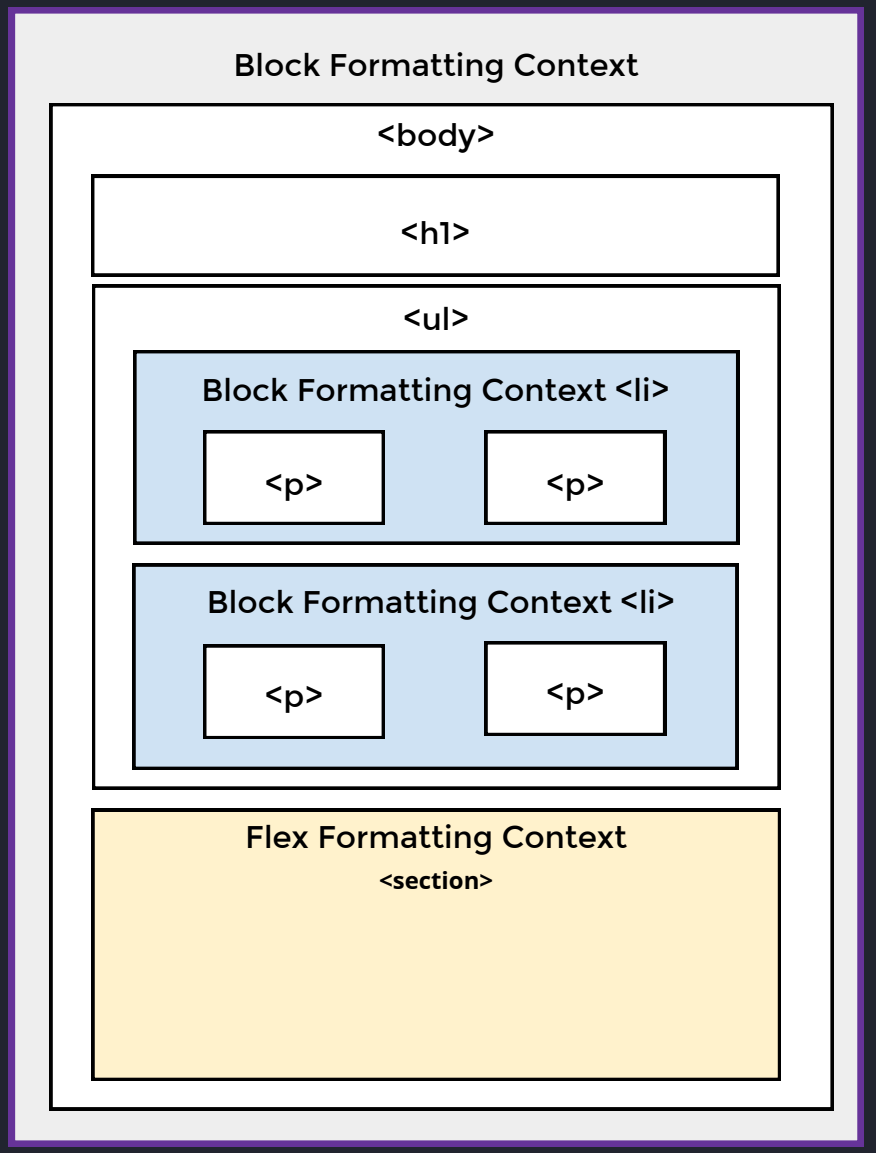

```
<html>
<body>
<h1>This is a Heading</h1>
<ul>
 <li style="display:inline-block">
    <p>Paragraph 1</p>
  <p>Paragraph 2</p>
 </li>
 <li style="display:inline-block">
    <p>Paragraph 3</p>
    <p>Paragraph 4</p>
 </li>
 <section style="display: flex; flex-direction: row">
   <p>Paragraph 5</p>
   <p>
     <span>Paragraph 6</span>
   </p>
 </section>
</ul>
</body>
</html>
```



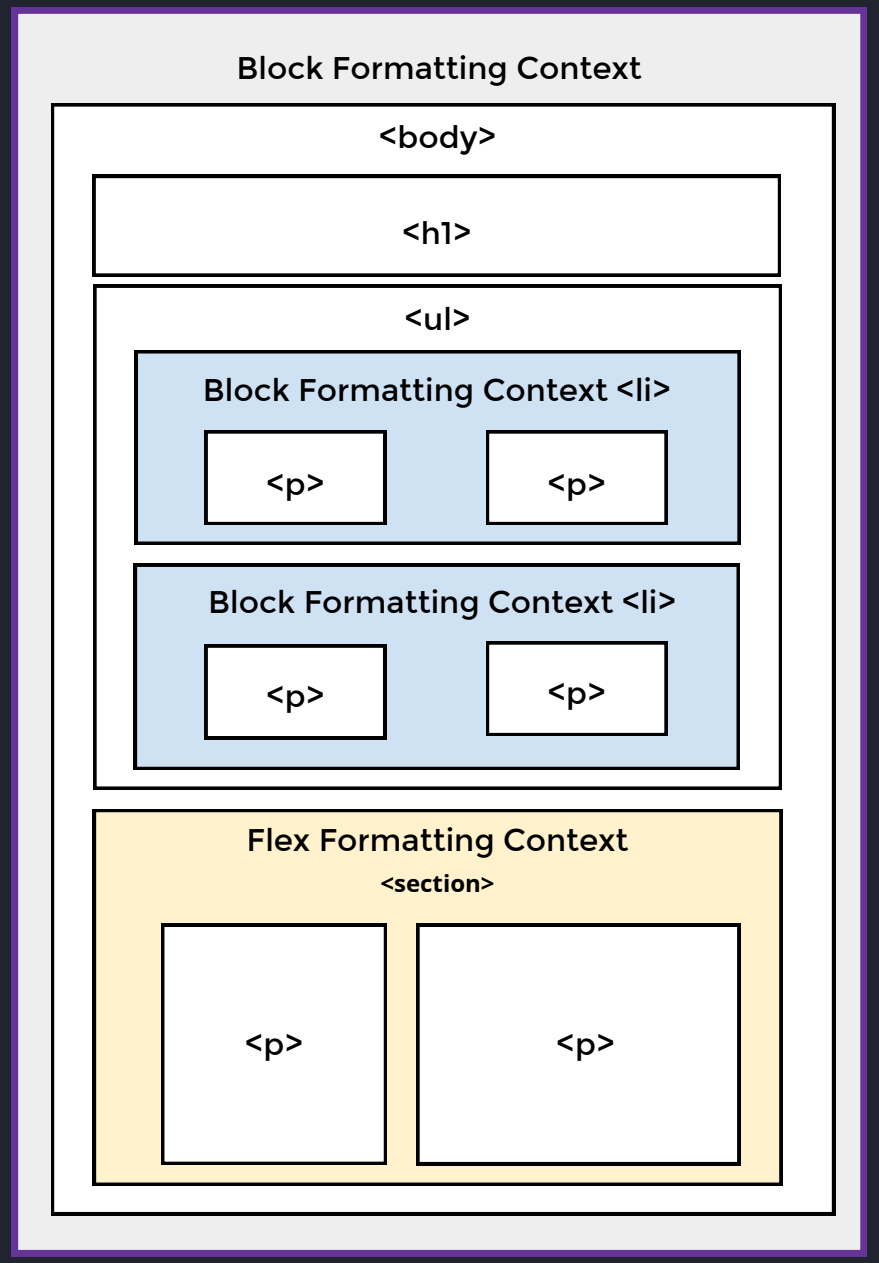

```
1  <html>
2  <body>
3  <h1>This is a Heading</h1>
4  <ul>
5   <li style="display:inline-block">
6      <p>Paragraph 1</p>
7    <p>Paragraph 2</p>
8   </li>
9   <li style="display:inline-block">
10     <p>Paragraph 3</p>
11     <p>Paragraph 4</p>
12  </li>
13  <section style="display: flex; flex-direction: row">
14    <p>Paragraph 5</p>
15    <p>
16      <span>Paragraph 6</span>
17    </p>
18  </section>
19 </ul>
20 </body>
21 </html>
```



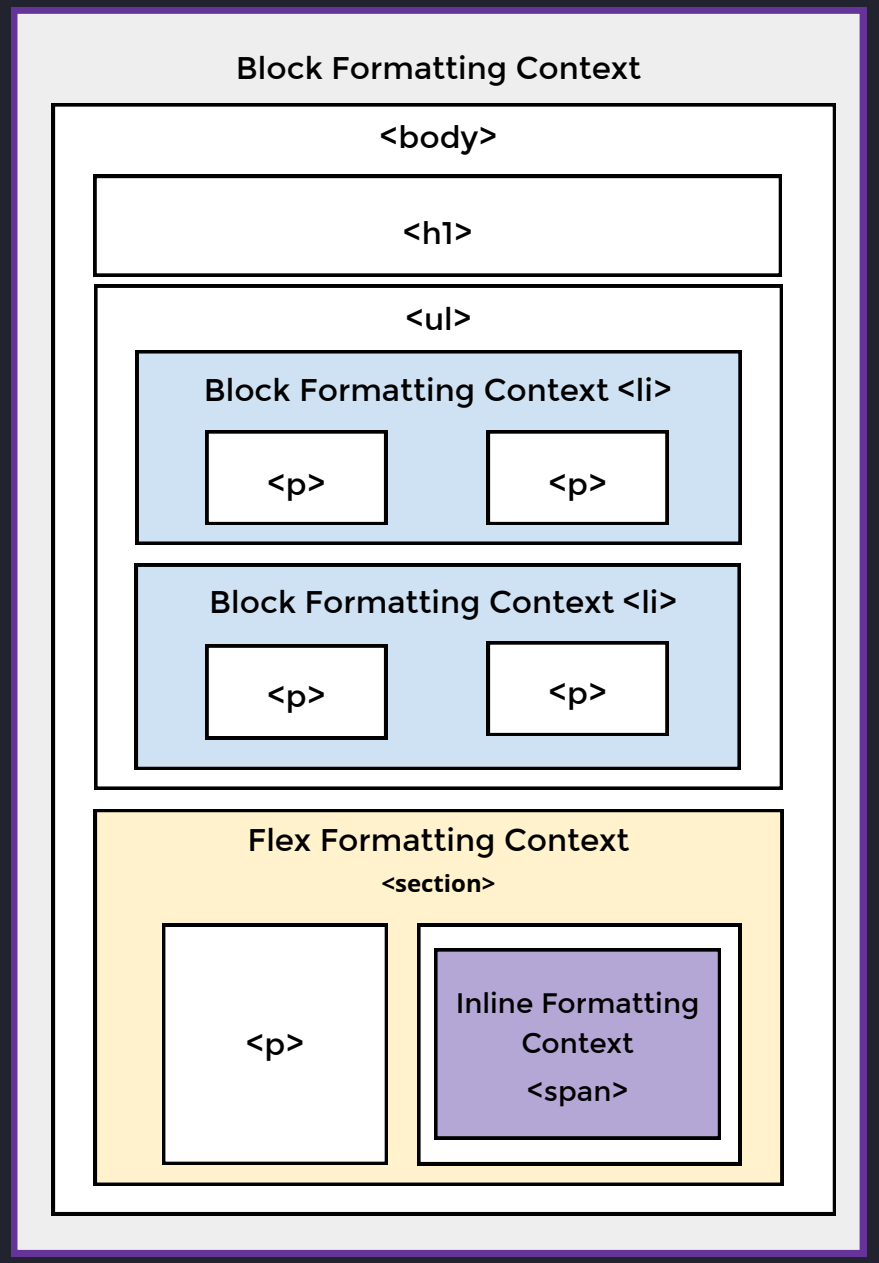

```
<html>
<body>
<h1>This is a Heading</h1>
<ul>
 <li style="display:inline-block">
    <p>Paragraph 1</p>
  <p>Paragraph 2</p>
 </li>
 <li style="display:inline-block">
    <p>Paragraph 3</p>
    <p>Paragraph 4</p>
 </li>
 <section style="display: flex; flex-direction: row">
   <p>Paragraph 5</p>
   <p>
     <span>Paragraph 6</span>
   </p>
 </section>
</ul>
</body>
</html>
```




```html
<html>
<body>
<h1>This is a Heading</h1>
<ul>
 <li style="display:inline-block">
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
 </li>
 <li style="display:inline-block">
  <p>Paragraph 3</p>
  <p>Paragraph 4</p>
 </li>
 <section style="display: flex; flex-direction: row">
  <p>Paragraph 5</p>
  <p>
    <span>Paragraph 6</span>
  </p>
 </section>
</ul>
</body>
</html>
```



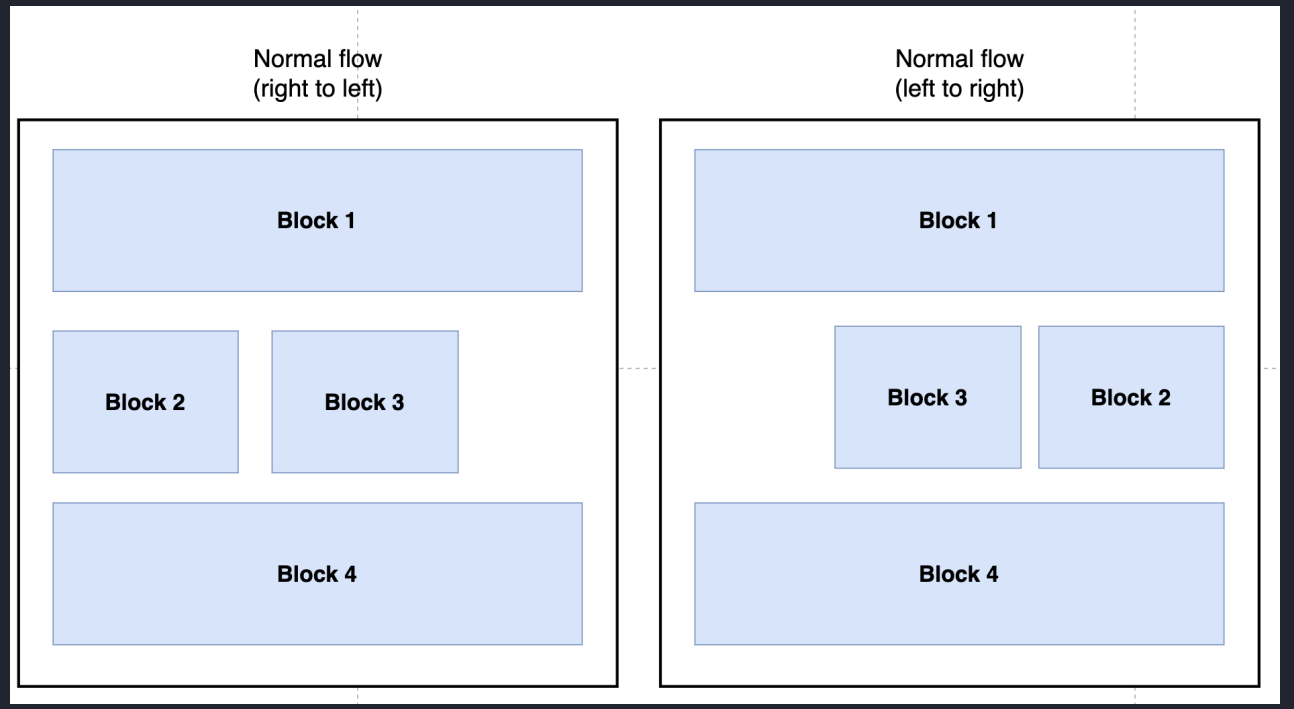

```html
<html>
<body>
<h1>This is a Heading</h1>
<ul>
 <li style="display:inline-block">
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
 </li>
 <li style="display:inline-block">
  <p>Paragraph 3</p>
  <p>Paragraph 4</p>
 </li>
 <section style="display: flex; flex-direction: row">
  <p>Paragraph 5</p>
  <p>
    <span>Paragraph 6</span>
  </p>
 </section>
</ul>
</body>
</html>
```

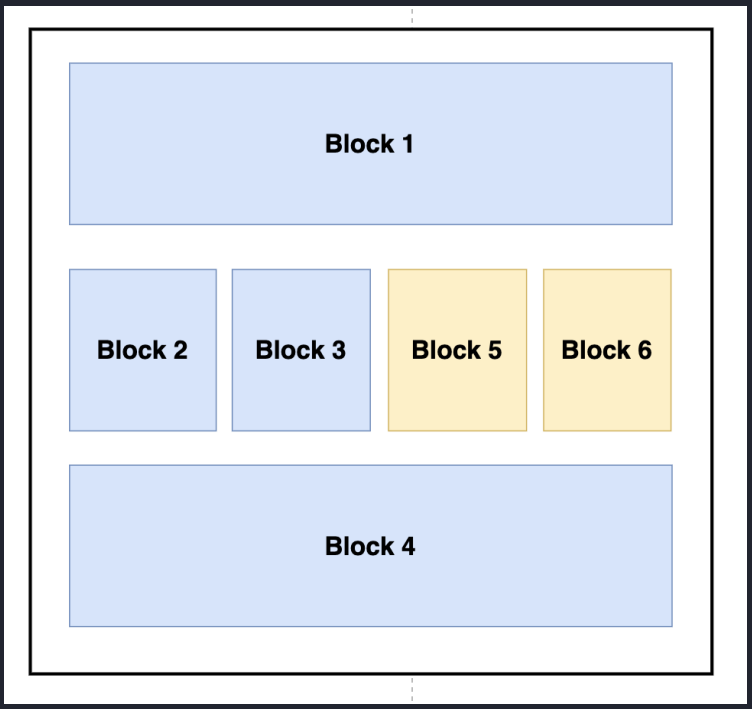



```html
<html>
<body>
<h1>This is a Heading</h1>
<ul>
 <li style="display:inline-block">
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
 </li>
 <li style="display:inline-block">
  <p>Paragraph 3</p>
  <p>Paragraph 4</p>
 </li>
 <section style="display: flex; flex-direction: row">
  <p>Paragraph 5</p>
  <p>
    <span>Paragraph 6</span>
  </p>
 </section>
</ul>
</body>
</html>
```

Browser Positioning System

Normal Flow


Normal flow
(right to left)
Block 1
Block 2
Block 3
Block 4
Normal flow
(left to right)
Block 1
Block 3
Block 2
Block 4


Block 1
Block 2
Block 3
Block 5
Block 6
Block 4

Changing Normal Flow

```
position: static | relative | absolute | sticky | fixed
```

The property `position` determines the variant of element positioning on the page, relative to the browser window or other elements on the page.

Most commonly used: `static`, `relative`, `absolute`

Position: static

```html
<main>
  <div class="box">
    Box Block
  </div>
  <div class="box inline">
    Box Inline
  </div>
  <div class="box inline">
    Box Inline
  </div>
  <div class="box">
    Box Block
  </div>
</main>
```

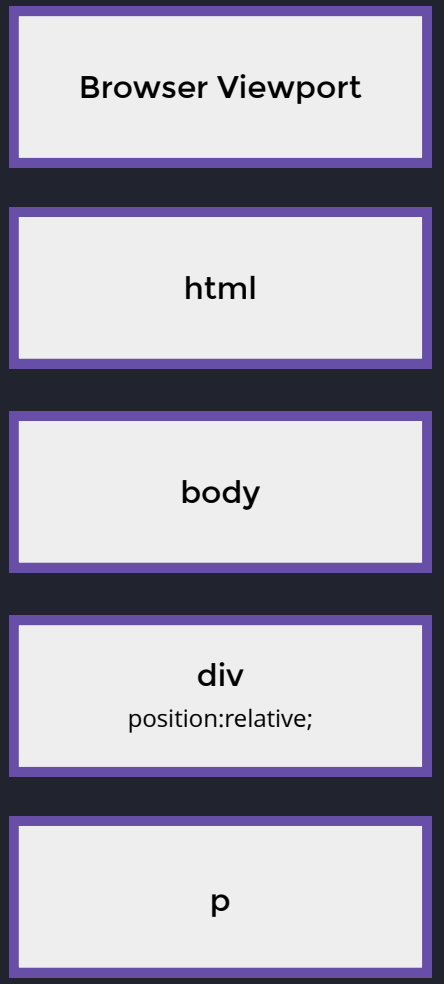
https://codepen.io/RayEuji/embed/WNWrWzV

Position: `relative`

1. The element is positioned according to the **normal flow** of the document
   - **offset** applied relative to itself based on the values of **`top`**, **`right`**, **`bottom`**, and **`left`**.
2. The **offset** does not affect the position of any other elements; thus, the space allocated for the element in the page layout remains the same as if the position were **`static`**.
3. This value creates a new **`stacking context`** when the value of **`z-index`** is not **`auto`**.

https://codepen.io/RayEuji/embed/gOyPyQq

Containing block



Containing Block of `<html>`

Rule: Viewport is always a containing block for the root html element

Containing Block of `<div>`

Rule: if the element has `position: relative`, its closest block-level ancestor element is the containing block

Containing Block of `<p>`

Rule: if the element has `position: relative`, it becomes a containing block

Position: absolute

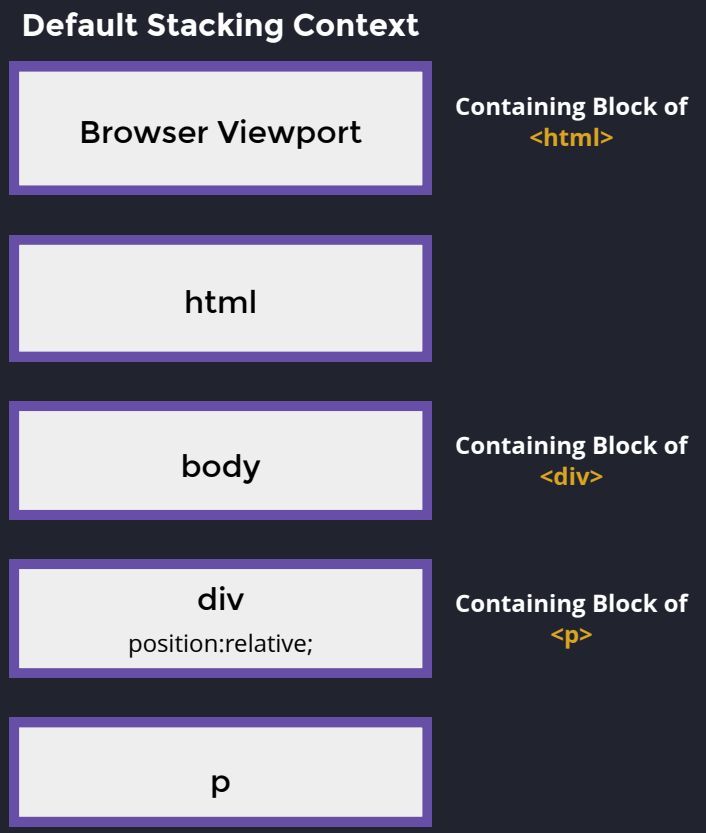
- The **element** is removed from the **normal document flow**

- no space is reserved for the element in the page layout.

- It is **positioned relative** to its closest **positioned ancestor** if one exists; otherwise, it is placed relative to the **initial containing block**.

- Final position is determined by the values of **`top`**, **`right`**, **`bottom`**, and **`left`**.

- This positioning creates a new **stacking context** when the **z-index** value is not **`auto`**.



Position: absolute

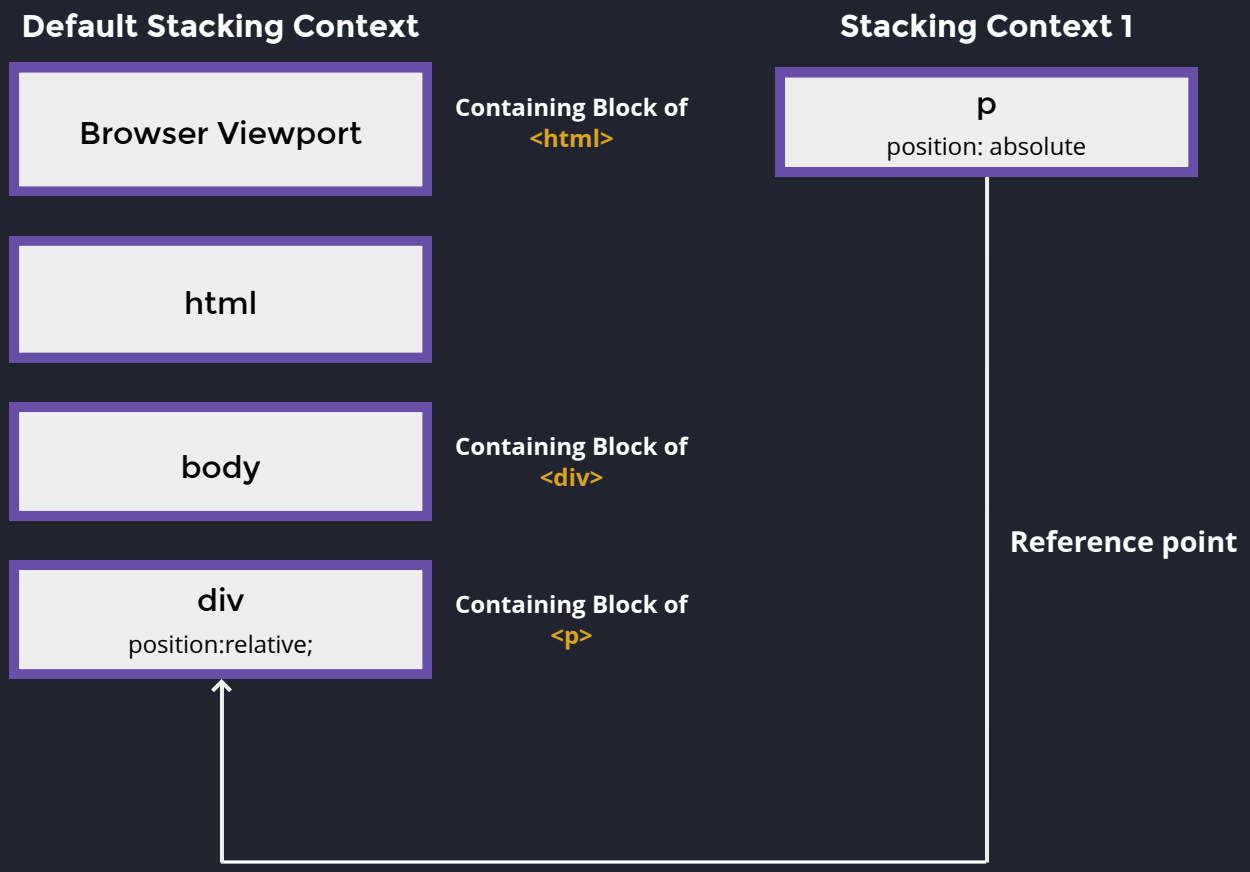
- The **element** is removed from the **normal document flow**

- no space is reserved for the element in the page layout.

- It is **positioned relative** to its closest **positioned ancestor** if one exists; otherwise, it is placed relative to the **initial containing block**.

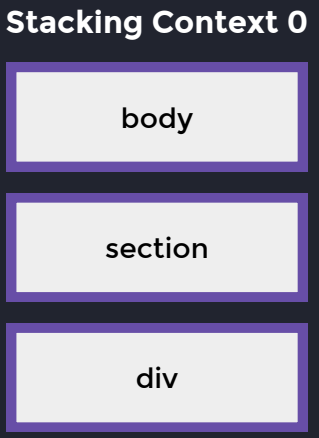
- Final position is determined by the values of **`top`**, **`right`**, **`bottom`**, and **`left`**.

- This positioning creates a new **stacking context** when the **z-index** value is not **`auto`**.



Position: absolute

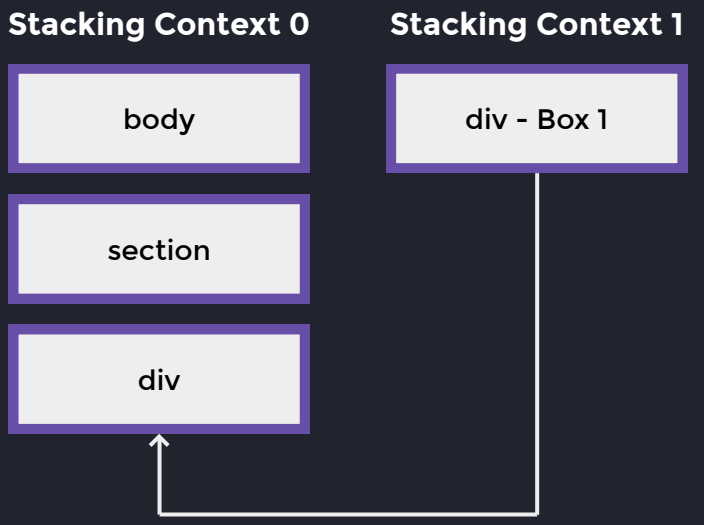
```
<body>
 <section class="container">
  <div style="position: relative; height: 150px;">
    <div class="box red absolute">Box 1</div>
    <div class="box blue absolute">Box 2</div>
  </div>
 </section>
</body>
```



Position: absolute

```html
<section class="container">
  <div style="position: relative; height: 150px;">
    <div class="box red absolute">Box 1</div>
    <div class="box blue absolute">Box 2</div>
  </div>
</section>
```

- **Box 1** was removed from the normal flow and placed in a separate stacking context.



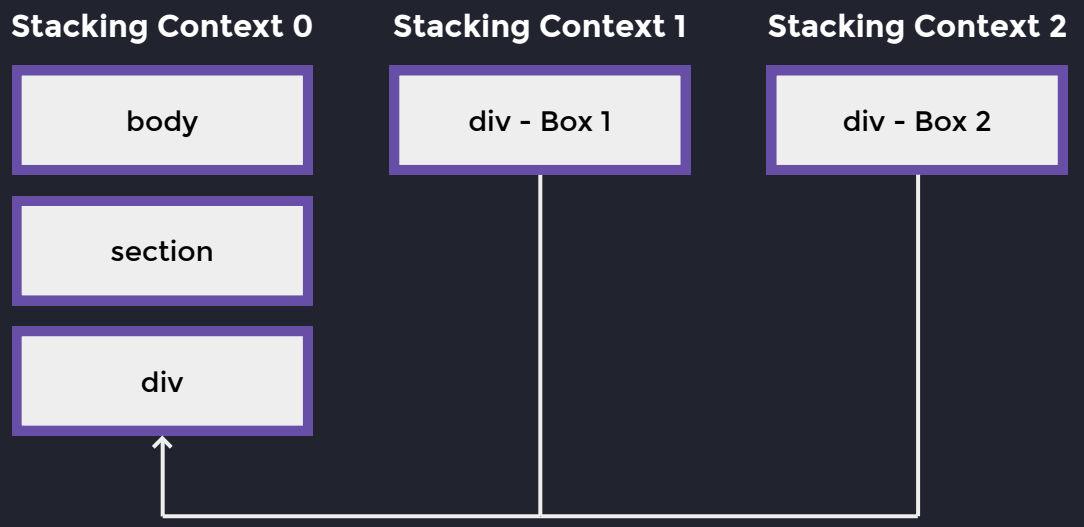
Position: absolute

```html
<section class="container">
  <div style="position: relative; height: 150px;">
    <div class="box red absolute">Box 1</div>
    <div class="box blue absolute">Box 2</div>
  </div>
</section>
```

https://codepen.io/RayEuji/embed/qBwbGpQ

- **Box 1** was removed from the normal flow and placed in a separate stacking context.
- **Box 2** was also removed from the normal flow and placed in the same stacking context as **Box 1**.

Since a later element is always stacked above the previous one, **Box 1** is positioned behind **Box 2**.



Key summary

The importance of these two types of positions — `relative` & `absolute` — lies in their ability to remove items from the **normal flow**

If we use positioning wise, we can achieve:

- **Isolation** - modifications made to elements positioned in this way will not affect other elements within the normal flow
- **Performance optimization** - key prerequisite for optimizing and minimizing updates to the **DOM tree**

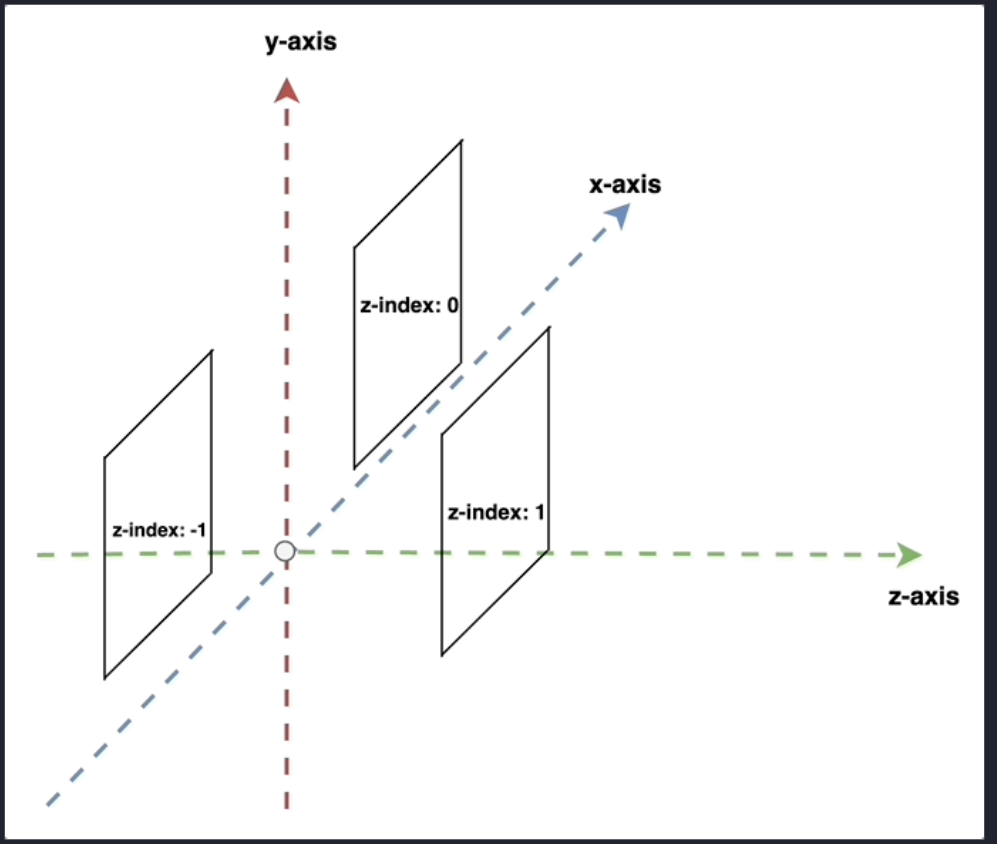
Stacking Context

When we build layouts, we work only with the **X** and **Y axis**, meaning everything is placed on a **single layer**.

However, when we engage in any **3D transformation**, **absolute positioning**, or any action that moves an element from **the normal flow**, we activate an additional axis known as the **Stacking Context** or **Z-axis.**

So, why does the browser need a stacking context?

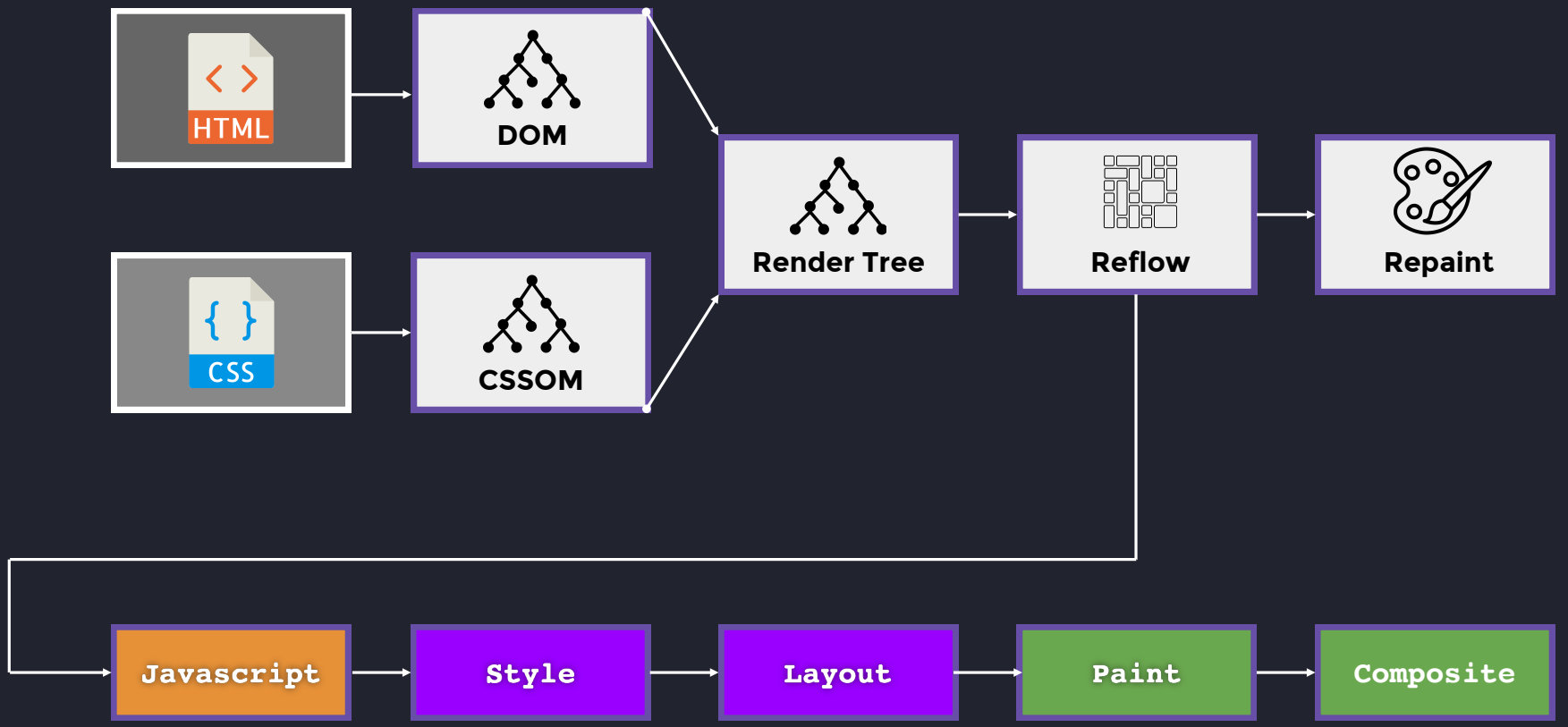
1. **Layering:** We need a way to represent layers in our layouts.
2. **Performance Optimization:**
    1. Elements removed from the **normal flow** are placed into a new stacking context.
    2. Modifications to every element within a separate **stacking context** do not impact any other elements within the **normal flow**.
    3. All CSS Transformations are GPU accelerated, meaning the browser doesn't need to recalculate the DOM Tree when such operations are performed. This minimizes the **reflow cycle**.



Stacking Context - 3D Example

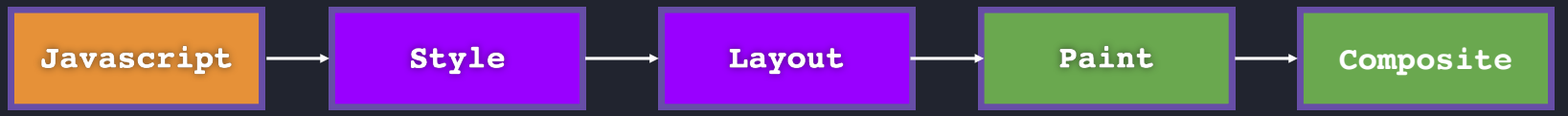
https://codepen.io/RayEuji/embed/QWMXreZ

Reflow

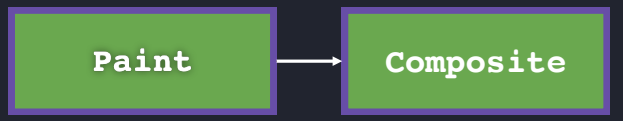

HTML
DOM
CSS
CSSOM
Render Tree
Reflow
Repaint
Javascript
Style
Layout
Paint
Composite

https://www.youtube.com/embed/dndeRnzkJDU?enablejsapi=1

Initial Pipeline

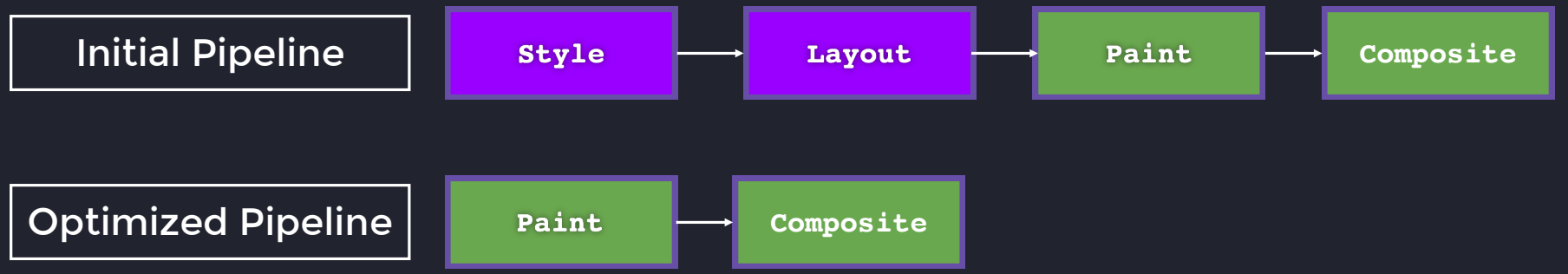


Optimized Pipeline



https://codepen.io/RayEuji/embed/JjyqQvw

4 lines of code: massive difference

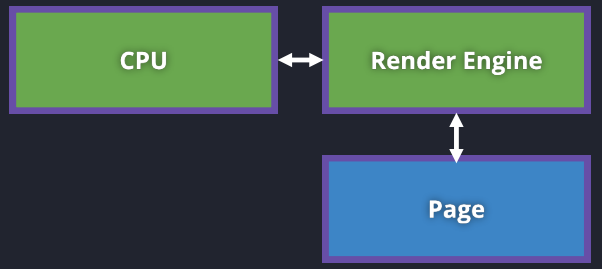



```
@keyframes moving-down-slow {
  from { margin-top: 0;}
  to { margin-top: 500px; }
}
```

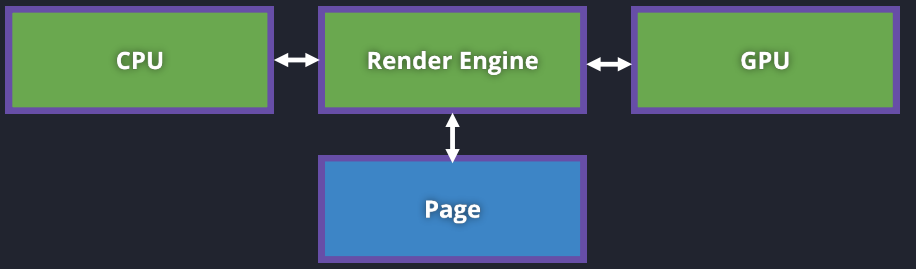
```
@keyframes moving-down-fast {
  from { transform: translateY(0px);}
  to { transform: translateY(500px);}
}
```

Composition Layers

Old Browsers



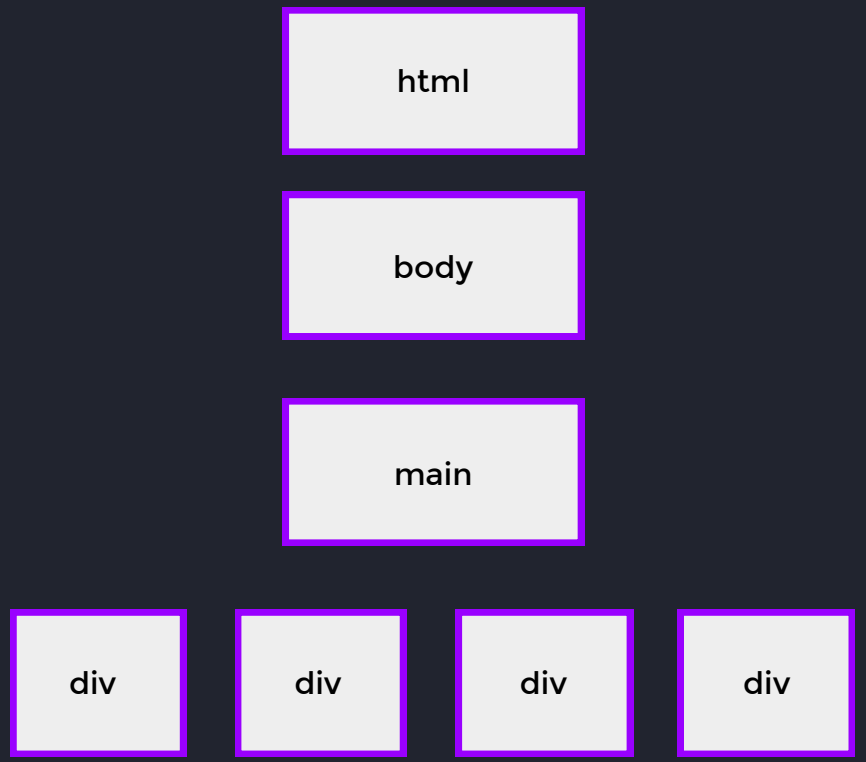
New Browsers



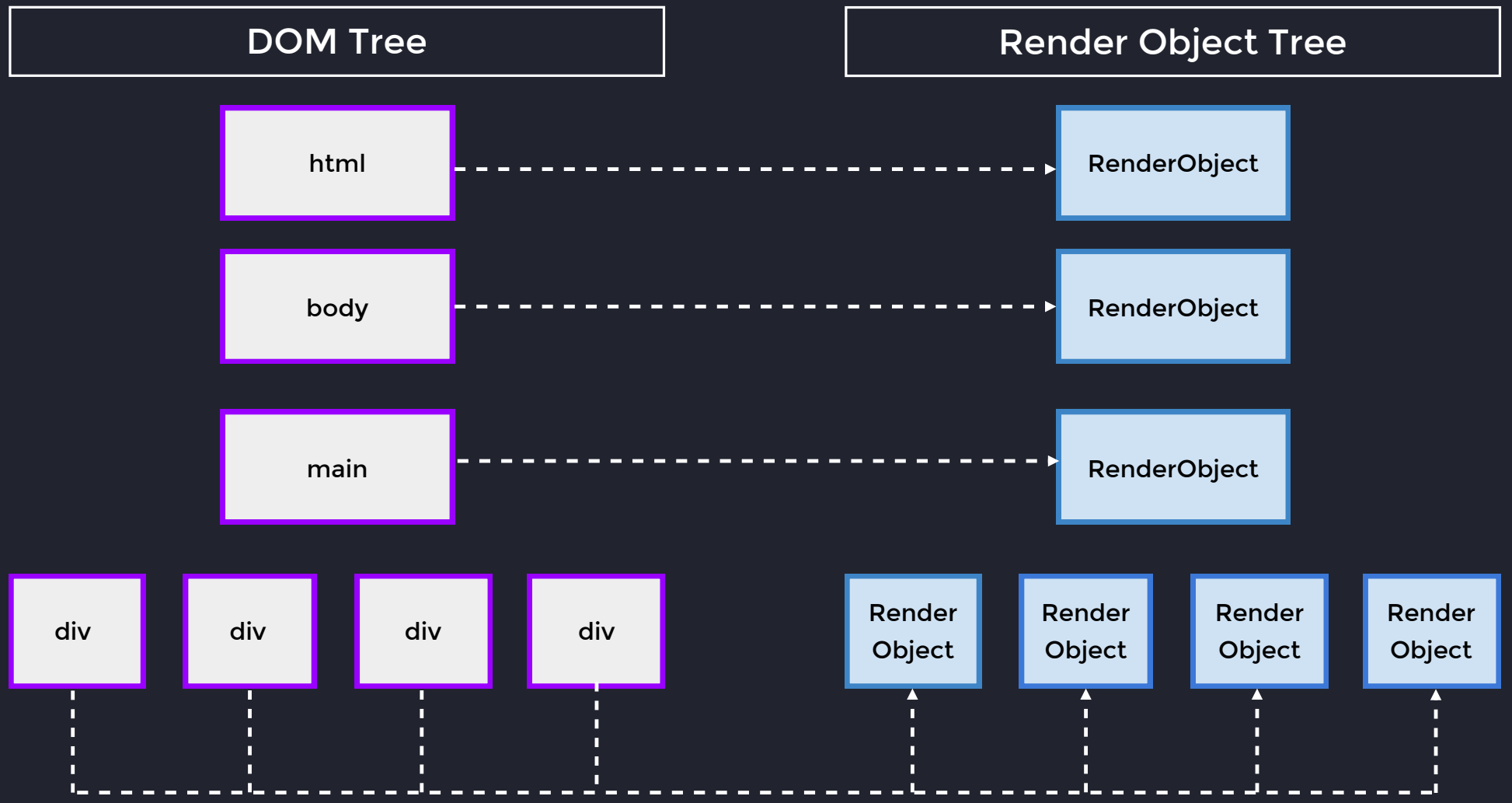
Composition Layers

```
1  <html>
2  <body>
3  <main>
4   <div class="box red"></div>
5   <div class="box yellow"></div>
6   <div class="box green"></div>
7   <div class="box blue"></div>
8  </main>
9  </body>
10 </html>
```

https://codepen.io/RayEuji/embed/qBwNbLm?editors=1100



Browser Graphics API: Render Object

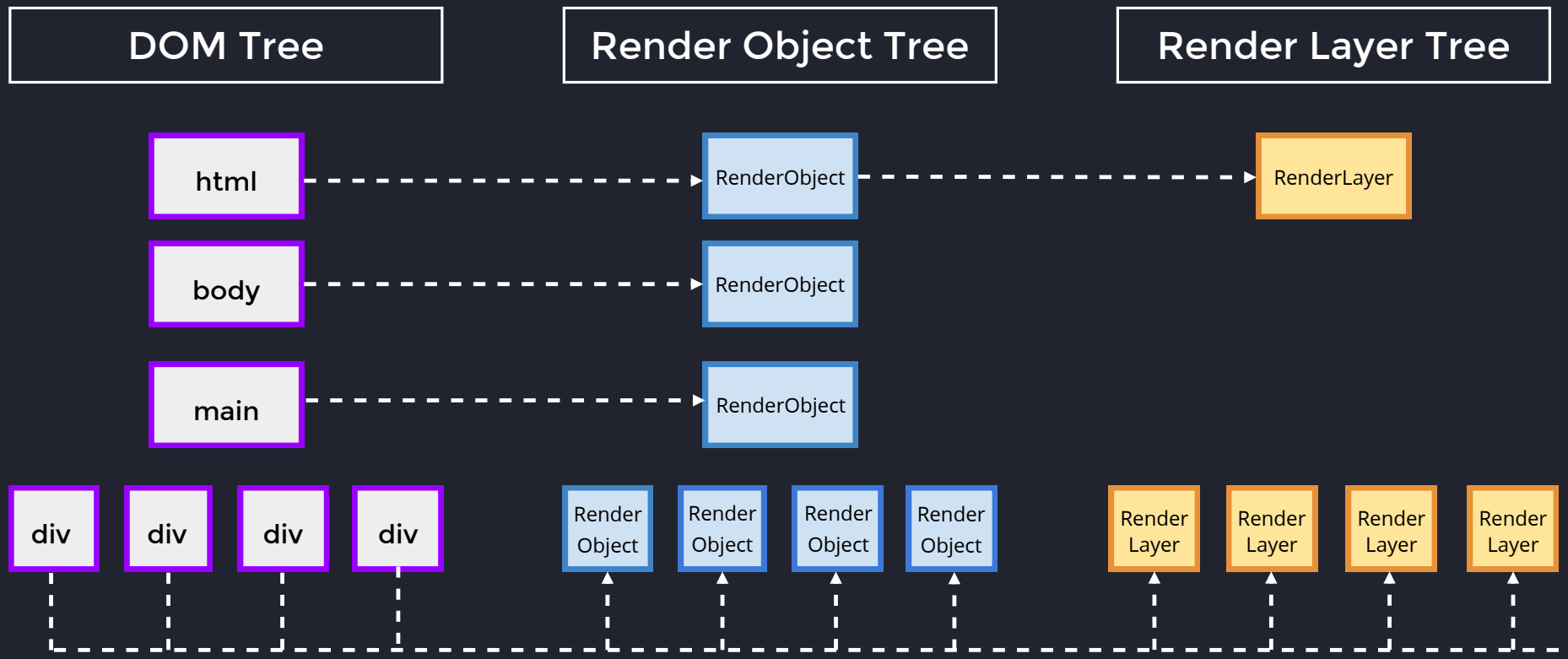


Browser Graphics API: Render Layer

Render Layer for element is constructed when element:

- It has explicit CSS properties:
  - `position: relative | absolute`
  - `transform`
- It's the root object for the page - `<html/>`
- It is **transparent**
- Has a **CSS filter**
- Corresponds to `<canvas>` element that has a 3D (WebGL) context or an accelerated 2D context
- Corresponds to a `<video>` element

