




# React Reconciler

Как написать собственный рендерер

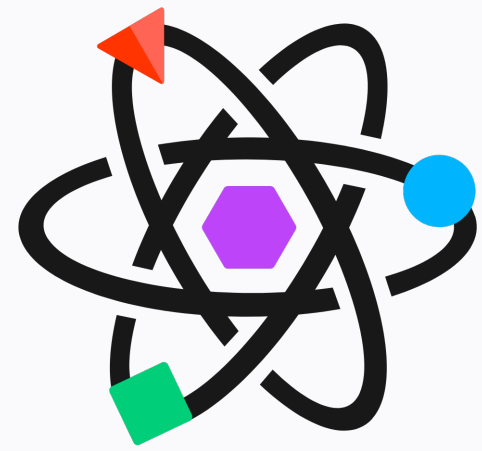


# Ярослав Лосев

Redis, креативное агентство

-  [LosYear](#)
-  [github.com/losyear](https://github.com/losyear)
-  [twitter.com/losyear](https://twitter.com/losyear)

# React Figma



[react-figma.now.sh](https://react-figma.now.sh)

[github.com/react-figma](https://github.com/react-figma)



@ilialesik



@losyear

```
figma-react [~/Workspace/figma-react] - .../examples/design-system/src/App.tsx
App.tsx
144     />
145   </Frame>
146   );
147 };
148
149 export const App = () => {
150   return (
151     <Page name="" style={{ flexDirection: 'row' }}>
152       <SpaceFrame />
153       <TypeFrame style={{ marginLeft: 50 }} />
154       <IconsFrame style={{ marginLeft: 50 }} />
155     </Page>
156   );
157 };
158
```