Browser Graphics API: Render Layer

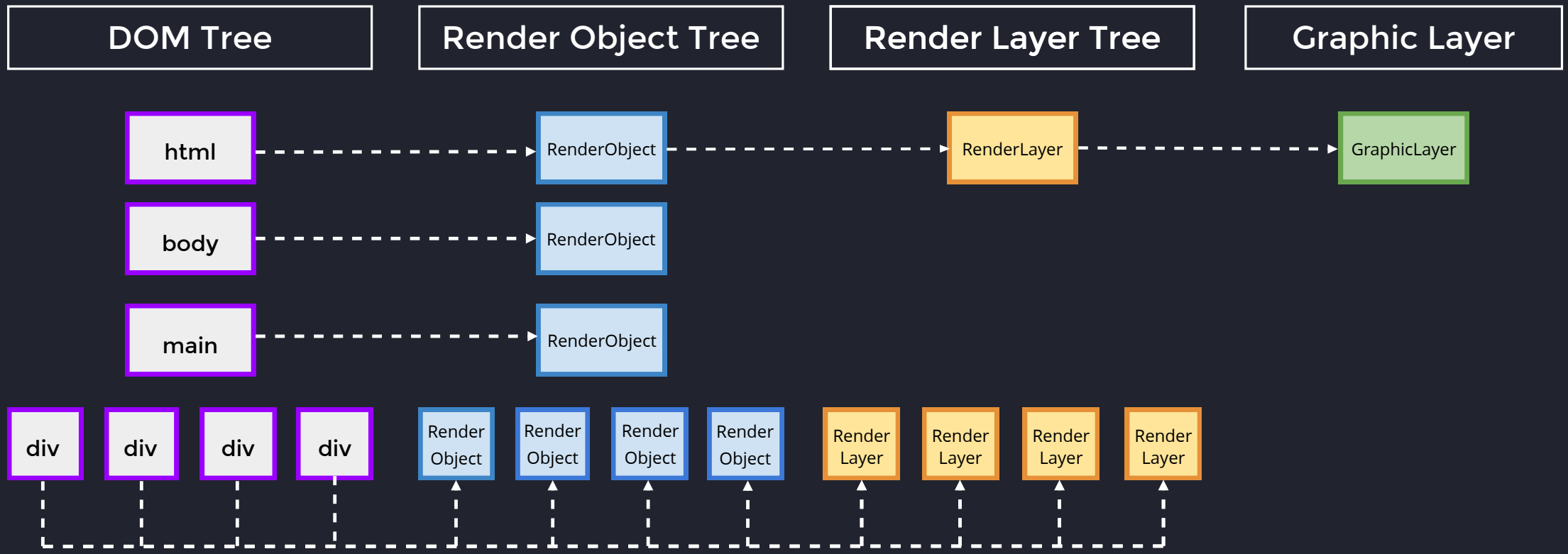


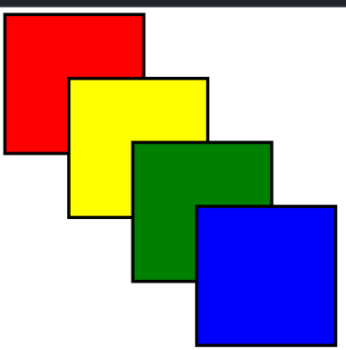
Browser Graphics API: Graphic Layer

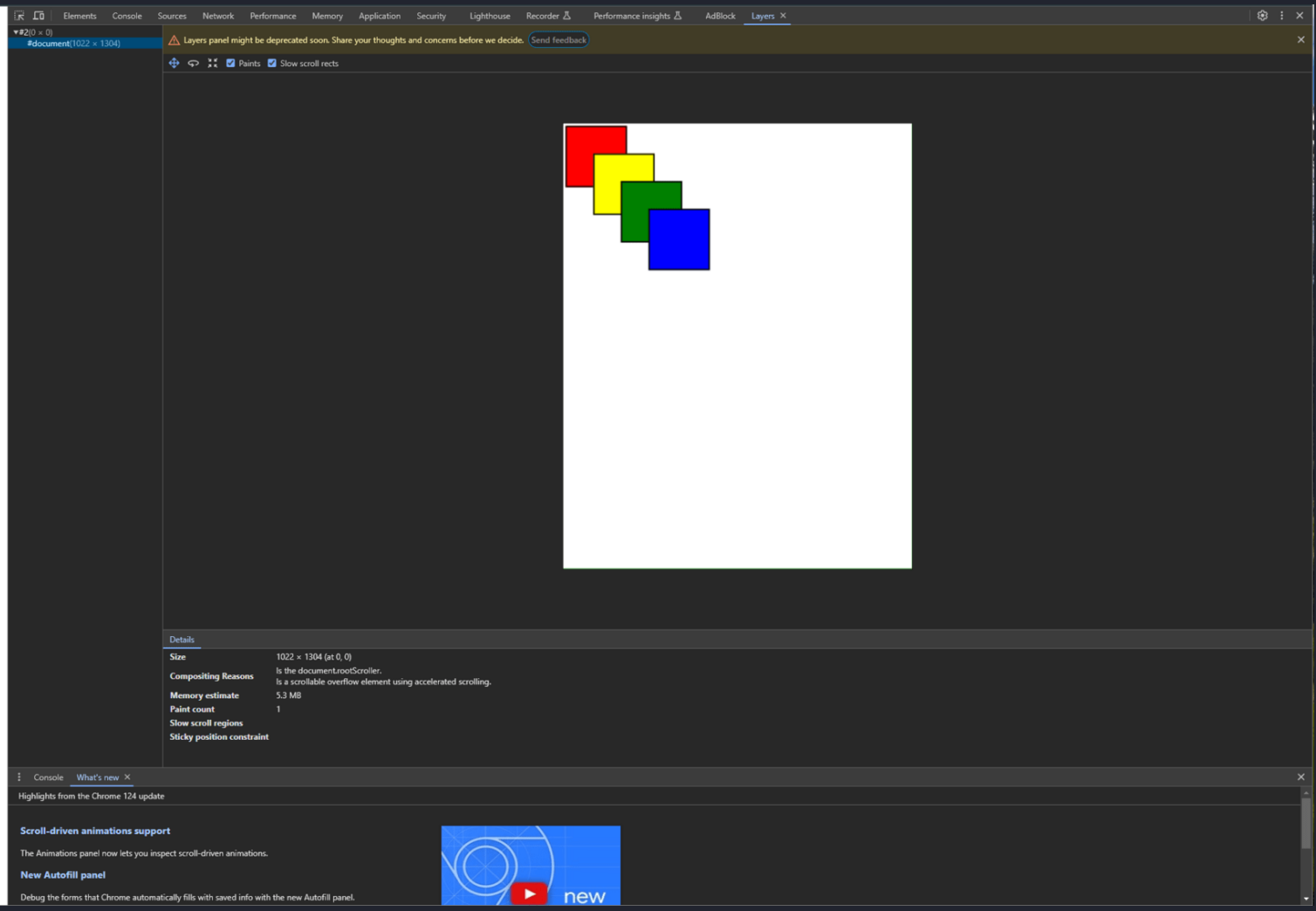
Graphic Layer is constructed when:

- **Render Layer** has **3D** or **perspective transform** CSS properties
- **Layer is used by:**
  - `<video>` element using accelerated video decoding
  - `<canvas>` with `3D/2D context`
- **Layer uses:**
  - CSS animation for its **opacity**
  - animated **web-kit transform**
- **Layer** uses accelerated `CSS filters`
- **Layer** has a descendant that is a **compositing layer**
- **Layer** has a sibling with a lower `z-index` which has a compositing layer (in other words the layer overlaps a composited layer and should be rendered on top of it)

Browser Graphics API: Graphic Layer






Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Performance insights
AdBlock
Layers
#document(1022 × 1304)
Layers panel might be deprecated soon. Share your thoughts and concerns before we decide.
Send feedback
Paints
Slow scroll rects
Details
Size
1022 × 1304 (at 0, 0)
Compositing Reasons
Is the document.rootScroller.
Is a scrollable overflow element using accelerated scrolling.
Memory estimate
5.3 MB
Paint count
1
Slow scroll regions
Sticky position constraint
Console
What's new
Highlights from the Chrome 124 update
Scroll-driven animations support
The Animations panel now lets you inspect scroll-driven animations.
New Autofill panel
Debug the forms that Chrome automatically fills with saved info with the new Autofill panel.
new

Caveats

"With great power comes great responsibility"

Graphic Layer is expensive object to initialize, overusing it can blow-up your device as it uses **VRAM** and **CPU**

Let's check the DEMO

Let's become a browser: Part 2

Our task is to:

- Build all **Formatting Contexts**
- Detect elements outside of **normal flow**
- Build a **Stacking context**
- Build **DOM**, **RenderObject**, **RenderLayer** and **GraphicLayer** Trees

```
<html>
    <body>
        <section style="display:flex">
            <div>Flex Item 1</div>
            <div>Flex Item 2</div>
        </section>
        <div style="position:absolute">
            <span>Modal</span>
        </div>
        <div style="display: inline-block">
            <div>List item</div>
        </div>
        <div style="position: absolute; transform: translateZ(0);">
    Transformed
    </div>
    </body>
```

Step 1

DOM Tree

<html>

Graphic Layer Tree

GraphicLayer

Stacking Context (Root)

Block Formatting Context (Root)

Render Object Tree

RenderObject

Render Layer Tree

RenderLayer

```
<html>
  <body>
    <section style="display:flex">
      <div>Flex Item 1</div>
      <div>Flex Item 2</div>
    </section>
      <div style="position:absolute">
         <span>Modal</span>
      </div>
      <div style="display: inline-block">
         <div>List item</div>
      </div>
      <div style="position: absolute; transform: translateZ(0);
        Transformed
     </div>
  </body>
</html>
```

Step 2

DOM Tree

<html>

<body>

Graphic Layer Tree

GraphicLayer

Stacking Context (Root)

Block Formatting Context (Root)

Render Object Tree

RenderObject

RenderObject

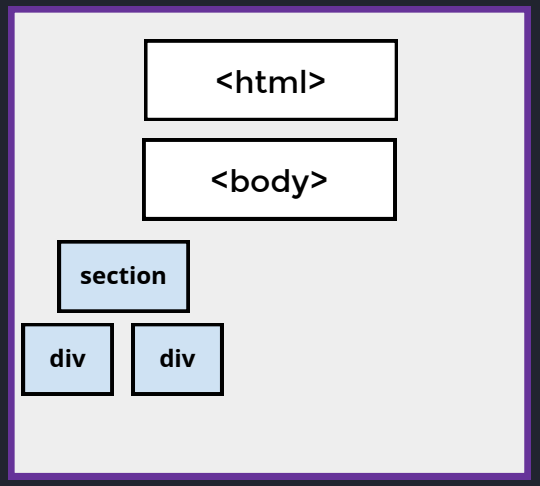
Render Layer Tree

RenderLayer

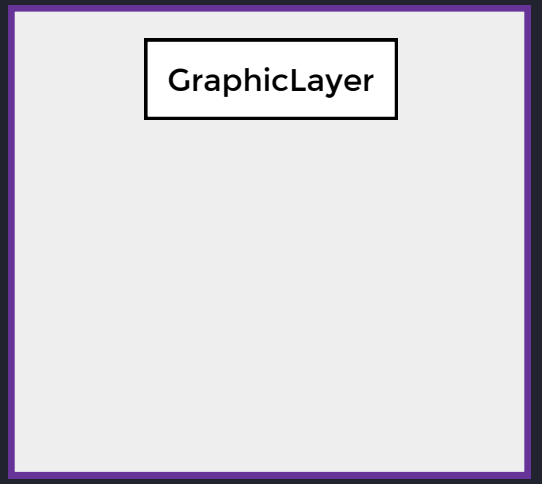
```
<html>
  <body>
    <section style="display:flex">
      <div>Flex Item 1</div>
      <div>Flex Item 2</div>
    </section>
      <div style="position:absolute">
         <span>Modal</span>
      </div>
      <div style="display: inline-block">
         <div>List item</div>
      </div>
      <div style="position: absolute; transform: translateZ(0);
        Transformed
      </div>
  </body>
</html>
```

Step 3: render <section>

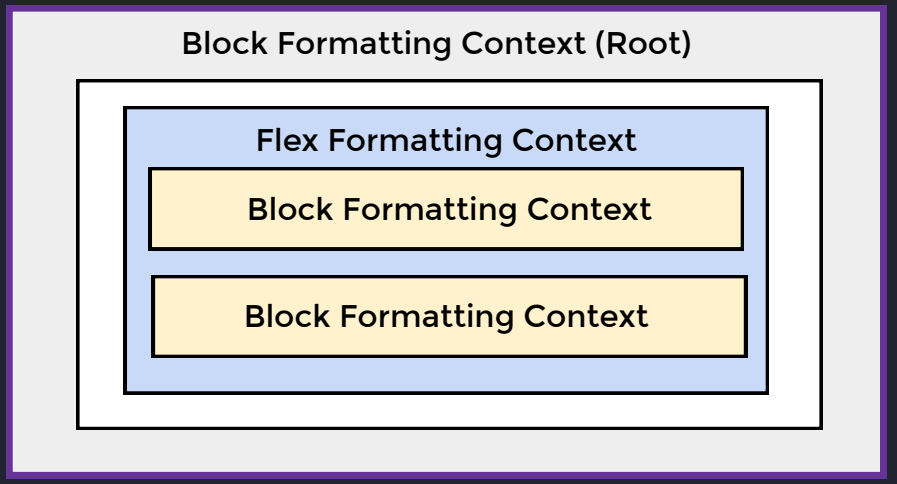
DOM Tree



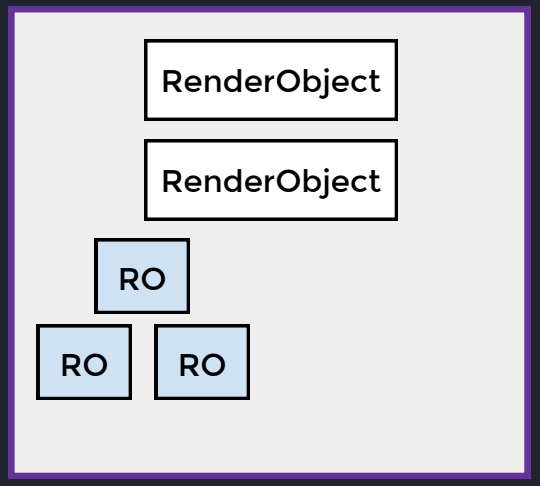
Graphic Layer Tree



Stacking Context (Root)



Render Object Tree

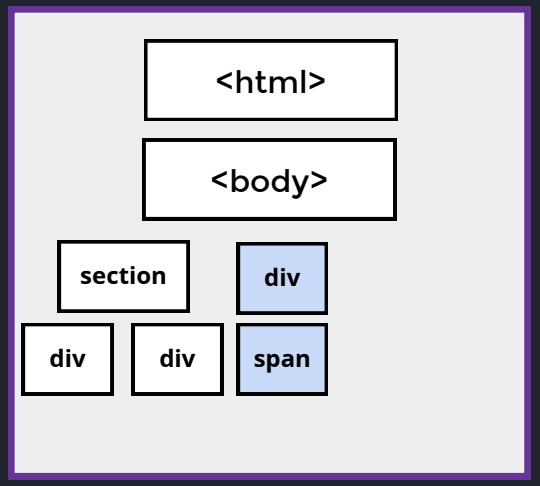


Render Layer Tree

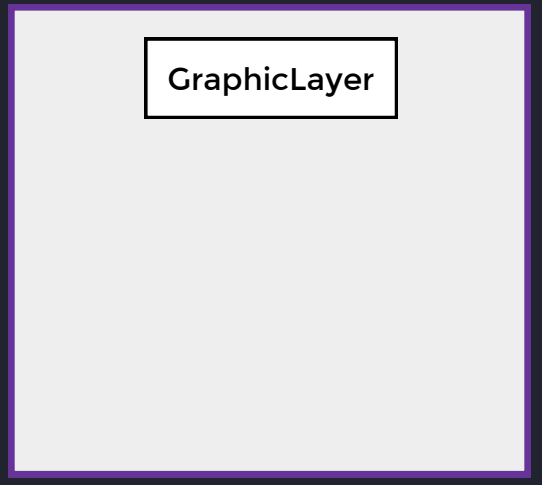

```html
<html>
 <body>
   <section style="display:flex">
     <div>Flex Item 1</div>
     <div>Flex Item 2</div>
   </section>
     <div style="position:absolute">
        <span>Modal</span>
     </div>
     <div style="display: inline-block">
        <div>List item</div>
     </div>
     <div style="position: absolute; transform: translateZ(0);
       Transformed
     </div>
 </body>
</html>
```

Step 4: render with absolute position

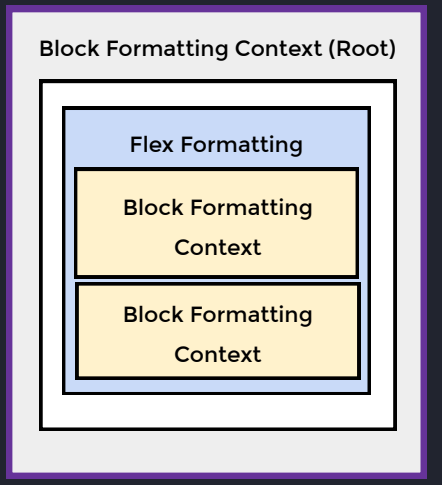
DOM Tree



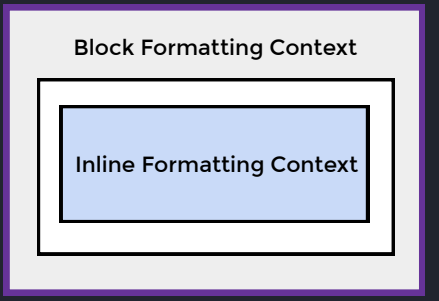
Graphic Layer Tree



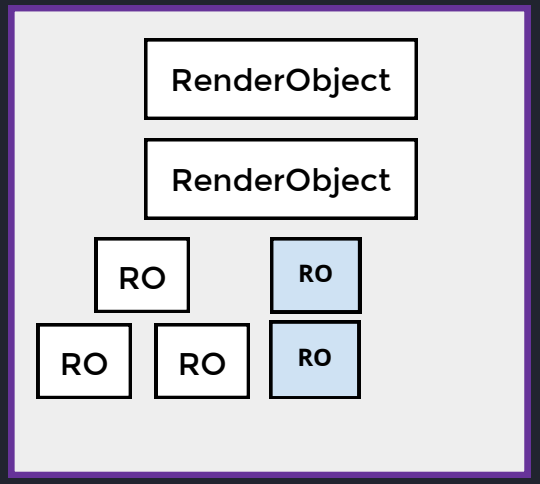
Stacking Context (Root)



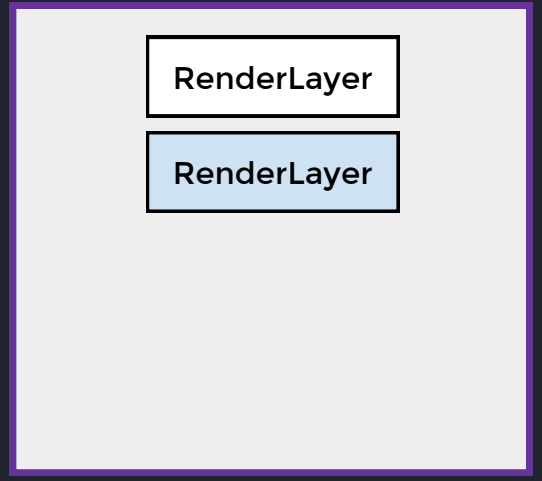
Stacking Context 1



Render Object Tree



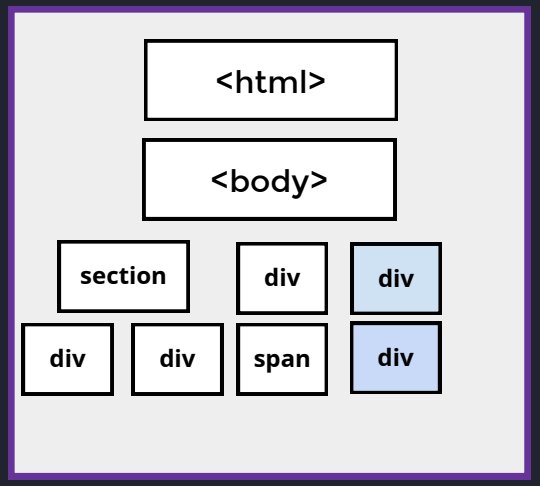
Render Layer Tree



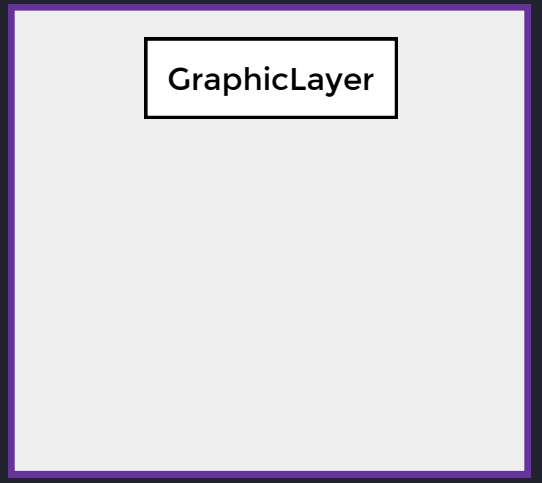

```
<html>
 <body>
   <section style="display:flex">
     <div>Flex Item 1</div>
     <div>Flex Item 2</div>
   </section>
     <div style="position:absolute">
        <span>Modal</span>
     </div>
     <div style="display: inline-block">
        <div>List item</div>
     </div>
     <div style="position: absolute; transform: translateZ(0);
       Transformed
     </div>
 </body>
</html>
```

Step 5: render with display: inline-block

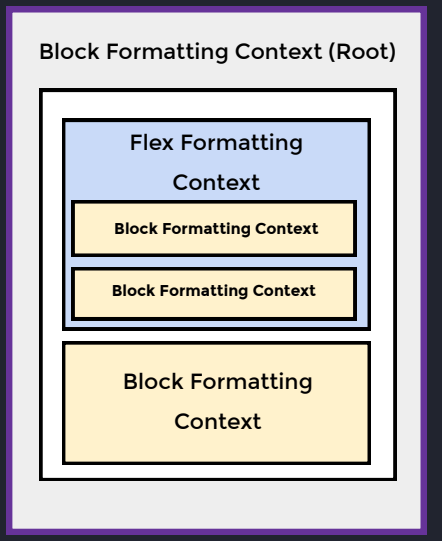
DOM Tree



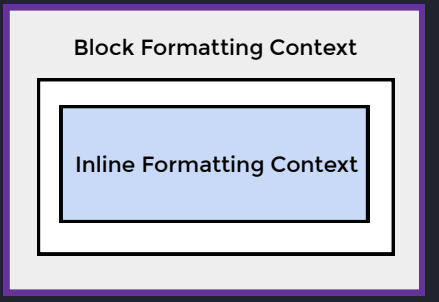
Graphic Layer Tree



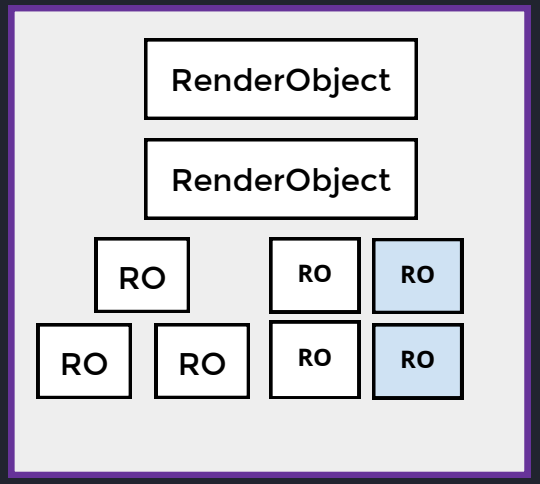
Stacking Context (Root)



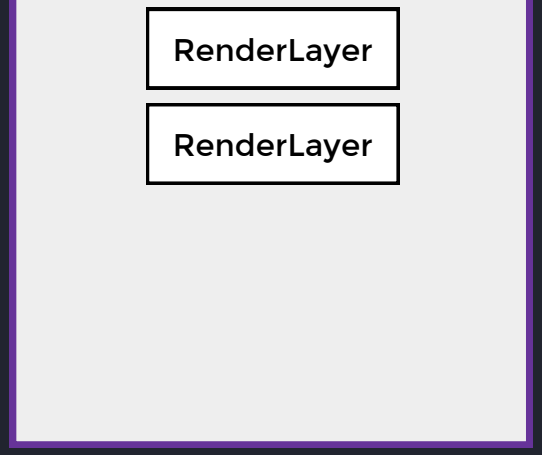
Stacking Context 1



Render Object Tree



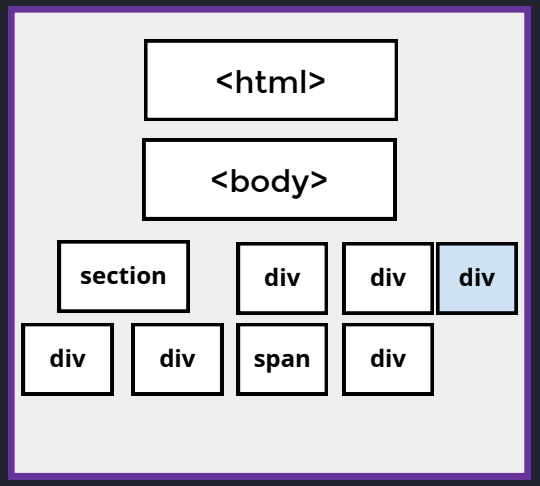
Render Layer Tree



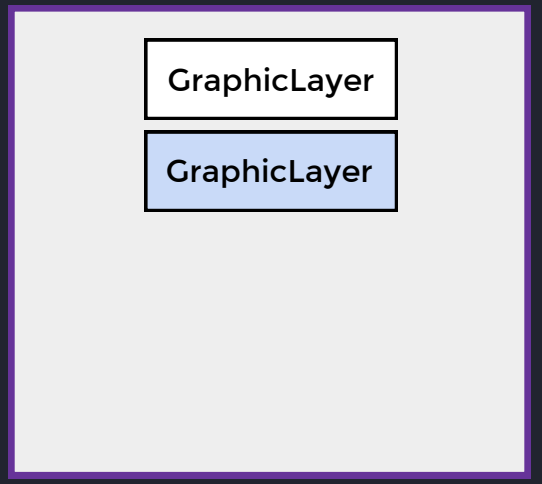

```
<html>
 <body>
   <section style="display:flex">
     <div>Flex Item 1</div>
     <div>Flex Item 2</div>
   </section>
     <div style="position:absolute">
        <span>Modal</span>
     </div>
     <div style="display: inline-block">
        <div>List item</div>
     </div>
     <div style="position: absolute; transform: translateZ(0);
       Transformed
    </div>
 </body>
</html>
```

Step 6: render with transform attribute

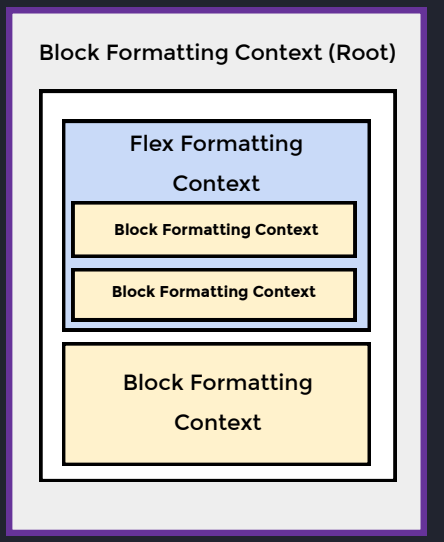
DOM Tree



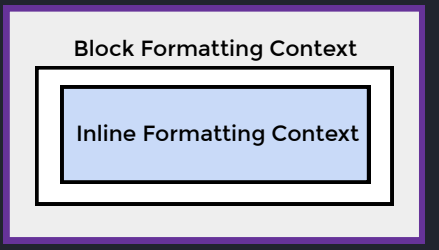
Graphic Layer Tree



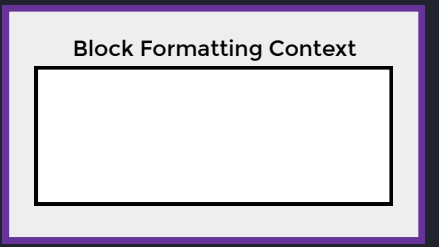
Stacking Context (Root)



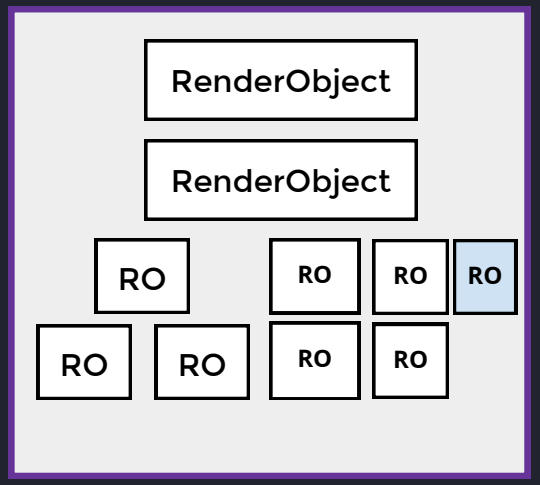
Stacking Context 1



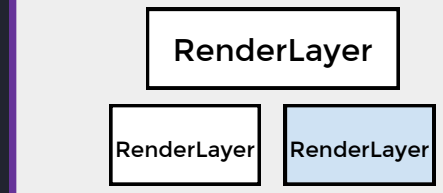
Stacking Context 2



Render Object Tree

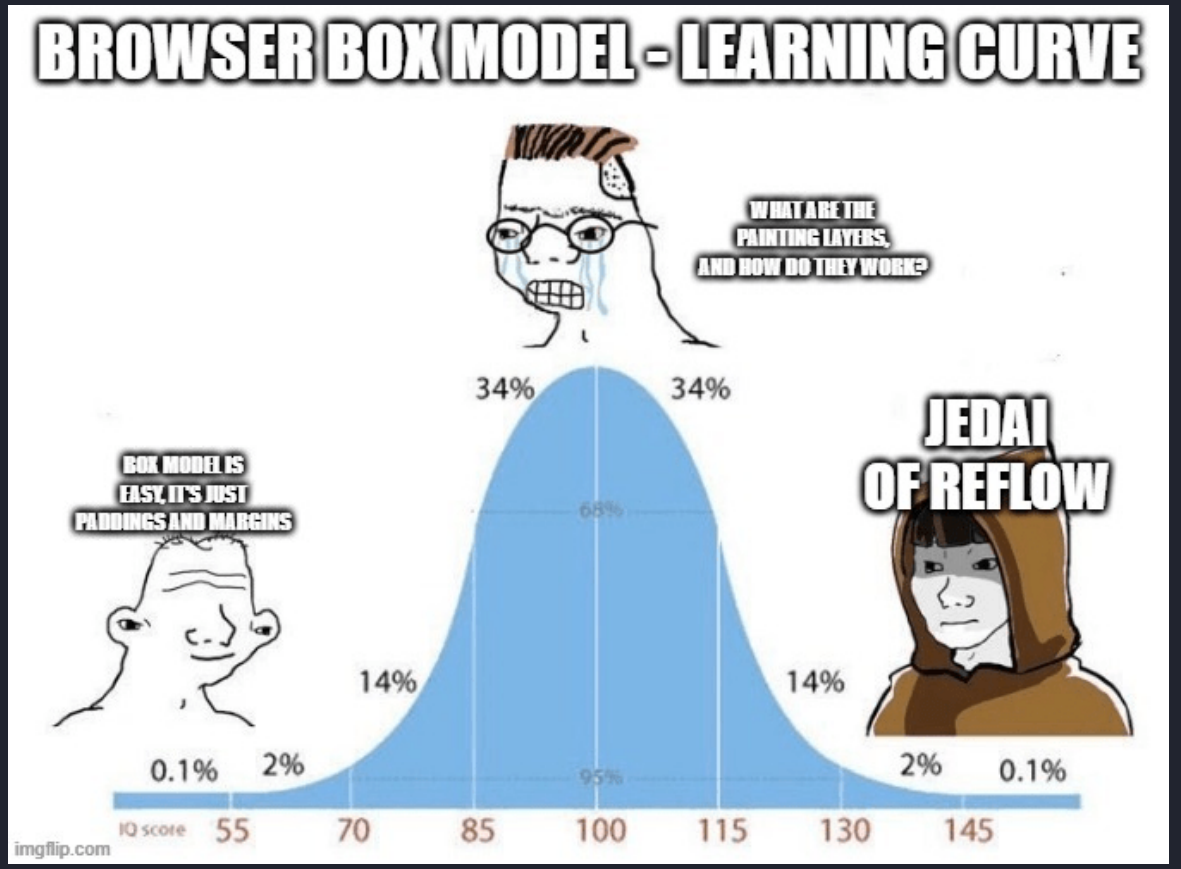


Render Layer Tree




```html
<html>
 <body>
   <section style="display:flex">
     <div>Flex Item 1</div>
     <div>Flex Item 2</div>
   </section>
     <div style="position:absolute">
        <span>Modal</span>
     </div>
     <div style="display: inline-block">
        <div>List item</div>
     </div>
     <div style="position: absolute; transform: translateZ(0);
       Transformed
    </div>
 </body>
</html>
```

DONE!


BROWSER BOX MODEL - LEARNING CURVE
WHAT ARE THE PAINTING LAYERS, AND HOW DO THEY WORK?
BOX MODEL IS EASY, IT'S JUST PADDINGS AND MARGINS
JEDAI OF REFLOW
34%
34%
14%
14%
0.1%
2%
2%
0.1%
68%
95%
IQ score
55
70
85
100
115
130
145
imgflip.com

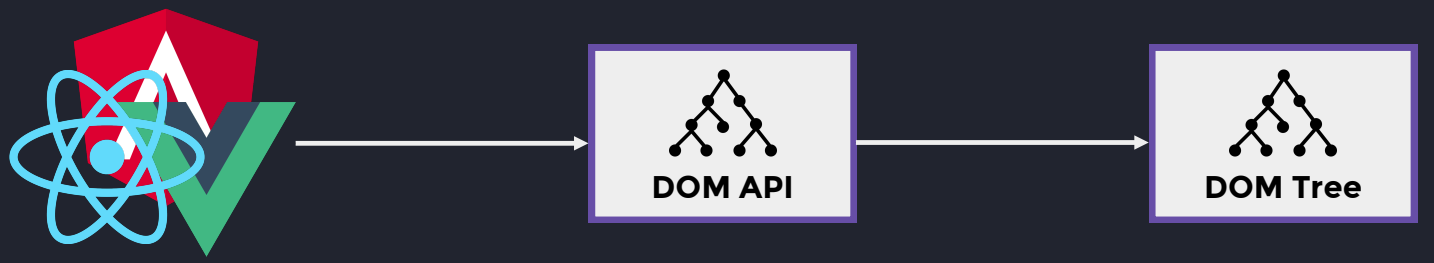
DOM API

DOM API

The **DOM API** is a set of methods we can utilise in JavaScript to manipulate the DOM.

When to use it?

1. Building a low-level library (e.g., virtualisation, DOM management, etc.).
2. Creating generic components (such as a Video Player or Chart Engine).
3. Developing minimal-blueprint apps.



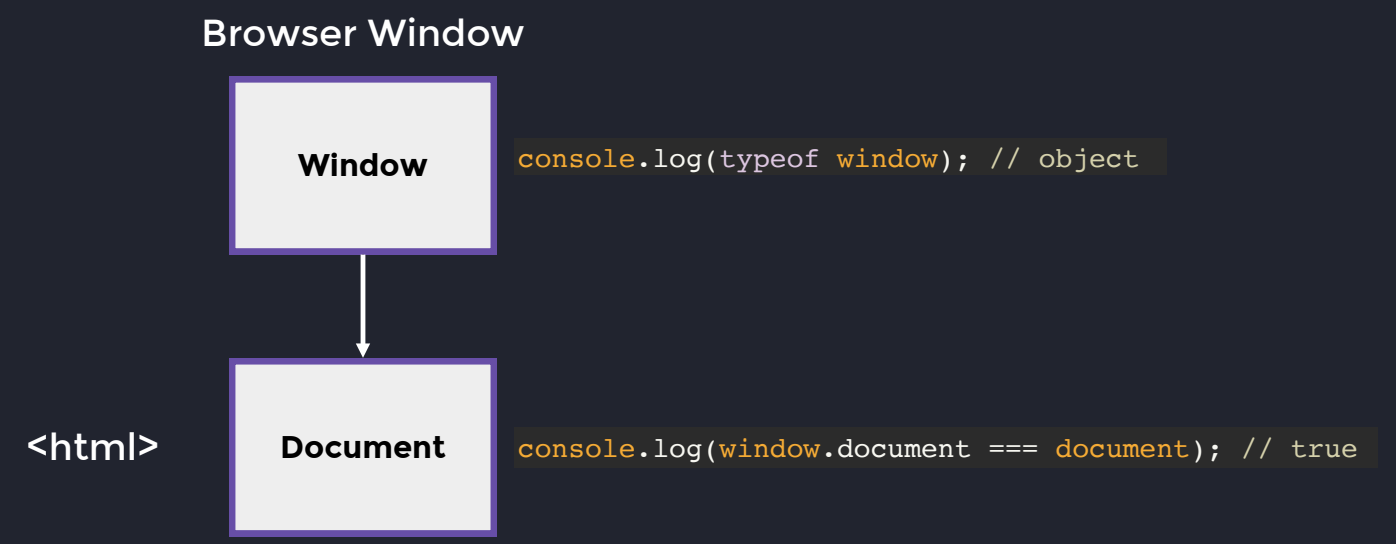
DOM API: Global Objects

Browser Window

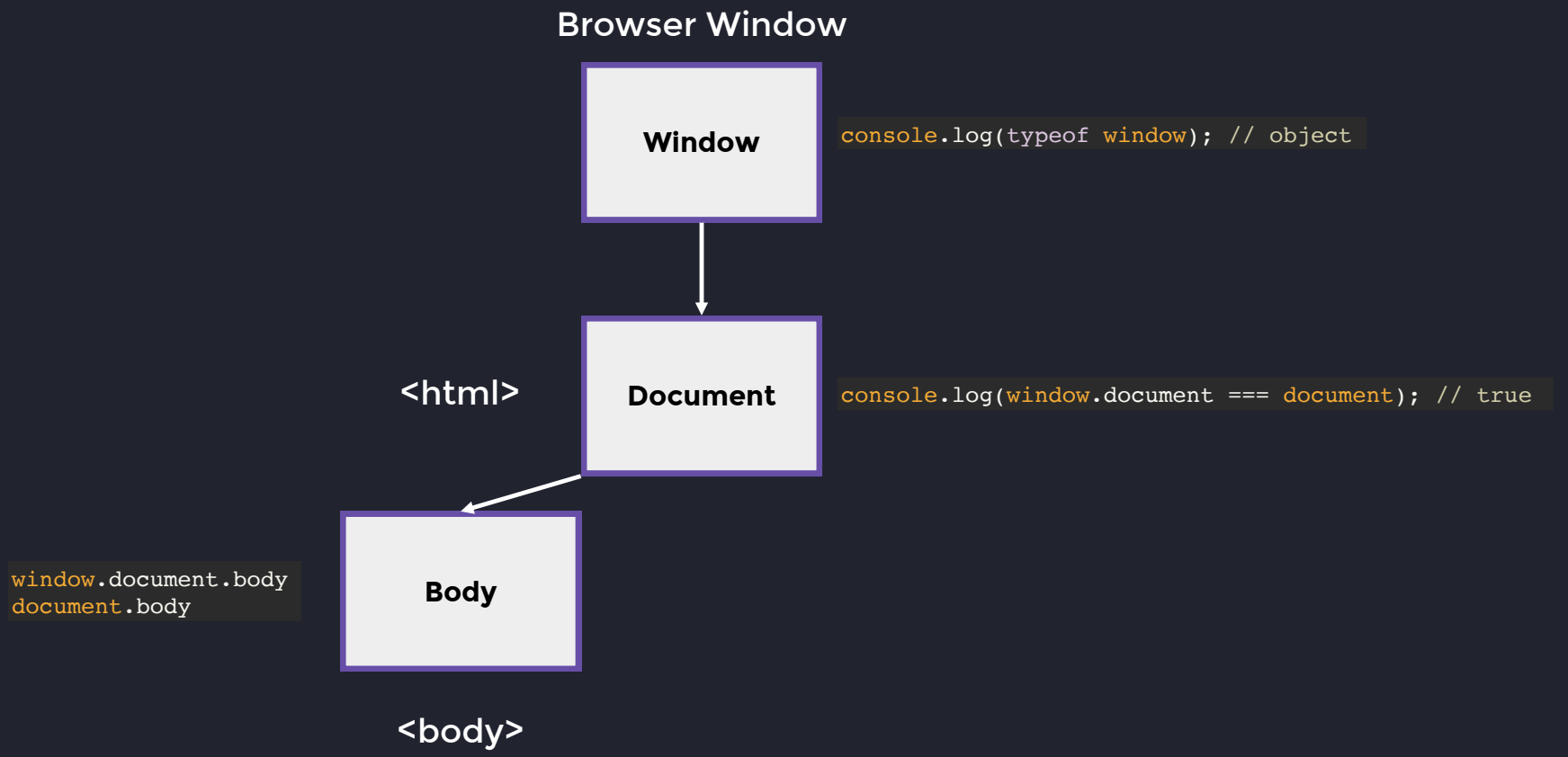
Window

```
console.log(typeof window); // object
```

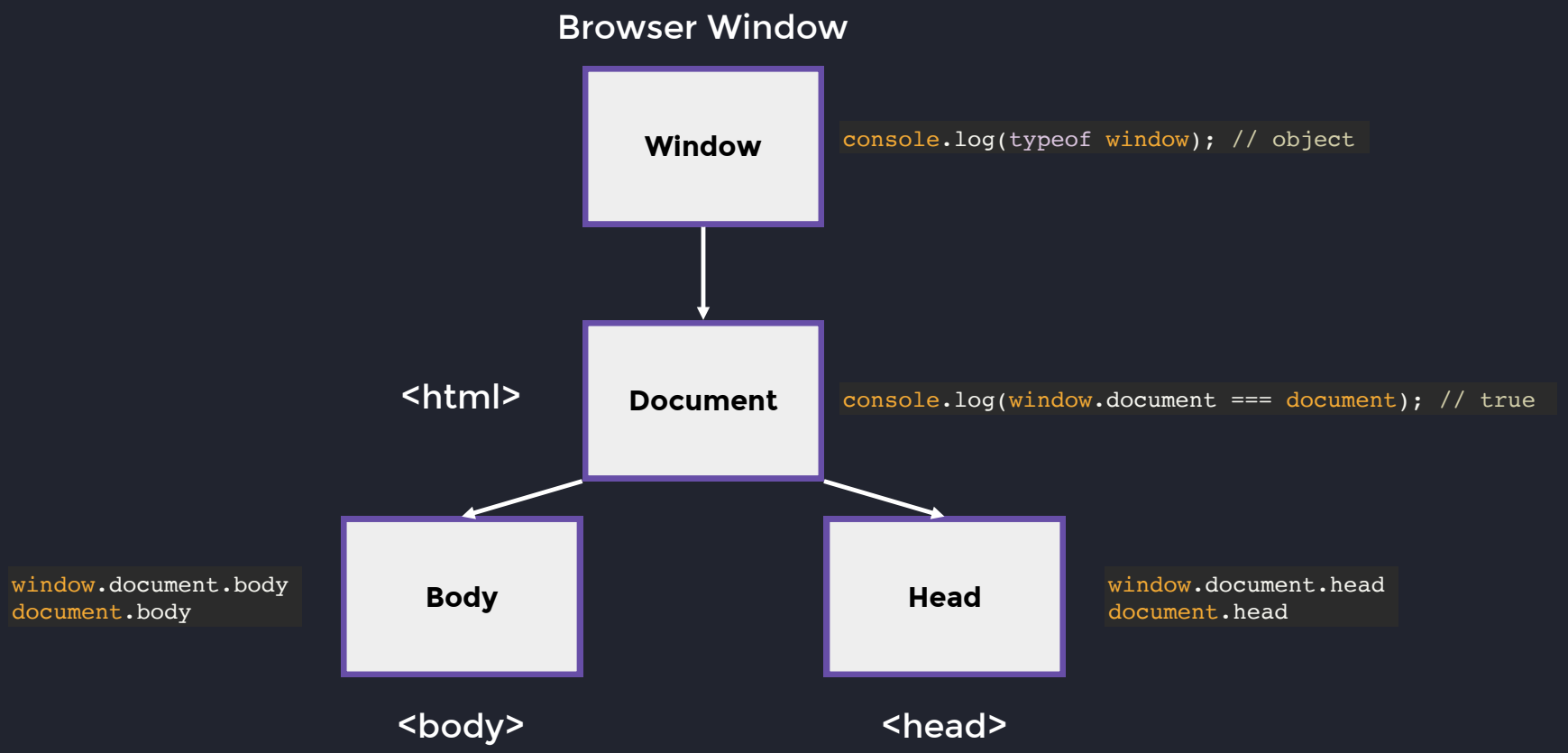
DOM API: Global Objects



DOM API: Global Objects



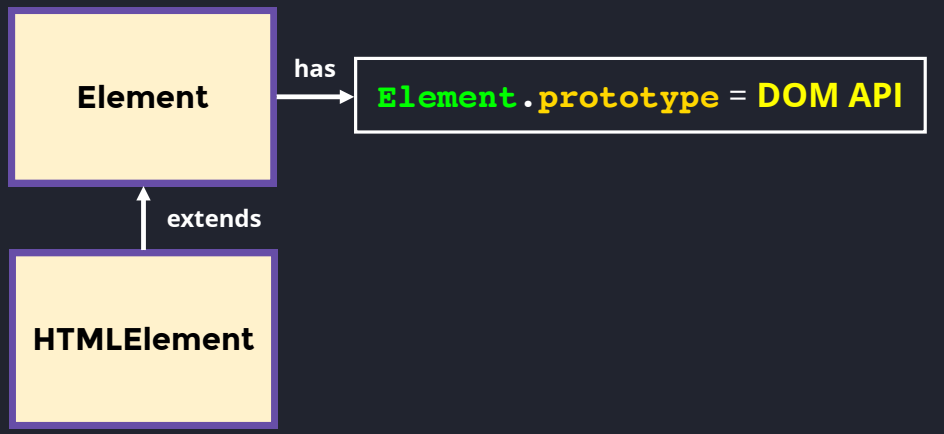
DOM API: Global Objects



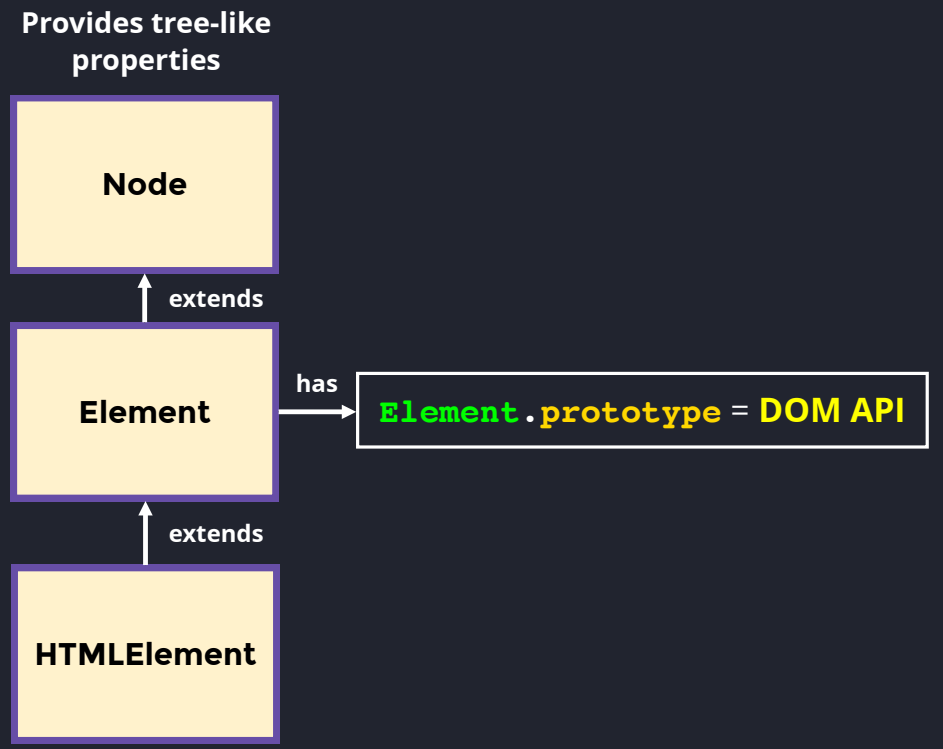
DOM API: Class Hierarchy

HTMLElement

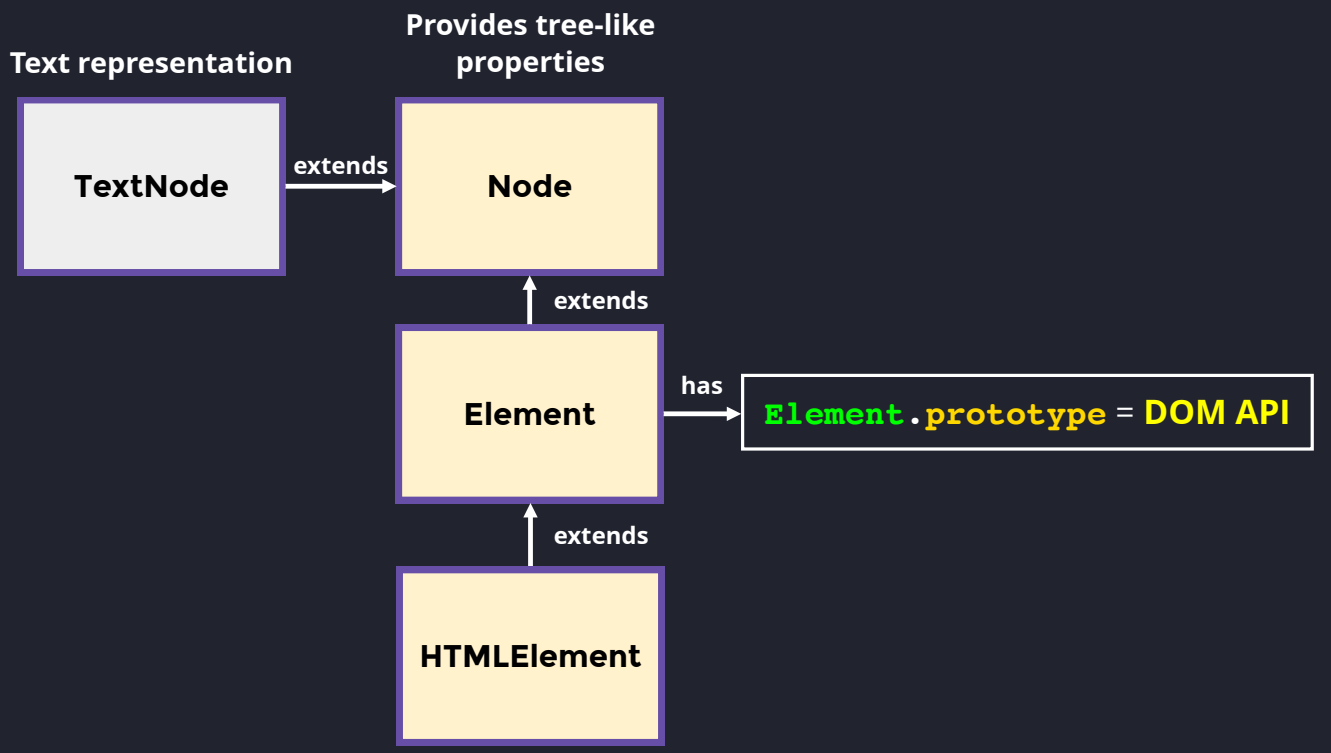
DOM API: Class Hierarchy



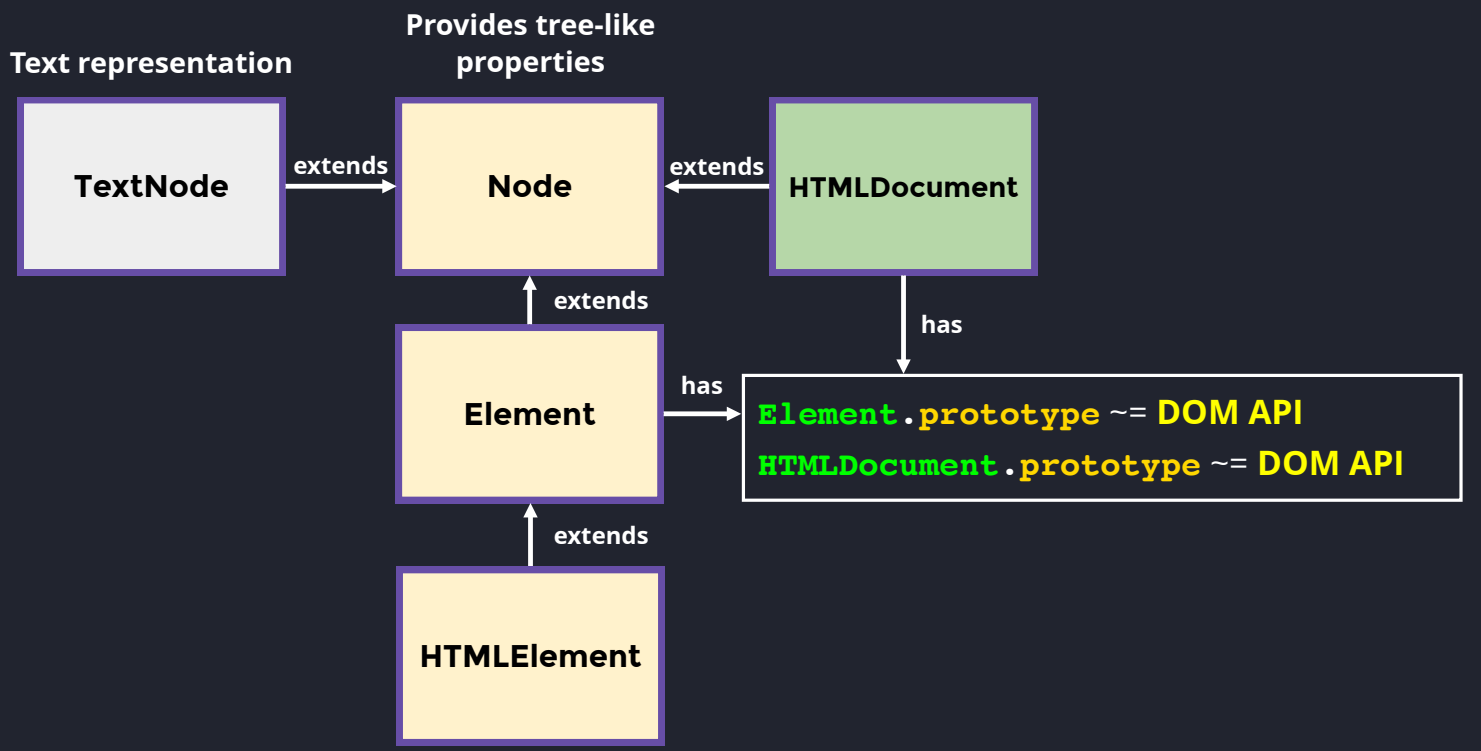
DOM API: Class Hierarchy



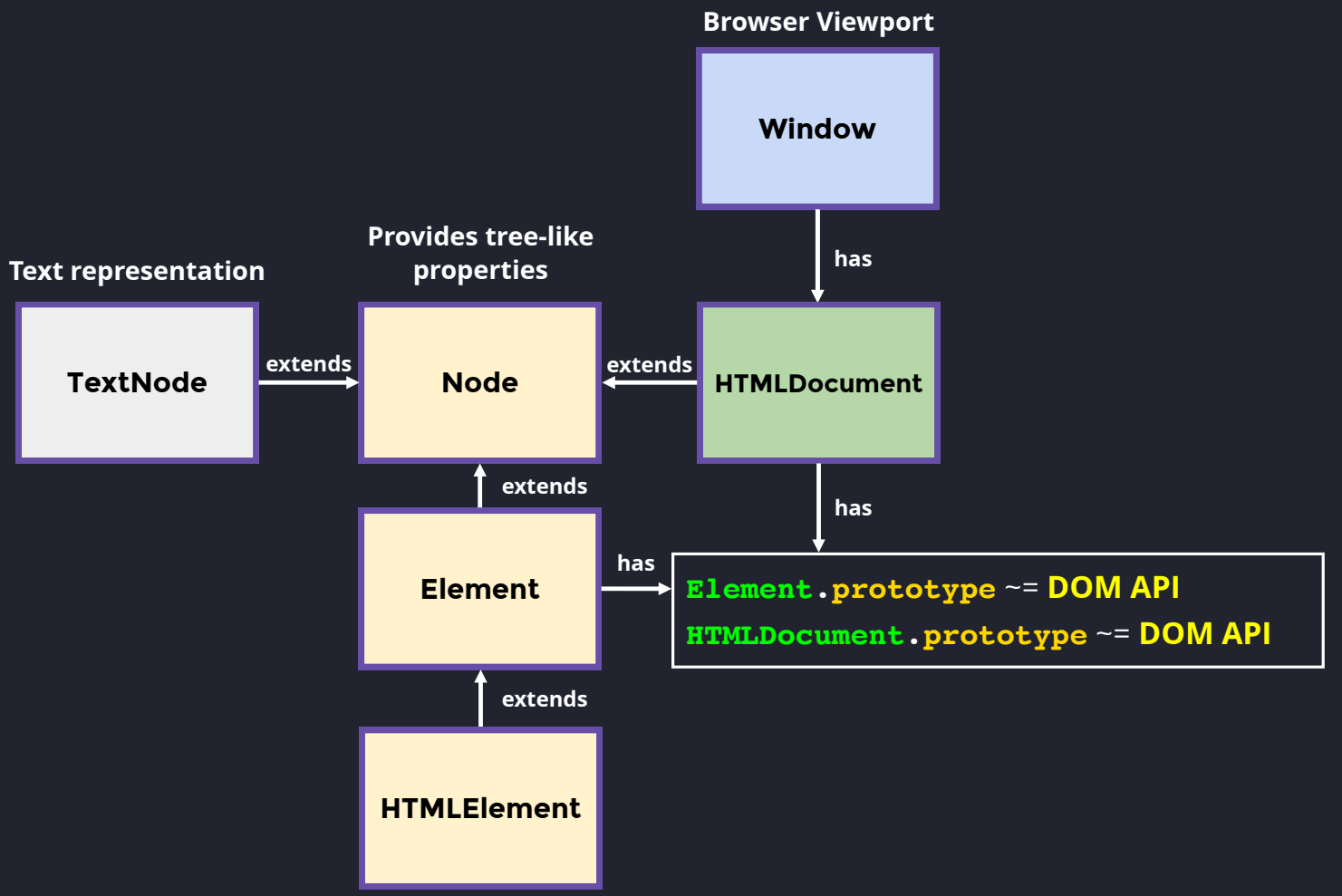
DOM API: Class Hierarchy



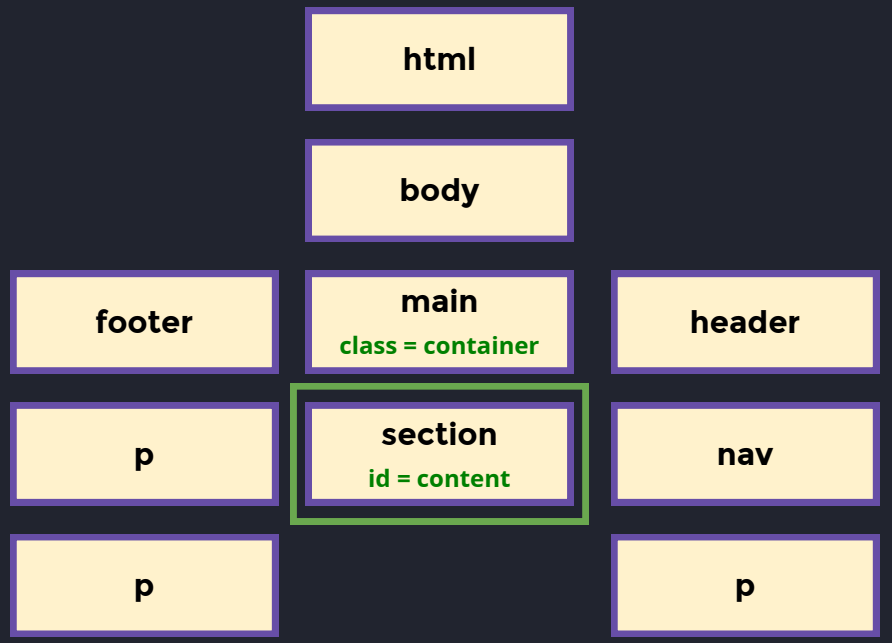
DOM API: Class Hierarchy



DOM API: Class Hierarchy



DOM Querying: getElementByID




```
document.getElementById("content");
```

ID	Ref
"content"	HTMLElement

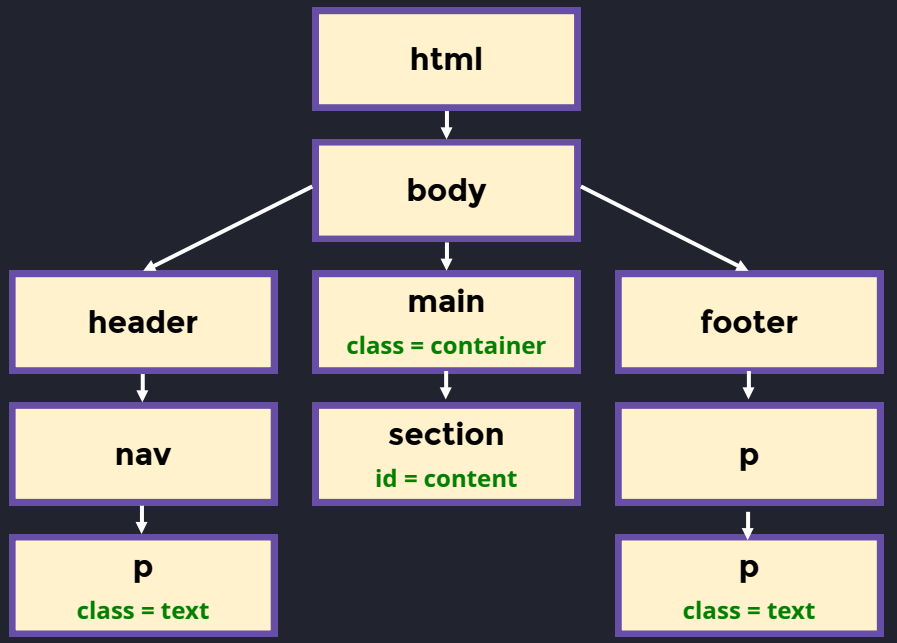
Return type: `Element`

Time Complexity: `O(1)`

Read Cost: `O(1)`

Memory Cost: `O(1)`

DOM Querying: `getElementsByClassName`




```
document.getElementsByClassName("text")
```

Return type: `HTMLCollection`

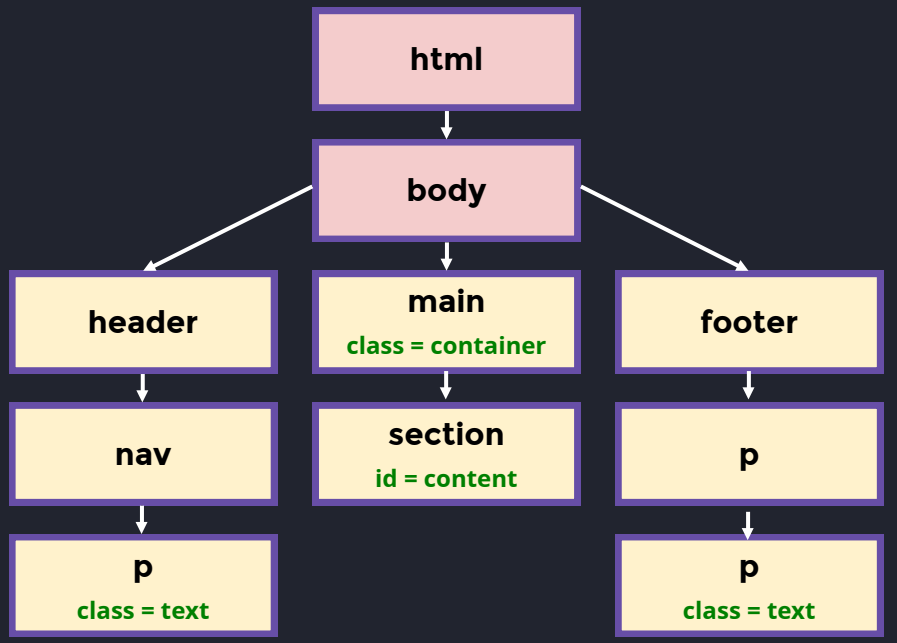
Time Complexity: `O(N)*`

Read Cost: `O(N)*`

Memory Cost: `Low`

Query Algorithm: `DFS + Hashmap`

DOM Querying: `getElementsByClassName`




```
document.getElementsByClassName("text")
```

Return type: `HTMLCollection`

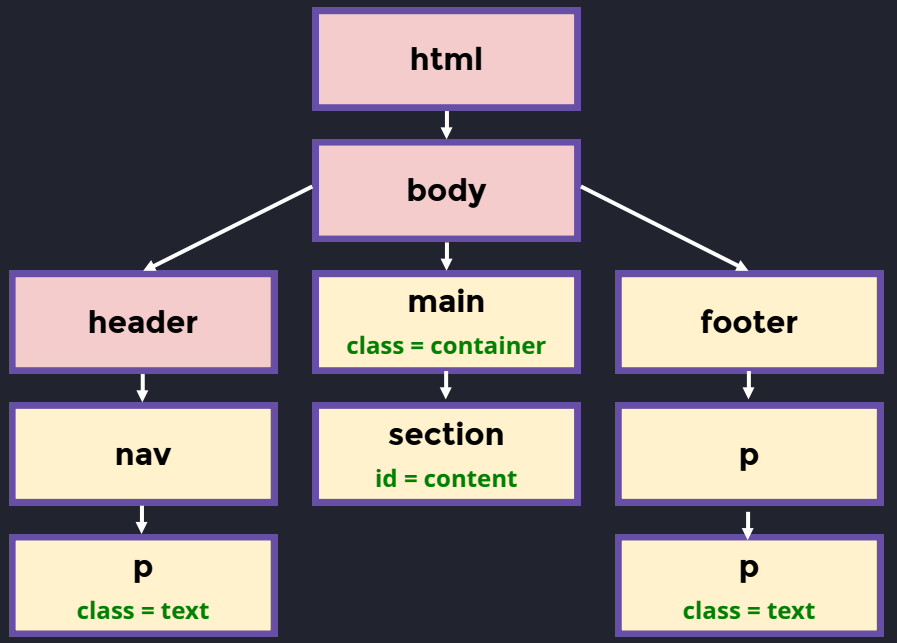
Time Complexity: `O(N)*`

Read Cost: `O(N)*`

Memory Cost: `Low`

Query Algorithm: `DFS + Hashmap`

DOM Querying: `getElementsByClassName`




```
document.getElementsByClassName("text")
```

Return type: `HTMLCollection`

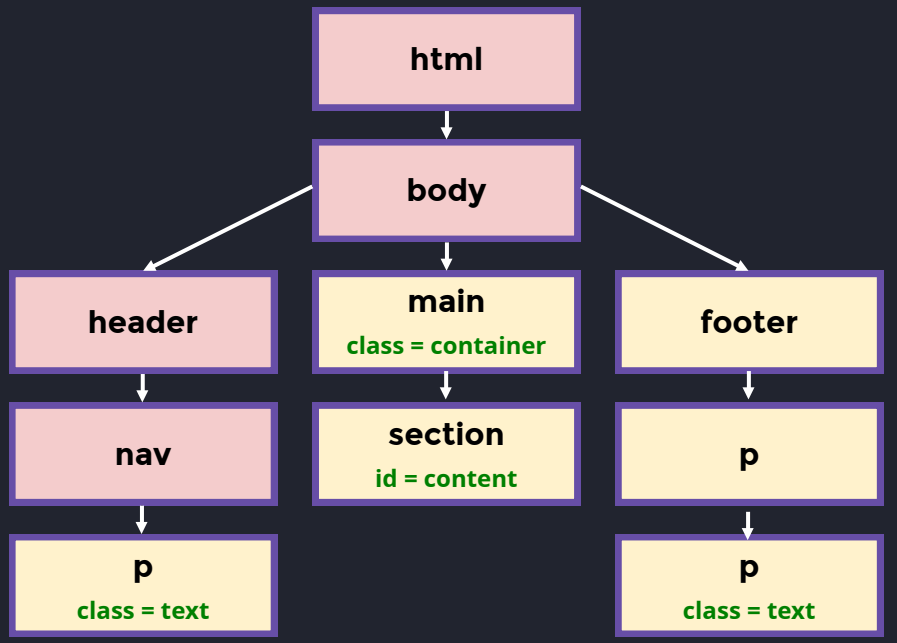
Time Complexity: `O(N)*`

Read Cost: `O(N)*`

Memory Cost: `Low`

Query Algorithm: `DFS + Hashmap`

DOM Querying: `getElementsByClassName`




```
document.getElementsByClassName("text")
```

Return type: `HTMLCollection`

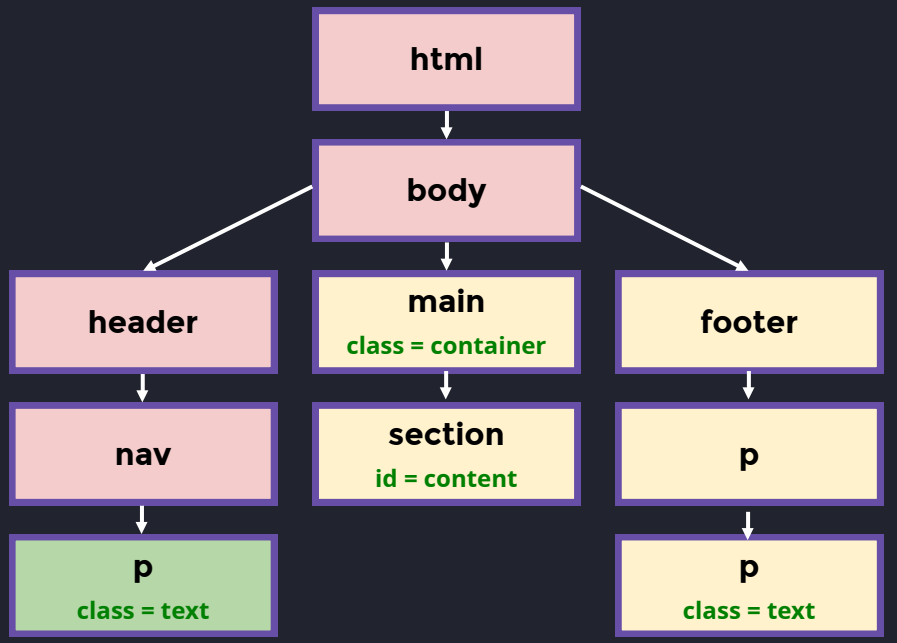
Time Complexity: `O(N)*`

Read Cost: `O(N)*`

Memory Cost: `Low`

Query Algorithm: `DFS + Hashmap`

DOM Querying: `getElementsByClassName`




```
document.getElementsByClassName("text")
```

Return type: `HTMLCollection`

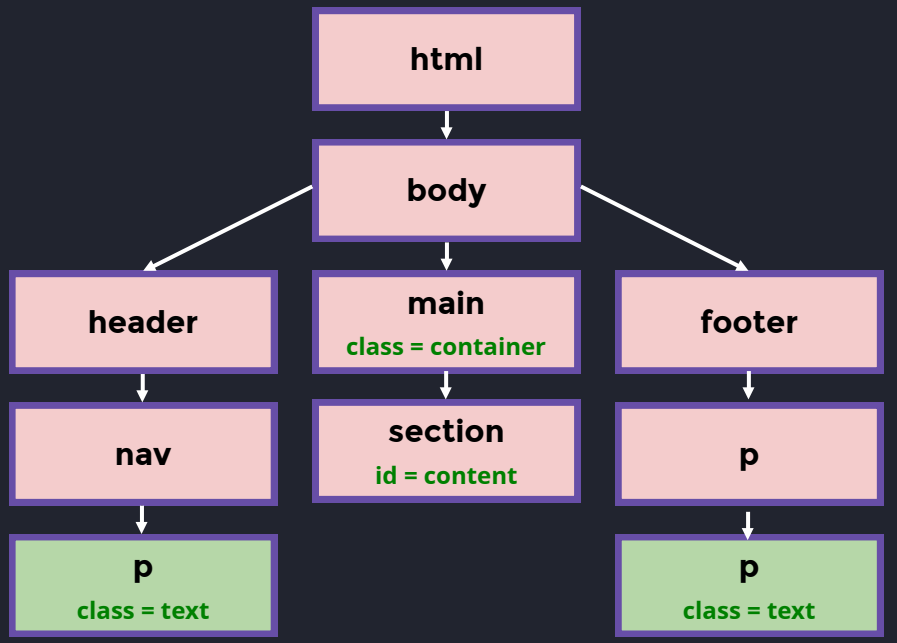
Time Complexity: `O(N)*`

Read Cost: `O(N)*`

Memory Cost: `Low`

Query Algorithm: `DFS + Hashmap`

DOM Querying: `getElementsByClassName`




```
document.getElementsByClassName("text")
```

Return type: `HTMLCollection`

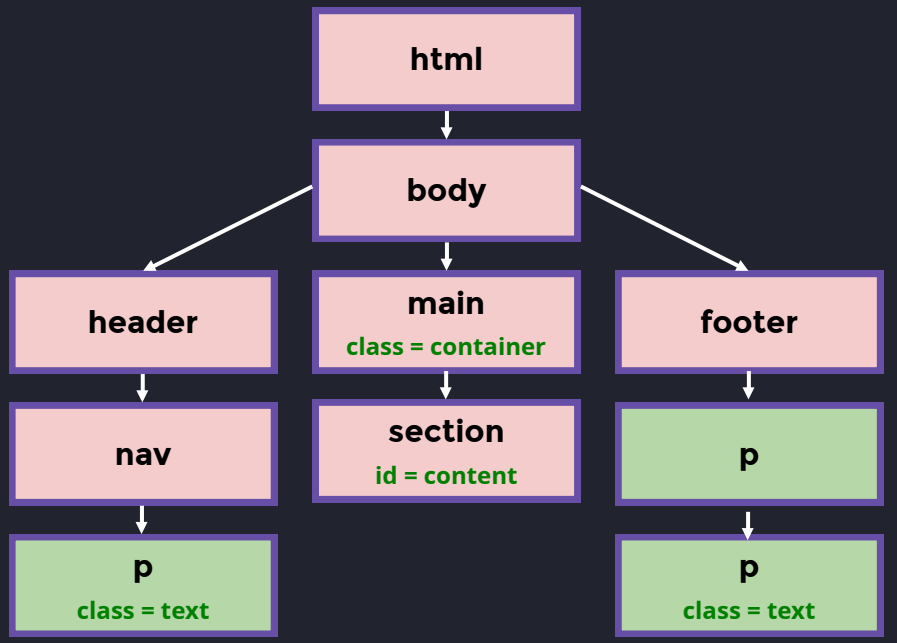
Time Complexity: `O(N)*`

Read Cost: `O(N)*`

Memory Cost: `Low`

Query Algorithm: `DFS + Hashmap`

DOM Querying: `getElementsByTagName`




```
document.getElementsByClassName("p");
```

Return type: `HTMLCollection`

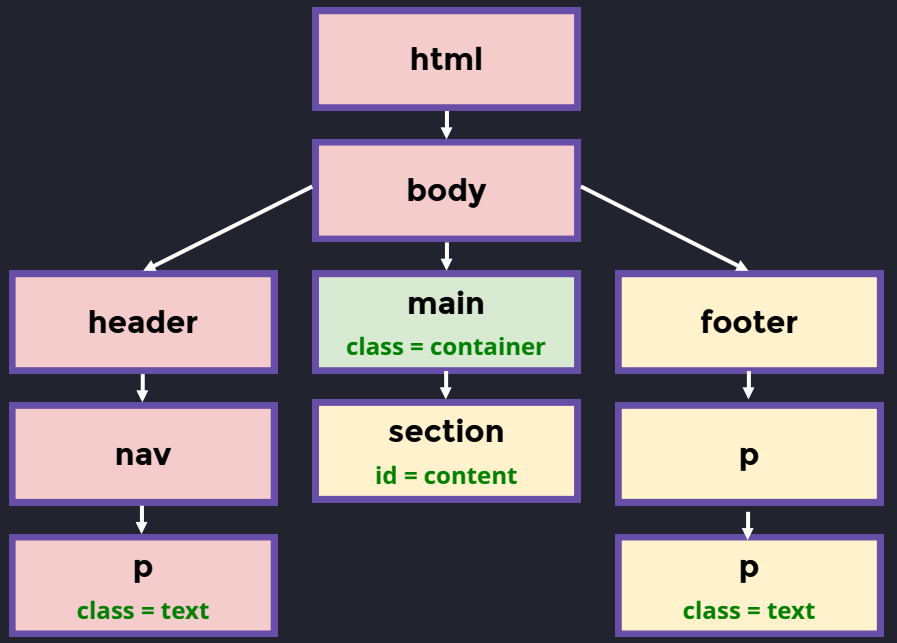
Time Complexity: `O(N)*`

Read Cost: `O(N)*`

Memory Cost: `Low`

Query Algorithm: `DFS + Hashmap`

DOM Querying: querySelector




```
document.querySelector("main.container"
```

Return type: `Element`

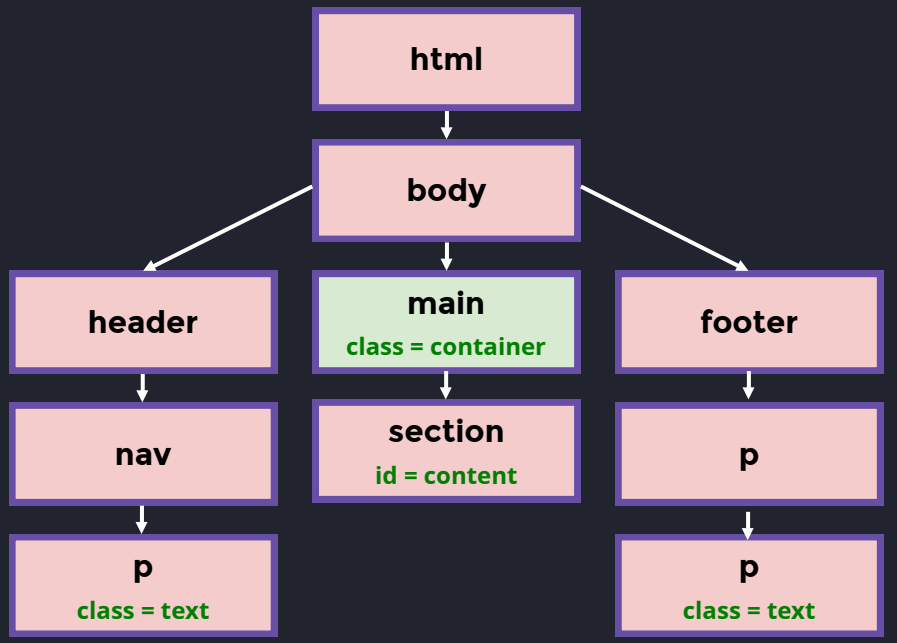
Time Complexity: `O(1)/O(N)` (depends on selector)

Read Cost: `O(1)`

Memory Cost: `O(1)`

Query Algorithm: `DFS + Hashmap`

DOM Querying: `querySelectorAll`




```
document.querySelectorAll("main.container")
```

Return type: `NodeList`

Time Complexity: `O(N)` (depends on selector)

Read Cost: `O(1)`

Memory Cost: `O(N)`

Query Algorithm: `DFS + Hashmap`

Performance overview

`getElementByID` - **provides the best performance** in terms of time and space complexity. The browser builds an index table, allowing instant access to the element.

Performance overview

`getElementByID` - **provides the best performance** in terms of time and space complexity. The browser builds an index table, allowing instant access to the element.

`getElementsByClassName`, `getElementsByTagName` - provide low-memory overhead since the browser doesn’t need to clone objects. Read access is high, although it's compensated by maintaining a live collection of elements.

Performance overview

`getElementByID` - **provides the best performance** in terms of time and space complexity. The browser builds an index table, allowing instant access to the element.

`getElementsByClassName`, **`getElementsByTagName`** - provide low-memory overhead since the browser doesn't need to clone objects. Read access is high, although it's compensated by maintaining a live collection of elements.

`querySelector` - Offers slightly worse performance compared to **`getElementByID`**. However, due to caching and browser optimizations, it is comparable on repetitive runs.

Performance overview

`getElementByID` - **provides the best performance** in terms of time and space complexity. The browser builds an index table, allowing instant access to the element.

`getElementsByClassName`, `getElementsByTagName` - provide low-memory overhead since the browser doesn't need to clone objects. Read access is high, although it's compensated by maintaining a live collection of elements.

`querySelector` - Offers slightly worse performance compared to `getElementByID`. However, due to caching and browser optimizations, it is comparable on repetitive runs.

`querySelectorAll` - Potentially high-memory overhead. However, since it returns a non-live collection, read access is cheap.

Performance best-practices

Simplify Selector: Complex selectors take time to compute, especially when querying a large set of elements. Simplify your selectors whenever possible.

```
div > div > section > .flex > .target


#container.flex > .target.
```

Performance best-practices

Simplify Selector: Complex selectors take time to compute, especially when querying a large set of elements. Simplify your selectors whenever possible.

```
div > div > section > .flex > .target

#container.flex > .target.
```

Use IDs of Core Containers: Utilize IDs for core containers whenever feasible. IDs provide direct and efficient access to specific elements.

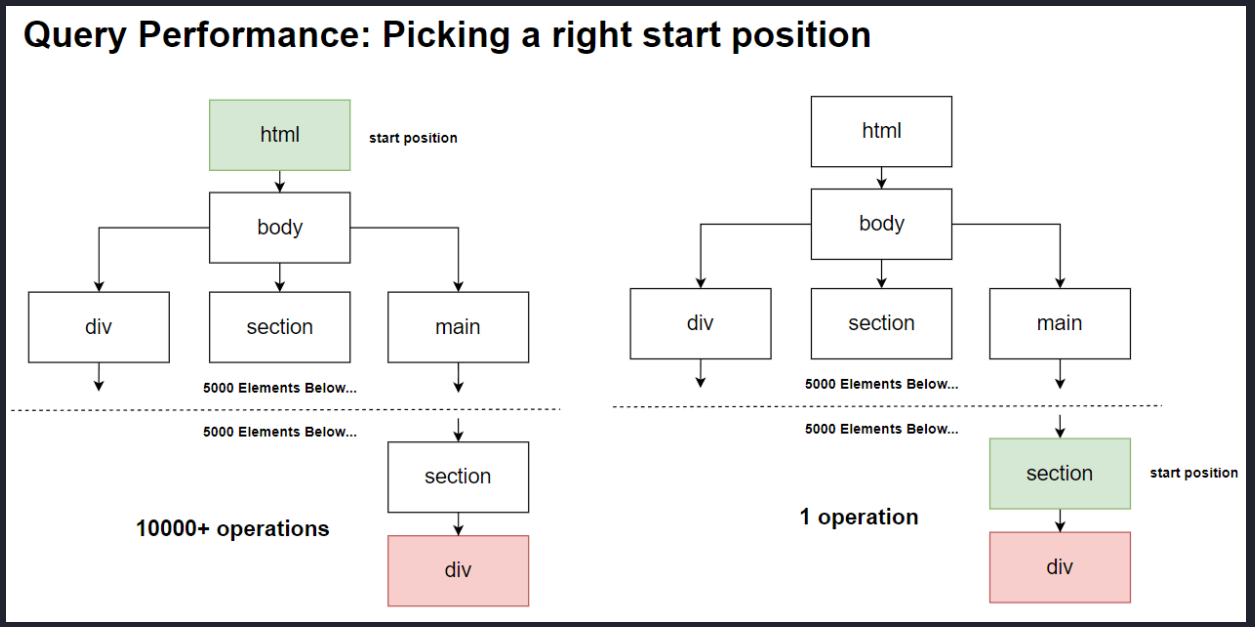
Performance best-practices

Simplify Selector: Complex selectors take time to compute, especially when querying a large set of elements. Simplify your selectors whenever possible.

```
div > div > section > .flex > .target

#container.flex > .target.
```

Use IDs of Core Containers: Utilize IDs for core containers whenever feasible. IDs provide direct and efficient access to specific elements.

Pick the Right Start Point: Choose the most appropriate starting point for your queries. Starting from a closer ancestor to the desired elements can significantly improve query performance.



Adding / Removing elements

Adding new elements

Method name	Perf Impact
`Element.prototype.innerHTML`	Extreme
`Element.prototype.insertAdjacentHTML(pos, html)`	Extreme
`Element.prototype.insertAdjacentElement(pos, html)`	High
`Element.prototype.appendChild(element)`	High

https://codepen.io/RayEuji/embed/GRLjrQJ

Removing elements

```
const el = document.getElementById('node');
el.remove();
```

```
const el = document.getElementById('node');
el.innerHTML = "";
```

Exercise time

Exercise time

```
1
2  const container = document.getElementById('container');
3
4  const html = `<article class="card">
5      <h3></h3>
6      <div class="card__body">
7        <div class='card__body__image'></div>
8        <section class='card__body__content'>
9        </section>
10     </div>
11   </article>`;
12
13
14 /**
15  * @param {string} title
16  * @param {string} body
17  *
18  * @return {HTMLElement
19  */
20 function createCardComponent(title, body) {
21   // @todo - Implement function
22 }
23
24 const component = createCardComponent(
25   "Frontend System Design: Fundamentals",
26   "This is a random body text"
27 );
28
29 container.appendChild(component);
30
31
```

Instruction

1. Checkout repository
2. Select branch - 1-dom-begin
3. Open folder - begin
4. Open file - **index.html**

Exercise: Solution 1 (not-recommended)

```
const container = document.getElementById("container");
let card = `
  <article class="card">
    <h3>_title_</h3>
    <div class="card__body">
      <div class='card__body__image'></div>
      <section class='card__body__content'>
         _content_
      </section>
    </div>
  </article>
`.trim();

card = card
  .replace("_title_", "Frontend System Design Fundamentals")
  .replace("_content_", "This is a random text");

container.innerHTML = card;
```

DocumentFragment

```
<template id="card_template">
 <article class="card">
   <h3></h3>
   <div class="card__body">
     <div class='card__body__image'></div>
     <section class='card__body__content'>
     </section>
   </div>
 </article>
</template>
```

1. **DocumentFragment** is a lightweight in-memory **HTMLElement** representation
2. Modifying it's content doesn't cause a **reflow**
3. It can be reused **many times**
4. It's **isolated** from the main **DOM Tree**
5. You can utilize HTML to create markup of component

Exercise: Solution 2

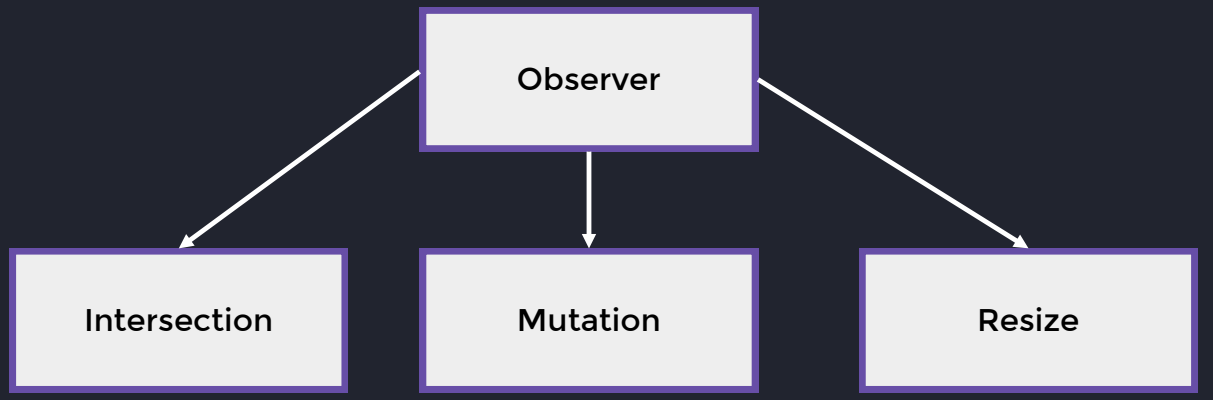
```html
<template id="card_template">
 <article class="card">
  <h3></h3>
  <div class="card__body">
    <div class='card__body__image'></div>
    <section class='card__body__content'>
    </section>
  </div>
 </article>
</template>
```

```js
const container = document.getElementById('container');

function createCardElement(title, body) {
  const template  = document.getElementById('card_template');
  const element   = template.content.cloneNode(true).firstElementChild;
  const cardTitle = element.querySelector("h3");
  const cardBody  = element.querySelector("section");
  [cardTitle.textContent, cardBody.textContent] = [title, body];
  return element;
}

container.appendChild(
  createCardElement(
    "Frontend System Design: Fundamentals",
    "This is a random content"
));

```

Summary: DOM API

Observer API

Observer API



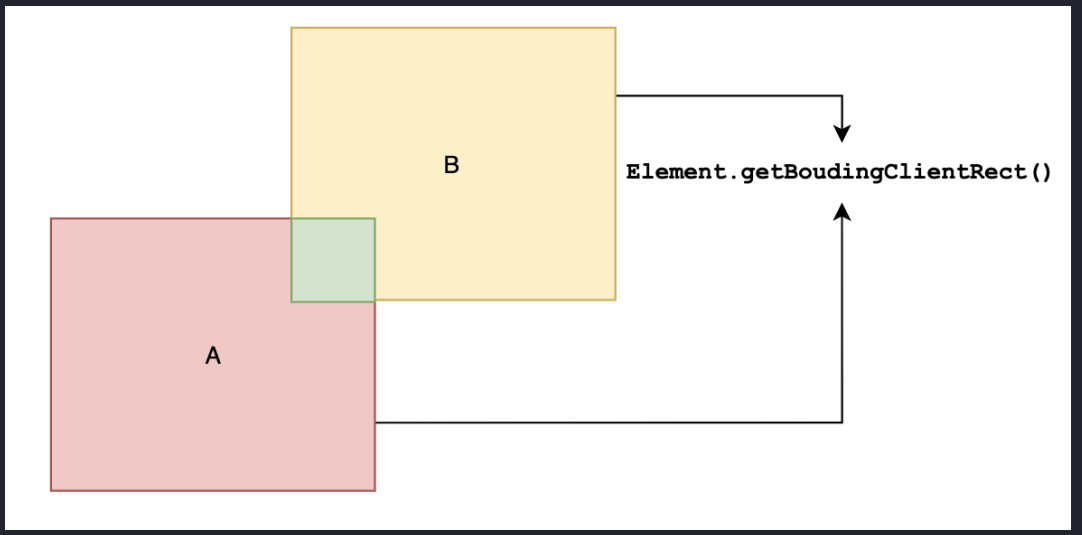
1. Virtualization
2. Lazy Components
3. Analytics
4. Dynamic UI Elements

1. Rich Text Editors
2. Drawing Tools

1. Adaptive Design
2. Charting tools
3. Drawing tools

Intersection Observer: vanilla approach

```
1  const A = document.getElementById('A');
2  const B = document.getElementByID('B');
3
4  setInterval(() => {
5    const rectA = A.getBoundingClientRect();
6    const rectB = B.getBoundingClientRect();
7    const isIntersection = rectA.bottom > rectB.top
8                        && rectA.right > rectB.left
9                        && rectA.top < rectB.bottom
10                       && rectA.left < rectB.right;
11   if(isIntersection) {
12     // do stuff
13   }
14 }, 50);
```

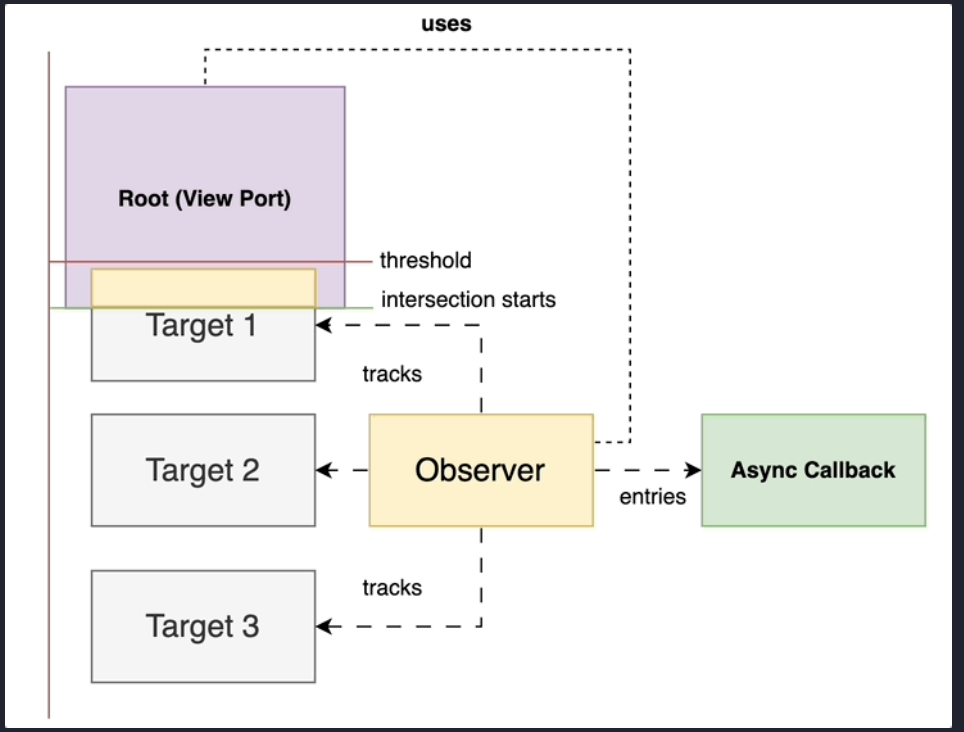


Intersection Observer

`target:` element we're tracking

`config:`

- `-> root` the "**window**" that we check the intersection against
- `-> threshold` the minimal intersection ratio that is required to trigger the callback
- `-> callback` function to execute on intersection



Intersection Observer: Creation

```
const callback = () => {};
const observer = new IntersectionObserver(
    callback,
    {
      root: document.getElementById("container"),
      threshold: 0.1
    }
);
```

config:

- **root** the "**window**" that we check the intersection against
- **threshold** the minimal intersection ratio that is required to trigger the callback
- **callback** function to execute on intersection

Intersection Observer: Creation

```
const callback = () => {};
const observer = new IntersectionObserver(
    callback,
    {
      root: document.getElementById("container"),
      threshold: 0.1
    }
);
```

config:

root the "**window**" that we check the intersection against

threshold the minimal intersection ratio that is required to trigger the callback

callback function to execute on intersection

```
const callback = (entries, observer) => {
  entries.forEach(entry => {
    if(entry.isIntersecting) {
      // logic is here
    }
  });
);
```

callback:

IntersectionObserverEntry[] - array of entries that were triggered.

Entry may intersect or may stop intersecting with object, so it's important to check **isIntersecting** property

observer - instance of Observer

Intersection Observer: Creation

```
const callback = () => {};
const observer = new IntersectionObserver(
    callback,
    {
      root: document.getElementById("container"),
      threshold: 0.1
    }
);
```

config:

root the "**window**" that we check the intersection against

threshold the minimal intersection ratio that is required to trigger the callback

callback function to execute on intersection

```
const callback = (entries, observer) => {
  entries.forEach(entry => {
    if(entry.isIntersecting) {
      // logic is here
    }
  });
};
```

callback:

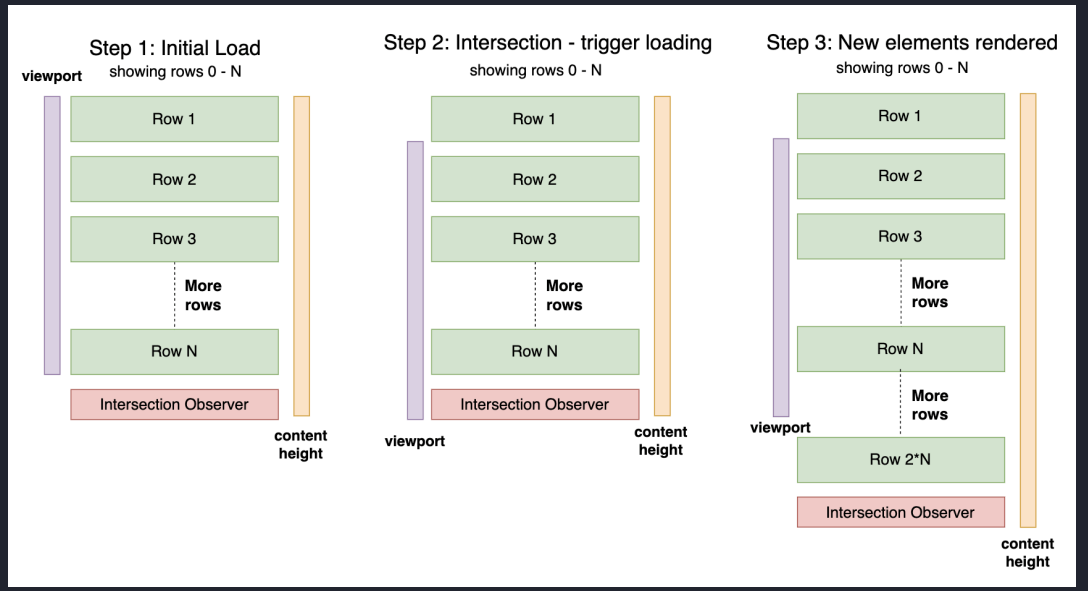
IntersectionObserverEntry[] - array of entries that were triggered.

Entry may intersect or may stop intersecting with object, so it's important to check **isIntersecting** property

observer - instance of Observer

```
observer.observe(
  document.getElementById('target');
)
```

Exercise Time: Intersection Observer



https://codepen.io/RayEuji/embed/wvZJvKN

Branch name: `2-intersection-observer-begin`

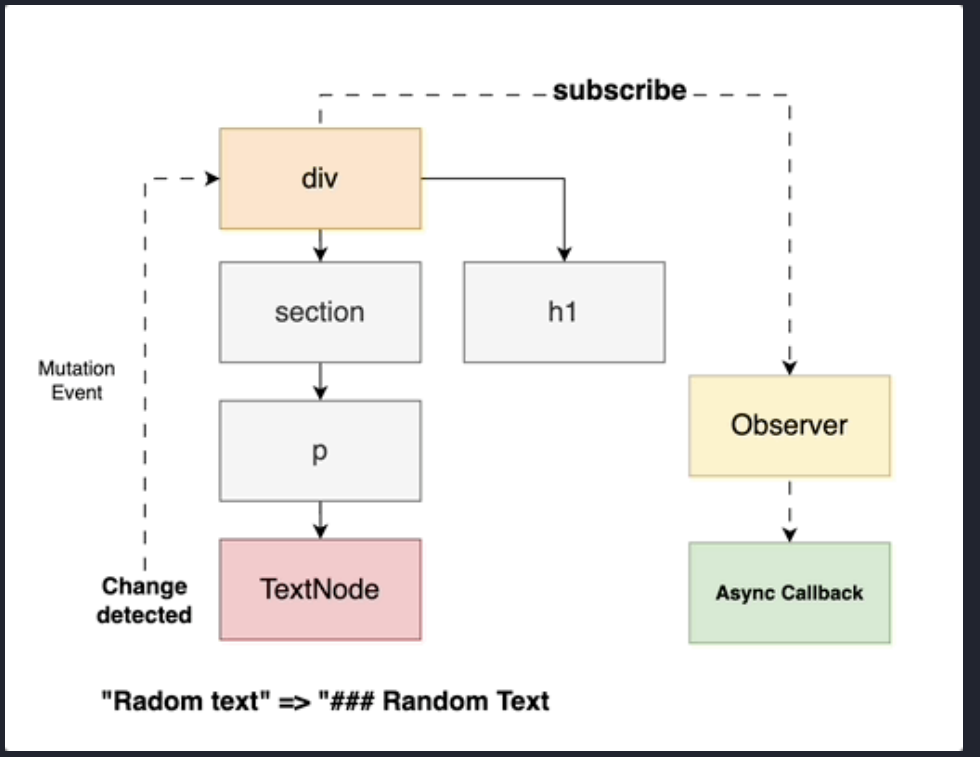
Solution: Intersection Observer

```
let page = 0;
const observer = new IntersectionObserver(async ([entry]) => {
   if(entry.isIntersecting) {
       const data = await db.getPage(page++);
       const fragment = new DocumentFragment();
       for (const datum of data) {
           const card = createCardElement(datum.title, datum.body)
           fragment.appendChild(card);
       }
       list.appendChild(fragment);
      }
    }, { threshold: 0.2 })

 observer.observe(observerElement);
```

https://codepen.io/RayEuji/embed/wvZJvKN

Mutation Observer

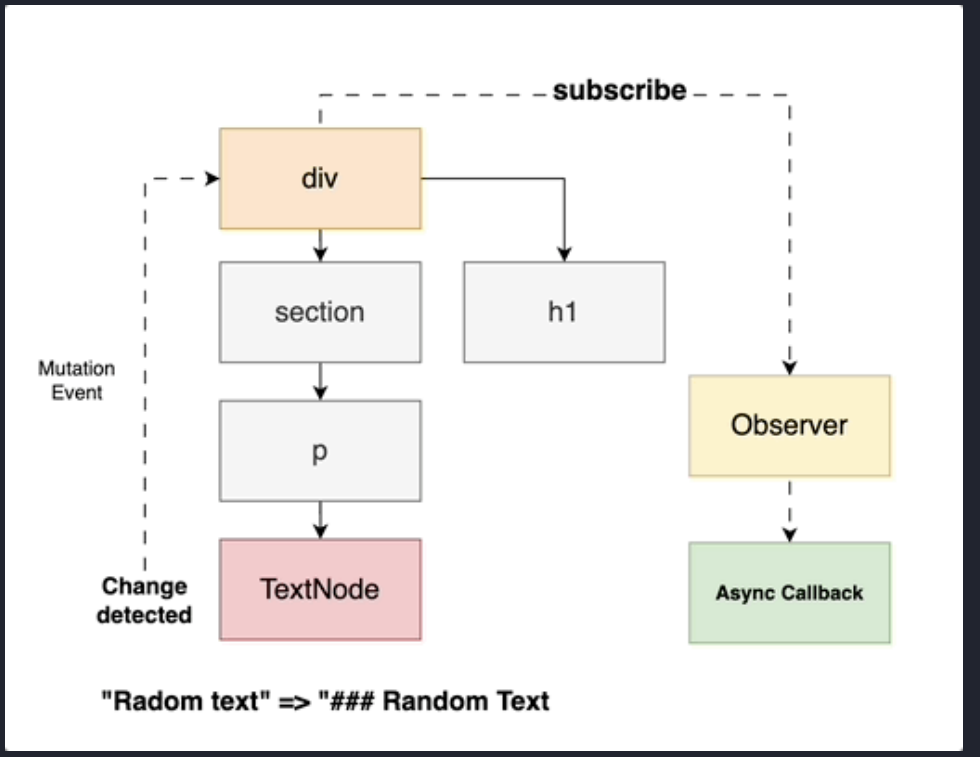
```
function callback (mutations) {}

let observer = new MutationObserver(callback);

observer.observe(targetNode, observerOptions);
```



Mutation Observer

```
function callback (mutations) {}

let observer = new MutationObserver(callback);

observer.observe(targetNode, observerOptions);
```

```
const observerOptions = {
  childList: false,                 // changes in the direct children of nod
  attributes: false,                // changes in element attirubutes
  characterData: true,              // changes in textContent of the Element
  subtree: false,                   // changes in any descendants
  attributeFilter: ['one', 'two'],  // attributes array to observe
```

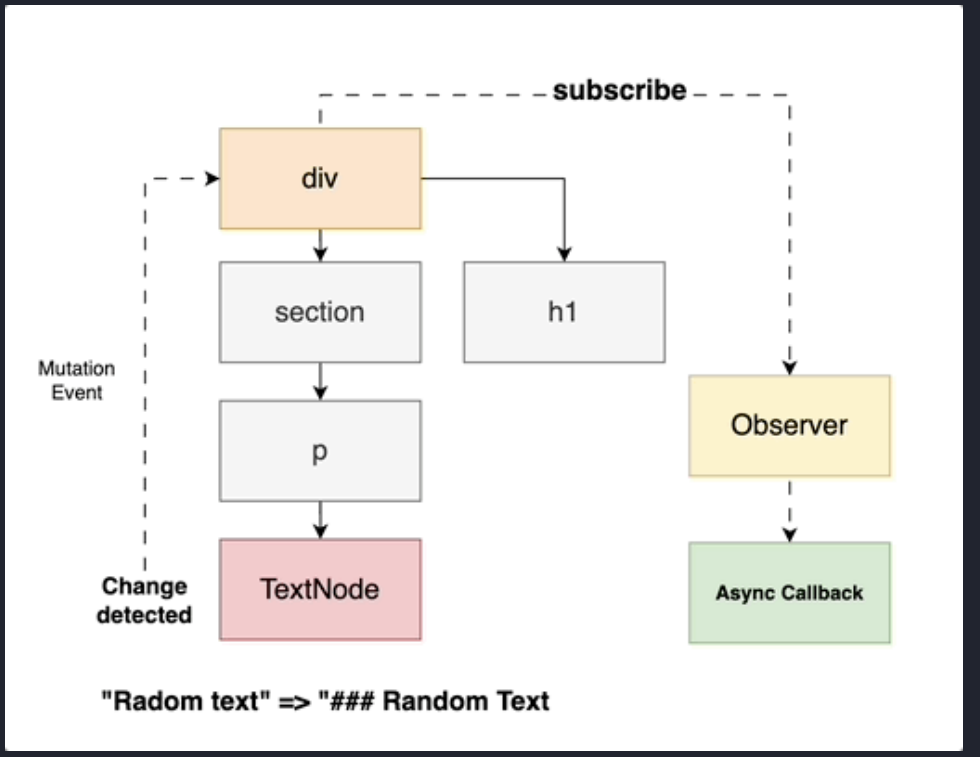


Mutation Observer

```
function callback (mutations) {}

let observer = new MutationObserver(callback);

observer.observe(targetNode, observerOptions);
```

```
const observerOptions = {
  childList: false,                 // changes in the direct children of nod
  attributes: false,                // changes in element attirubutes
  characterData: true,              // changes in textContent of the Element
  subtree: false,                   // changes in any descendants
  attributeFilter: ['one', 'two'],  // attributes array to observe
```

```
type MutationRecord ={
  type: "attributes"
  | "characterData"
  | "childList";              // What to track
  target: Node;               // Which object to track
  addedNodes: NodeList;       // Nodes that were added with mutation
  removedNodes: NodeList;     // Nodes that were added with mutation
  oldValue: ?string;          // Previous value
`
```



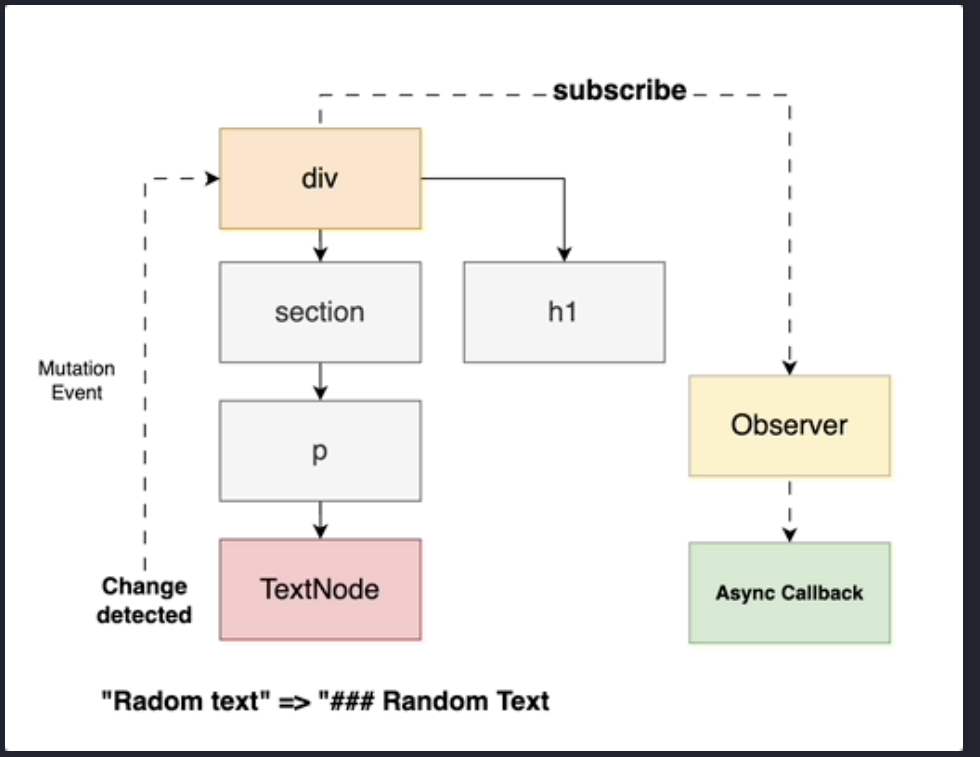
Mutation Observer

```
function callback (mutations) {}

let observer = new MutationObserver(callback);

observer.observe(targetNode, observerOptions);
```

```
const observerOptions = {
  childList: false,                 // changes in the direct children of nod
  attributes: false,                // changes in element attirubutes
  characterData: true,              // changes in textContent of the Element
  subtree: false,                   // changes in any descendants
  attributeFilter: ['one', 'two'],  // attributes array to observe
```

```
type MutationRecord ={
  type: "attributes"
  | "characterData"
  | "childList";              // What to track
  target: Node;               // Which object to track
  addedNodes: NodeList;       // Nodes that were added with mutation
  removedNodes: NodeList;     // Nodes that were added with mutation
  oldValue: ?string;          // Previous value
`
```

```
function callback (mutations) {
  for(let mutation of mutations) {
    if(mutation.type === 'characterData') {
      // Your logic
    }
  }
`
```



Exercise time: Mutation Observer

1. Add `contenteditable` attribute to `section` with class `card__body__content`
2. Create `MutationObserver` with the following parameters that would allow us to track text changes in **TextNodes**:
   1. `characterData: true`
   2. `subtree: true`
3. Implement callback for `MutationObserver`
4. Observer text section for each card
5. Use single observer for all rendered cards

https://codepen.io/RayEuji/embed/ExJWaEO

Branch name: 3-mutation-observer-start

Solution: Mutation Observer

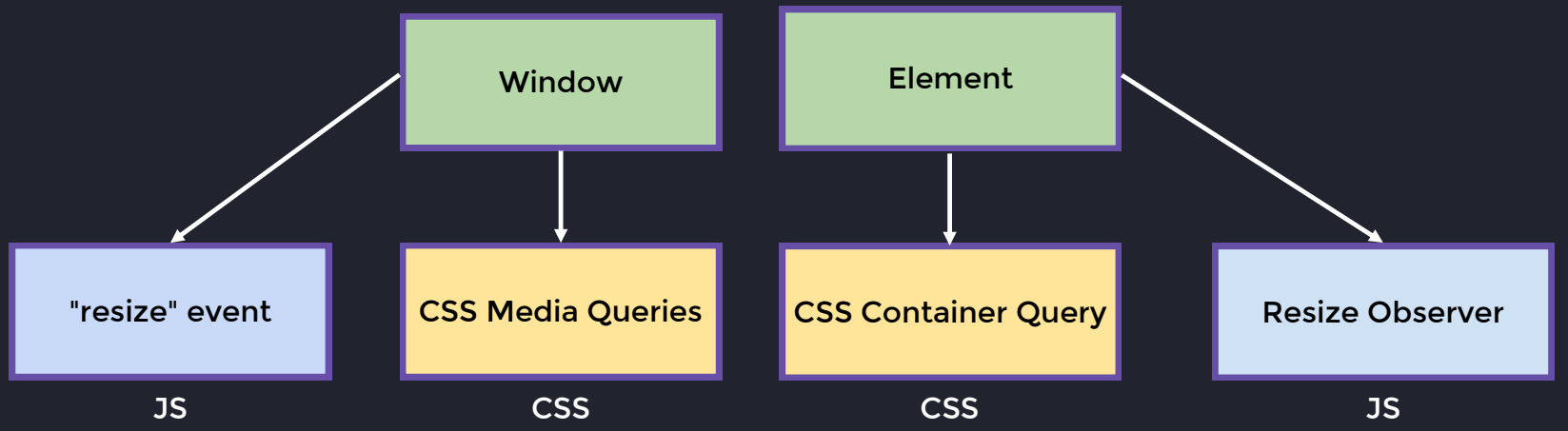
```
const mutationObserver = new MutationObserver((mutations) => {
    for (const {target, type} of mutations) {
        if (
            type === 'characterData' &&
            SUPPORTED_ELEMENTS.has(target?.textContent)
        ) {
            const element = getHeading(target);
            target.replaceWith(element);
            element.focus();
        }
    }
})
```

Branch name: 3-mutation-observer-end

https://codepen.io/RayEuji/embed/ExJWaEO

Resize Observer

How to track resize of elements



1. Fires whenever "Window" size changes
2. Can fire too many times and may introduce performance issues

1. Allows to change the appearance of the UI based on container size
2. Best approach performance wise, however to way to run a callback

1. Allows to change the appearance of the UI based on container size
2. Allows to run a JS callback on resize

When to use each

	Performance	Callback	Element tracking	When to use
CSS Media Queries	Super Fast ✅	❌	❌	General adaptive layout cases when there is no need to run any JS Code due to main Window resize
CSS Container Query	Super Fast ✅	❌	✅	Combine with CSS Media Queries for general adaptive layout cases when running JavaScript code is unnecessary.
"resize" event	Potentially Slow ❌	✅	❌ (window only)	**Use with caution:** If ResizeObserver is not supported and you need to run logic when the main window size changes, be wary.
Resize Observer	Fast ✅ (10x faster then resize)	✅	✅	Combine with **CSS Media Queries** for general adaptive layout cases when running JavaScript code is unnecessary.

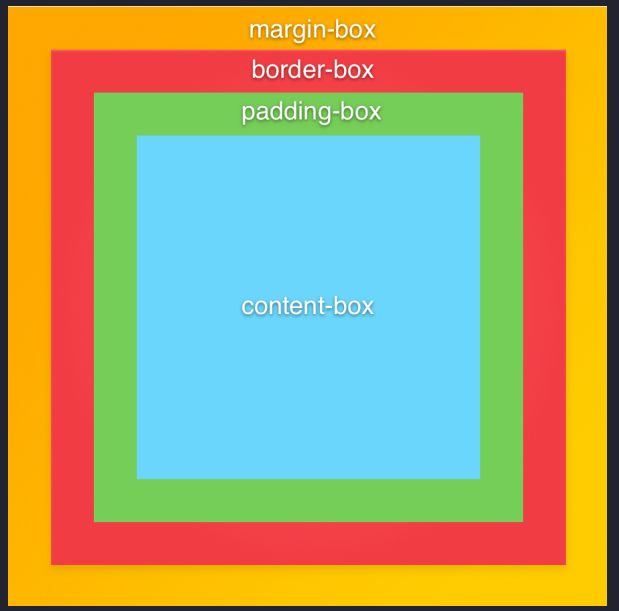
ResizeObserver API

```
const callback = () => {};
const observer = new ResizeObserver(callback);
```

ResizeObserver API

```
const callback = () => {};
const observer = new ResizeObserver(callback);
```

```
const observerOptions = {
  box:   'content-box' | //  Size of the content area as defined in CSS
         'border-box'    //  Size of the border area as defined in CSS
};
observer.observe(
  document.body,
  { box: 'border-box' }
`
```

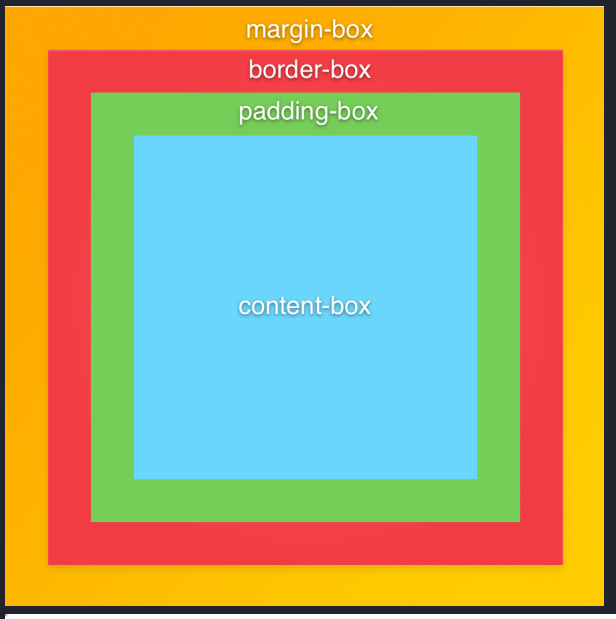


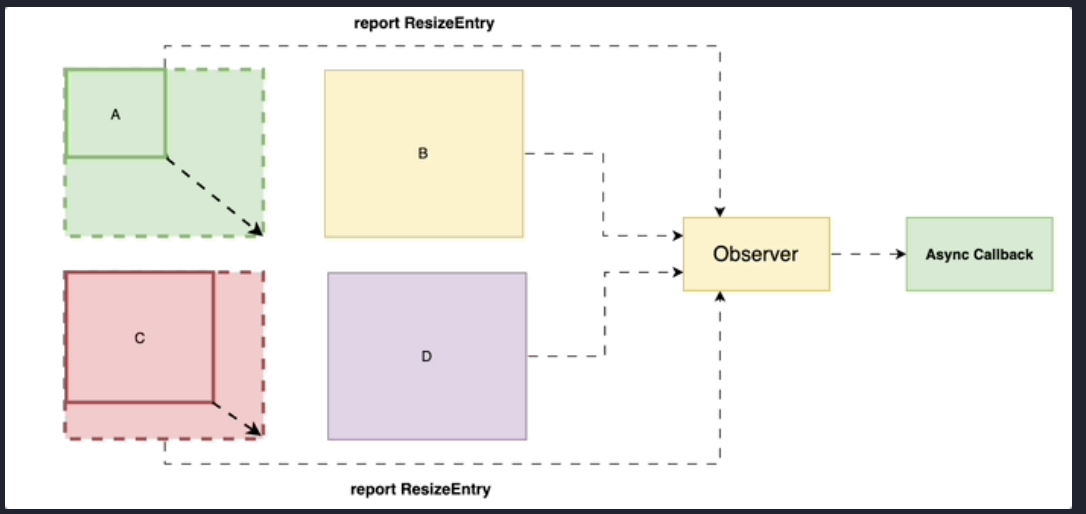
ResizeObserver API

```
const callback = () => {};
const observer = new ResizeObserver(callback);
```

```
const observerOptions = {
  box:   'content-box' | //  Size of the content area as defined in CSS
         'border-box'    //  Size of the border area as defined in CSS
};
observer.observe(
  document.body,
  { box: 'border-box' }
`
```

```
type ResizeObserverEntry = {
  borderBoxSize: Array<Box>;
  contentBoxSize: Array<Box>;
  contentRect: DOMRectReadOnly;
  target: Element;
```





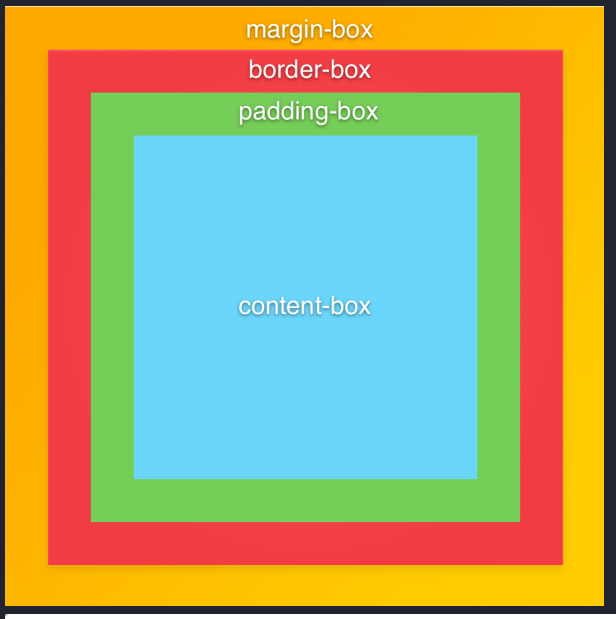
ResizeObserver API

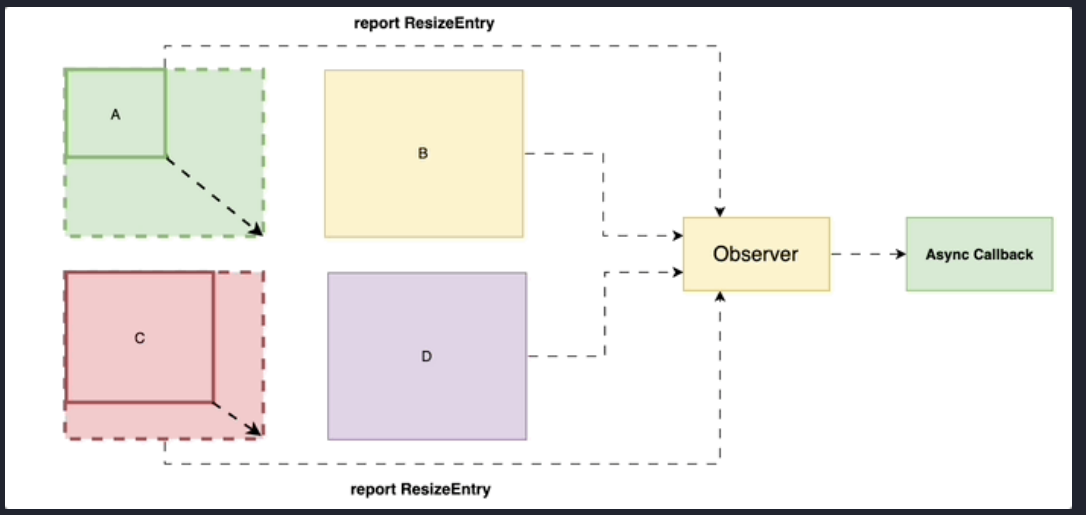
```
const callback = () => {};
const observer = new ResizeObserver(callback);
```

```
const observerOptions = {
  box:   'content-box' | //  Size of the content area as defined in CSS
         'border-box'    //  Size of the border area as defined in CSS
};
observer.observe(
  document.body,
  { box: 'border-box' }
`
```

```
type ResizeObserverEntry = {
  borderBoxSize: Array<Box>;
  contentBoxSize: Array<Box>;
  contentRect: DOMRectReadOnly;
  target: Element;
```

```
const callback = (entries) => {
  for (const entry of entries) {
   const [width, height] = [
     entry.borderBoxSize[0].inlineSize,
     entry.borderBoxSize[0].blockSize
   ];
   // do stuff
  }
}).
```





Exercise time: ResizeObserver

1. You're given 4 resizable boxes - A, B, C, D
2. Create ResizeObserver so each box becomes a circle when width and height of the box is less than 150px

```html
<main id="container">
    <div class="box">
        A
    </div>
    <div class="box">
        B
    </div>
    <div class="box">
        C
    </div>
    <div class="box">
        D
    </div>
</main>
```

https://codepen.io/RayEuji/embed/QWPprNE

Branch-name: 4-resize-observer-start

Solution: ResizeObserver

```
const observer = new ResizeObserver((entries) => {
    for(let entry of entries) {
        const {
          borderBoxSize: [
            {
              inlineSize, blockSize
            }
          ],
          target
        } = entry;
        if(inlineSize < 150 && blockSize < 150) {
           target.style.borderRadius = "100%";
           target.style.borderWidth = "4px"
       } else {
           target.style.borderRadius = "unset";
           target.style.borderWidth = "unset"
        }
    }
 })
 const elements = document.querySelectorAll('.box');
 elements.forEach(el => observer.observe(el));
```

https://codepen.io/RayEuji/embed/QWPprNE

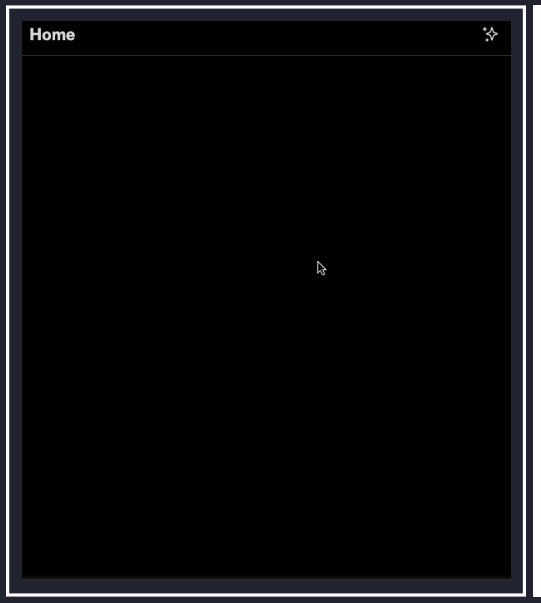
Summary: Observer API

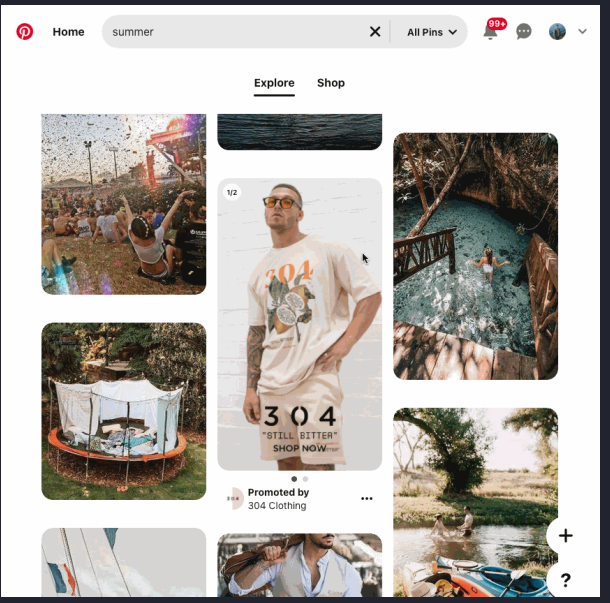
Virtualisation

Virtualization is a **UI optimization technique** that involves maintaining a data in memory while rendering only a limited subset, often referred to as a **'sliding window'**

Goals of the pattern:

1. Minimize number of elements rendered in the DOM Tree
2. Minimize number of DOM Mutations
3. Minimize CPU & Memory usage that is required to maintain a DOM Tree

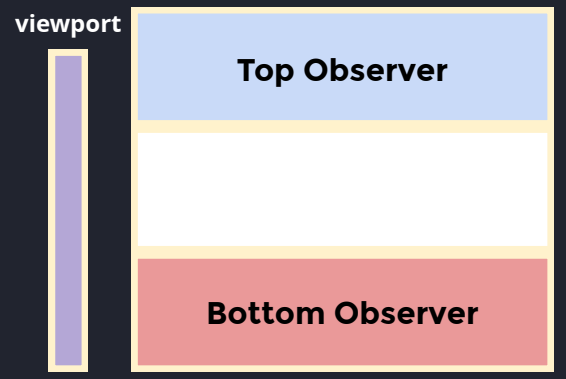




Virtualisation: High-level overview

Pool Initiation Phase

Step 1 - Initial Loading

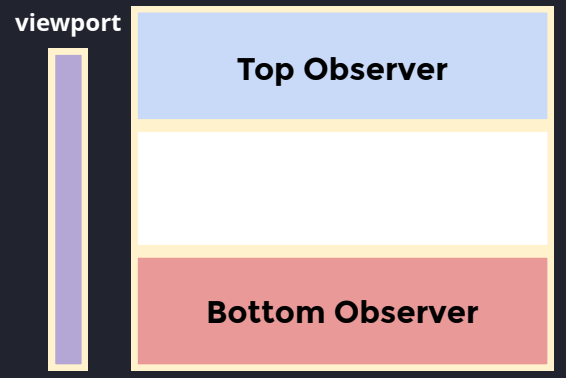


Empty List - Bottom Observer is triggered

Virtualisation: High-level overview

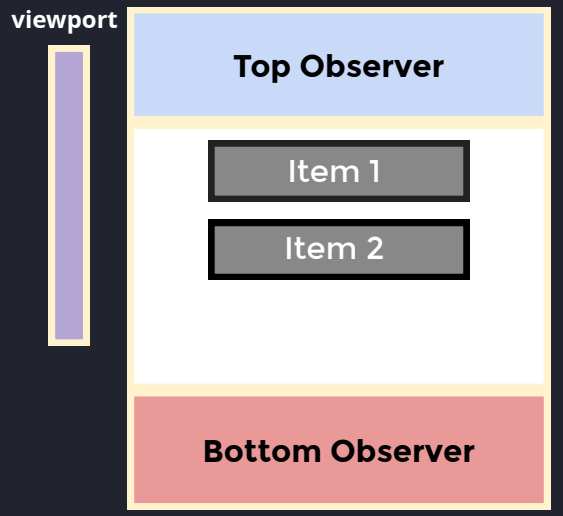
Pool Initiation Phase

Step 1 - Initial Loading



Empty List - Bottom Observer is triggered

Step 2 - Rendering first page

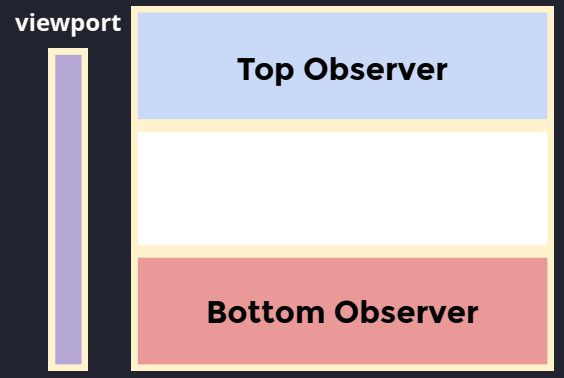


First page of data is rendered

Virtualisation: High-level overview

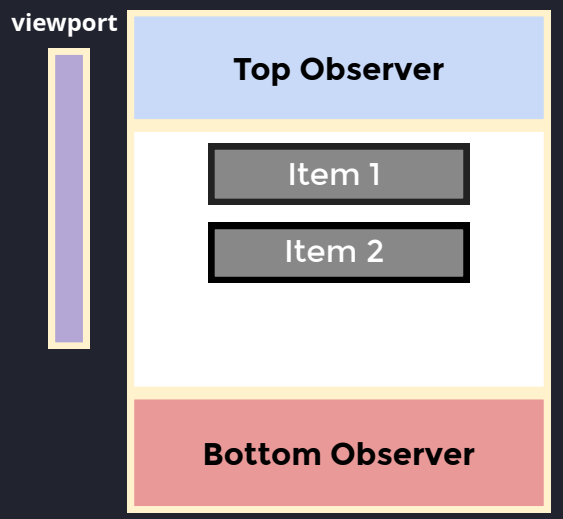
Pool Initiation Phase

Step 1 - Initial Loading



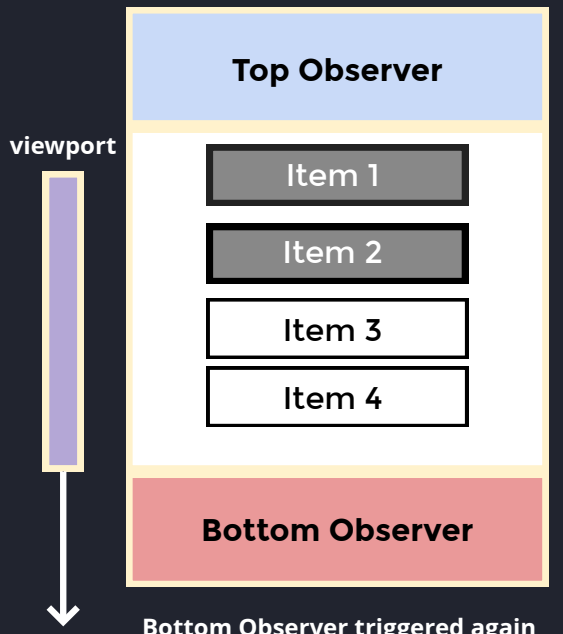
Empty List - Bottom Observer is triggered

Step 2 - Rendering first page



First page of data is rendered

Step 3 - Scrolldown

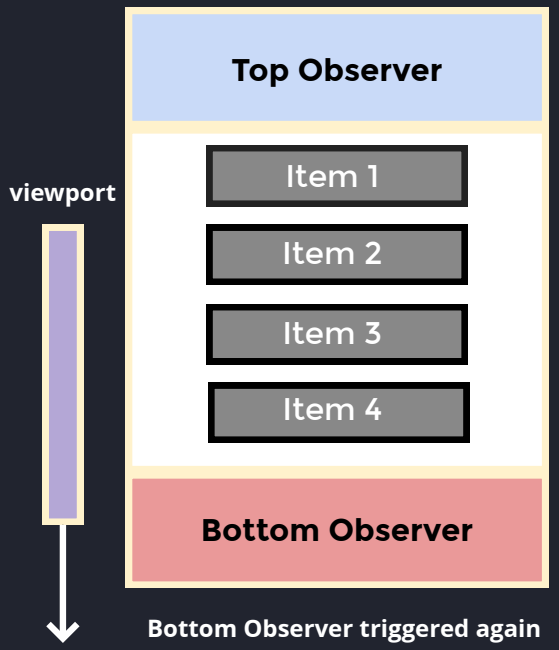


Bottom Observer triggered again

Virtualisation: High-level overview

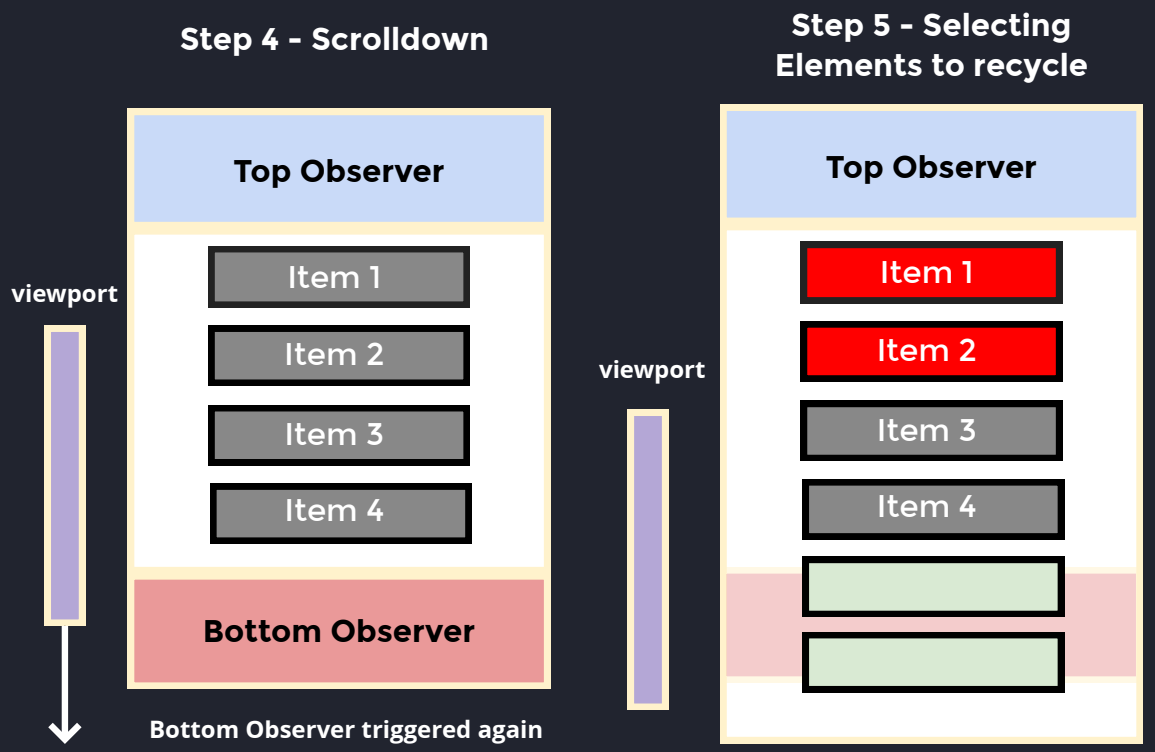
Recycling Phase

Step 4 - Scrolldown



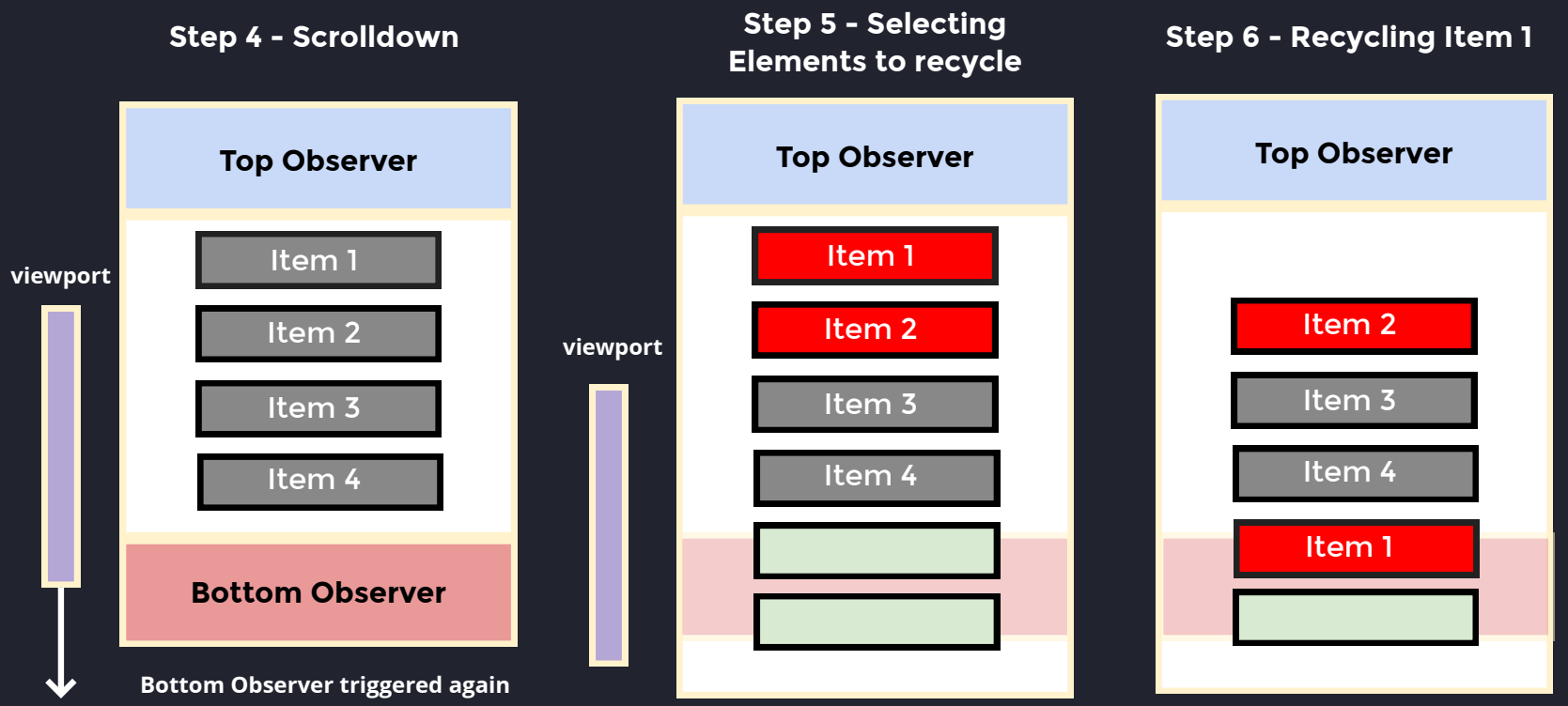
Virtualisation: High-level overview

Recycling Phase



Virtualisation: High-level overview

Recycling Phase



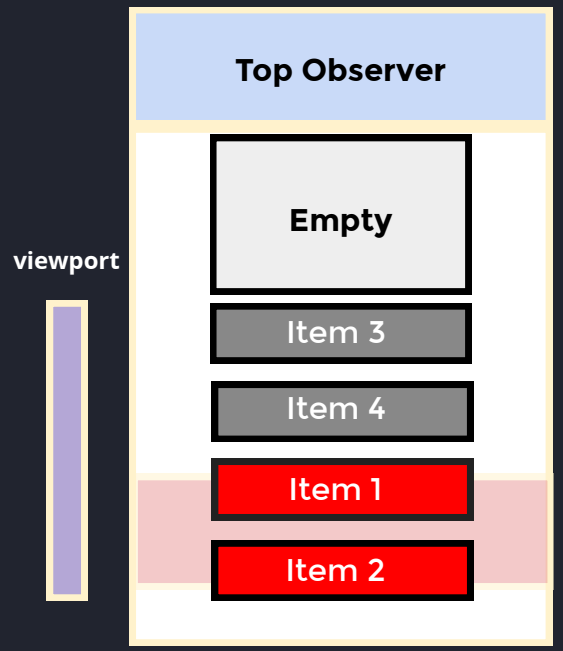
Virtualisation: High-level overview

Recycling Phase

Virtualisation: High-level overview

Update Phase

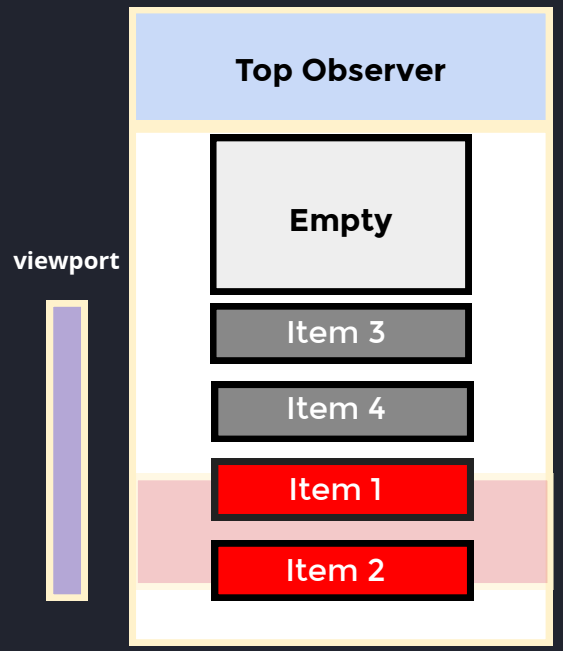
Step 6 - Recycling finished



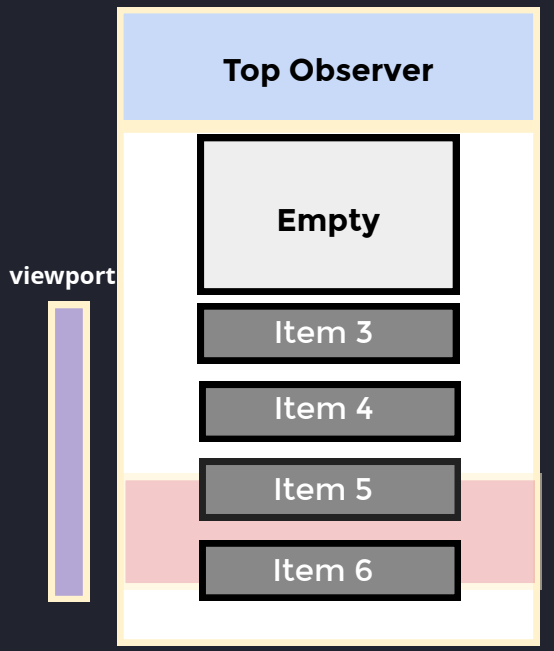
Virtualisation: High-level overview

Update Phase

Step 6 - Recycling finished



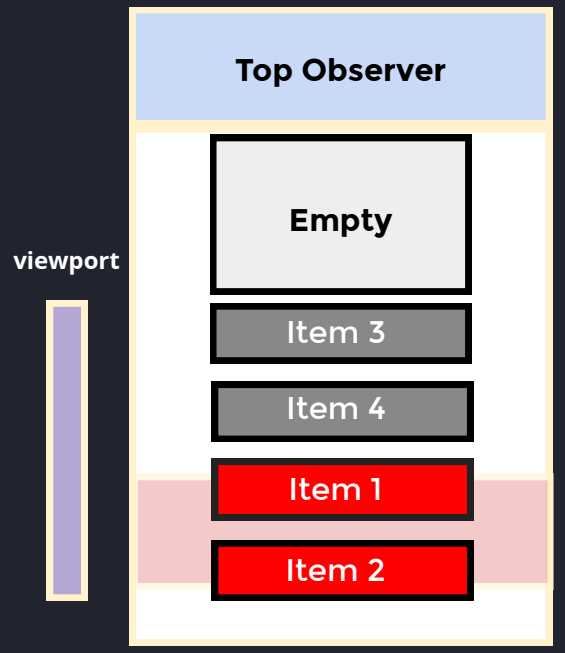
Step 7 - Update the data



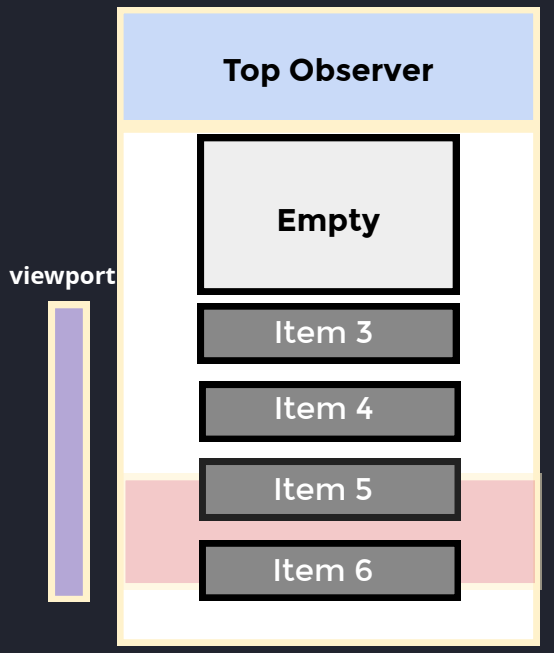
Virtualisation: High-level overview

Update Phase

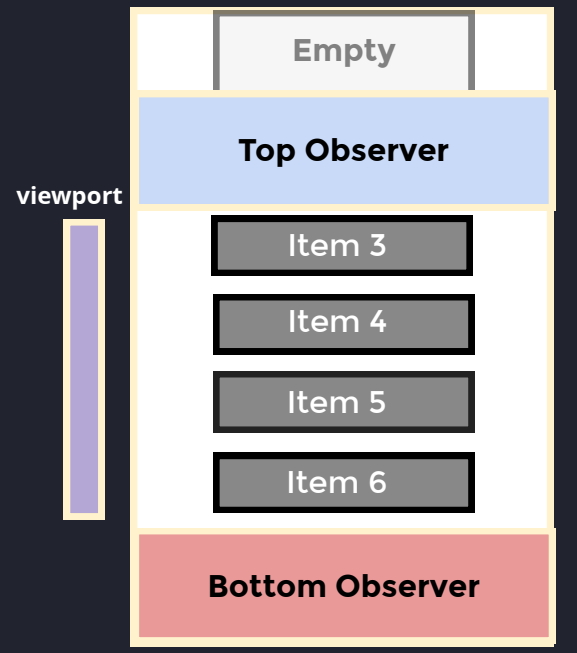
Step 6 - Recycling finished



Step 7 - Update the data



Step 8 - Update Observers position



Virtualisation: Live Coding

1. Check-out the branch - **5.1-virtualization-skeleton-start**
2. Open a filename - **5-virtualisation/5-1-skeleton/begin/virtual-list.js**
3. Repo url: **https://shorturl.at/Hc6p0**

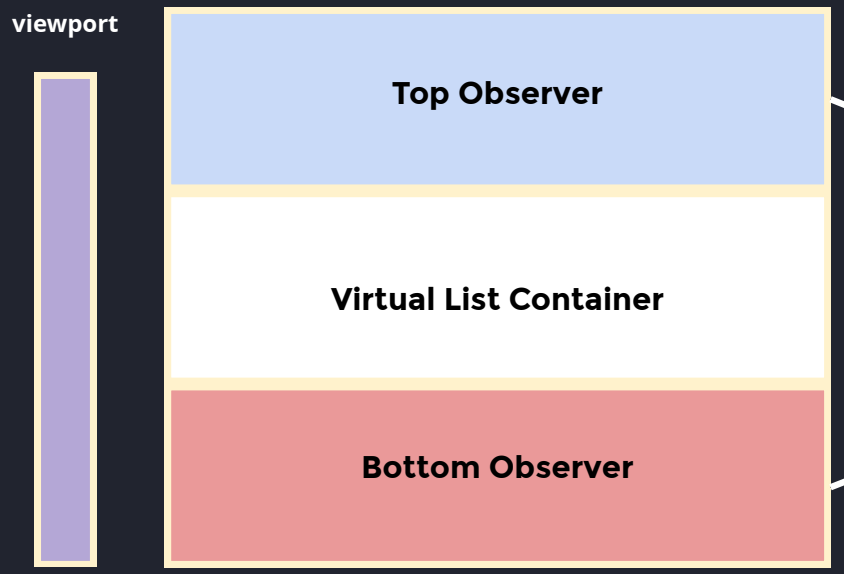
Branch

- 5-1-virtualisation-skeleton-end
- 5-1-virtualisation-skeleton-start
- 5-6-scroll-preservation-end
- 5-6-scroll-preservation-begin
- 5-5-top-observer-end
- 5-5-top-observer-begin
- 5-4-2-elements-pool-and-recycing-rendering-end
- 5-4-2-elements-pool-and-recycing-rendering-start
- 5-4-1-elements-pool-and-recycing-preparation-end
- 5-4-1-elements-pool-and-recycing-preparation-begin
- 5-3-property-model-and-loading-end
- 5-3-property-model-and-loading-begin
- 5-2-observer-handling-end
- 5-2-observer-handling-start

Virtualisation: Live Coding

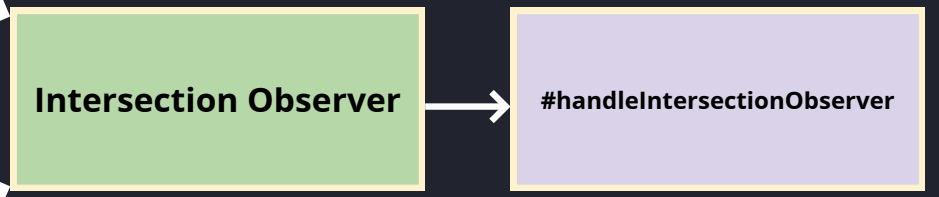
Part 1 and 2: Skeleton

Step 1 - Skeleton



Solution: **5.1-virtualization-skeleton-end**

Step 2 - Observer creation

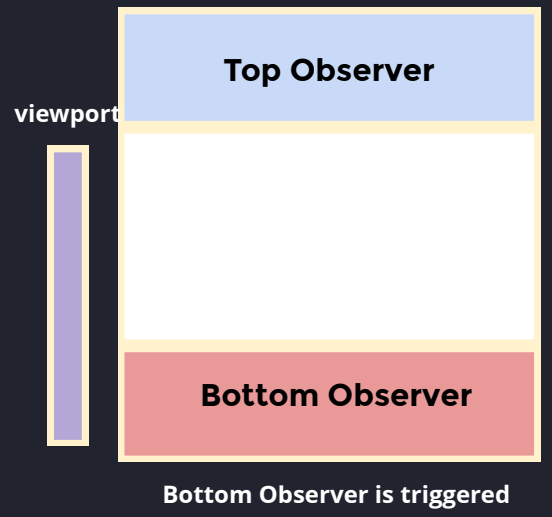


Solution: **5.2-observer-handling-end**

Virtualisation: Live Coding

Part 3: Loading data and filling up the pool

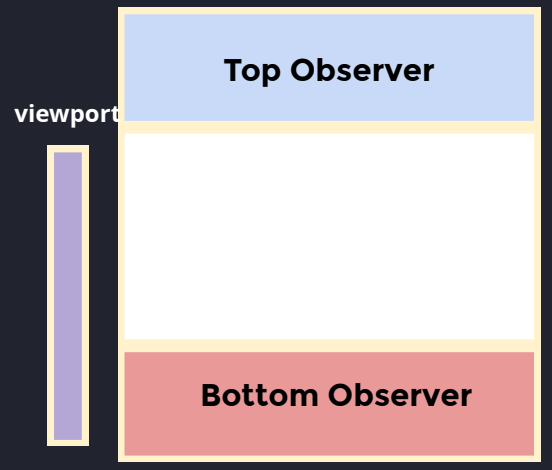
Step 3 - Scrolldown - Load new data



Virtualisation: Live Coding

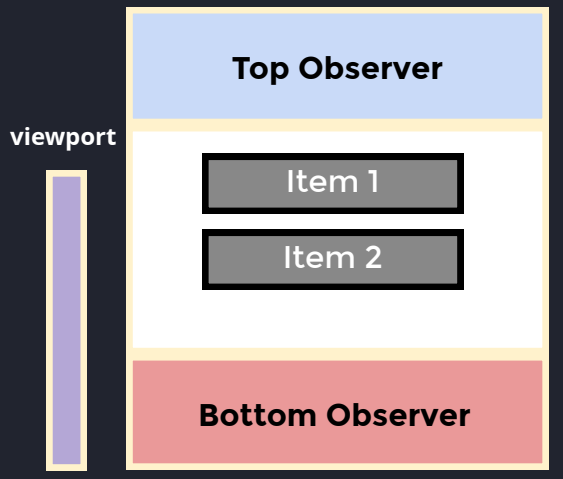
Part 3: Loading data and filling up the pool

Step 3 - Scrolldown - Load new data



Bottom Observer is triggered

Step 4 - Load more data



Bottom Observer triggered again

start

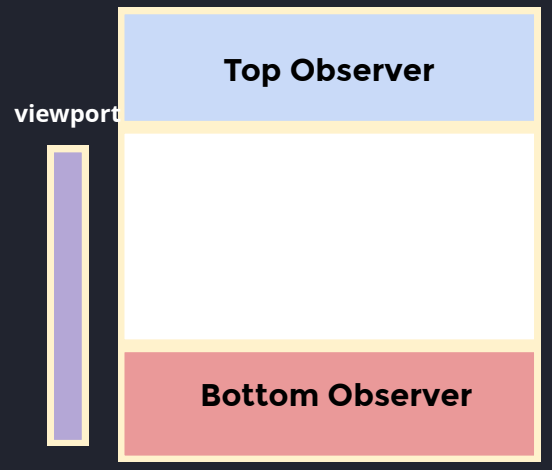
Page 1

end

Virtualisation: Live Coding

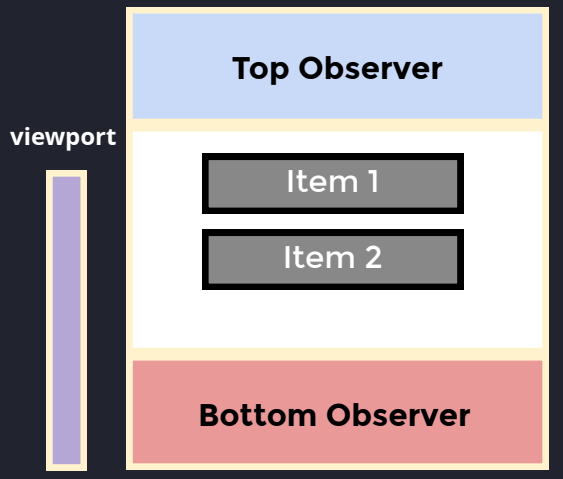
Part 3: Loading data and filling up the pool

Step 3 - Scrolldown - Load new data



Bottom Observer is triggered

Step 4 - Load more data



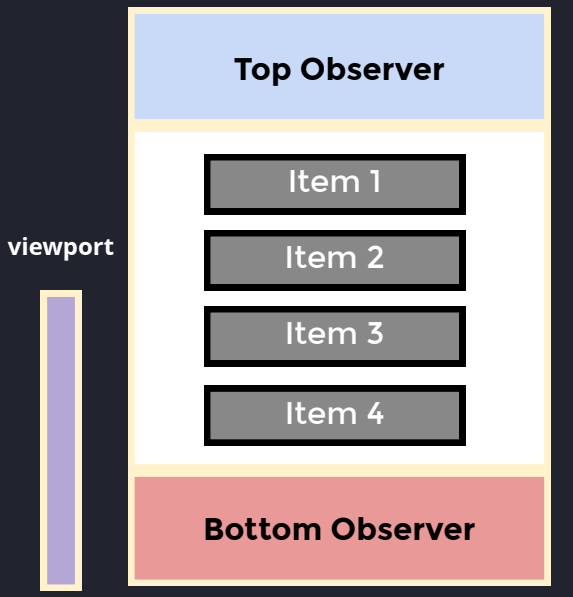
Bottom Observer triggered again

start

Page 1

end

Step 5 - Load more data



Bottom Observer triggered again

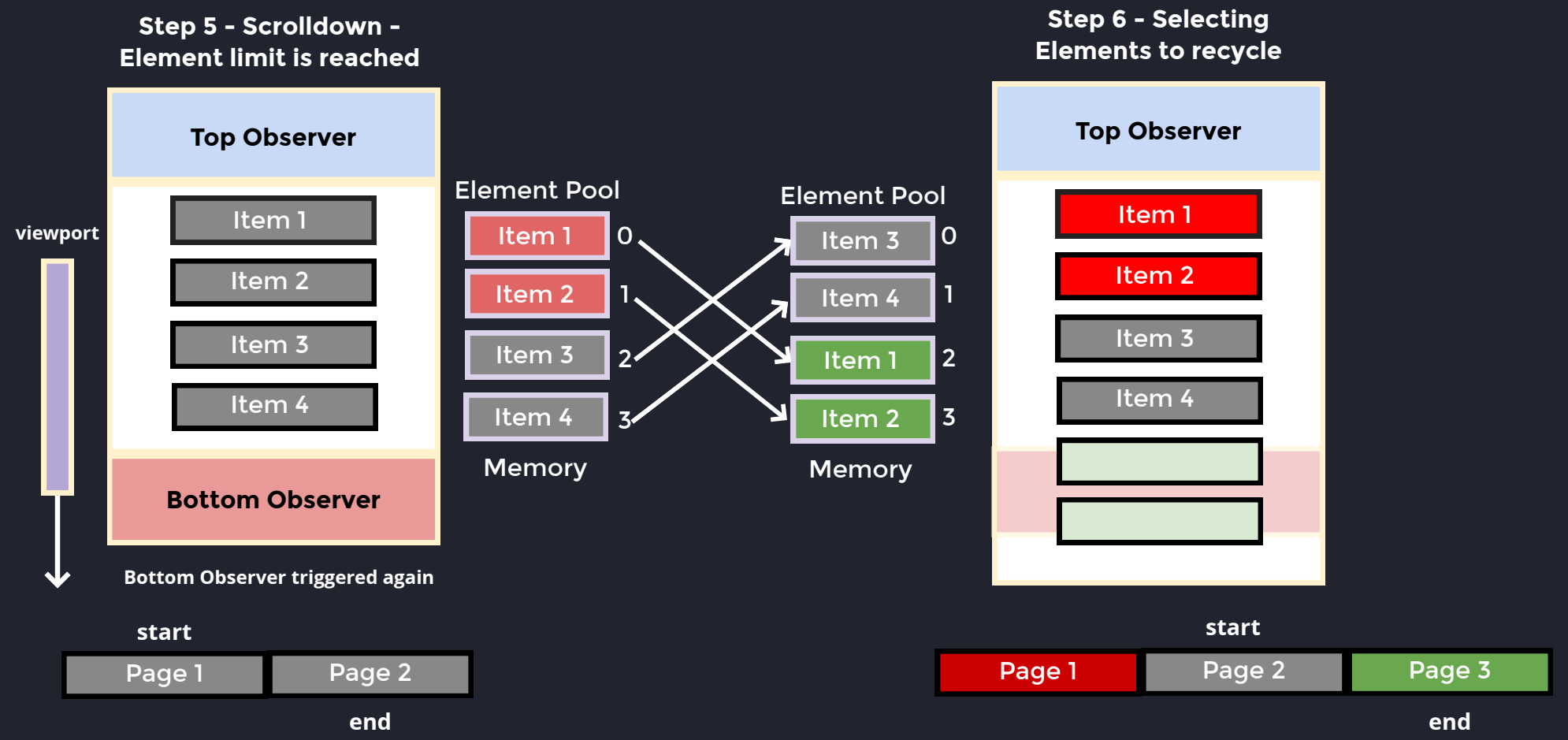
start

Page 1 | Page 2

end

Virtualisation: Live Coding

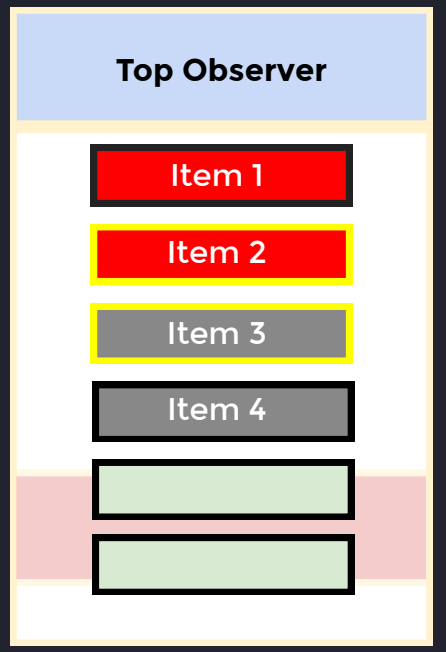
Part 4: Selecting and Preparing elements for recycling

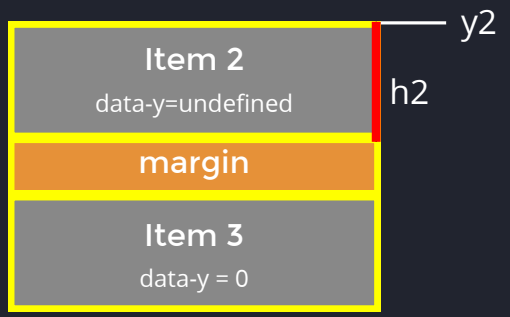


Virtualisation: Live Coding

Part 5: Handle bottom virtialization

Step 6 - Selecting Elements to recycle



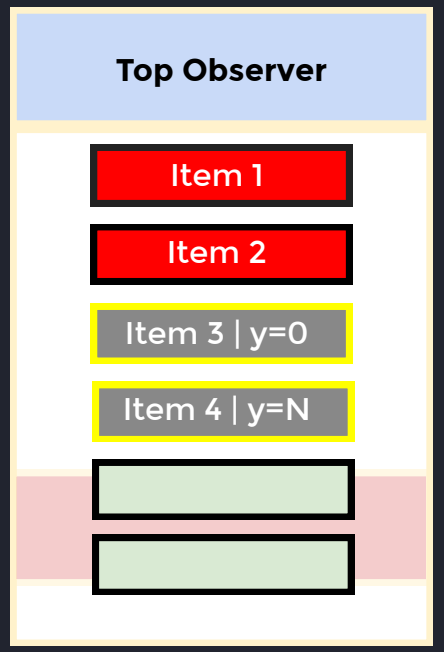


Prev Element data-y	
!= null	y3 = y2 + h2 + margin
== null	y3 = 0

Virtualisation: Live Coding

Part 5: Handle bottom virtialization

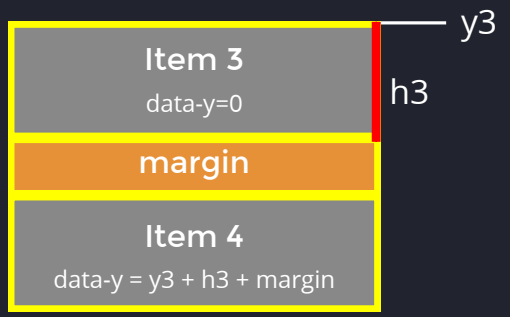
Step 7 - Looping through the pool



Element Pool

prev	Item 3	0
curr	Item 4	1
	Item 1	2
	Item 2	3

Memory



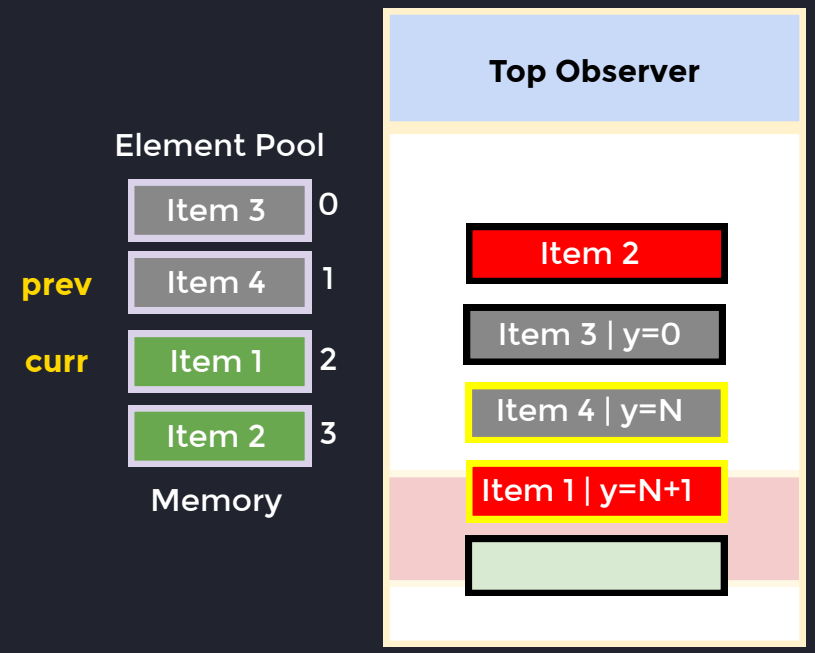
Prev Element data-y	
!= null	y4 = y3 + h3 + margin
== null	y3 = 0

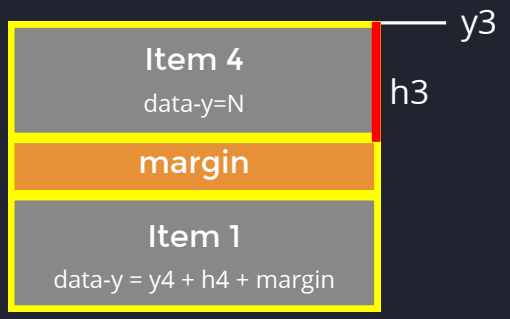
```
current.style.transform = 'translateY(${N}px)'
```

Virtualisation: Live Coding

Part 5: Handle bottom virtialization

Step 8 - Looping through the pool





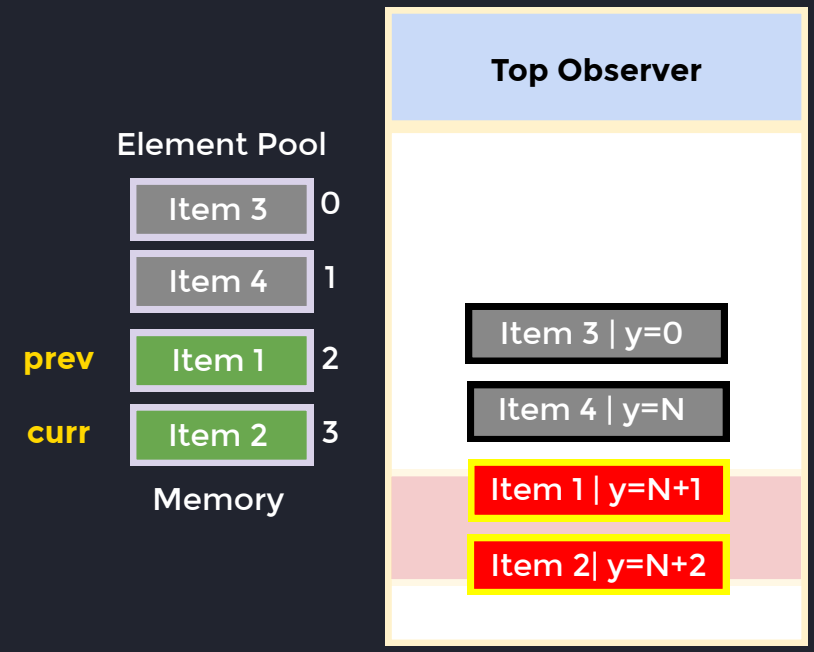
Prev Element data-y	
!= null	y = y4 + h4 + margin
== null	y4 = 0

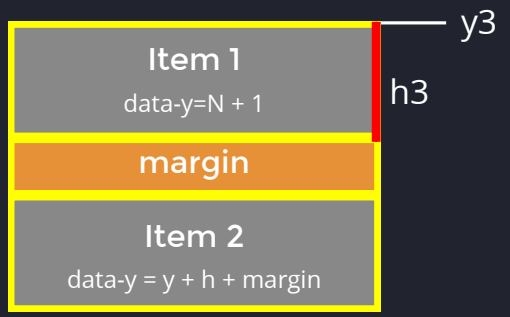
```
current.style.transform = 'translateY(${N}px)'
```

Virtualisation: Live Coding

Part 5: Handle bottom virtialization

Step 9 - Looping through the pool





Prev Element data-y	
!= null	y = y(N-1) + h + margin
== null	y = 0

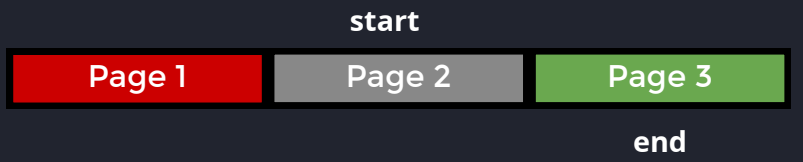
```
current.style.transform = 'translateY(${N}px)'
```

Virtualisation: Live Coding

Part 5: Handle bottom virtialization

Step 10 - Update observers and data

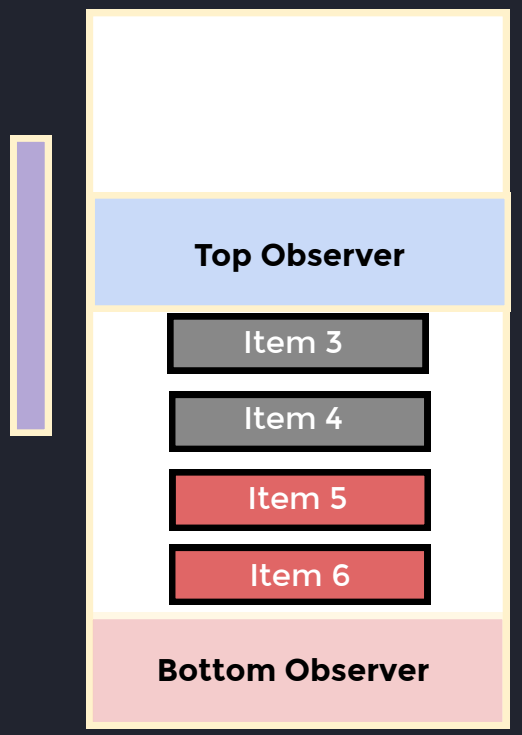
Observer position	
top	Y(0)
bottom	Y(N) + H(N) + M

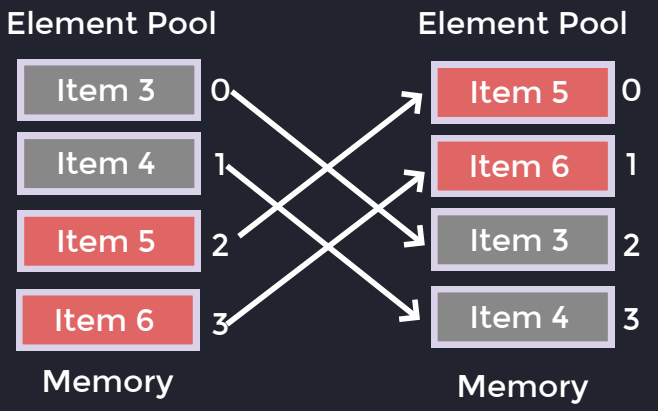


Virtualisation: Live Coding

Part 6: Handle top virtualization

Step 10 - Top observer is triggered

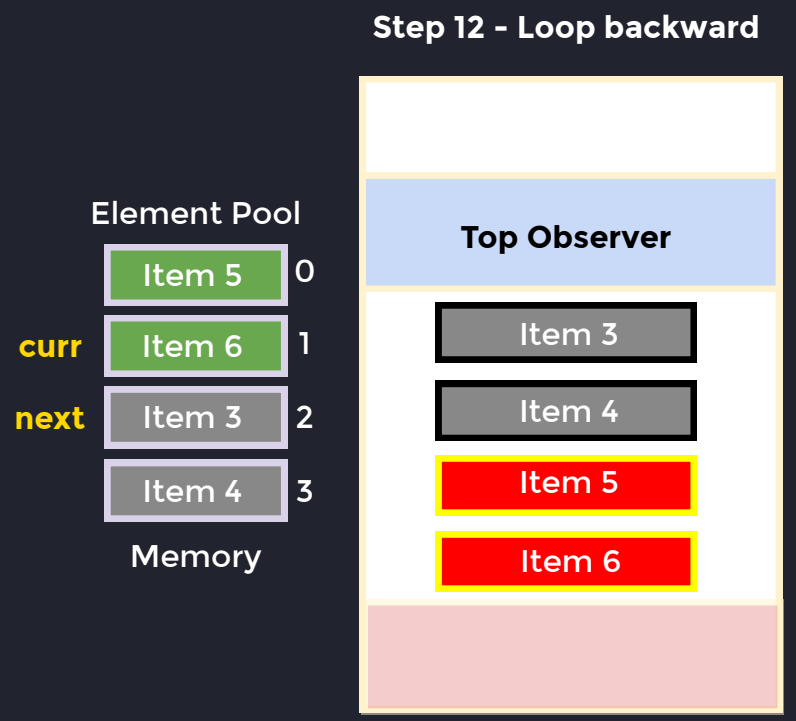


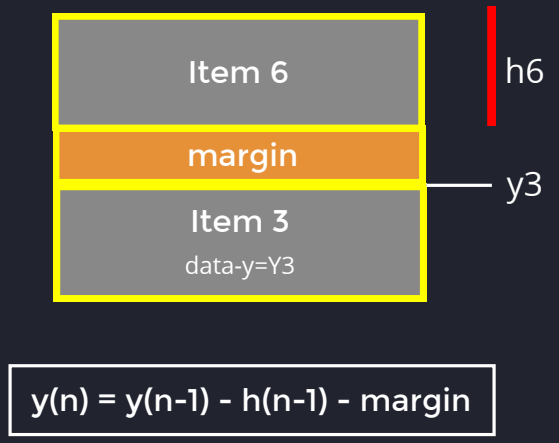


Step 11 - Prepare elements for recycling

Virtualisation: Live Coding

Part 6: Handle top virtualization

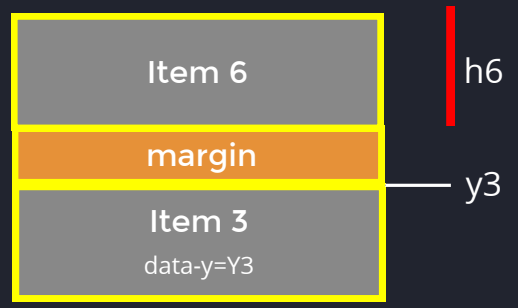




Virtualisation: Live Coding

Part 6: Handle top virtualization

Step 12 - Loop backward

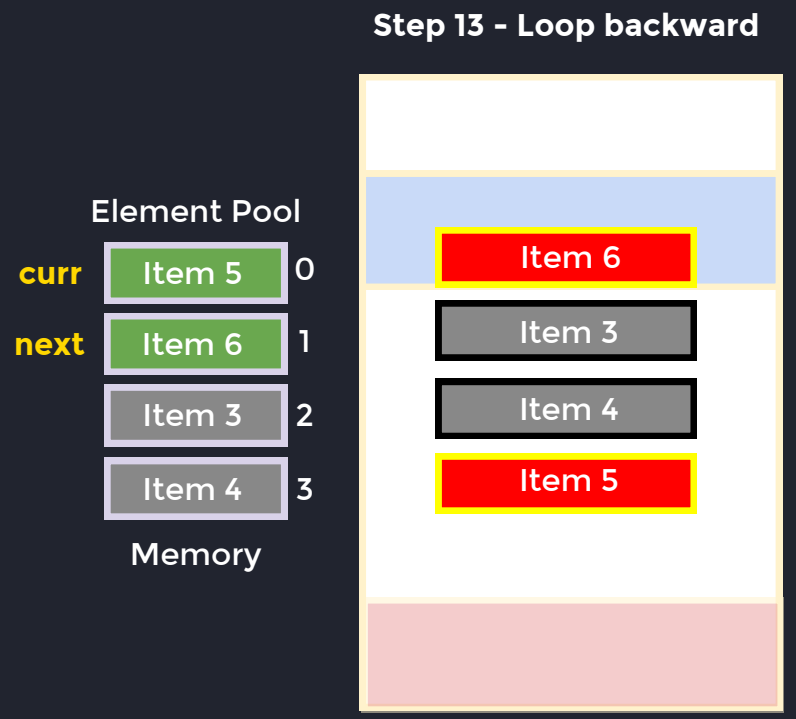


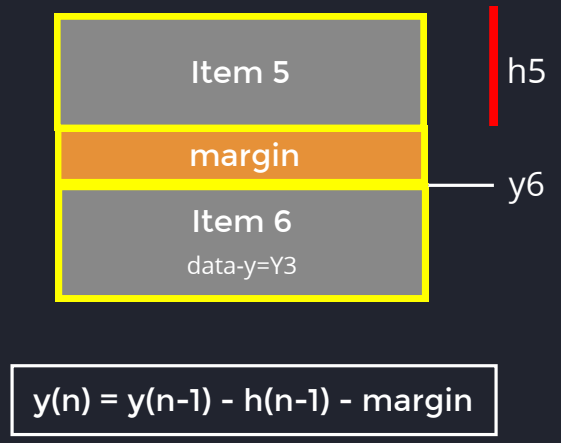
$$y(n) = y(n-1) - h(n-1) - margin$$

```
current.style.transform = 'translateY(${N}px)'
```

Virtualisation: Live Coding

Part 6: Handle top virtualization

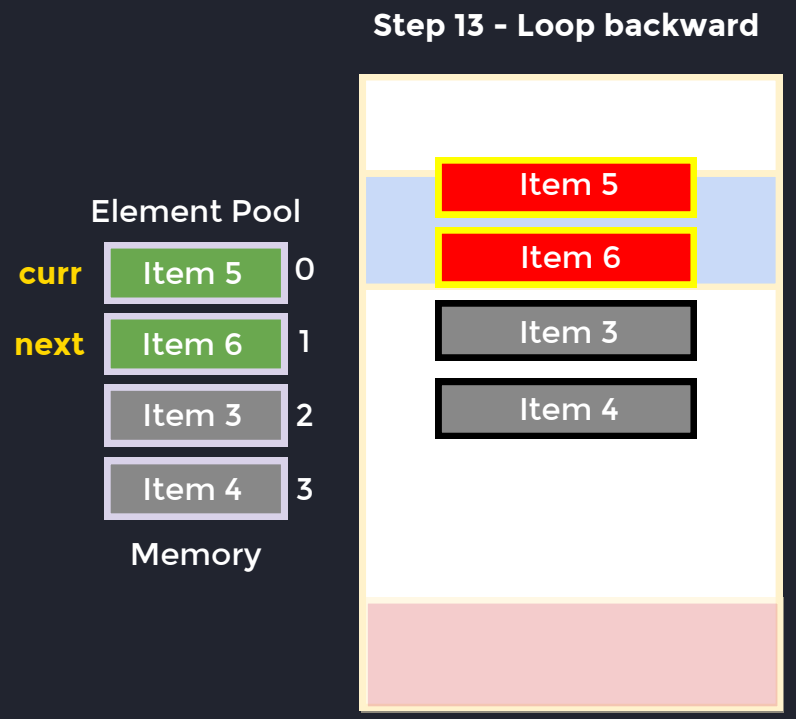




$$y(n) = y(n-1) - h(n-1) - margin$$

Virtualisation: Live Coding

Part 6: Handle top virtualization

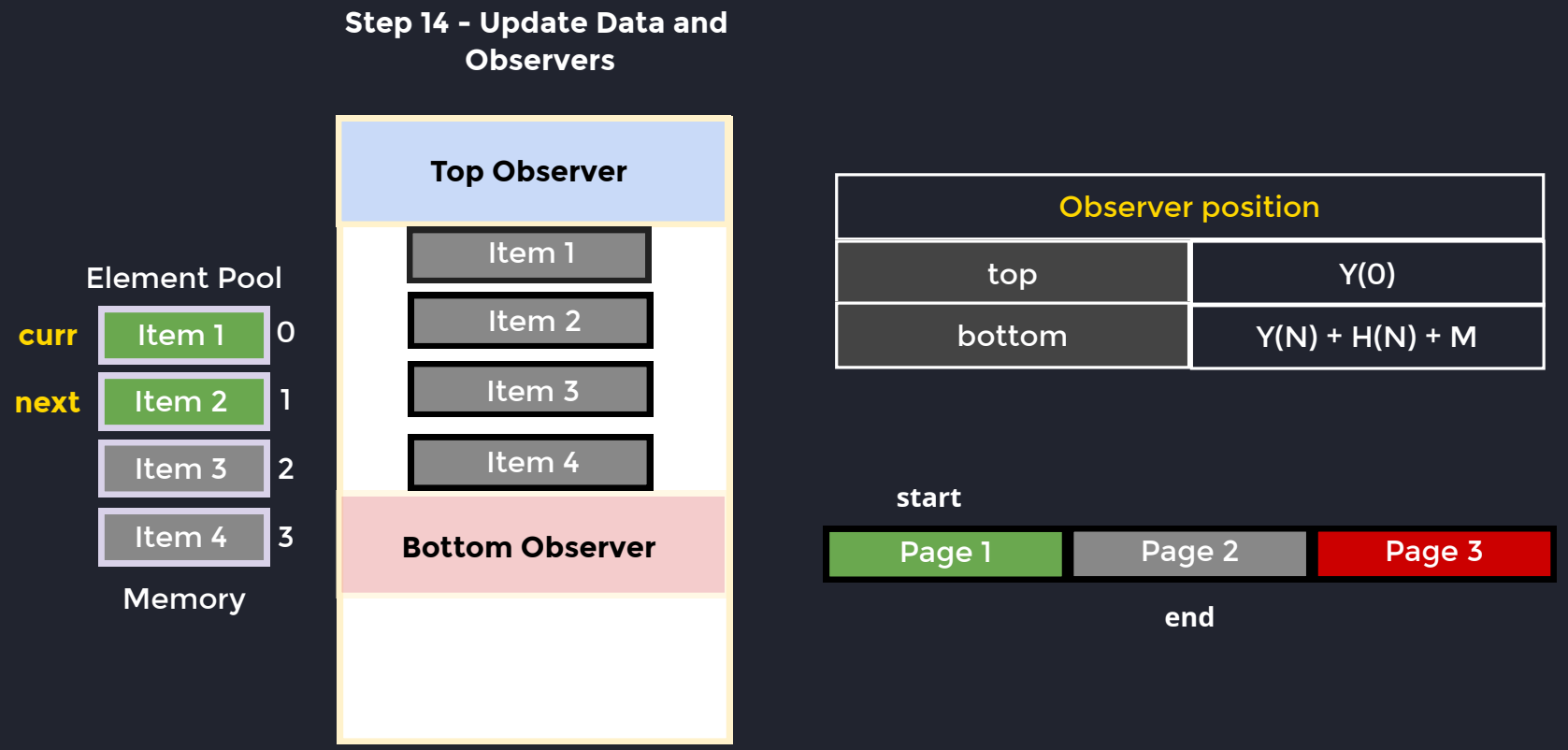


$$y(n) = y(n-1) - h(n-1) - margin$$

```
current.style.transform = 'translateY(${N}px)'
```

Virtualisation: Live Coding

Part 6: Handle top virtualization



Virtualisation: DONE!!!

Application State Design

Application State: Data classes and properties

Data Classes

UI State

Data Properties

Data properties

Application State: Data classes and properties

Data Classes

UI State

App Config

1. User theme
2. Locale
3. font-size
4. accessibility settings

Data Properties

Data properties

Application State: Data classes and properties

Data Classes

UI State

App Config

UI Elements State

1. User theme
2. Locale
3. font-size
4. accessibility settings

1. Selected controls
2. Selected Text formatting (Google Docs)
3. Any other elements' configuration
4. Entered text etc

Data Properties

Data properties

Application State: Data classes and properties

Data Classes

UI State

App Config

1. User theme
2. Locale
3. font-size
4. accessibility settings

UI Elements State

1. Selected controls
2. Selected Text formatting (Google Docs)
3. Any other elements' configuration
4. Entered text etc

Server Data

1. Messages
2. Posts

Data Properties

Data properties

Application State: Data classes and properties

Data Classes

UI State

App Config

1. User theme
2. Locale
3. font-size
4. accessibility settings

UI Elements State

1. Selected controls
2. Selected Text formatting (Google Docs)
3. Any other elements' configuration
4. Entered text etc

Server Data

1. Messages
2. Posts

Data Properties

Data properties

Access level

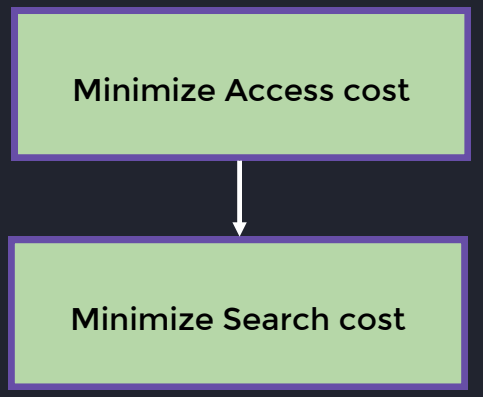
Read/ Write Frequency

Size

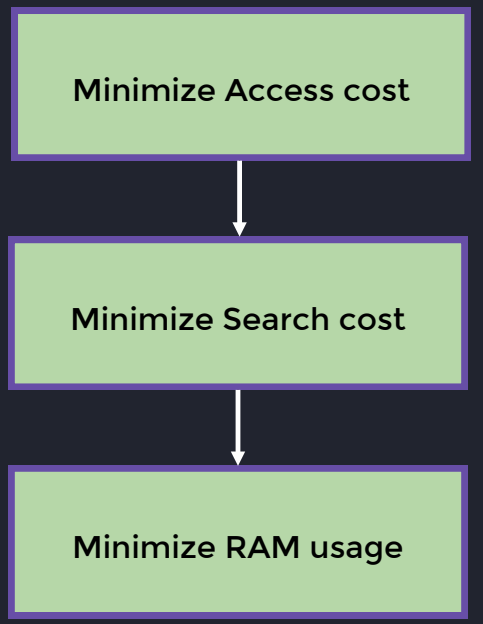
Application State: General Principles

Minimize Access cost

Application State: General Principles

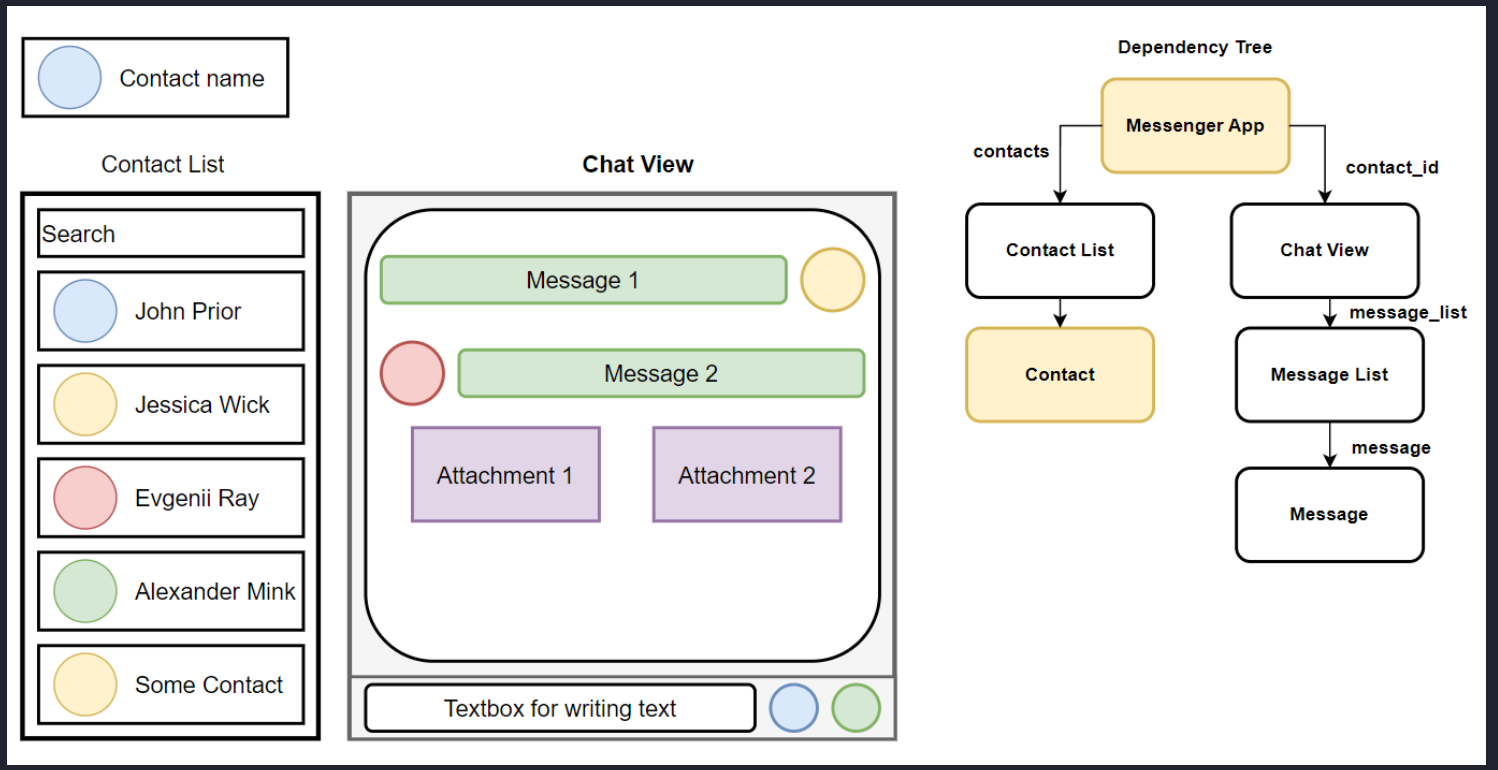


Application State: General Principles



Application State: Minimize data access cost

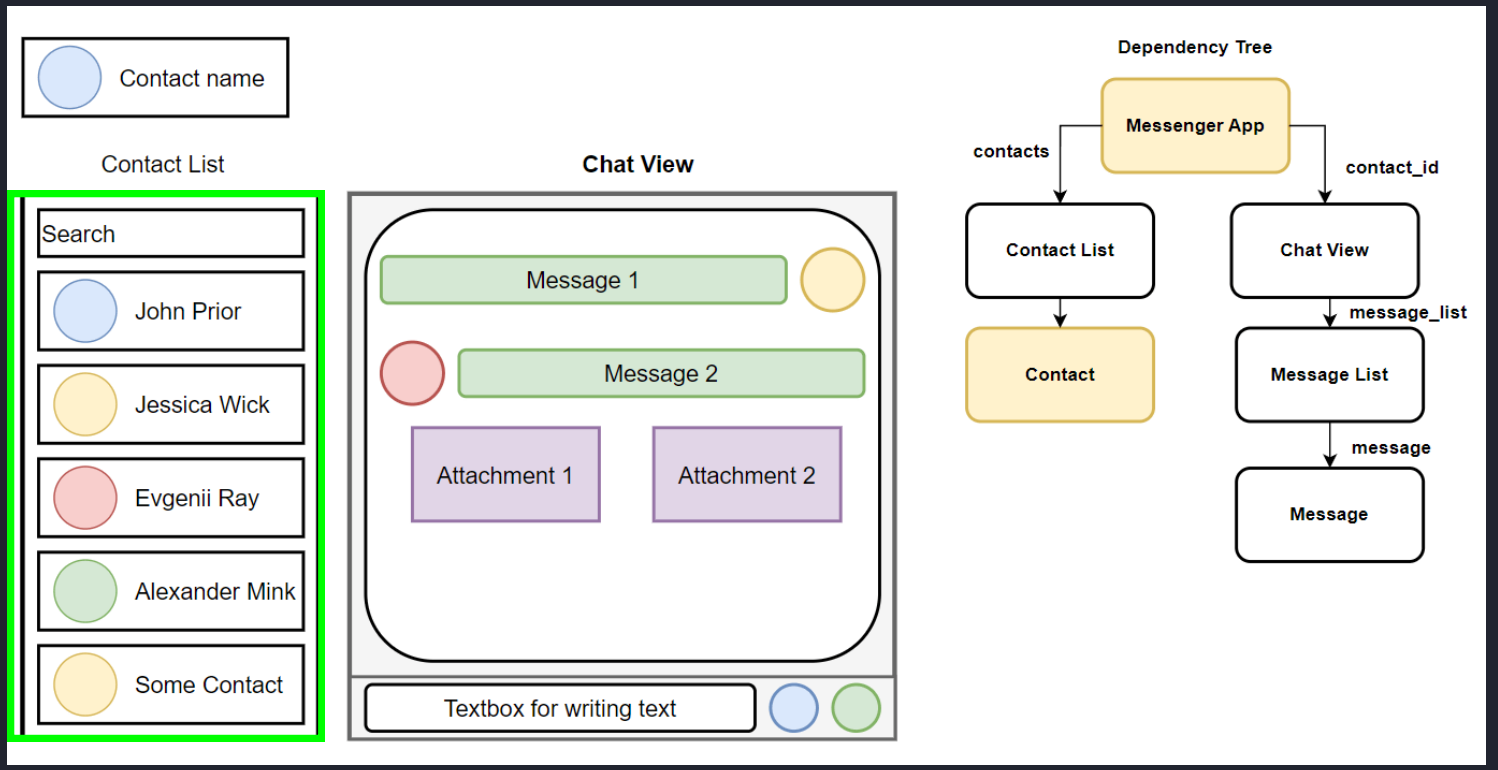
```
type TUser = {
  contacts: TContact[];
}

type TContact = {
    id: string;
    name: string;
    conversation: TConversation;
}

type TConversation = {
    messages: TMessage[];
}

type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    timestamp: number;
    img: URL;
    title: string;
    content: string;
}

type TAppGlobalState = {
  contacts: TContact[];
}
```



Application State: Minimize data access cost

```ts
type TUser = {
  contacts: TContact[];
}

type TContact = {
    id: string;
    name: string;
    conversation: TConversation;
}
```

```ts
type TConversation = {
    messages: TMessage[];
}

type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    timestamp: number;
    img: URL;
    title: string;
    content: string;
}

type TAppGlobalState = {
  contacts: TContact[];
}
```

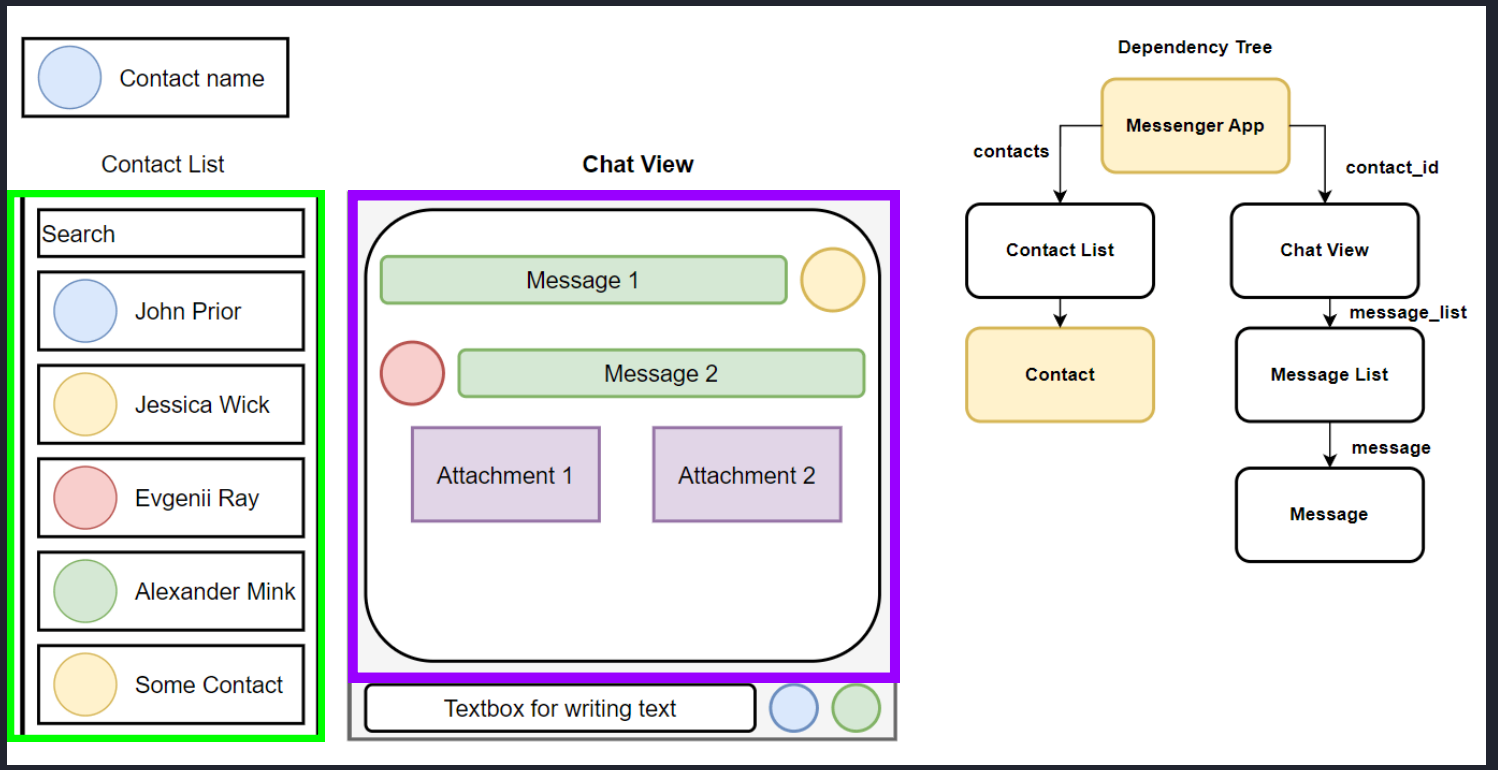


Application State: Minimize data access cost

```ts
type TUser = {
  contacts: TContact[];
}

type TContact = {
    id: string;
    name: string;
    conversation: TConversation;
}
```

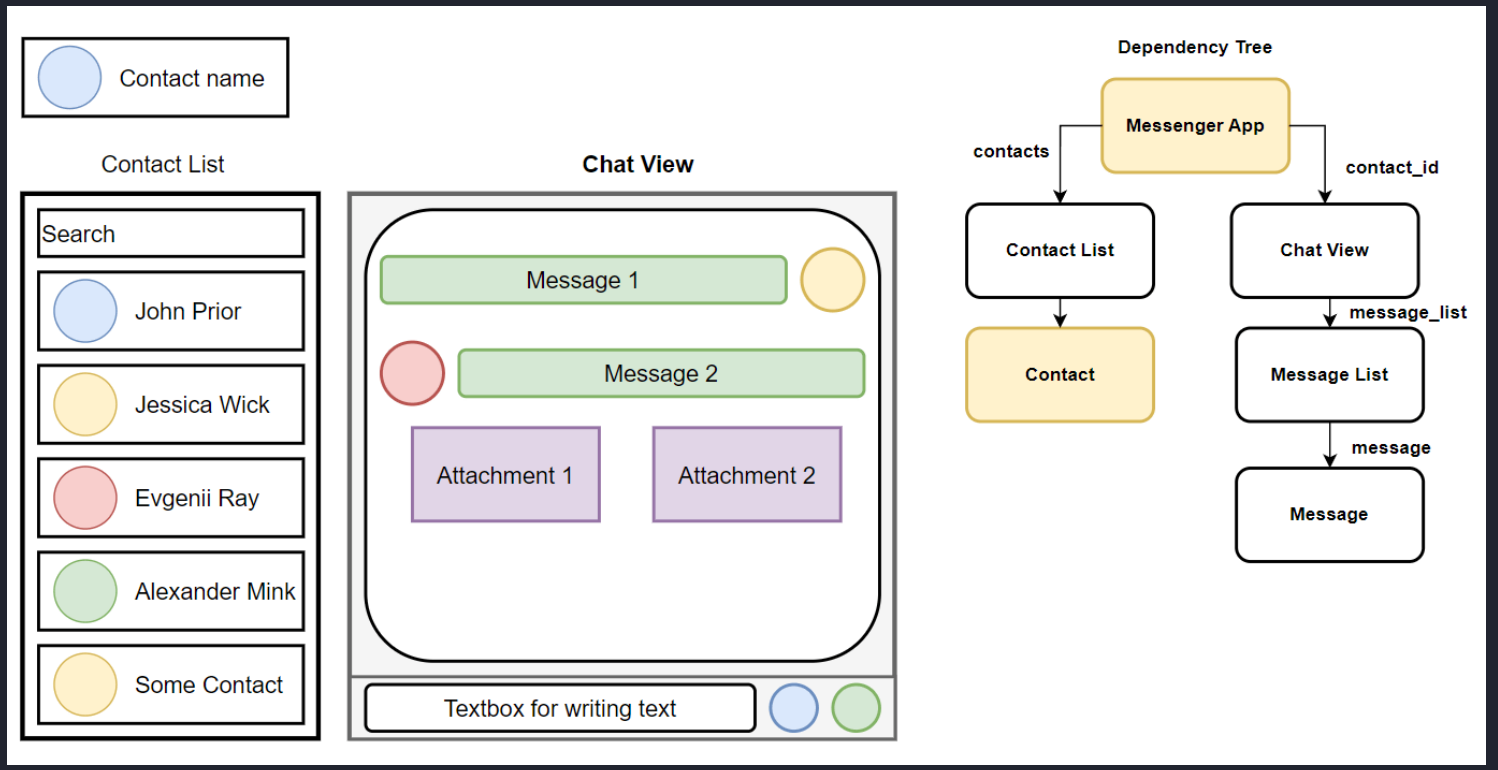
```ts
type TConversation = {
    messages: TMessage[];
}

type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    timestamp: number;
    img: URL;
    title: string;
    content: string;
}
```

```ts
type TAppGlobalState = {
  contacts: TContact[];
}
```



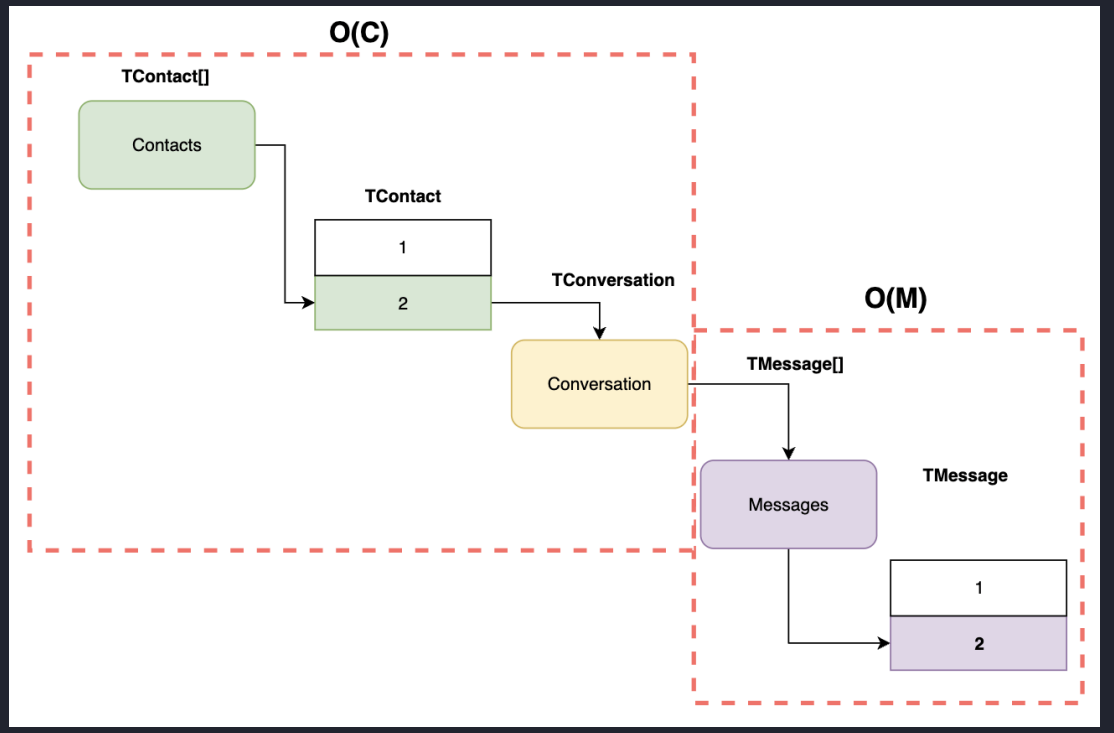
Application State: Minimize data access cost

```
type TUser = {
  contacts: TContact[];
}

type TContact = {
    id: string;
    name: string;
    conversation: TConversation;
}

type TConversation = {
    messages: TMessage[];
}

type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    timestamp: number;
    img: URL;
    title: string;
    content: string;
}

type TAppGlobalState = {
  contacts: TContact[];
}
```



Application State: Minimize data access cost

```
// Get contact entity
const contact = state.contacts.find(c => c.id === 'jane_smith');

// Get Message entity
const message = contact.conversation.messages.find(m => m.id === 5);
```



1. Accessing contact - `O(C)`
2. Accessing message - `O(M)`

Access cost: `O(C)+O(M)`

Application State: Minimize data access cost

Data Normalization

Goals of Data Normalization

1. Optimized Access Performance
2. Optimized Storage Structure
3. High Developer Readability and Maintenance

Application State: Minimize data access cost

Data Normalization

Non NF

```
{
  id: "1",
  name: "Evgenii",
  job: {
    id: "UIE",
    title: "UI Engineer",
    department: "Engineering"
  },
  location: { code: "UK", name: "United Kingdom" }
}
```

1NF

1. Data is atomic
2. It has primary key

```
{
  id: "1",
  name: "Evgenii",
  job_id: "UIE",
  job_title: "UI Engineer",
  department: "Engineering",
  country_code: "UK",
  country_name: "United Kingdom"
}
```

Application State: Minimize data access cost

Data Normalization

Non NF

```
{
  id: "1",
  name: "Evgenii",
  job: {
    id: "UIE",
    title: "UI Engineer",
    department: "Engineering"
  },
  location: { code: "UK", name: "United Kingdom" }
}
```

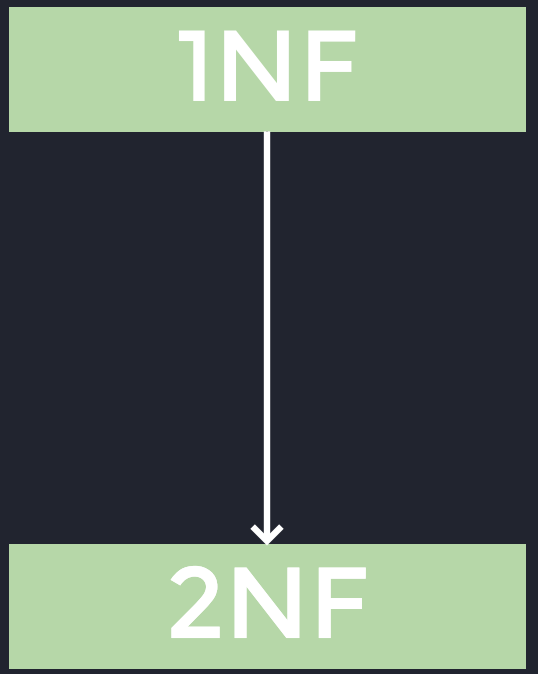
1NF

1. Data is atomic
2. It has primary key

```
{
  id: "1",
  name: "Evgenii",
  job_id: "UIE",
  job_title: "UI Engineer",
  department: "Engineering",
  country_code: "UK",
  country_name: "United Kingdom"
}
```

Application State: Minimize data access cost

Data Normalization



1. Data is atomic
2. It has primary key

```
{
  id: "1",
  name: "Evgenii",
  job_id: "UIE",
  job_title: "UI Engineer",
  department: "Engineering",
  country_code: "UK",
  country_name: "United Kingdom"
}
```

1NF + non-primary keys depend on entity primary key

Application State: Minimize data access cost

Data Normalization

1NF

1. Data is atomic
2. It has primary key

```
{
  id: "1",
  name: "Evgenii",
  job_id: "UIE",
  job_title: "UI Engineer",
  department: "Engineering",
  country_code: "UK",
  country_name: "United Kingdom"
}
```

2NF

1NF + non-primary keys depend on entity primary key

```
const users = {
  "1": {
    name: "Evgenii",
    job_id: 'UIE',
    country_code: "UK",
    country_name: "United Kingdom"
  }
}
const jobs = {
  UIE: {
    title: "UI Enginer",
    department: 'Engineering'
};
const user_jobs = { "1": "UIE" };
```

Application State: Minimize data access cost

Data Normalization

1NF

1. Data is atomic
2. It has primary key

```
{
  id: "1",
  name: "Evgenii",
  job_id: "UIE",
  job_title: "UI Engineer",
  department: "Engineering",
  country_code: "UK",
  country_name: "United Kingdom"
}
```

2NF

1NF + non-primary keys depend on entity primary key

```
const users = {
  "1": {
    name: "Evgenii",
    job_id: 'UIE',
    country_id: "UK",
  }
}

const jobs = {
  UIE: {
    title: "UI Enginer",
    department: 'Engineering'
};

const user_jobs = { "1": "UIE" };

const countries = { UK: "United Kingdom" };
```

Application State: Minimize data access cost

Data Normalization

2NF

1. 1NF
2. non-primary keys depend on entity primary key

3NF

1. 2NF
2. non-primary keys **ONLY** depend on entity primary key

```
const users = {
  "1": {
    name: "Evgenii",
    job_id: 'UIE',
    country_id: "UK",
  }
}
const jobs = {
  UIE: {
    title: "UI Enginer",
    department: 'Engineering'
};
const user_jobs = { "1": "UIE" };
```

```
const users = {
  "1": {
    name: "Evgenii",
    job_id: 'UIE',
    country_id: "UK",
  }
}
const jobs = { UIE: "UI Enginer" }

const department = { UIE: "Engineering"}

const user_jobs = { "1": "UIE" };

const countries = { UK: "United Kingdom" };
```

Application State: 1NF and 2NF

Non NF

```
type TContact = {
    id: string;
    name: string;
    conversation: TConversation;
}
```

```
type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    timestamp: number;
    img: URL;
    title: string;
    content: string;
```

```
type TConversation = {
    id: string;
    title: string;
    messages: TMessage[];
}
```

Application State: 1NF and 2NF

Non NF

```
type TContact = {
    id: string;
    name: string;
    conversation: TConversation
}
```

```
type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    timestamp: number;
    img: URL;
    title: string;
    content: string;
}
```

```
type TConversation = {
    id: string;
    title: string;
    messages: TMessage[];
}
```

Application State: 1NF and 2NF

Non NF

```
type TContact = {
    id: string;
    name: string;
    conversation: TConversation
}
```

```
type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    timestamp: number;
    img: URL;
    title: string;
    content: string;
}
```

```
type TConversation = {
    id: string;
    title: string;
    messages: TMessage[];
}
```

Atomic Fields

```
type TContact = {
    id: string;
    name: string;
    conversation_id: string
}
```

```
type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    conversation_id: string
    timestamp: number;
    img: URL;
    content: string;
}
```

```
type TConversation = {
  id: string;
  title: string;
}
```

Application State: 1NF and 2NF

Non NF

```
type TContact = {
    id: string;
    name: string;
    conversation: TConversation
}
```

```
type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    timestamp: number;
    img: URL;
    title: string;
    content: string;
}
```

```
type TConversation = {
    id: string;
    title: string;
    messages: TMessage[];
}
```

Atomic Fields

```
type TContact = {
    id: string;
    name: string;
    conversation_id: string
}
```

```
type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    conversation_id: string
    timestamp: number;
    img: URL;
    content: string;
}
```

```
type TConversation = {
  id: string;
  title: string;
}
```

Primary Keys

```
type TContact = {
    id: string;
    name: string;
    conversation_id: string
}
```

```
type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    conversation_id: string
    timestamp: number;
    img: URL;
    content: string;
}
```

```
type TConversation = {
  id: string;
  title: string;
}
```

Application State: 1NF and 2NF

Non NF

```
type TContact = {
    id: string;
    name: string;
    conversation: TConversation
}
```

```
type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    timestamp: number;
    img: URL;
    title: string;
    content: string;
}
```

```
type TConversation = {
    id: string;
    title: string;
    messages: TMessage[];
}
```

Atomic Fields

```
type TContact = {
    id: string;
    name: string;
    conversation_id: string
}
```

```
type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    conversation_id: string
    timestamp: number;
    img: URL;
    content: string;
}
```

```
type TConversation = {
  id: string;
  title: string;
}
```

Primary Keys

```
type TContact = {
    id: string;
    name: string;
    conversation_id: string
}
```

```
type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    conversation_id: string
    timestamp: number;
    img: URL;
    content: string;
}
```

```
type TConversation = {
  id: string;
  title: string;
}
```

App State

```
type TAppState = {
  messages: {
    [message_id: string]: TMessage;
  }
  conversations: {
    [convo_id: string]: TConversation;
  },
  contacts: {
    [user_id: string]: TContact;
  }
}
```

Application State: Minimize search cost

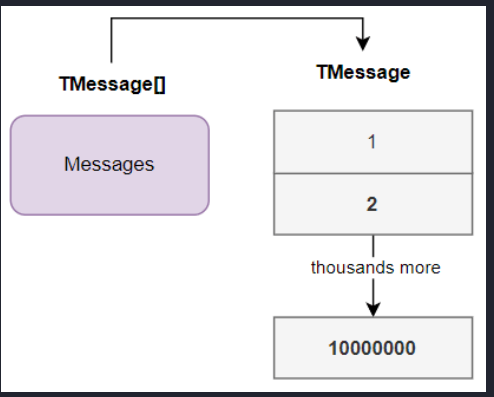
App State

```
type TAppState = {
  messages: {
   [message_id: string]: TMessage;
  }
  conversations: {
   [convo_id: string]: TConversation;
  },
  contacts: {
   [user_id: string]: TContact;
  }
```

Message Entity

```
type TMessage = {
    receiver_id: string;
    sender_id: string;
    message_id: string;
    conversation_id: string
    timestamp: number;
    img: URL;
    content: string;
```

Search



Application State: Minimize search cost

Indexing - Inverted Index Table

Messages

ID	Content	Timestamp
1	Hello, Jane	1000
2	Hey, How are u?	2000

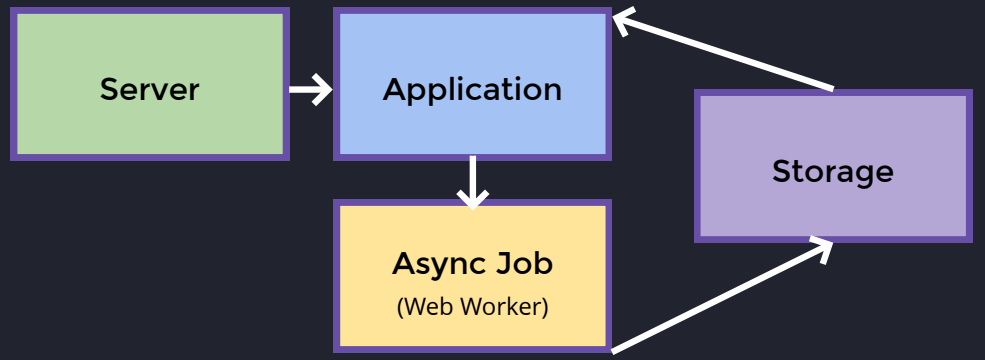
Inverted Index Table

Key	Messages
Jane	[[1, 1000]]
Hey	[[2, 2000]]

```
type TMessageInvertedIndex = {
  [key: string]: Array<
   [
     message_id: string,
     timestamp: number
   ]
  >
}
```

Application State: Minimize search cost

Indexing - Inverted Index Table



Messages

ID	Content	Timestamp
1	Hello, Jane	1000
2	Hey, How are u?	2000

Inverted Index Table

Key	Message Tuple
Jane	[[1, 1000]]
Hey	[[2, 2000]]

```
type TMessageInvertedIndex = {
  [key: string]: Array<
   [
     message_id: string,
     timestamp: number
   ]
  >
}
```

Application State: Minimize search cost

Indexing - Composite Inverted Index Table

Normalized State

Messages

ID	Content	Timestamp
1	Hello, Jane	1000
2	Hey, How are u?	2000

Inverted Index Table

Simple Keys

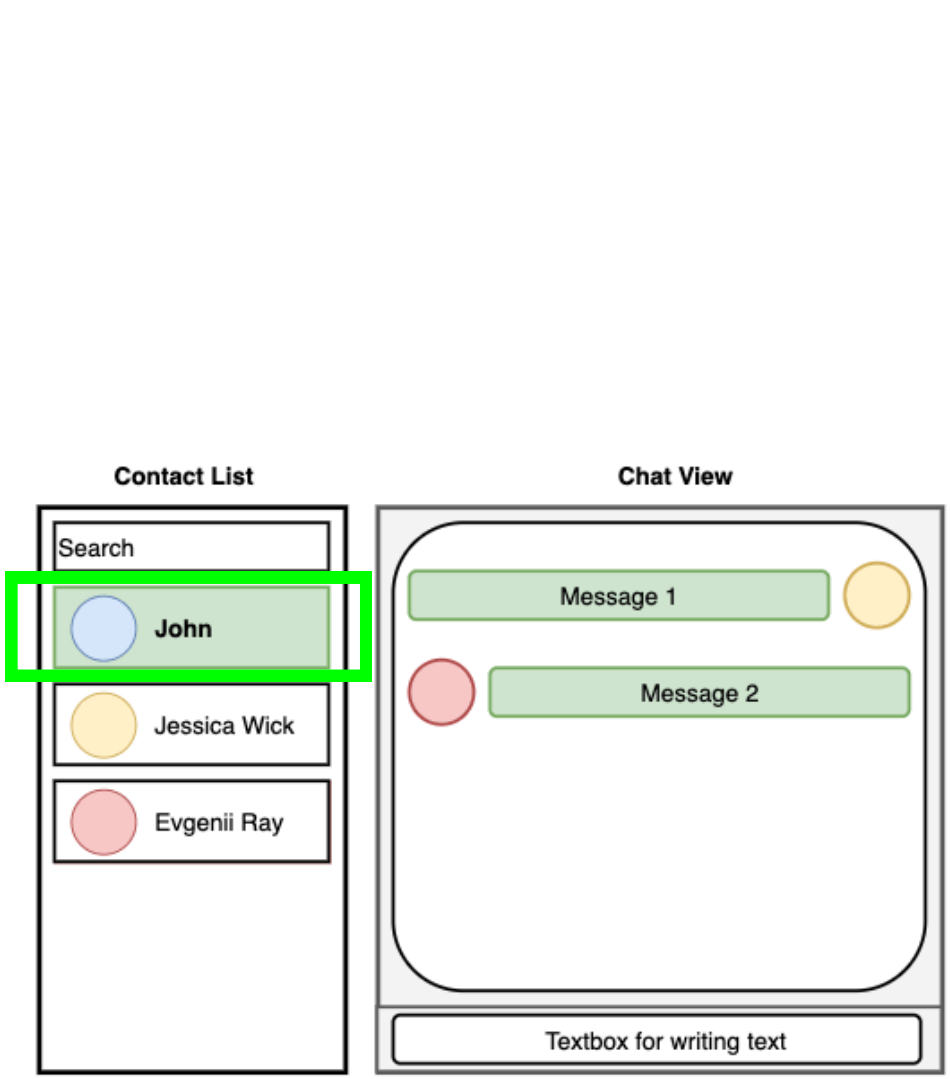
Key	Message
Jane	[[1, 1000]]
Hey	[[2, 1000]]

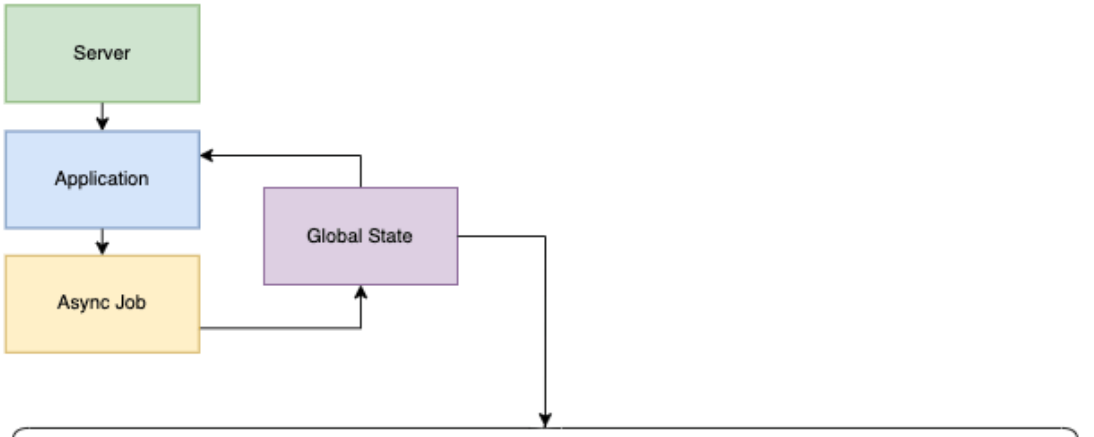
Composite Keys

Key	Message
[J, Ja, Jan, Jane]	[[1, 1000]]
[H, He, Hey]	[[2, 1000]]

```
type TAppState = {
  messages: {
   [message_id: string]: TMessage;
  }
  conversations: {
   [convo_id: string]: TConversation;
  },
  contacts: {
   [user_id: string]: TContact;
  }
  message_index: TMessageInvertedIndex;
```

Application State: Memory Offloading





messages

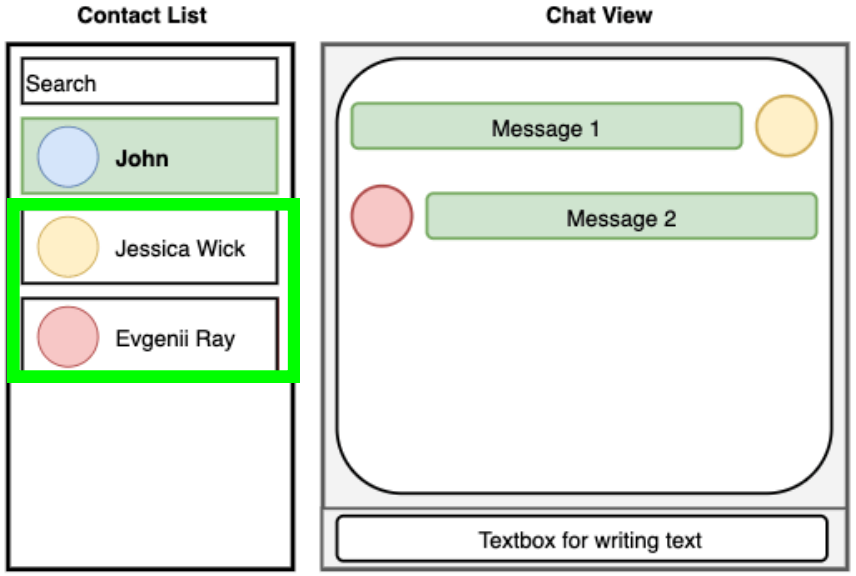
ID	Content	Conversation ID	Timestamp
1	Hello, Jane	1	1000
2	Hey, how are u?	1	2000
3	Pizza is good	2	10000
4	I prefer broccoli	2	11000
5	JS is weird	3	7000
6	I love it	3	8000

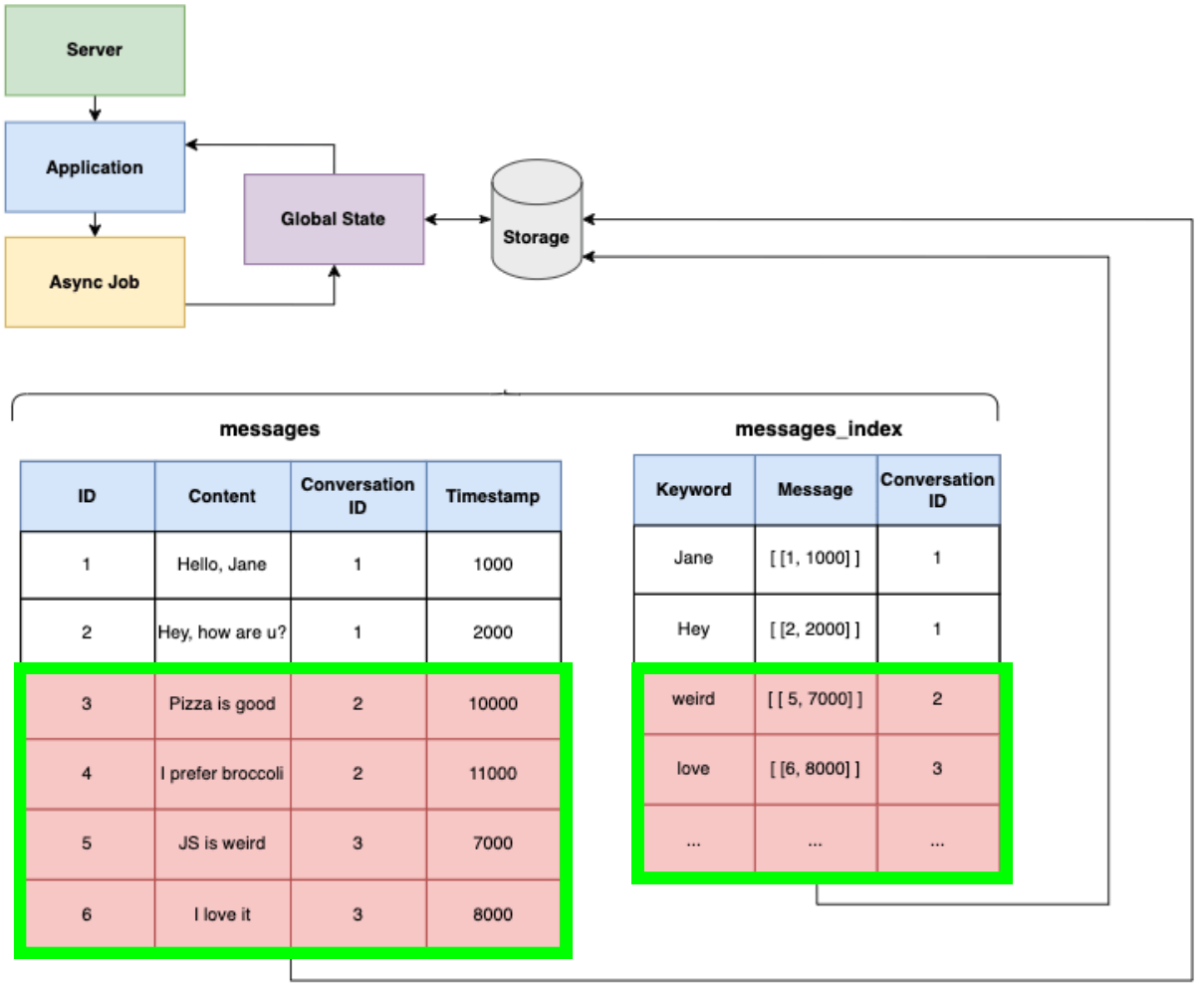
messages_index

Keyword	Message	Conversation ID
Jane	[[1, 1000]]	1
Hey	[[2, 2000]]	1
weird	[[5, 7000]]	2
love	[[6, 8000]]	3
...	...	...

Application State: Memory Offloading

Identifying un-used data





messages

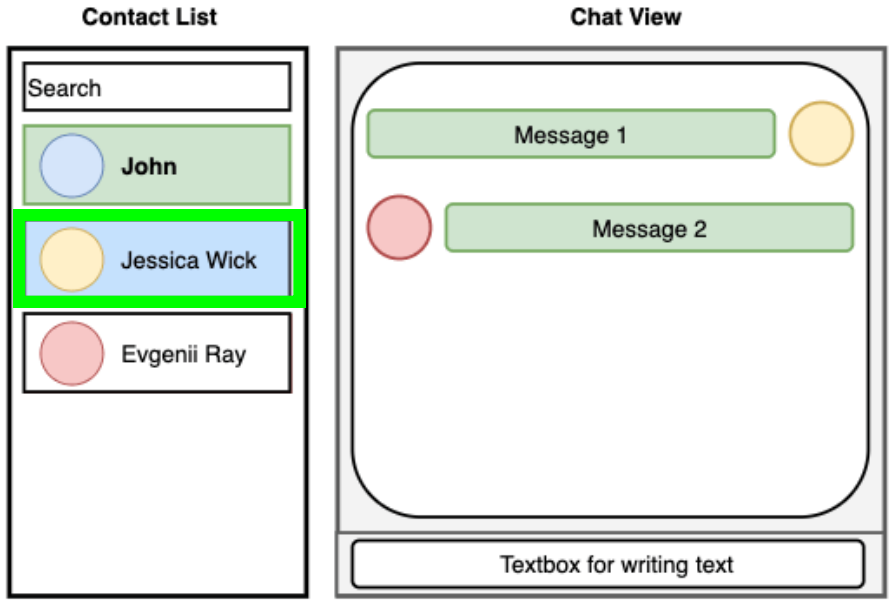
ID	Content	Conversation ID	Timestamp
1	Hello, Jane	1	1000
2	Hey, how are u?	1	2000
3	Pizza is good	2	10000
4	I prefer broccoli	2	11000
5	JS is weird	3	7000
6	I love it	3	8000

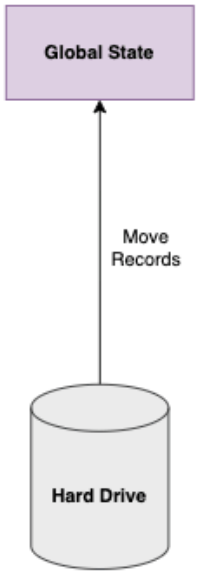
messages_index

Keyword	Message	Conversation ID
Jane	[[1, 1000]]	1
Hey	[[2, 2000]]	1
weird	[[5, 7000]]	2
love	[[6, 8000]]	3
...	...	...

Application State: Memory Offloading

Fetching data from hard-drive





messages

ID	Content	Conversation ID	Timestamp
1	Hello, Jane	1	1000
2	Hey, how are u?	1	2000

messages_index

Keyword	Message	Conversation ID
Jane	[[1, 1000]]	1
Hey	[[2, 2000]]	1

messages

ID	Content	Conversation ID	Timestamp
3	Pizza is good	2	10000
4	I prefer broccoli	2	11000
5	JS is weird	3	7000
6	I love it	3	8000

messages_index

Keyword	Message	Conversation ID
weird	[[5, 7000]]	2
love	[[6, 8000]]	3
...	...	...

Memory offloading: Choosing a storage

Storage Properties

Type	Indexed DB	Local Storage	Session Storage
Storage Cap	Unlimited	5 Mb	5 mb
Indexing	Yes	No	No
Advanced search	Yes	No	No
Data Types	number, date, string, binary, or array	string	string
Blocking thread	No	Yes	Yes

Use-cases

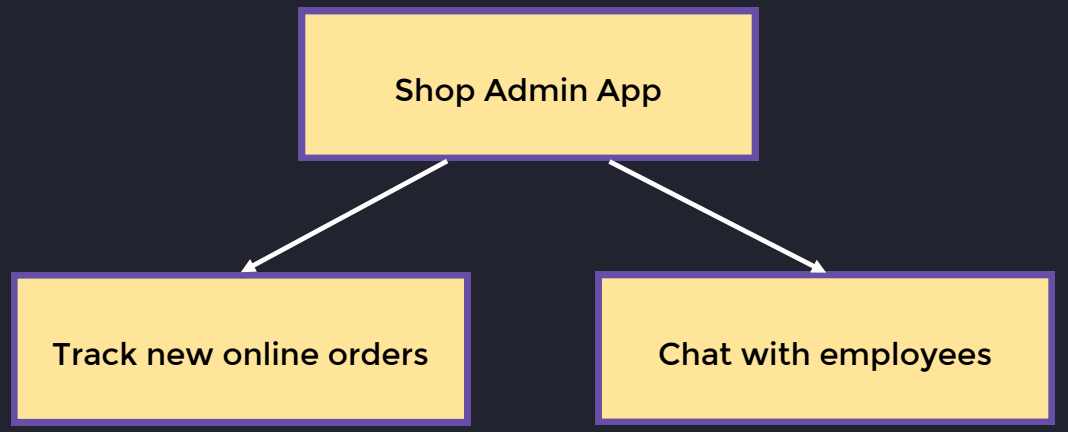
Usecase	IndexedDB	Local Storage	Session Storage
User preferences, small configs			
Non-persistent data			
Large data with query support			
Offline Mode			
High Read / Writes			
Multi-storage support			

Summary: Application state design

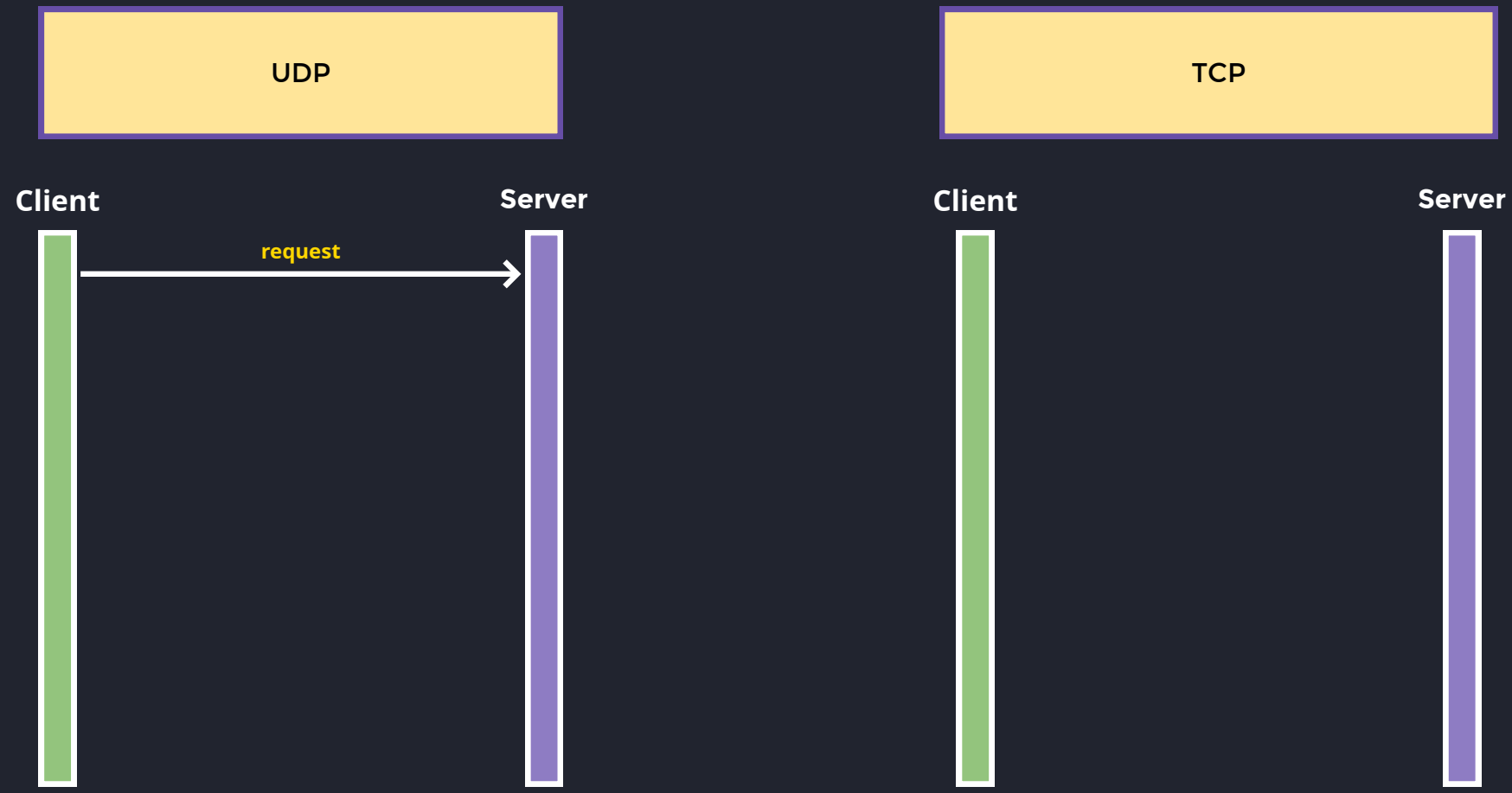
1. **Know your scale** - optimize accordingly
2. Always start with how you **structure your data**
3. Use **Normal forms** to optimize access cost
4. Use **Indexes** if in-app search is required
5. **Offload data** to hard-drive when it's needed
6. Pick a **suitable storage**

Network Connectivity

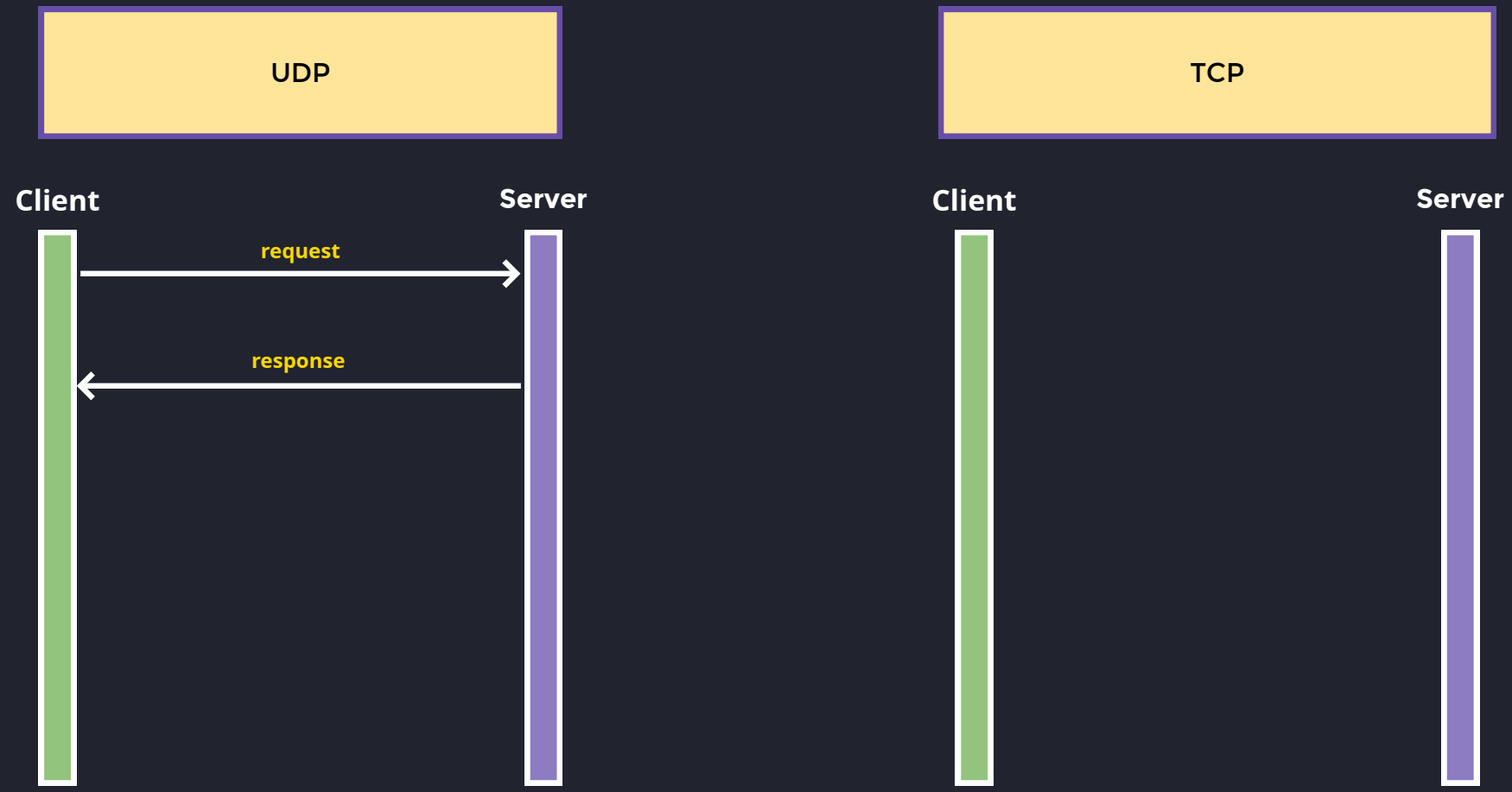
Network Connectivity



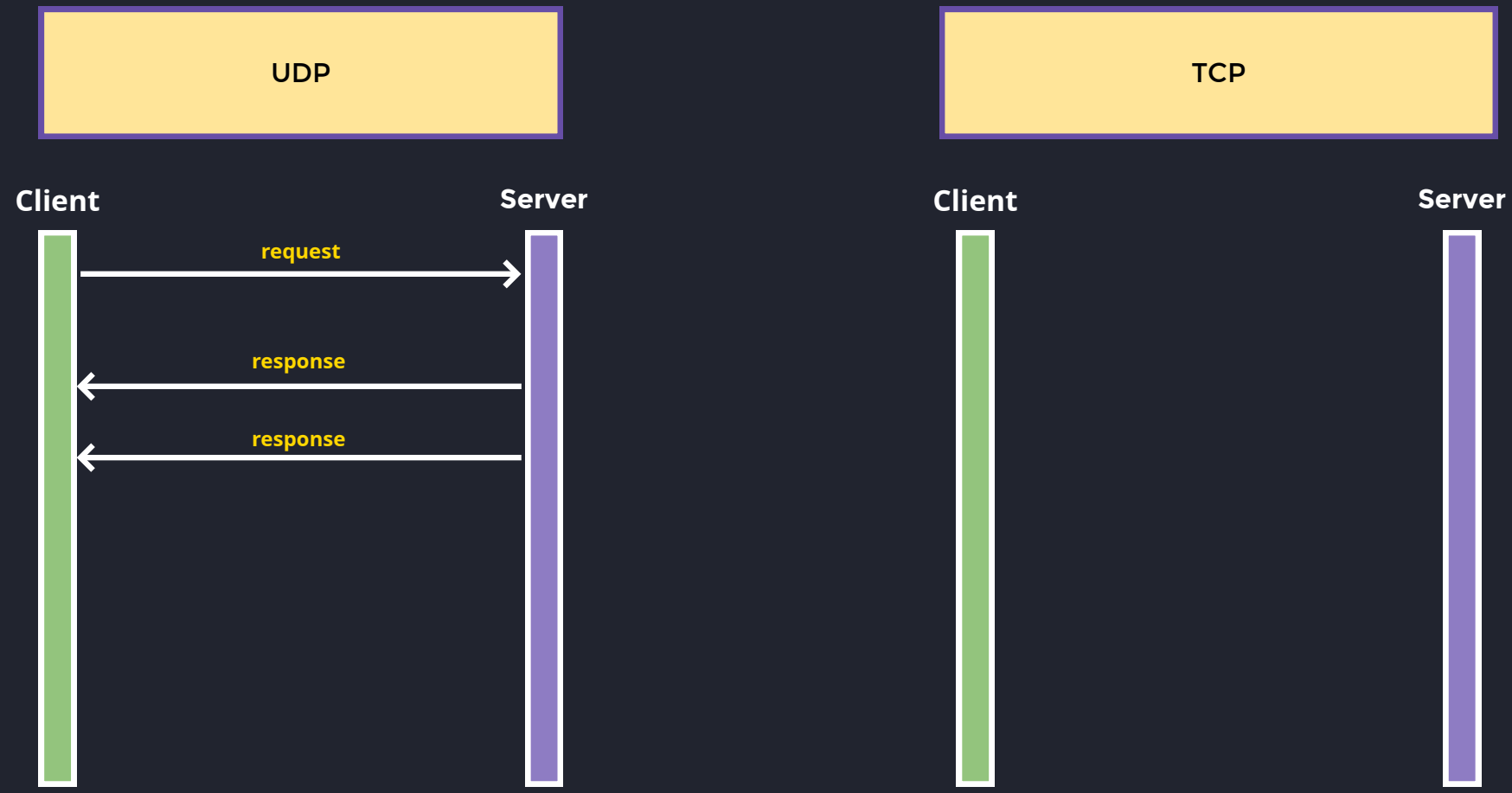
Network Connectivity: Protocols overview



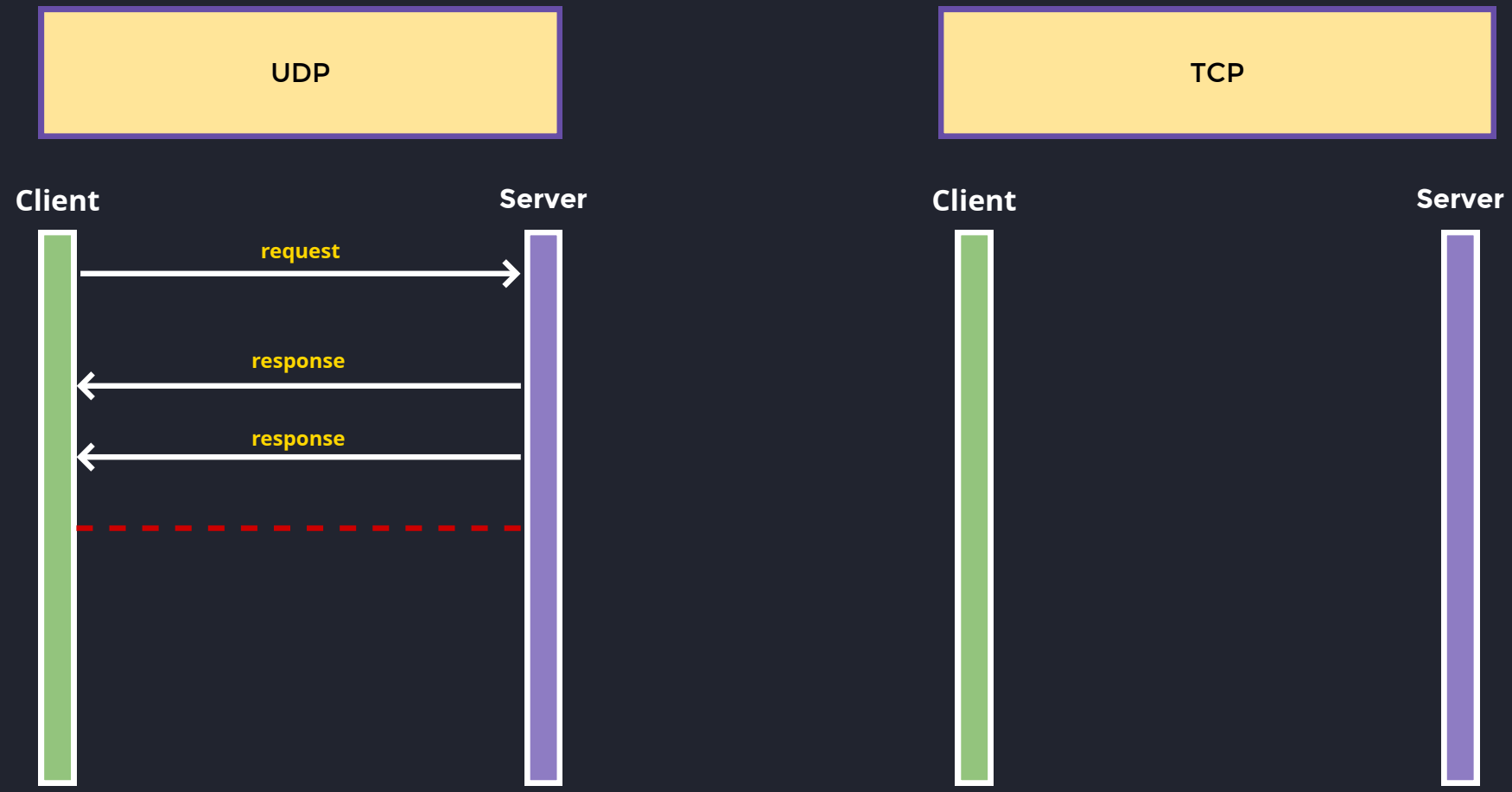
Network Connectivity: Protocols overview



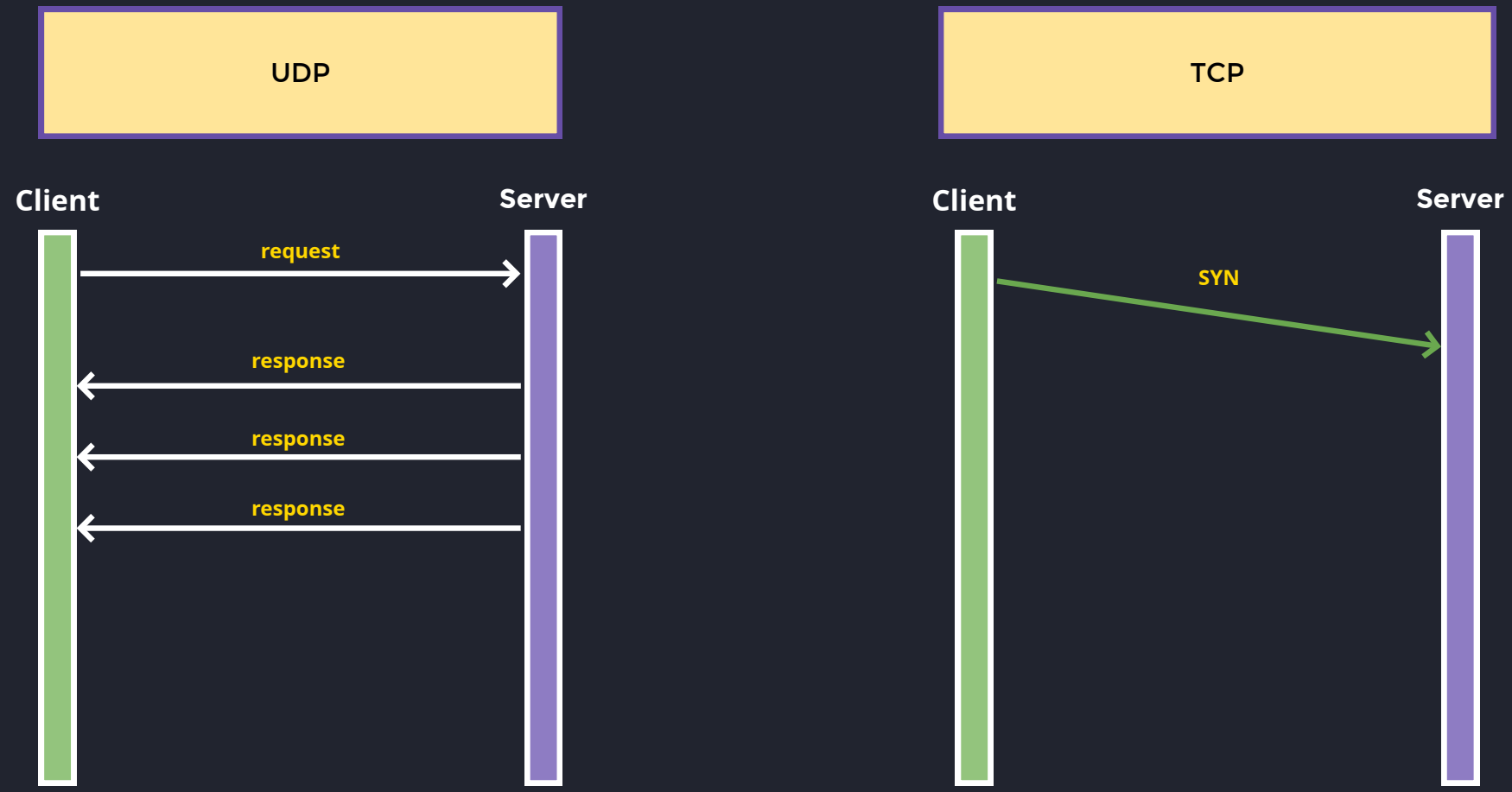
Network Connectivity: Protocols overview



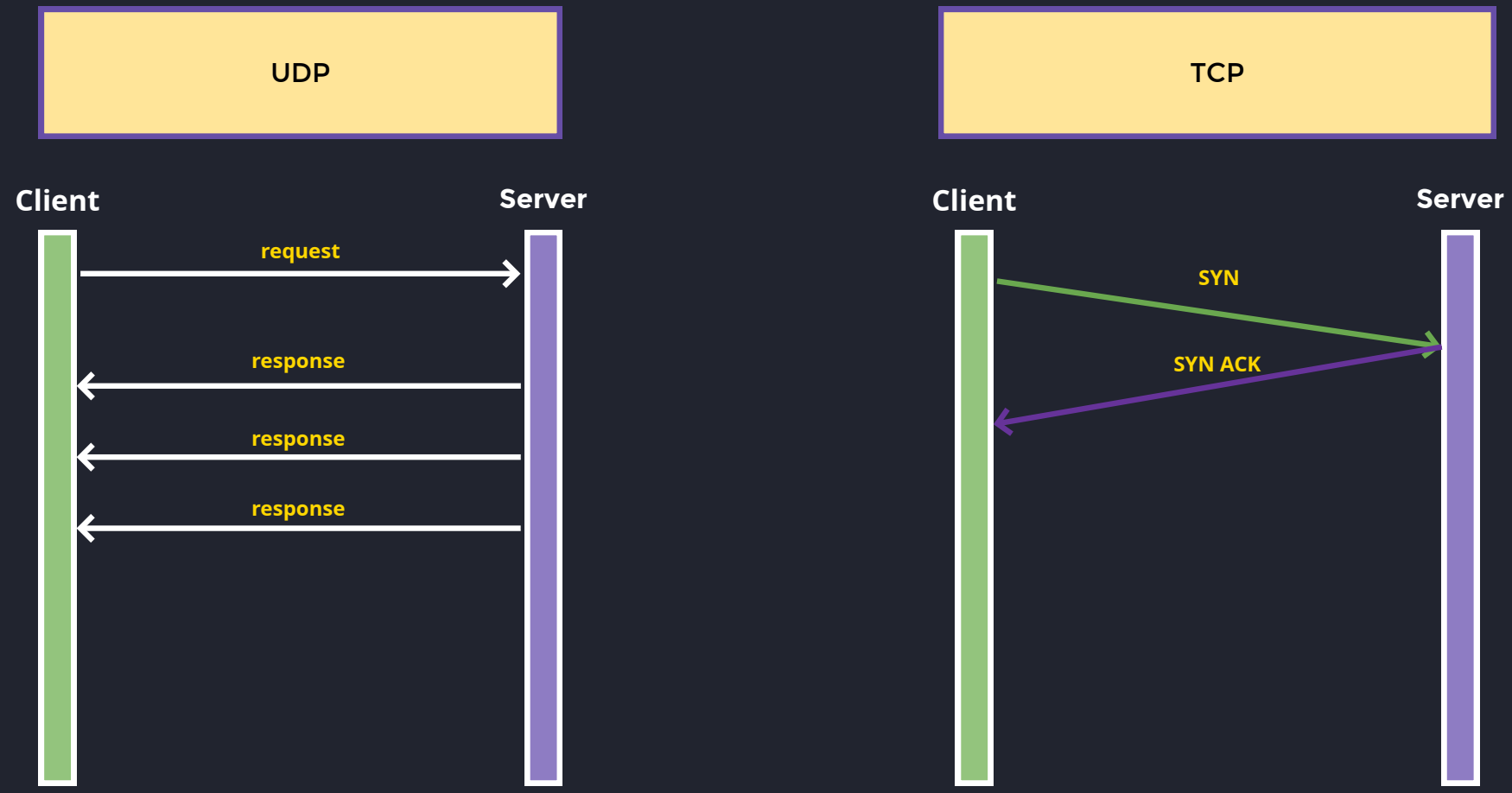
Network Connectivity: Protocols overview



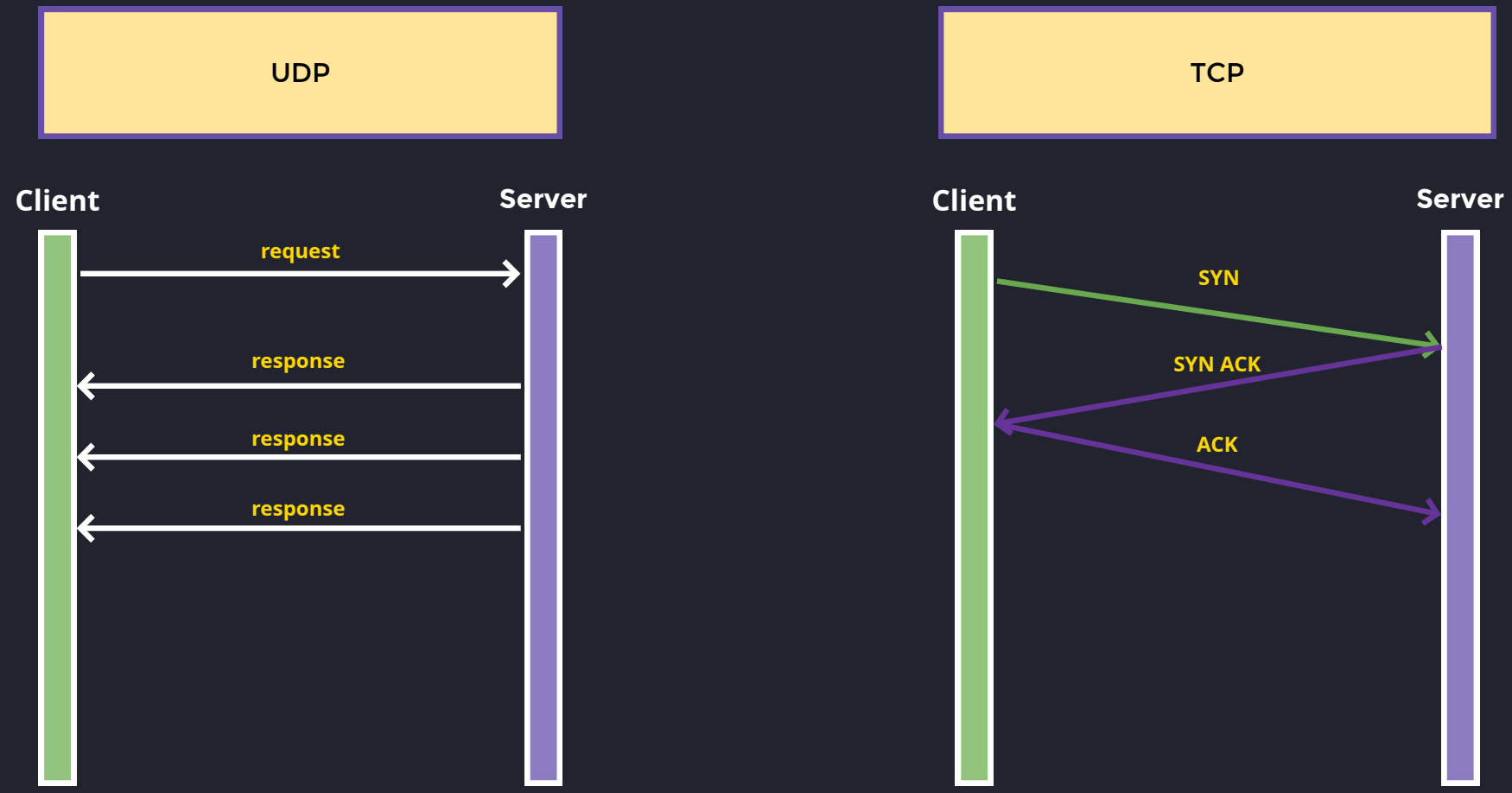
Network Connectivity: Protocols overview



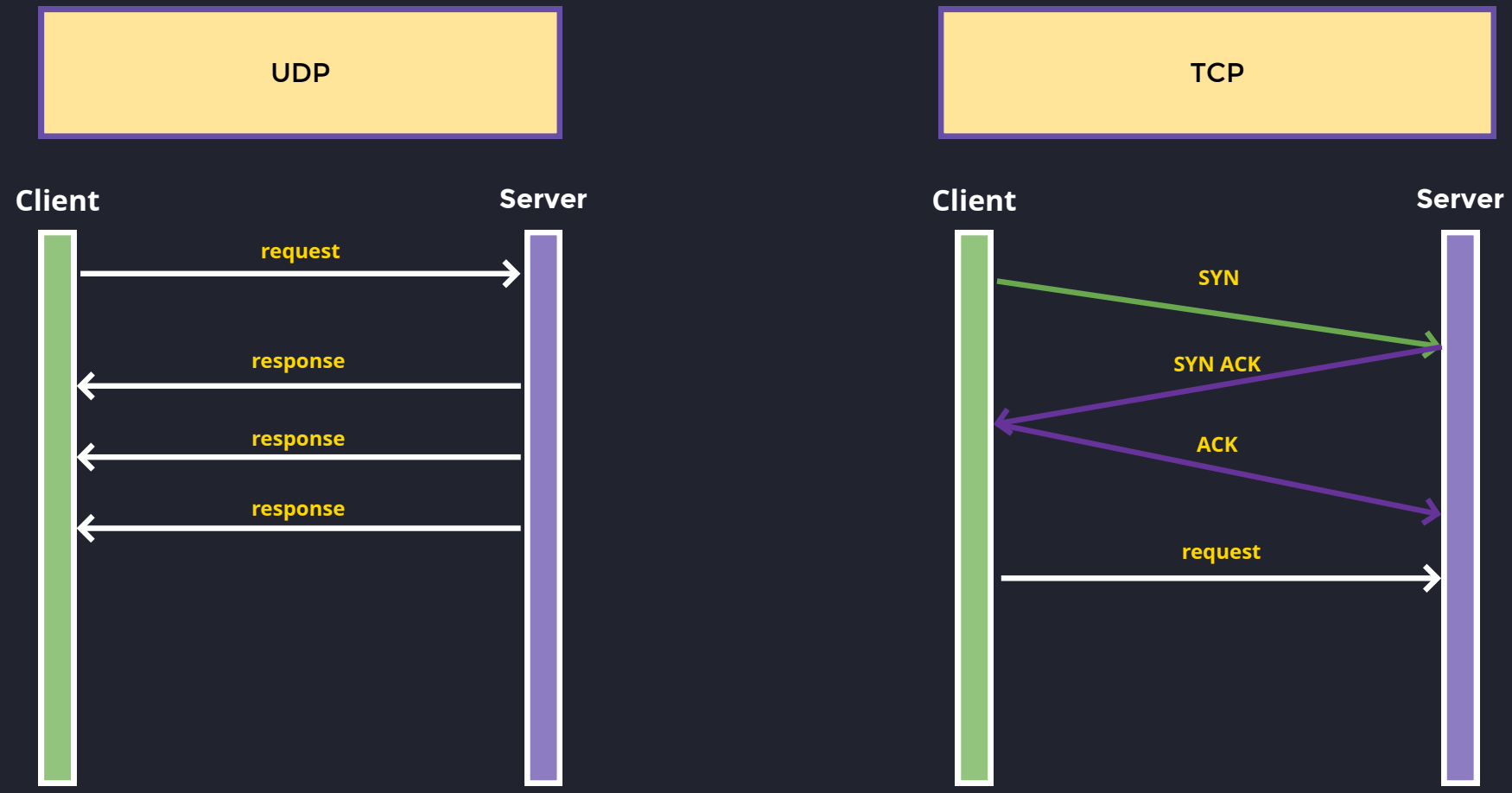
Network Connectivity: Protocols overview



Network Connectivity: Protocols overview



Network Connectivity: Protocols overview

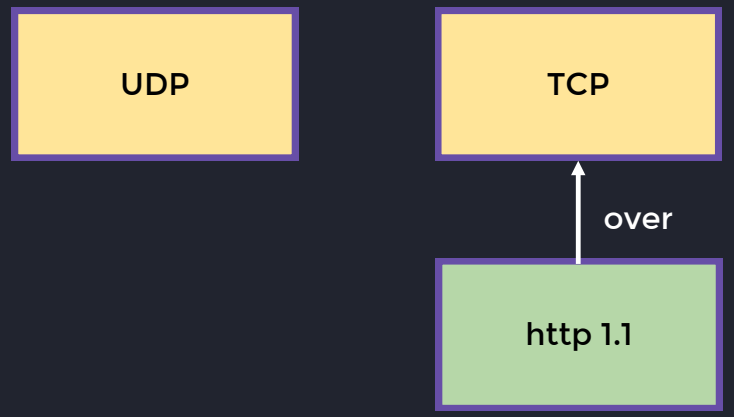


Network Connectivity: Protocols overview

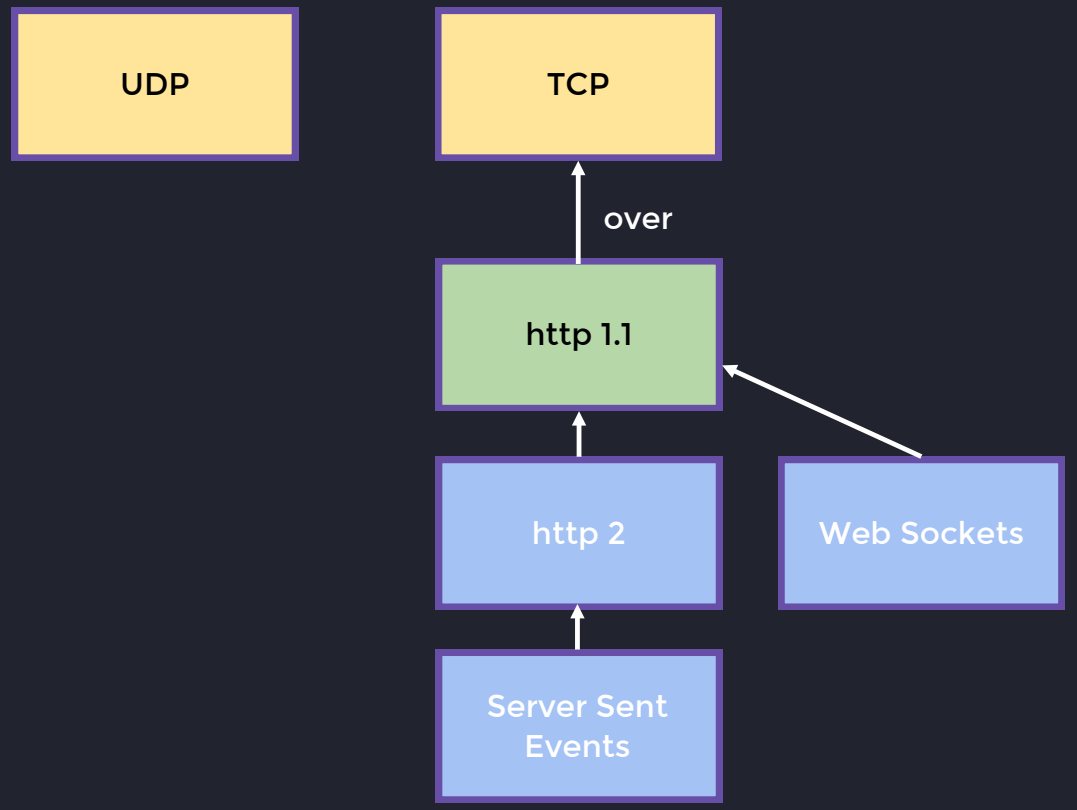
UDP

TCP

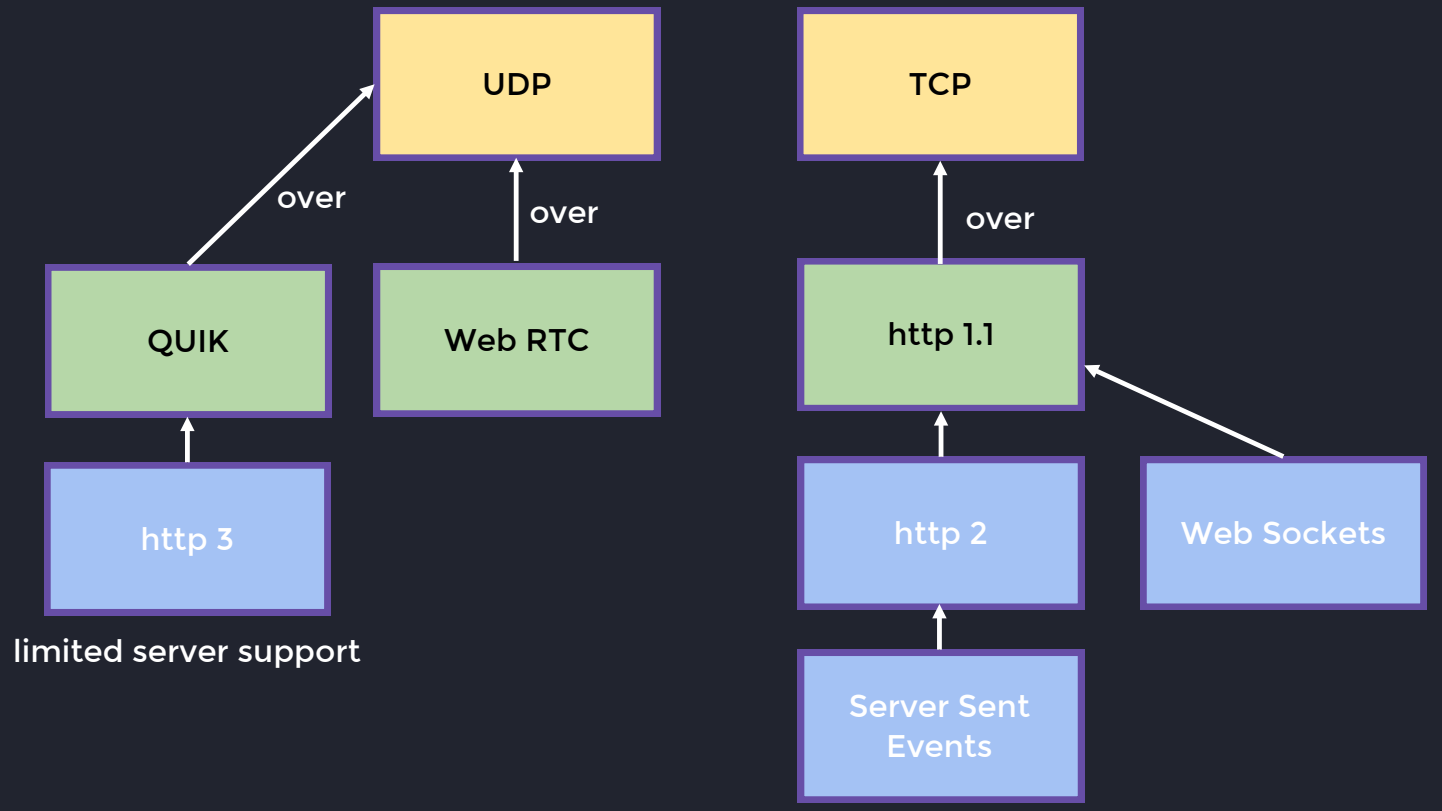
Network Connectivity: Protocols overview



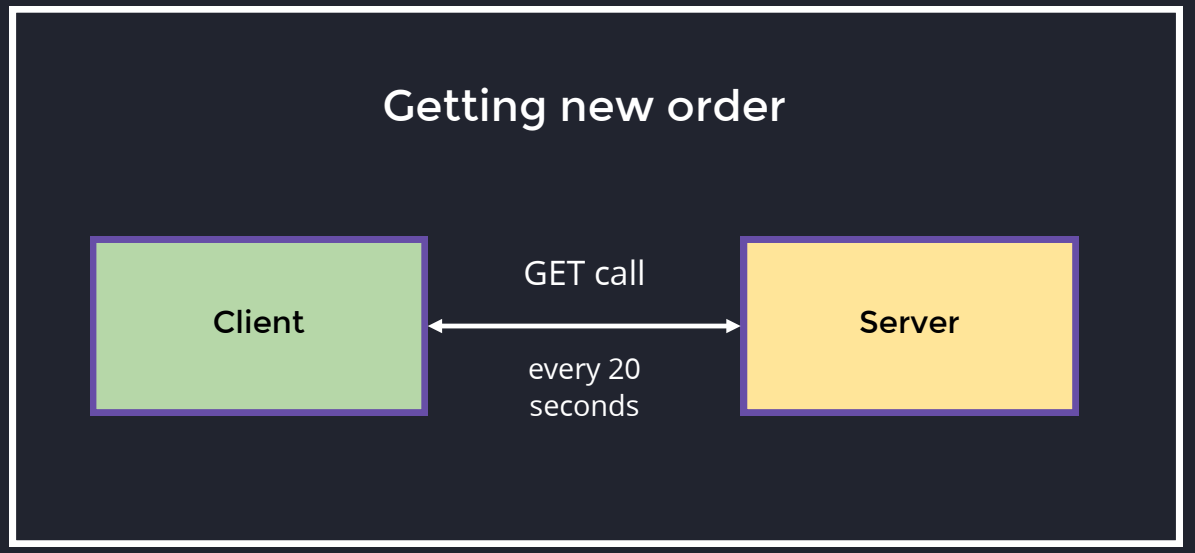
Network Connectivity: Protocols overview



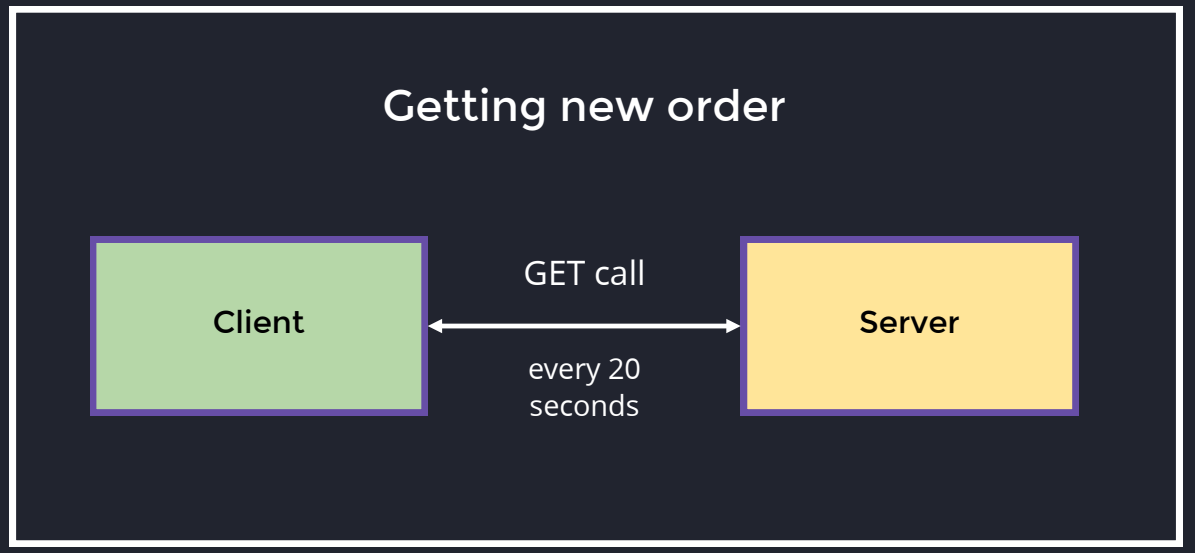
Network Connectivity: Protocols overview



Network Connectivity: Long polling

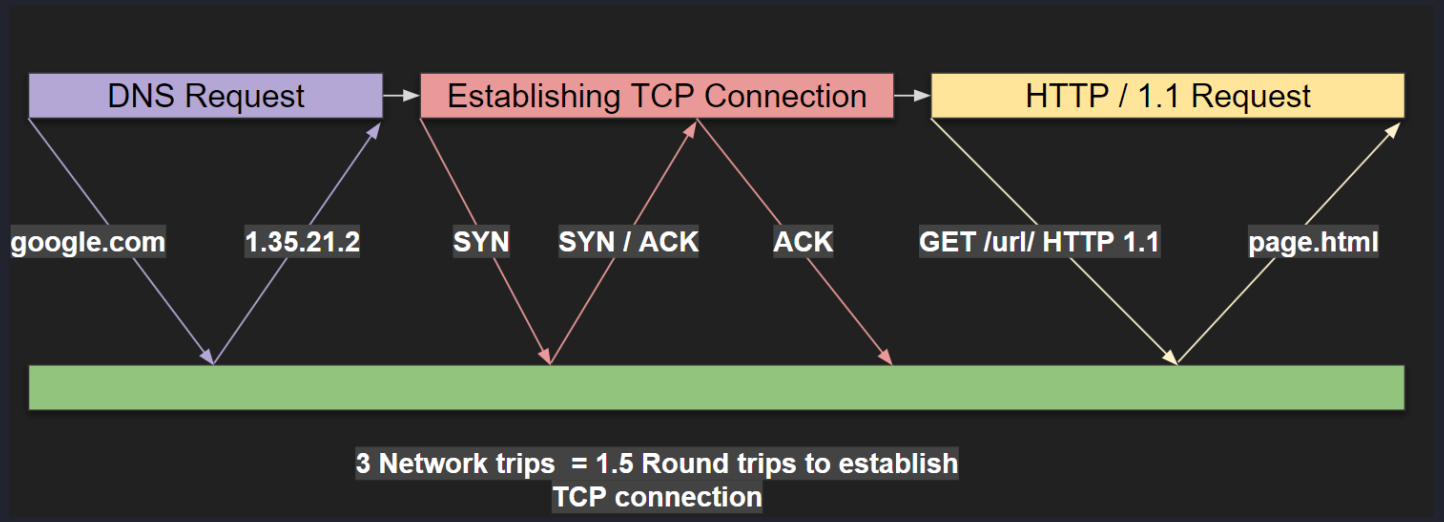


Network Connectivity: Long polling

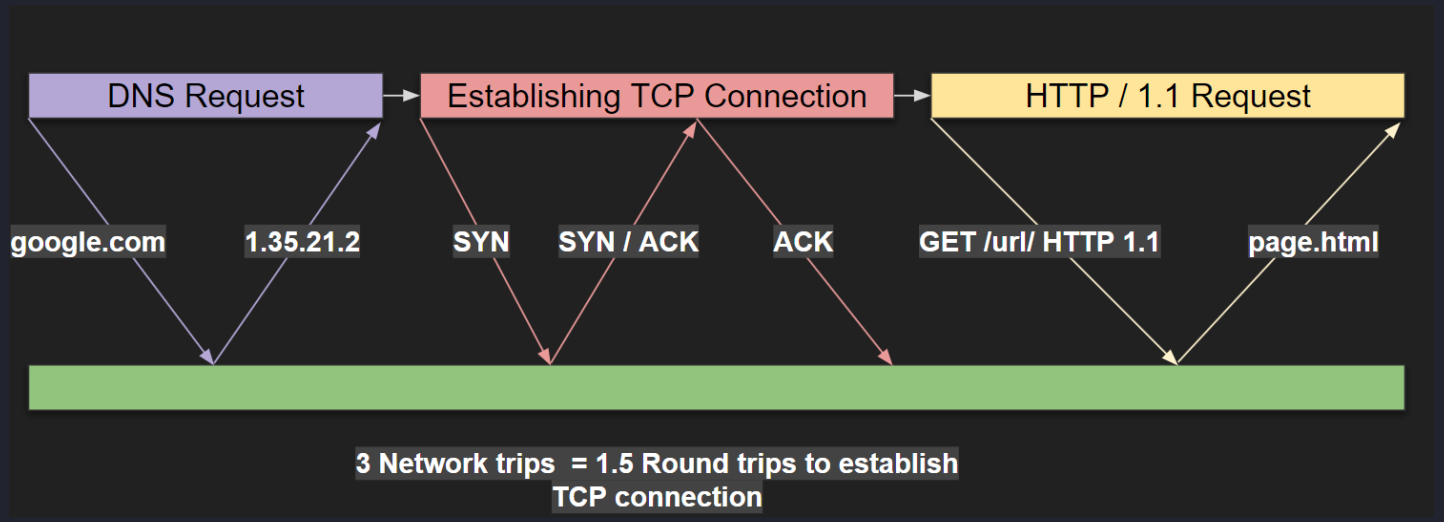



```
1 setInterval(() => {
2   fetch(api, params).then(updateOrders);
3 }, timeout);
```

Problem 1: Energy Consumption



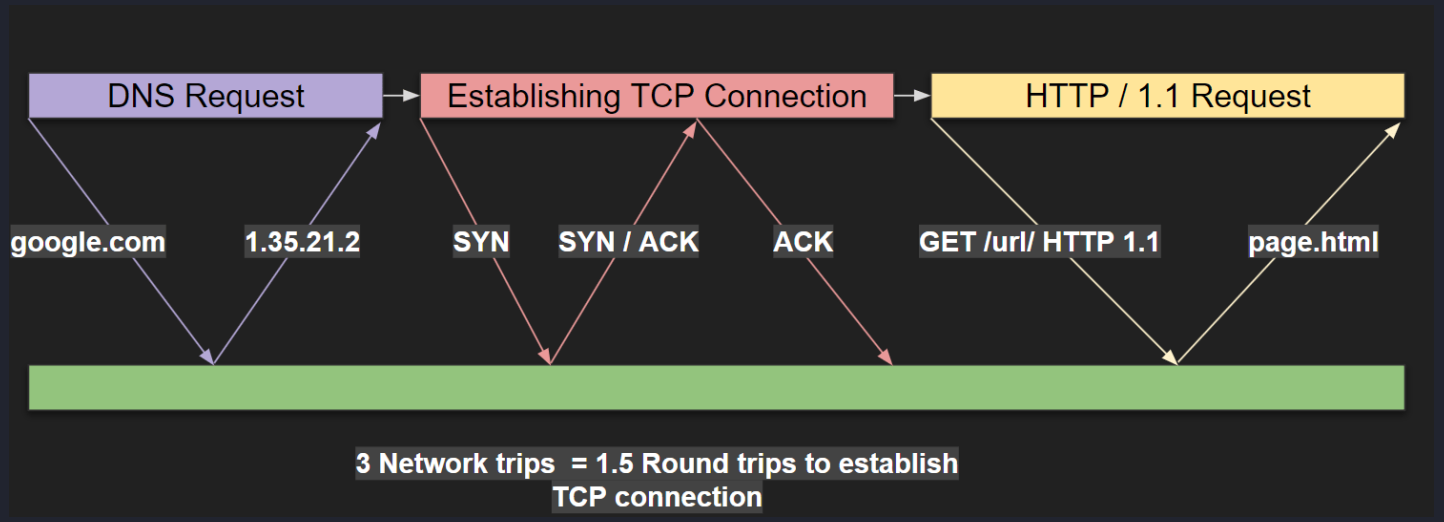
Problem 1: Energy Consumption



Establishing the connection:

- It costs **a lot of CPU time**.
- **Drains energy** as a result
- **Inefficient network usage** since you need to send the header data in each request

Problem 1: Energy Consumption



Establishing the connection:

- It costs **a lot of CPU processing**.
- **Drains energy** as a result
- **Inefficient network usage** since you need to send the header data in each request

Energy drain estimate (link):

Timeout: 30 seconds

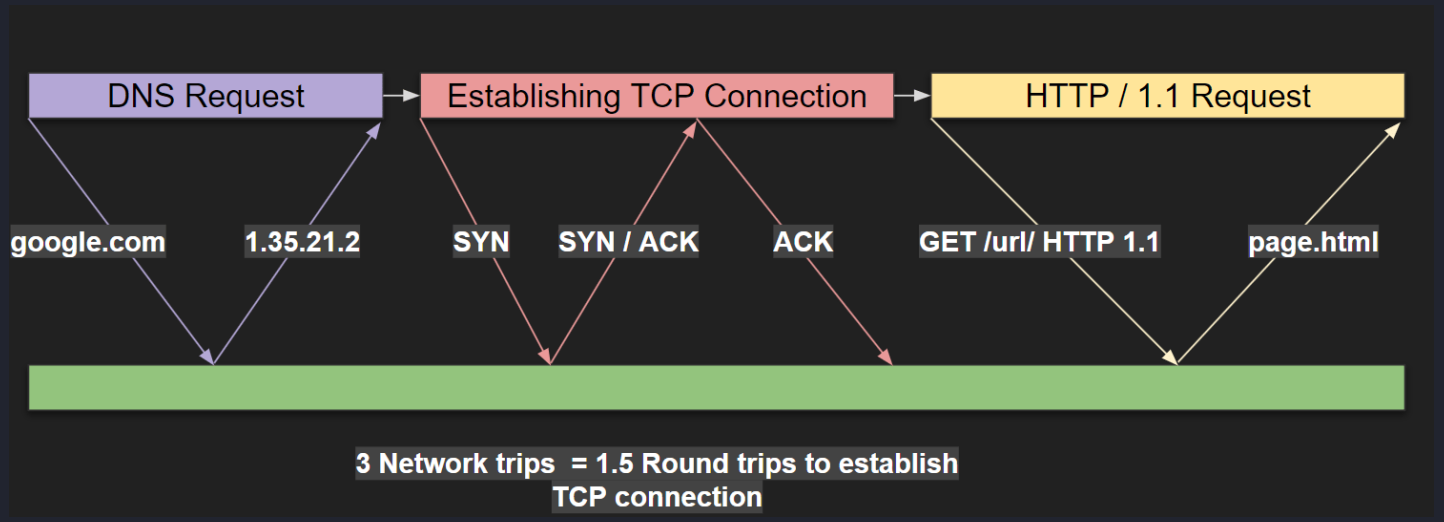
Timeframe: 5 minutes

Battery: 2000 mAh / 3.7V

Energy use: 600 joules / 5-minute

Battery life: 3.7 hours

Problem 1: Energy Consumption



Establishing the connection:

- It costs **a lot of CPU processing**.
- **Drains energy** as a result
- **Inefficient network usage** since you need to send the header data in each request

Energy drain estimate (link):

Timeout: 30 seconds

Timeframe: 5 minutes

Battery: 2000 mAh / 3.7V

Energy cost: 600 joules / 5-minute

Battery life: 3.7 hours

Mobile Network Module

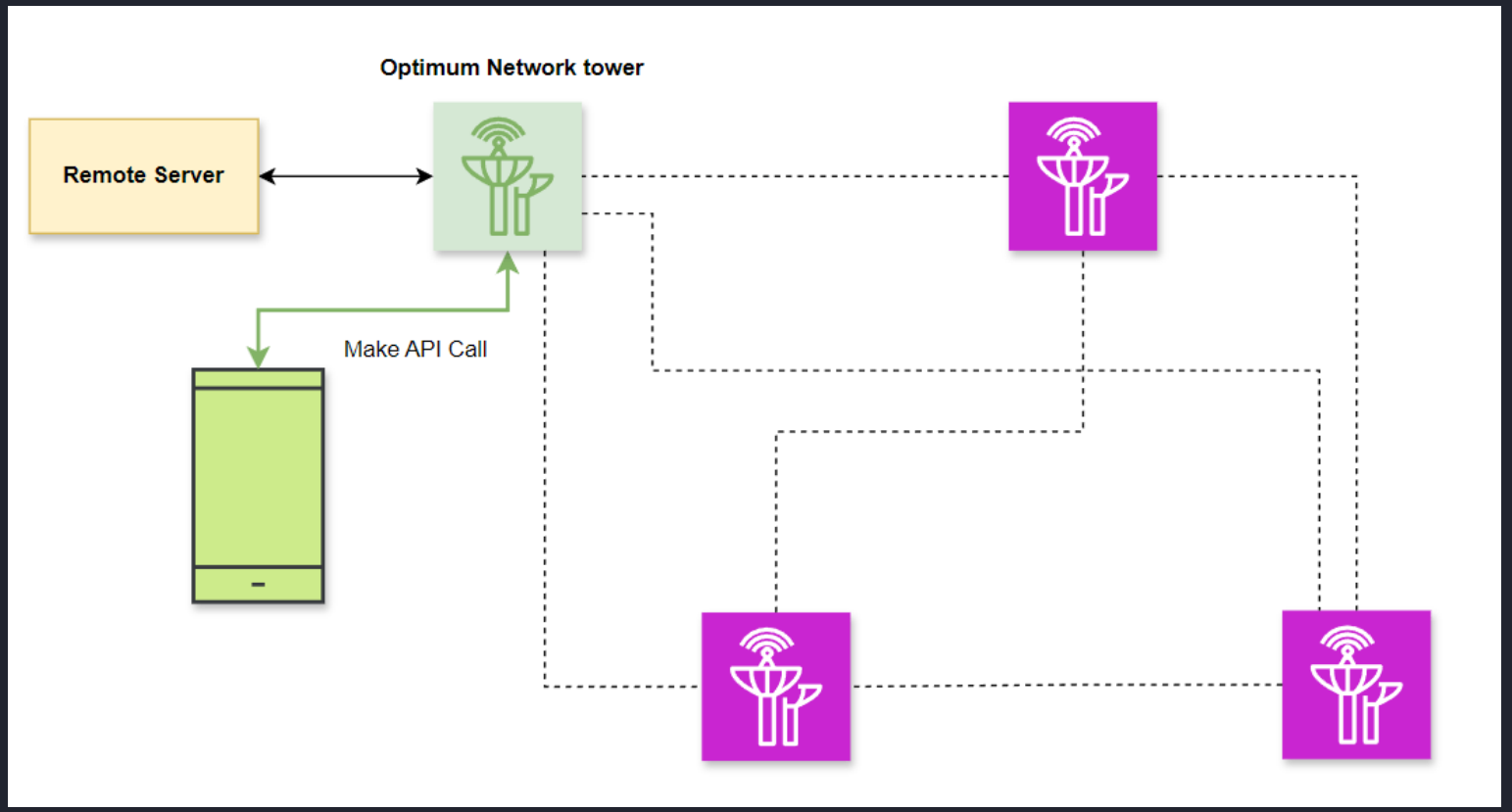
Mono (receive-only)

Energy efficient, but can only receive data

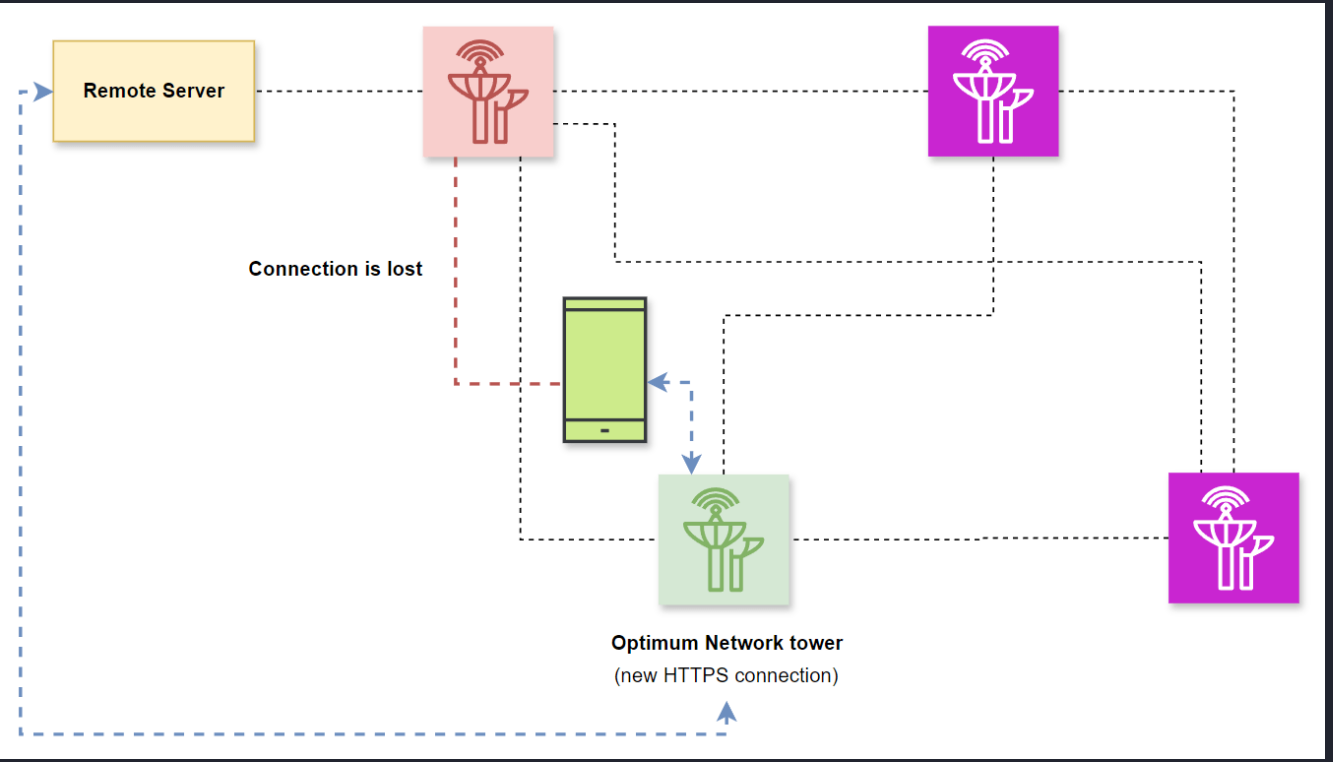
Duplex (bi-directional)

Can receive and sent the data, but use a lot of energy

Problem 2: Latency



Problem 2: Latency



Reconnection complexities:

1. Server needs to keep a state
2. Reconnection needs to be implemented on the client
3. New TCP connection (3 way hand-shake)

Long Polling: Summary

Pros:

1. Easy & cheap to implement
2. No additional infrastructure is needed.
3. 99.9% of servers will support that

Cons:

1. Battery inefficient (High CPU usage due to open TCP connection and transmitter usage)
2. Network & Data inefficient
3. HTTP 1.1 requires request headers to be sent with every request
4. Latency can degrade very quick on mobile networks.

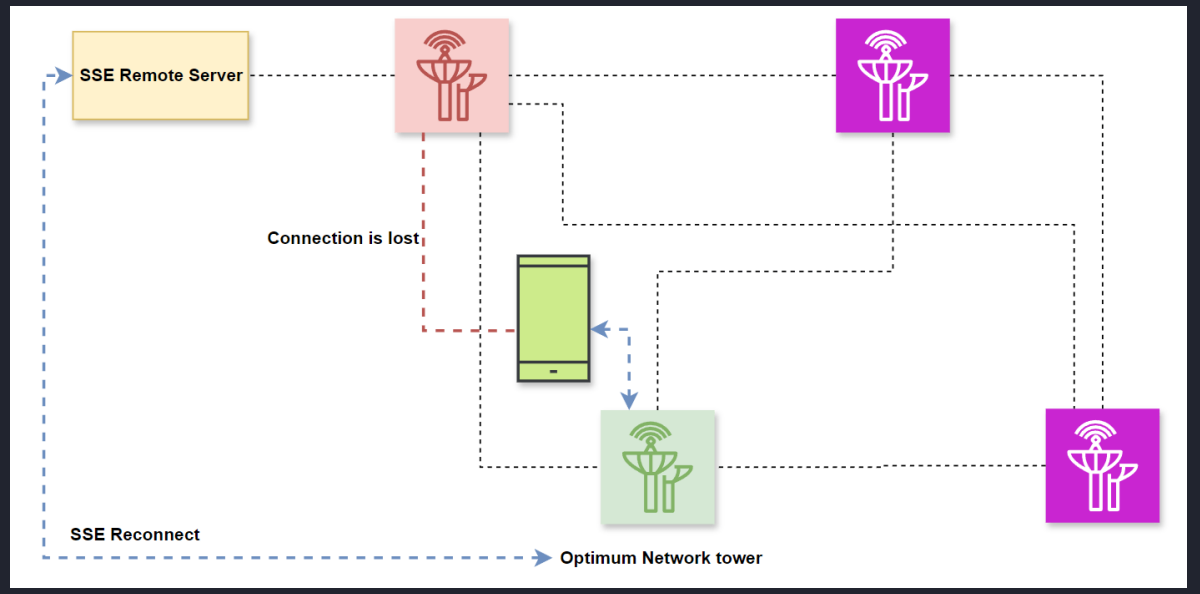
When to use it

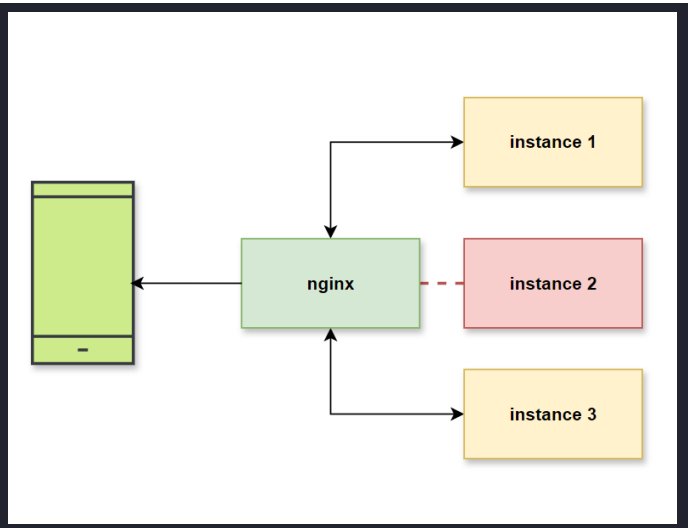
A desktop web application where some delays are acceptable. For instance, loading new group posts is totally fine with long polling on a desktop.

When to avoid

Mobile web application

Network Connectivity: Server Sent Events





Benefits of SSE

1. Duplex communication is only used when establishing initial connection
2. It doesn't send junk data (unnecessary headers)
3. Reconnection is **handled automatically**
4. Easy to scale since servers don't need to know the state
5. Since SSE is HTTP2 based, it can re-use existing TCP connection with a server

Summary: Server Sent Events

Pros:

1. Automatic reconnection handling
2. Battery efficient (uses only receiver antenna)
3. Relatively easy to scale in terms of infrastructure
4. Minimal network overhead (you only receive the actual data without header overhead)
5. Fast

Cons:

1. You can't push the data to the server.
2. Only string data is supported - you'll have to parse the payload

When to use it

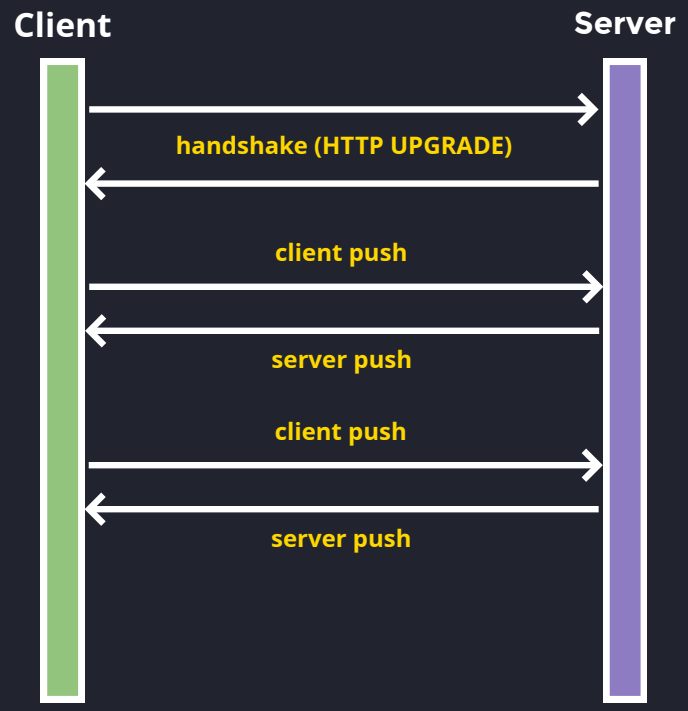
1. A desktop / mobile web application where you must receive data with minimum latency.
2. It can also be an alternative to `Web-Sockets` when some minor latency is acceptable.
3. Large text-data streaming

When to avoid

Simple desktop apps, you'll be fine with long-polling

Network Connectivity: Web-Sockets

Anatomy



1. The client sends a handshake request with **`UPGRADE`** headers.
2. The server responds with a successful request.
3. Browser upgrades the protocol to **Web-Sockets**
4. Client and Server has a **`bi-directional`** communication meaning that in will use TCP connection to send binary packages

Network Connectivity: Web-Sockets

HTTP vs Web-Sockets

How web-sockets are different from HTTP requests?

1. HTTP is used make only initial **handshake**
2. Pure TCP connection is used afterwards
3. TCP-protocol allows to establish ~65K connections within one socket

Pros:

1. web-sockets provide almost `real-time` communication mechanism
2. Unlimited number of connections

Cons:

1. **Infrastructure cost.**
2. **Engineering cost.**
3. **Reconnection is not implemented.**
4. **Web-Sockets** are stateful.
5. **Computing resource ineffeciency.**
    1. needs to maintain a constant TCP connection,
    2. uses duplex antenna
    3. drains energy and utilizes CPU

Summary: Web-Sockets

When to use it

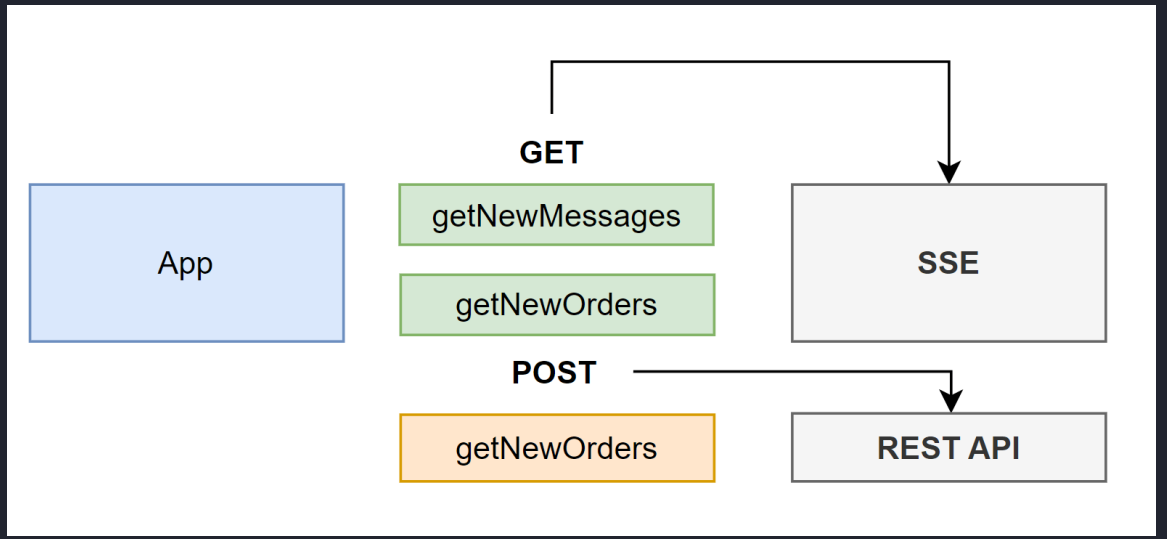
`Real-time` communication environments. This is a huge advantage of the protocol.

Environments such as:

1. Work with machine sensors / controls
2. Online gaming
3. Trading
4. Precise location tracking

Summary: API Design example

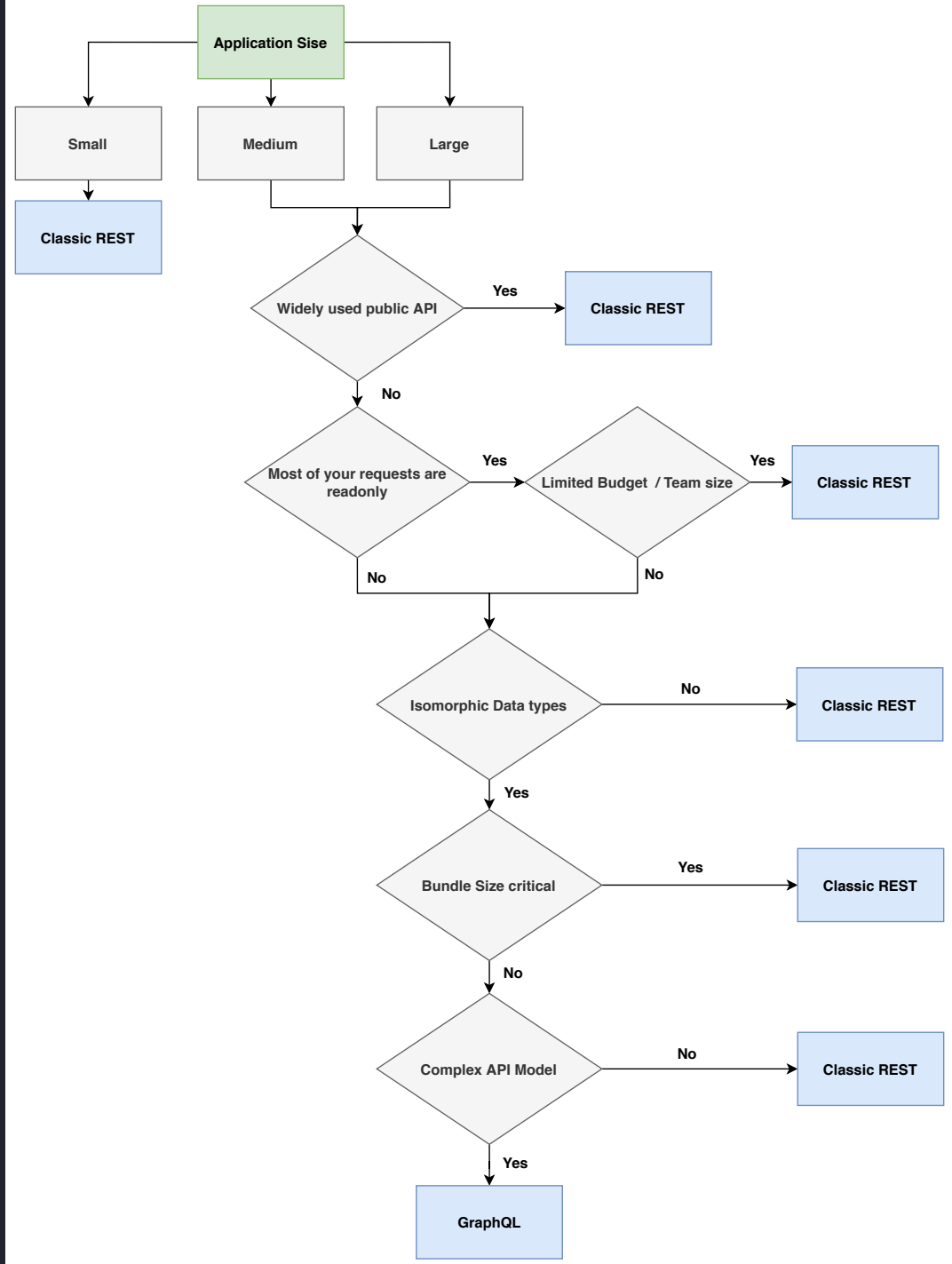
	Mobile	Desktop	
getShoppingOrders	SSE	Long-polling, SSE	`getShoppingOrders(timestamp: number, count: number, token): Order`
getNewMessages	SSE	Short-polling, SSE	`getNewMessages(timestamp: number, count: number, token): Message`
sendMessage	HTTP POST	HTTP POST	`sendMessage(message, token): Message`



Classic REST / GraphQL

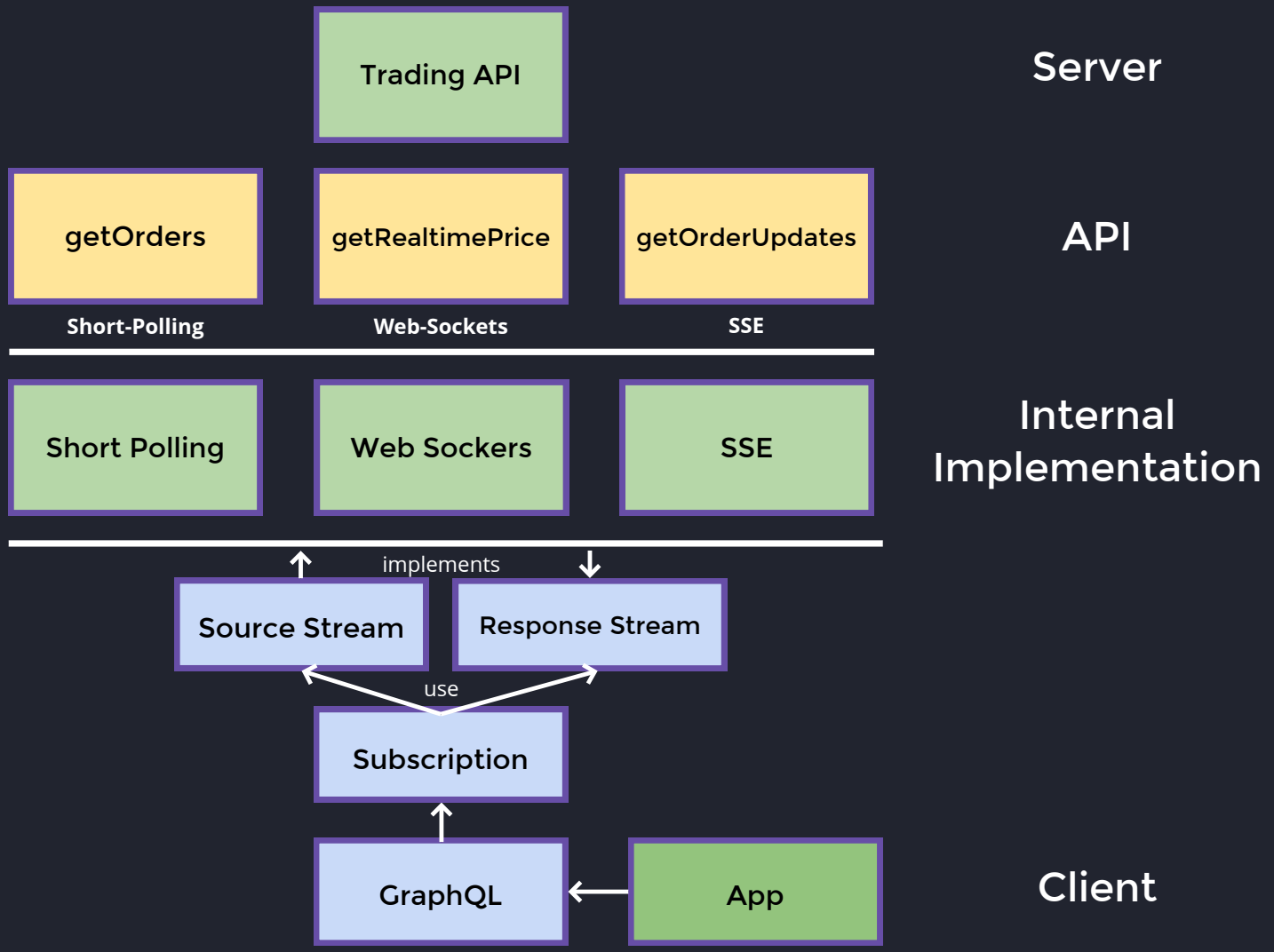
GraphQL introduces additional complexity:

1. Additional **client library** to work with server **GraphQL API**
2. Additional **client caching** layer
3. Additional **state manager** - **GraphQL Client** is responsible for syncing state between client and the server
4. Potential impact on your web-bundle size


Application Sise
Small
Medium
Large
Classic REST
Widely used public API
Yes
Classic REST
No
Most of your requests are readonly
Yes
Limited Budget / Team size
Yes
Classic REST
No
No
Isomorphic Data types
No
Classic REST
Yes
Bundle Size critical
Yes
Classic REST
No
Complex API Model
No
Classic REST
Yes
GraphQL

GraphQL provides the most value in Complex Apps.

GraphQL can reduce the complexity

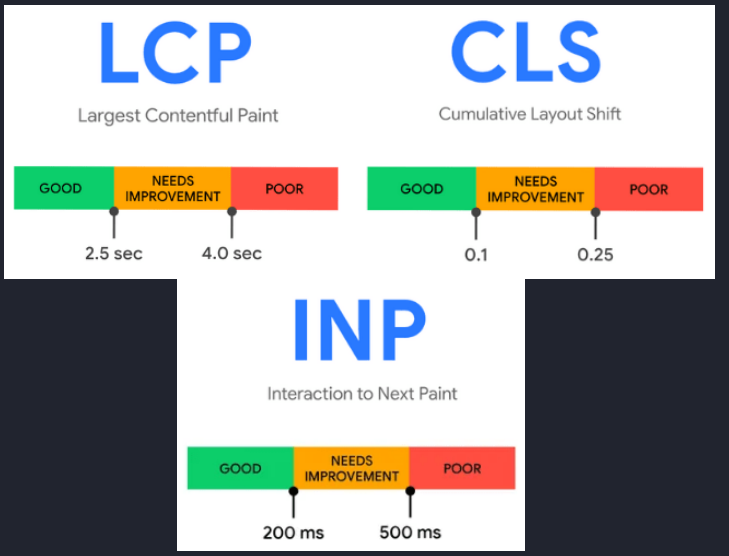


Summary: Network Connectivity

Performance Optimization

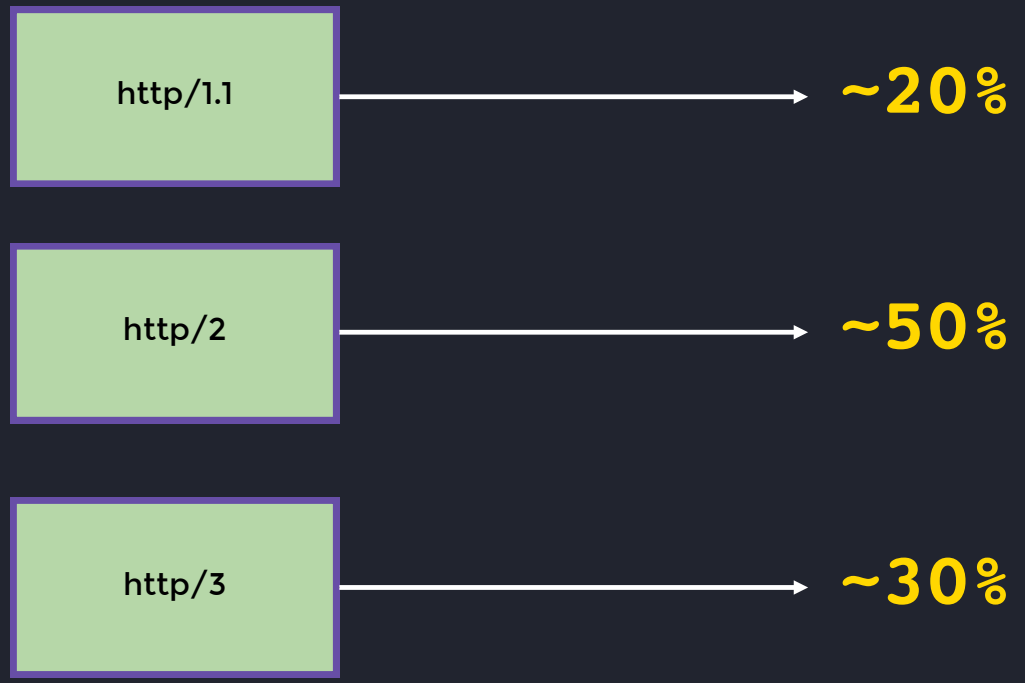
Performance Optimization: Web Vitals

1. **Largest Contentful Paint (LCP):**
   - **What it measures:** Loading performance.
   - **Goal:** LCP <=2.5 seconds of page load start for a good user experience.
2. **Interaction to Next Paint (INP):**
   - **What it measures:** Interactivity.
   - **Goal:** INP <= 200 milliseconds or less for optimal user experience.
3. **Cumulative Layout Shift (CLS):**
   - **What it measures:** Visual stability.
   - **Goal**: Maintain a CLS of 0.1 or less to ensure a visually stable experience.



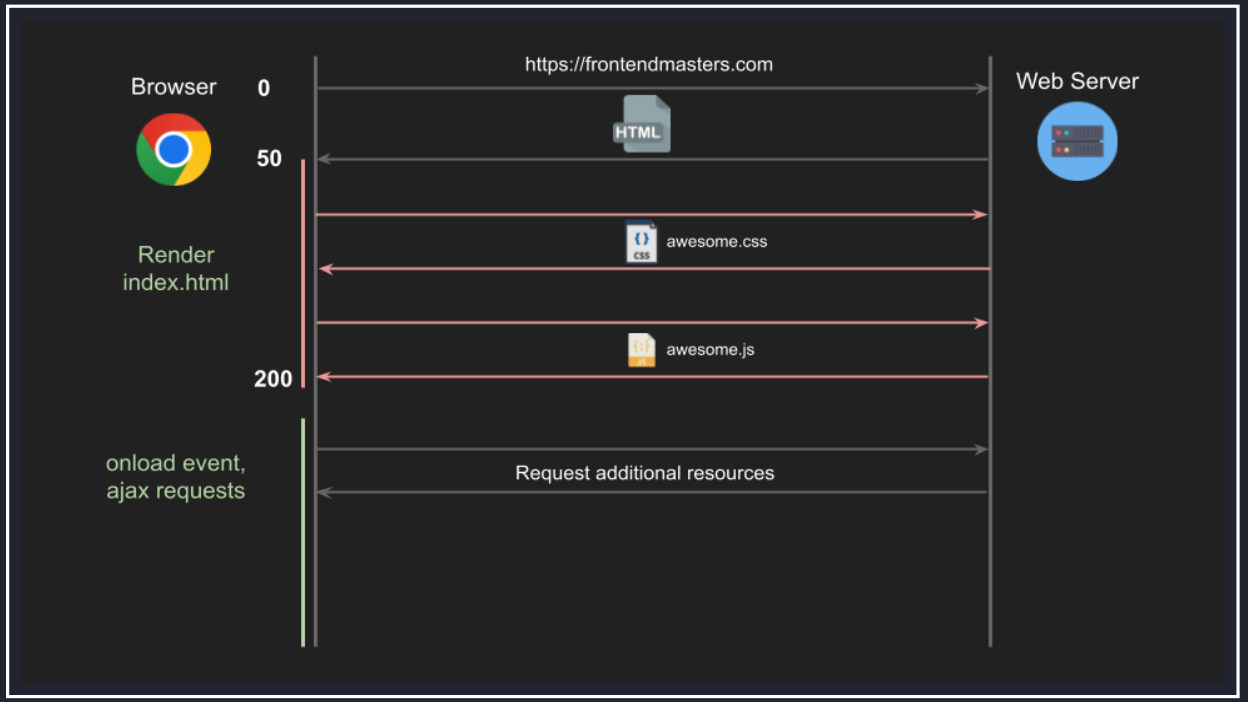
Performance Optimization: Network Performance

Communication Protocol



Performance Optimization: Network Performance

http/1.1 - Loading efficiency limitation

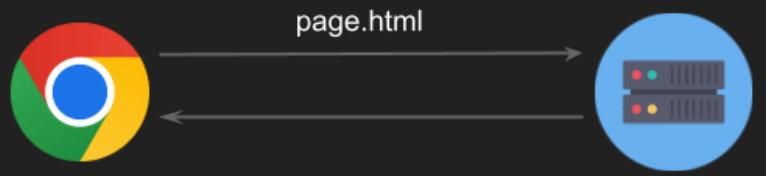


1. we request some **`html`** page
2. the server responds with the requested page
3. Browser start to render a web-page and sequentially load every resource
4. The page is rendered only when all critical resources on the html page are fully loaded
5. After rendering is completed, browser will be executing some additional request made by async scripts

Performance Optimization: Network Performance

http/1.1 - data overahead

```
GET /page.html HTTP/1.1↵

Accept: text/html,application/xhtml+xml,image/jxr/,*/*↵
Accept-Encoding: gzip, deflate, br↵
Accept-Language: en-GB,en-US;q=0.8,en;q=0.6↵
Host: www.frontend-engineer.com↵
User-Agent: Chrome 1.1↵↵
```



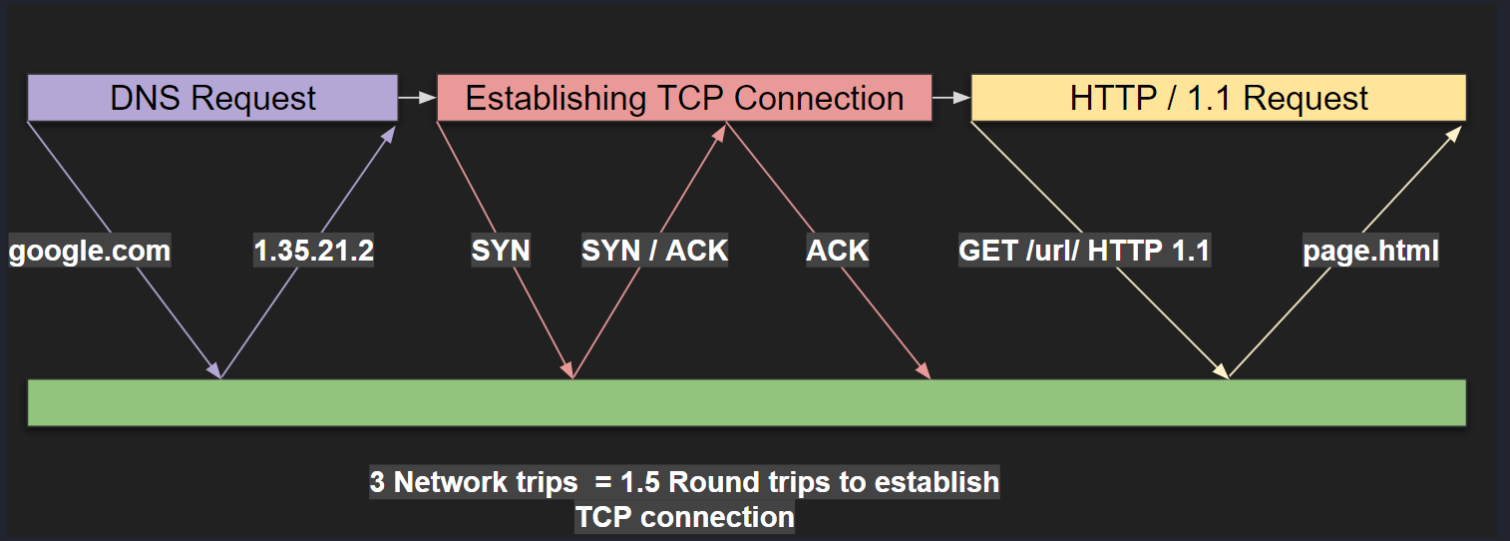
Performance Optimization: Network Performance

http/1.1 - maintance cost

HTTP/1.1 server costs more for business in the long run

Performance Optimization: Network Performance

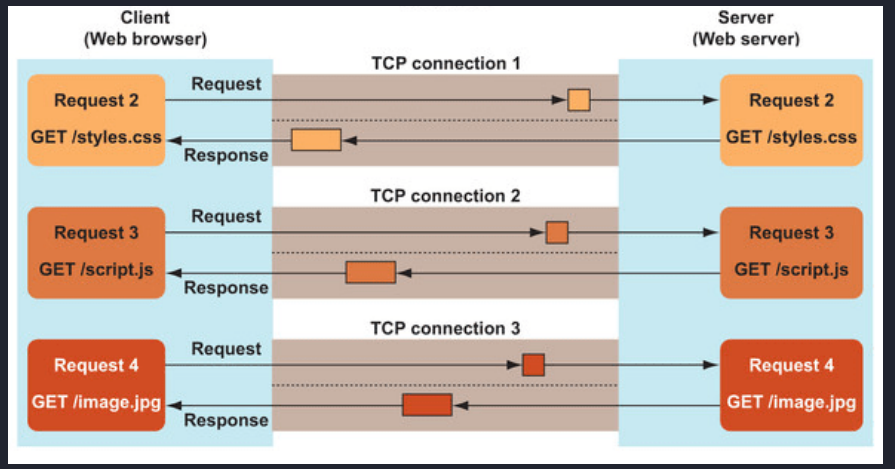
Benefits of http/2 and http/3



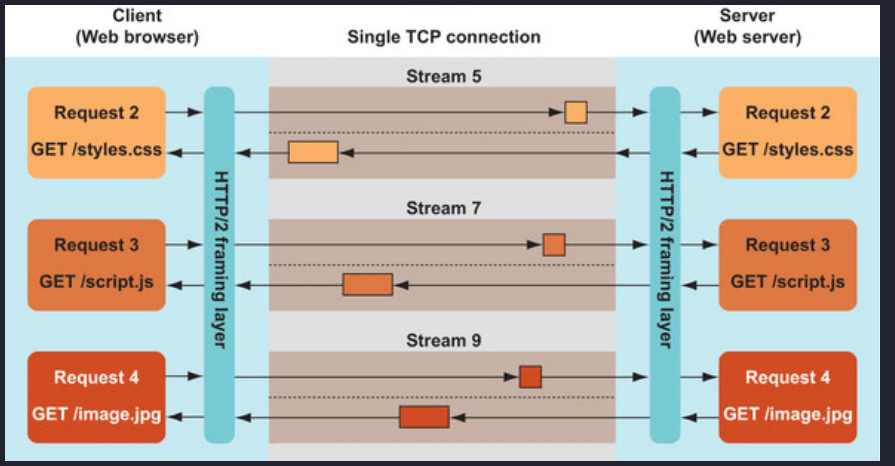
Performance Optimization: Network Performance

Benefits of http/2 and http/3 - Multiplexing

HTTP/1



HTTP/2



Performance Optimization: Network Performance

Benefits of http/2 and http/3 - Header Compression

HTTP/2+ provides 98% header compression

HTTP / 1.1 Header overhead	**5KB**
HTTP / 2 Header overhead	**12.5 bytes**

Performance Optimization: Network Performance

Javascript Bundle Optimization

Guess the bundle size?

```
"FrontendMasters".matchAll(/Frontend/);
```

Performance Optimization: Network Performance

Javascript Bundle Optimization

Guess the bundle size?

```
"FrontendMasters".matchAll(/Frontend/);
```

Function	ES5 Bundle Size	ES2020 Bundle size
String.prototype.matchAll	16.9 KB	69 B

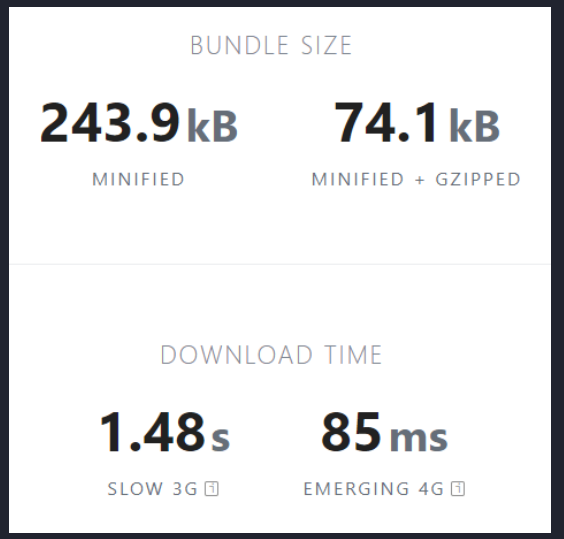
Performance Optimization: Network Performance

Javascript Bundle Optimization

Guess the bundle size?

```
"FrontendMasters".matchAll(/Frontend/);
```

Function	ES5 Bundle Size	ES2020 Bundle size
String.prototype.matchAll	16.9 KB	69 B



Core JS

Performance Optimization: Network Performance

Javascript Bundle Optimization

Do we actually need polyfills?

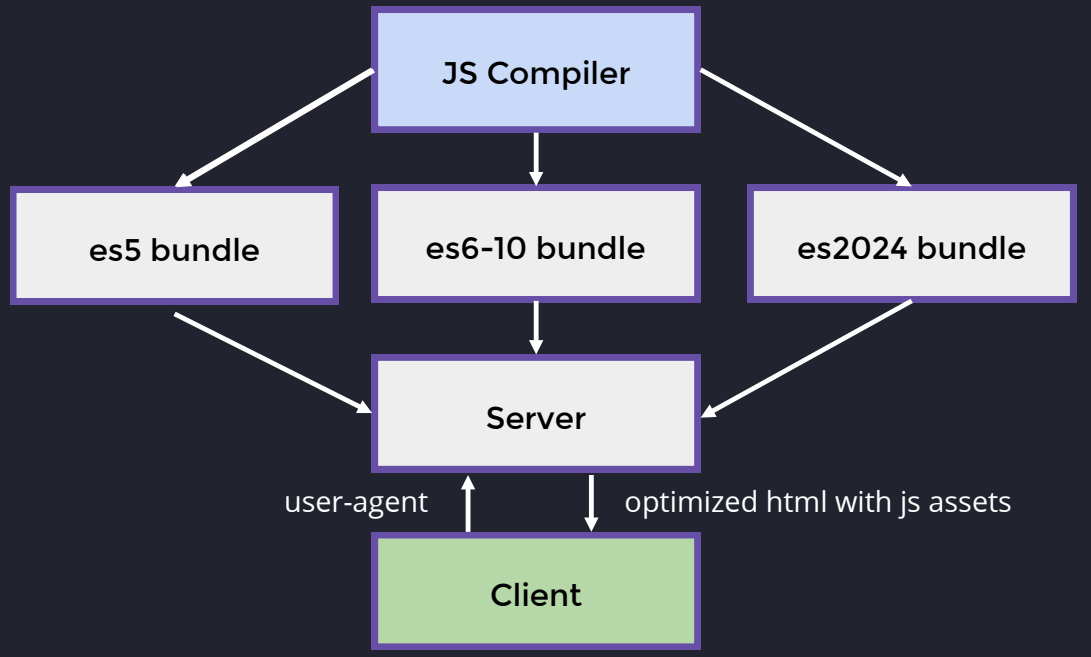
Here are a few facts:

1. **ES6** has a support of 98.2%
2. **ES7**-10 has a support of 96%
3. **ES11** - 90%
4. **ES12** - 89%

Performance Optimization: Network Performance

Javascript Bundle Optimization

Multi-bundle approach

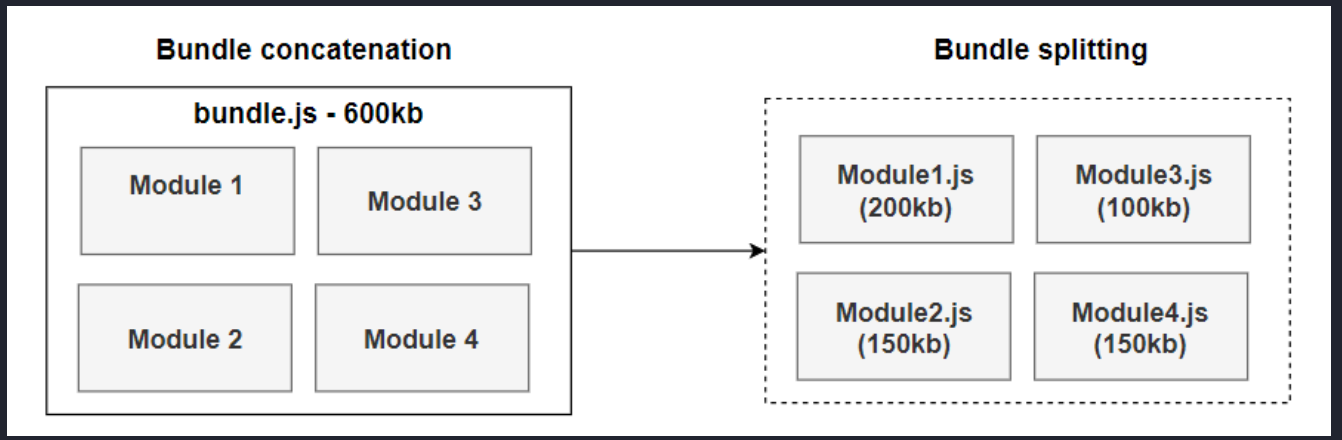


Performance Optimization: Network Performance

Javascript Bundle Optimization

Code Split

```
const cta = document.getElementById("next-button");
cta.addEventListener('click', async () => {
    const {render} = await import('./module2.js');
    render(root);
  }
);
```



Performance Optimization: Network Performance

Javascript Bundle Optimization

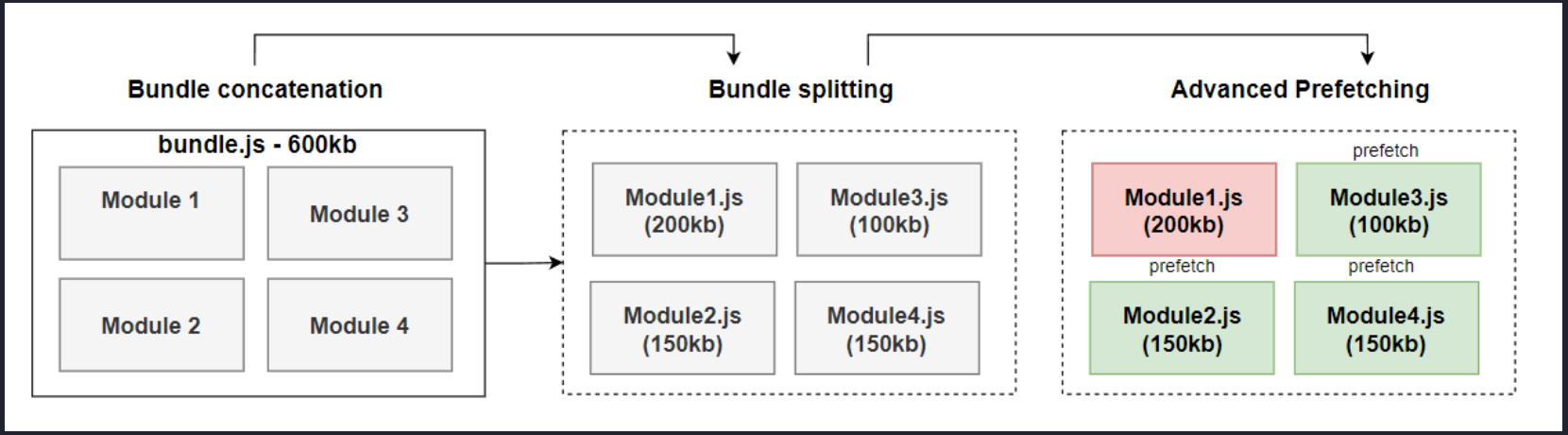
Code Pre-fetch

Two values that are particular useful in these case are:

`<link rel="preload">` - preloads a resource in background with a high priority.

`<link rel="prefetch">` - preloads and caches a resource in background with a low priority.

```
<link rel="prefetch" href="./module2.js">
<link rel="prefetch" href="./module3.js">
<link rel="prefetch" href="./module4.js">
```



Performance Optimization: Network Performance

Javascript Bundle Optimization

Code Minification & Compression

```
function helloWorld() {
  const longVariableName = 5 + 5;
  return longVariableName;
}
```

```
function h(){return 5+5}
```

Average size reduction ~ 20%

1MB -> 800KB

Performance Optimization: Network Performance

Javascript Bundle Optimization

Code Minification & Compression

```
function helloWorld() {
  const longVariableName = 5 + 5;
  return longVariableName;
}
```

```
function h(){return 5+5}
```

Average size reduction ~ 20%

1MB -> 800KB

Performance Optimization: Network Performance

Javascript Bundle Optimization

Code Minification & Compression

There are two main options

`brotli` is 20-30% more efficient, but less supported by older browsers

`gzip` is much faster and is supported in most of the browser

Library	Algorithm	Initial Size	Compressed	Ratio
jquery-3.7.1.js	Brotli	302 KiB	69 KiB	77%
jquery-3.7.1.js	gzip	302 KiB	83 KiB	73%
jquery-3.7.1.min.js	Brotli	85 KiB	27 KiB	68%
jquery-3.7.1.min.js	gzip	85 KiB	30 KiB	65%
lodash-4.17.21.js	Brotli	531 KiB	73 KiB	86%
lodash-4.17.21.js	gzip	531 KiB	94 KiB	82%
lodash-4.17.21.min.js	Brotli	71 KiB	23 KiB	68%
lodash-4.17.21.min.js	gzip	71 KiB	25 KiB	65%

Performance Optimization: Network Performance

Javascript Bundle Optimization

Code Minification & Compression

There are two main options

`brotli` is 20-30% more efficient, but less supported by older browsers

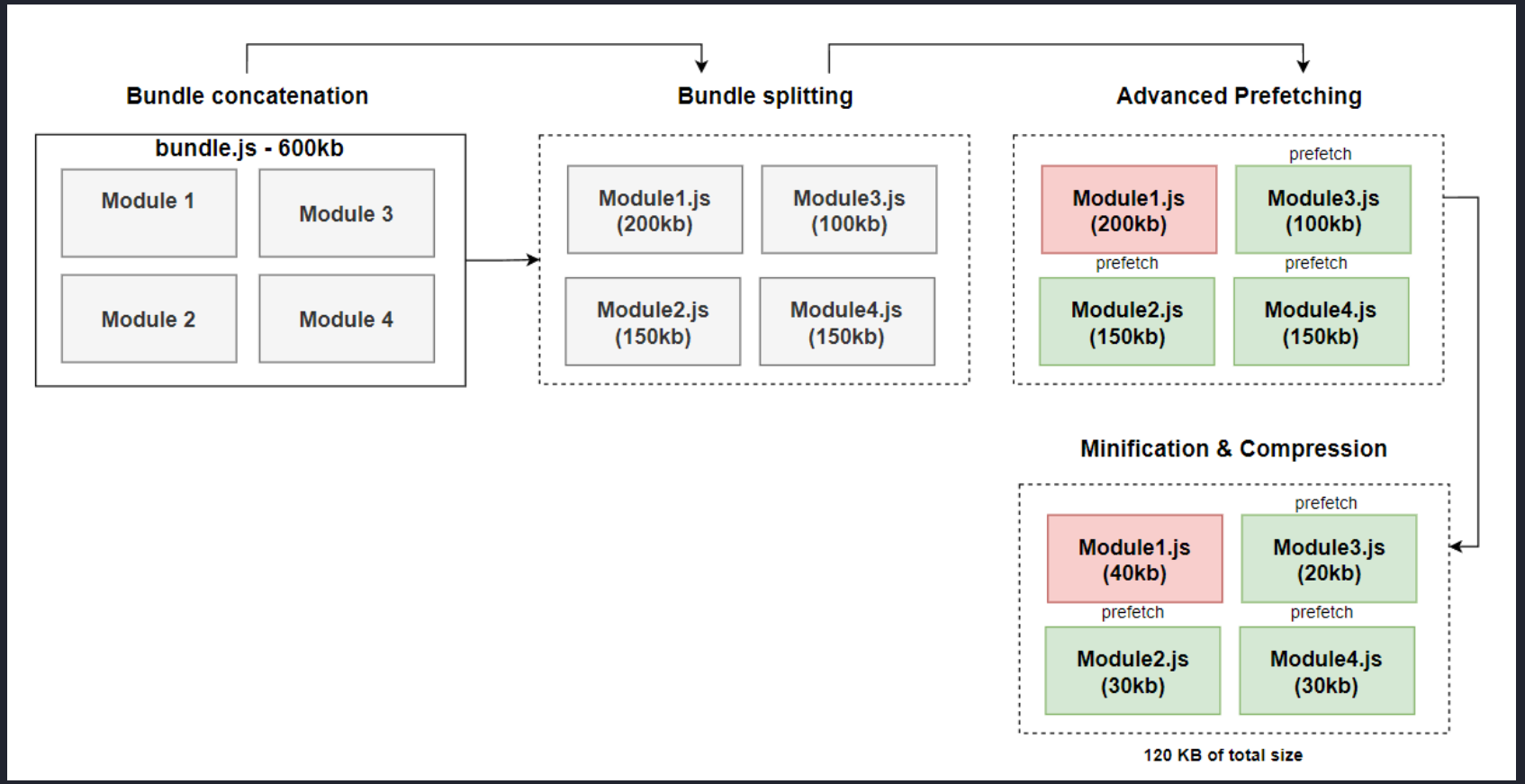
`gzip` is much faster and is supported in most of the browser

Library	Algorithm	Initial Size	Compressed	Ratio
jquery-3.7.1.js	Brotli	302 KiB	69 KiB	77%
jquery-3.7.1.js	gzip	302 KiB	83 KiB	73%
jquery-3.7.1.min.js	Brotli	85 KiB	27 KiB	68%
jquery-3.7.1.min.js	gzip	85 KiB	30 KiB	65%
lodash-4.17.21.js	Brotli	531 KiB	73 KiB	86%
lodash-4.17.21.js	gzip	531 KiB	94 KiB	82%
lodash-4.17.21.min.js	Brotli	71 KiB	23 KiB	68%
lodash-4.17.21.min.js	gzip	71 KiB	25 KiB	65%

Performance Optimization: Network Performance

Javascript Bundle Optimization

Code Minification & Compression



Performance Optimization: Network Performance

Javascript Bundle Optimization

Deferred Load

```
<script src="module1.js"></script>
<script src="analytics.js"></script>
<script src="telemetry.js"></script>
<script src="module2.js"></script>
```

1
2
3
4

Normal loading order

```
<script src="module1.js"></script>
<script src="analytics.js" defer></script>
<script src="telemetry.js" defer></script>
<script src="module2.js"></script>
```

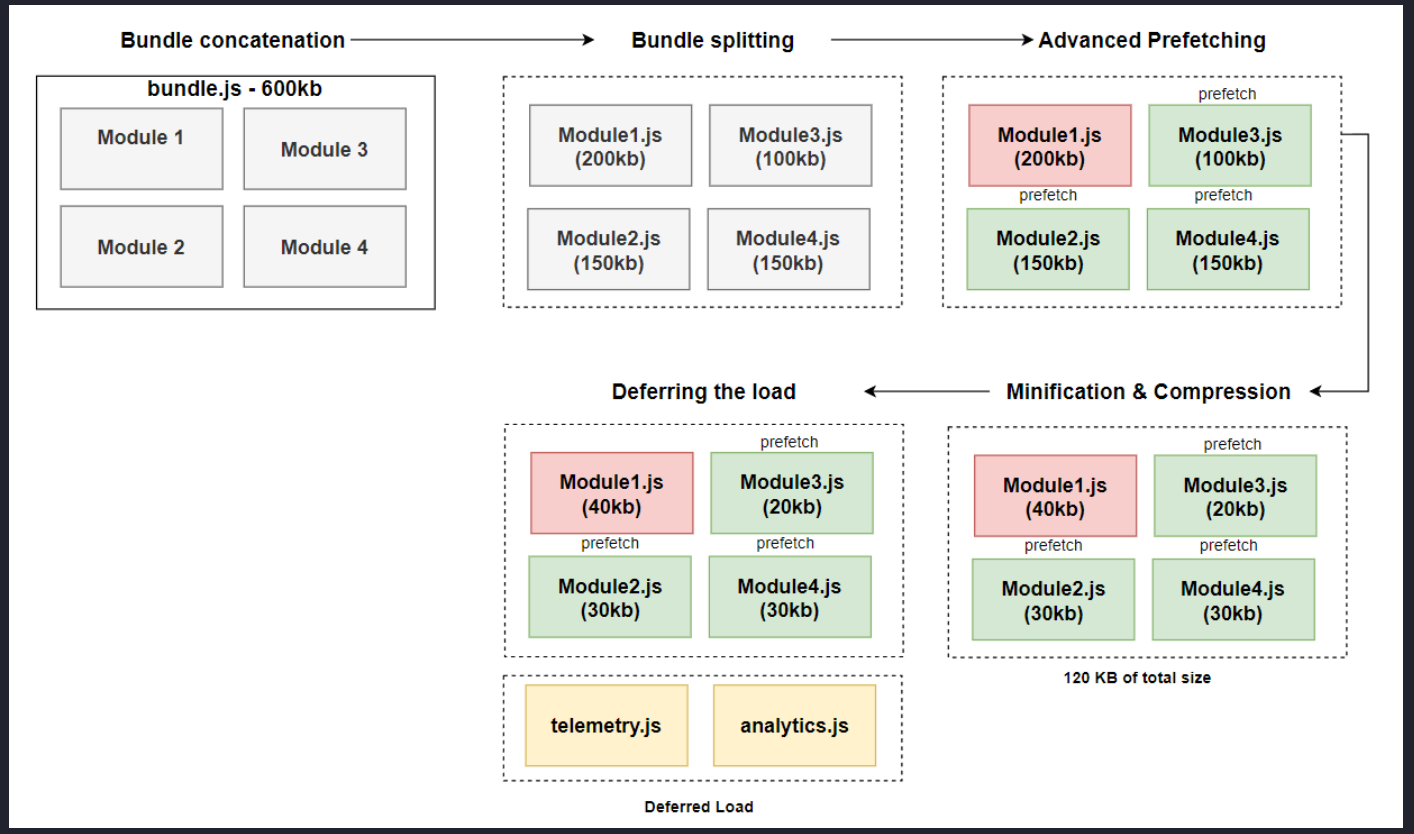
1
3
4
2

Load when every other asset is ready

Performance Optimization: Network Performance

Javascript Bundle Optimization

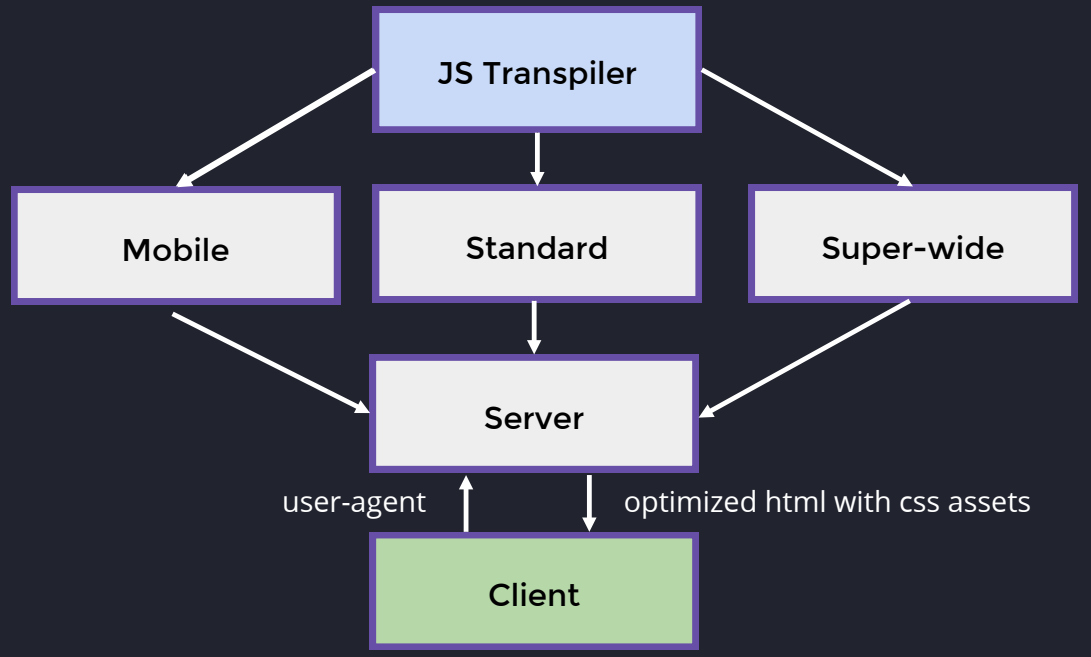
Deferred Load



Performance Optimization: Network Performance

CSS Bundle Optimization

Bundle Split



Performance Optimization: Network Performance

CSS Bundle Optimization

Minification and compression

Tailwind

Uncompressed	Minified	Gzip	Brotli
2413.4kB	1967.4kB	190.2kB	46.2kB

Performance Optimization: Network Performance

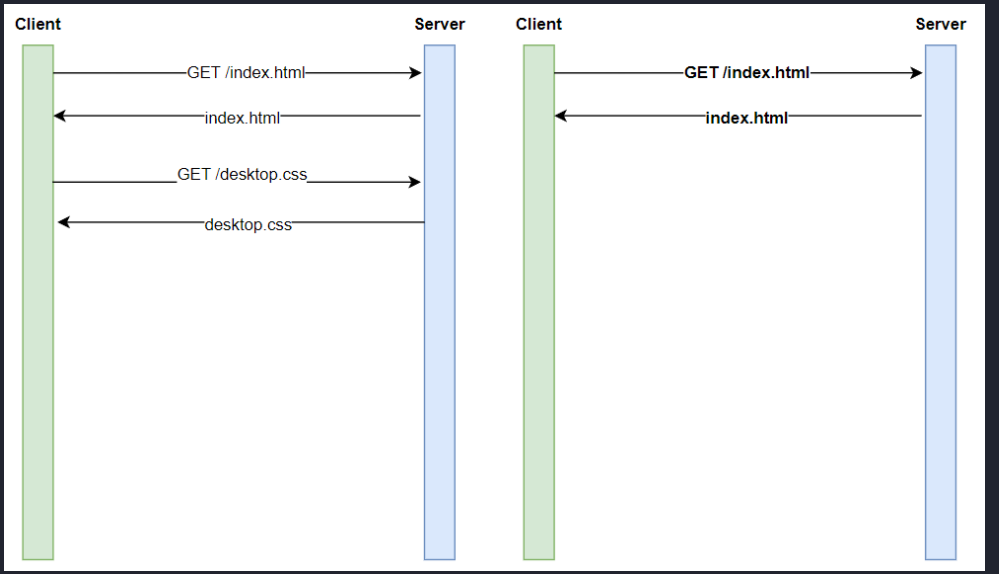
CSS Bundle Optimization

Critical style extraction

There are two types of styles within the app:

`critical` - application won't be rendered properly and can't be used

`non-critical` - pop-ups, advanced graphic features, inactive pages. Application will be able to work properly without it



Performance Optimization: Network Performance

CSS Bundle Optimization

Critical style extraction

Non-optimized

```
<html>
  <head>
    <link rel="stylesheet"
          href="https://cdn.com/desktop.css"/>
  </head>
  <body></body>
</html>
```

Optimized

```
<html>
 <head>
  <style>
    #root {  // Styles }
  </style>
 </head>
 <body></body>
</html>
```

Performance Optimization: Network Performance

CSS Bundle Optimization

Fetching non-critical styles

Using media=print

```
<html>
 <head>
 <style>
    #root {  // Critical Styles }
 </style>
 <link rel="stylesheet"
       href="https://cdn.com/non-critical.css"
       media="print"
       onload="this.media='all'"/>
 </head>
 <body></body>
</html>
```

Using "preload" marker

```
<html>
 <head>
 <style>
   #root {  // Critical Styles }
 </style>
 <link rel="preload"
       as="style"
       href="https://cdn.com/non-critical.css"/
 </head>
<body></body>
</html>
```

Performance Optimization: Network Performance

CSS Bundle Optimization

Summary

Split the bundle when nessacary

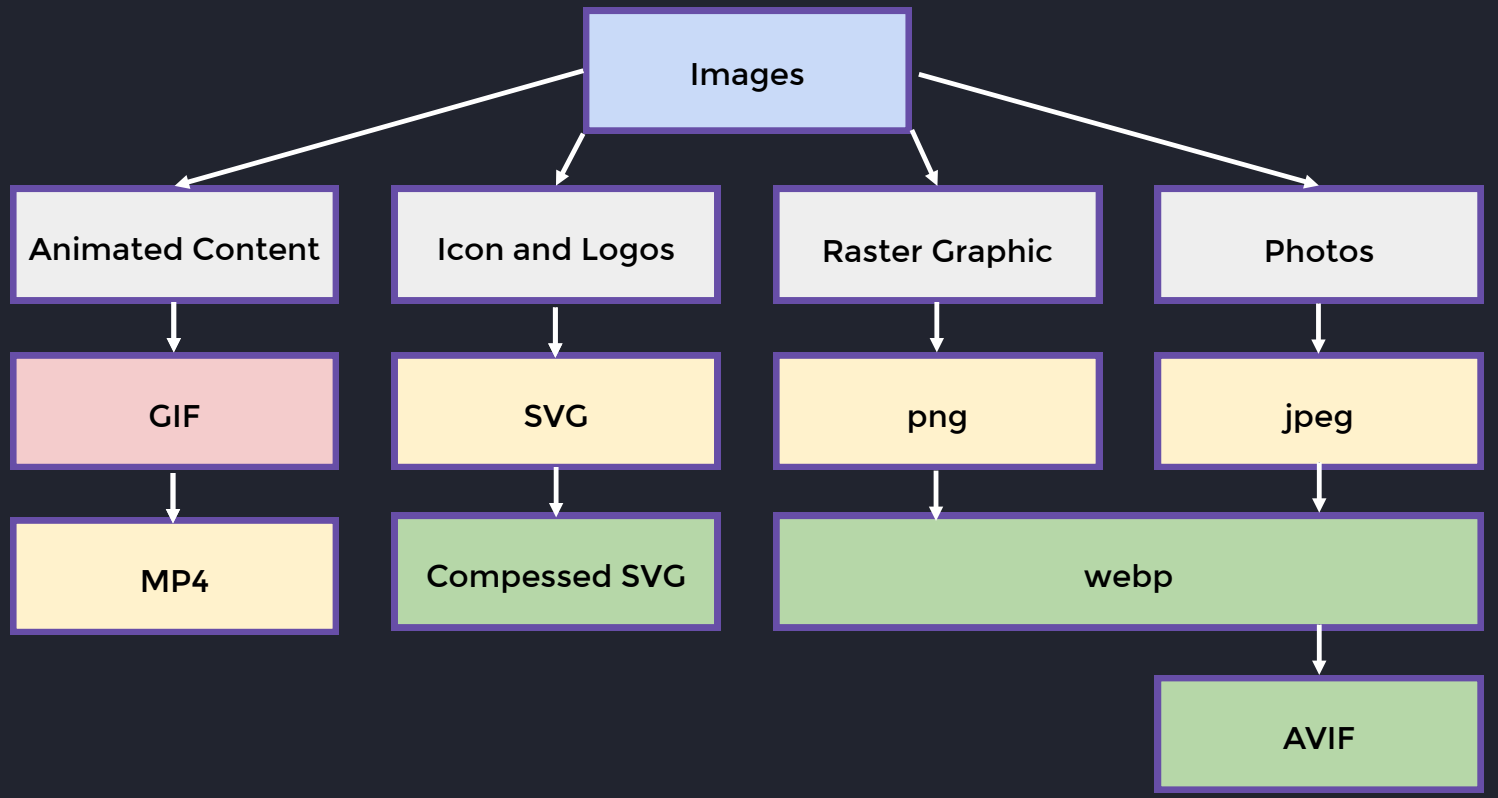
Minify and Compress

Inline critical resource

Non-critical resource can be downloaded on background

Performance Optimization: Network Performance

Image Assets Optimization: Choosing right format



Performance Optimization: Network Performance

Image Assets Optimization: Choosing right format

16KB GIF

5KB webp

Performance Optimization: Network Performance

Image Assets Optimization: Choosing right format

webp — designed to replace png , jpg and gif for usage on the web pages

AVIF — New image encoding format has the same benefits of web with even better compression and picture quaility

Quality	Size	Format	Result
Original	10 MB	jpeg	0
Original	7.5 MB	webp	-25%
Original	6.2 MB	avif	-38%
50%	1.2 MB	jpeg	-88%
50%	1.1 MB	webp	-89%
50%	1.1 MB	avif	-89%

AVIF (Uncompressed) 6.2MB

WEBP (Uncompressed) 7.5MB

Performance Optimization: Network Performance

Image Assets Optimization: Choosing right format

WebP image format - OTHER

Image format (based on the VP8 video format) that supports lossy and lossless compression, as well as animation and alpha transparency. WebP generally has better compression than JPEG, PNG and GIF and is designed to supersede them. AVIF and JPEG XL are designed to supersede WebP.

Current aligned | Usage relative | Date relative | Filtered | All

Usage: Global — % of all users: 96.35% + 0.51% = 96.86%

Chrome	Edge	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS	Samsung Internet	Opera Mini	Opera Mobile	UC Browser for Android	Android Browser	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
4-8				10.1												
1 9-22		3.1-13.1		1 11.5								2.1-3				
2 23-31	12-17	3 14-15.6	2-64	3 12.1-18			3.2-13.7					1 4-4.1				
32-123	18-122	16.0-17.3	65-124	19-108	6-10		14-17.3	4-23		12-12.1		4.2-4.4.4				2.5
124	123	17.4	125	109	11	123	17.4	24	all	80	15.5	123	124	14.9	13.52	3.1
125-127		17.5-TP	126-128				17.5									

AVIF image format - OTHER

Baseline 2024 Newly available across major browsers

A modern image format based on the AV1 video format. AVIF generally has better compression than WebP, JPEG, PNG and GIF and is designed to supersede them. AVIF competes with JPEG XL which has similar compression quality and is generally seen as more feature-rich than AVIF.

Current aligned | Usage relative | Date relative | Filtered | All

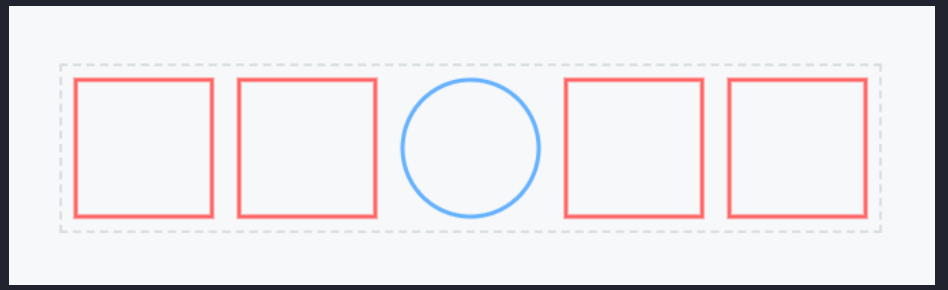
Usage: Global — % of all users: 92.95% + 0.22% = 93.16%

Chrome	Edge	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS	Samsung Internet	Opera Mini	Opera Mobile	UC Browser for Android	Android Browser	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
			2-76													
	12-113		1 77-92													
	3 114-117	3.1-16.0	2 93-110				3.2-15.8									
4-84	118-120	2 3 4 16.1-16.3	5 111-112	10-70			2 3 4 16.0-16.3	4-13.0								
85-123	121-122	16.4-17.3	113-124	71-108	6-10		2 16.4-17.3	14.0-23		12-12.1		2.1-4.4.4				2.5
124	123	17.4	125	109	11	123	2 17.4	24	all	80	15.5	123	124	14.9	13.52	3.1
125-127		17.5-TP	126-128				2 17.5									

Performance Optimization: Network Performance

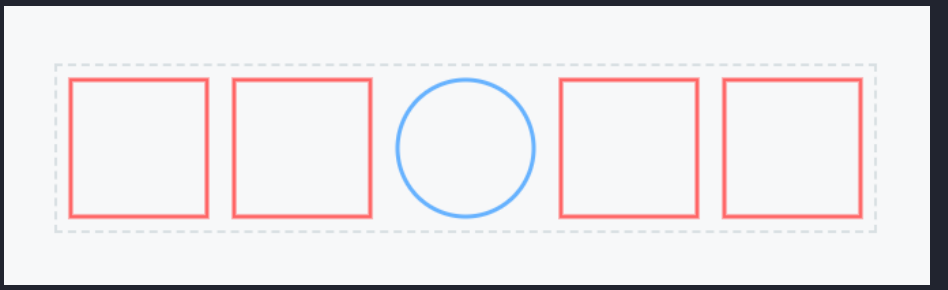
Image Assets Optimization: SVG Path compression

Uncompressed - 0.7KB



```svg
<!-- 480 characters -->
<svg width="600" height="120" fill="none">
    <rect x="10" y="10" width="100"
          height="100" stroke-width="3"
          stroke="#ff6666">
    </rect>
    <rect x="130" y="10"
          width="100" height="100"
          stroke-width="3" stroke="#ff6666">
    </rect>
    <ellipse cx="300" cy="60"
             rx="50" ry="50"
             stroke-width="3" stroke="#66b2ff">
    </ellipse>
    <rect x="370" y="10"
          width="100" height="100"
          stroke-width="3" stroke="#ff6666">
    </rect>
    <rect x="490" y="10"
          width="100" height="100"
          stroke-width="3" stroke="#ff6666">
    </rect>
</svg>
```

Compressed- 0.2KB



```svg
<!-- 263 characters -->
<svg width="600" height="120" fill="none" stroke-width="3">
    <path d="M10 10h100v100H10zm120 0h100v100H130z"
          stroke="#f66">
    </path>
    <circle cx="300" cy="60"
            r="50" stroke="#66b2ff"></circle>
    <path d="M370 10h100v100H370zm120 0h100v100H490z"
          stroke="#f66">
    </path>
</svg>
```

Performance Optimization: Network Performance

Image Assets Optimization

Summary

1. Compress images for web
2. Use optimized formats
3. Use SVG Path compression

Performance Optimization: Network Performance

Font Loading Optimization

Allow browser to display content when custom font is downloading

```
@font-face {
  src: "https://my-custom-font.com/font/";
  font-display: 'auto';
}
```

Browser waits for 3 seconds to load

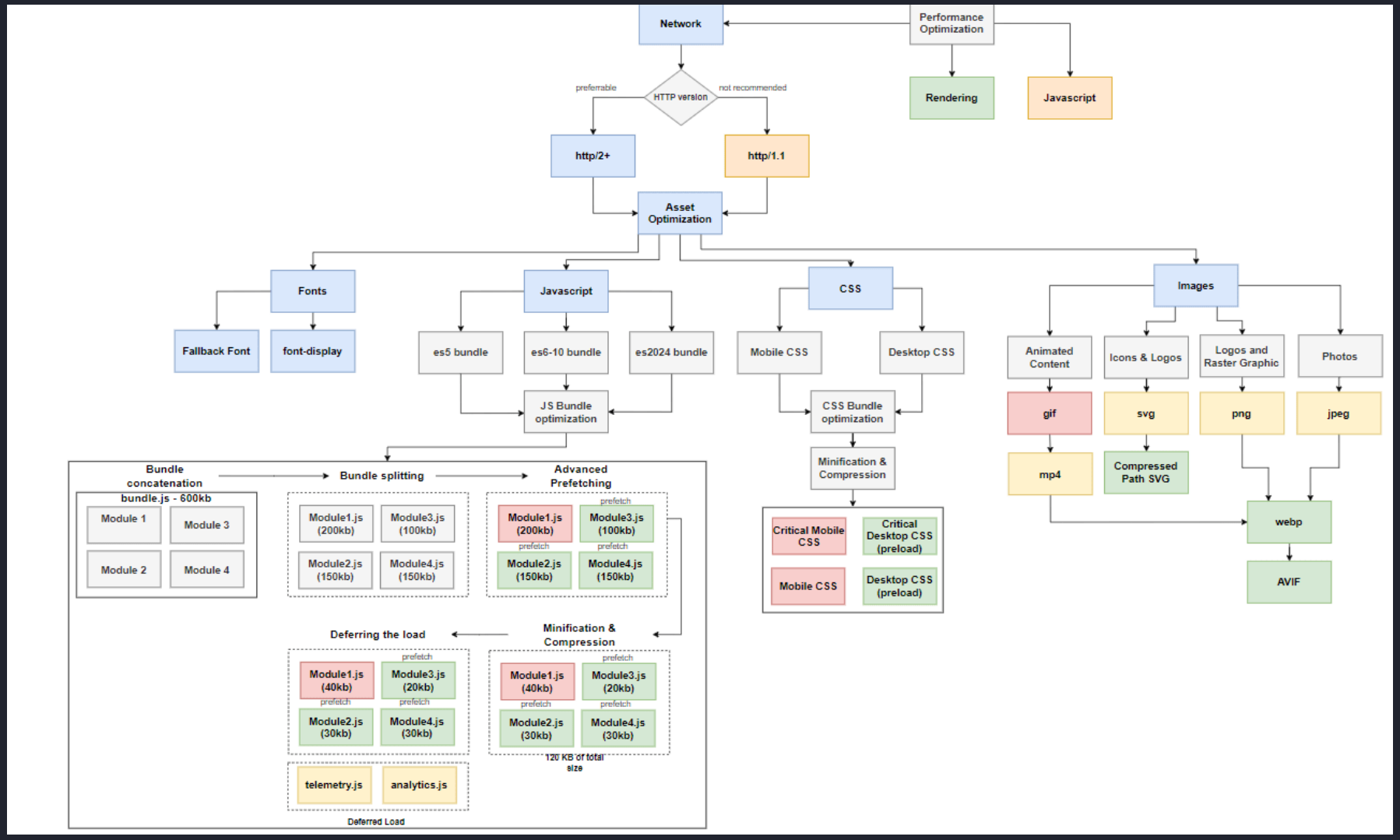
```
@font-face {
  src: "https://my-custom-font.com/font/";
  font-display: 'fallback';
}
```

Render unstyled text immediately, switch if font is loaded within 3 seconds

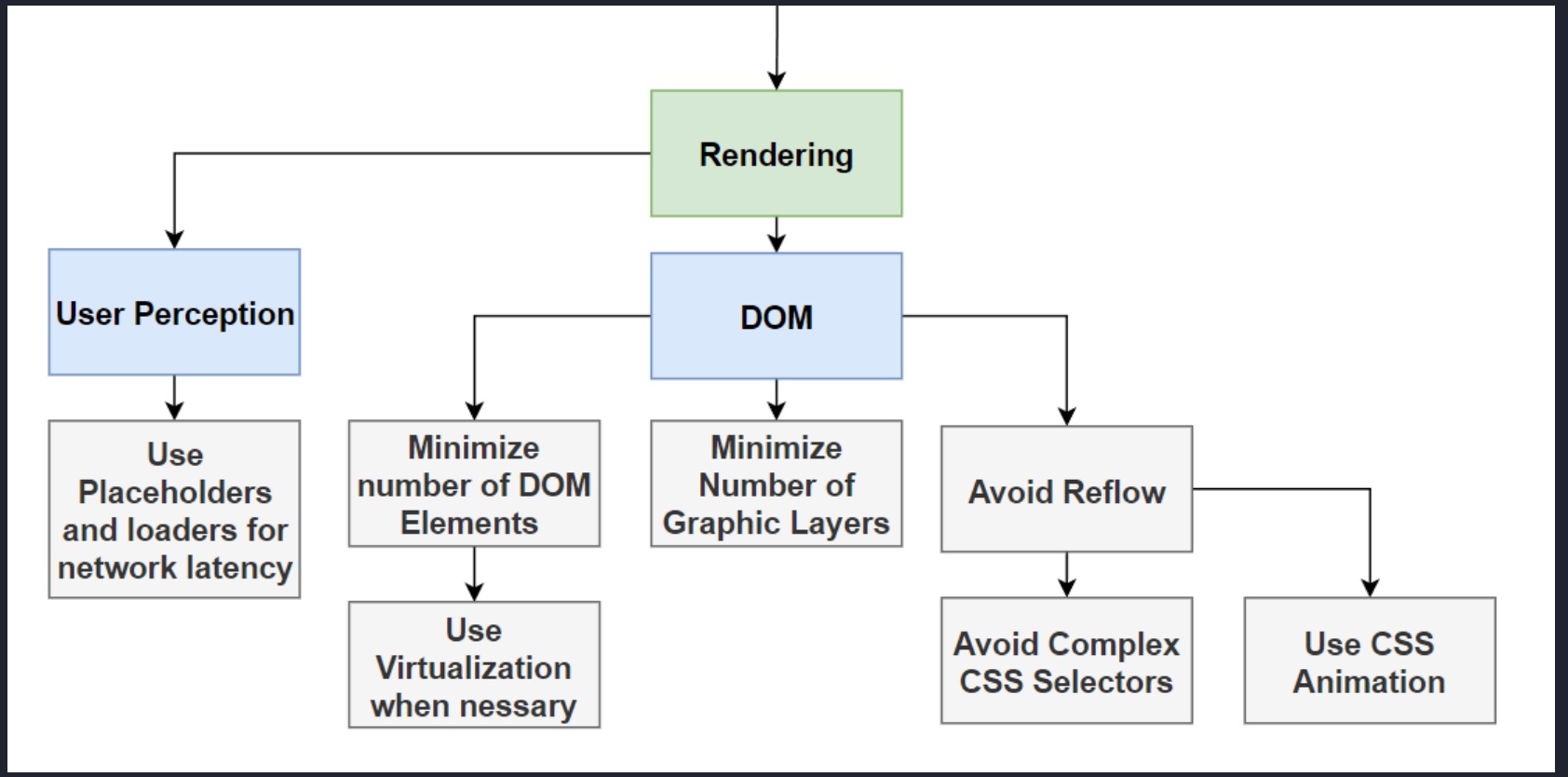
```
@font-face {
  src: "https://my-custom-font.com/font/";
  font-display: 'optional';
}
```

Render unstyled text immediately, switch only on refresh if it's downloaded

Network Performance - Summary

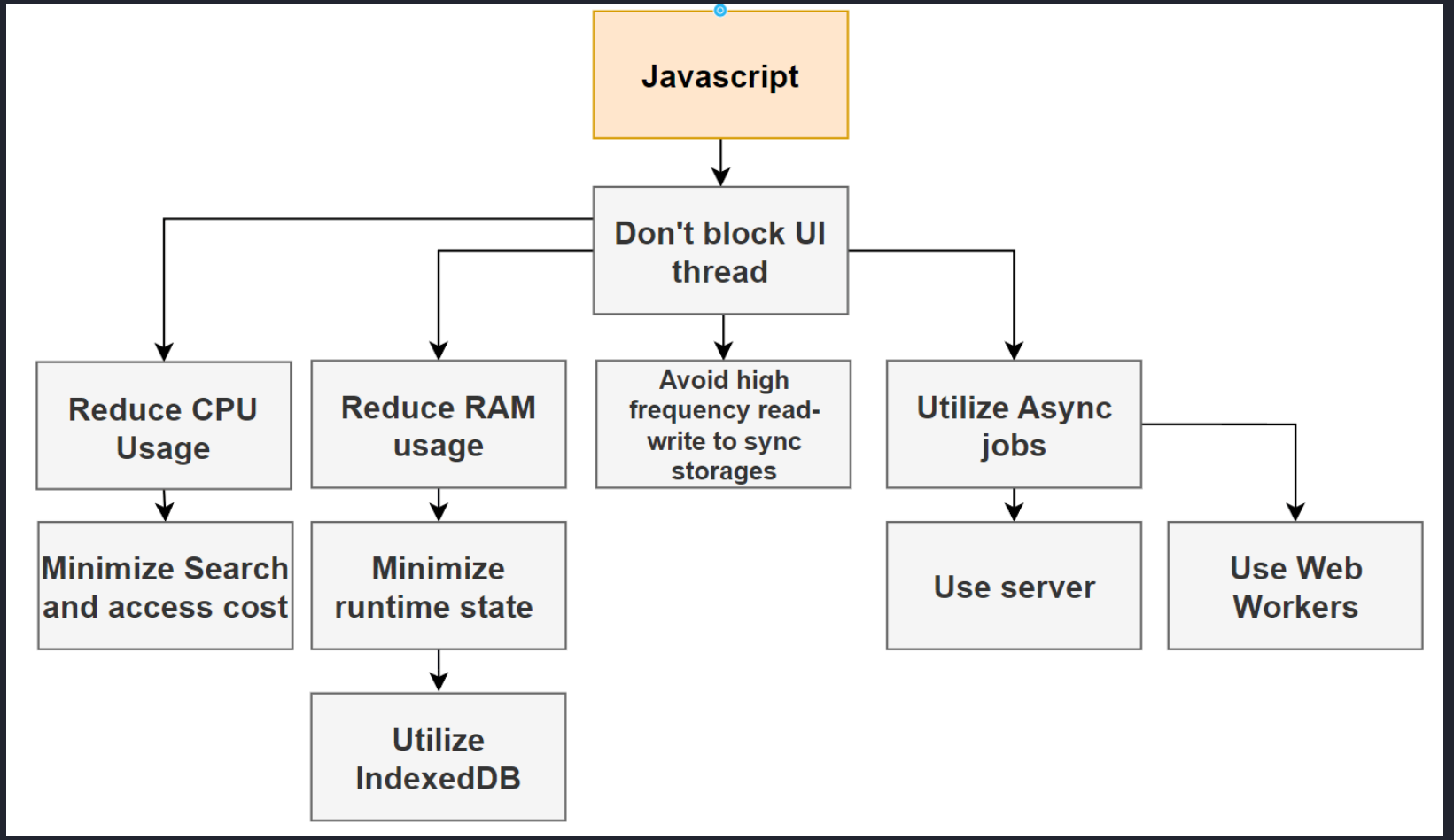


Rendering Performance - Summary



Javascript Performance - Summary

The main rule of thump in building UI apps - `Don't block a UI thread.`



Workshop Summary

Bonus Section - System Design Time!

Let's build **News Feed!**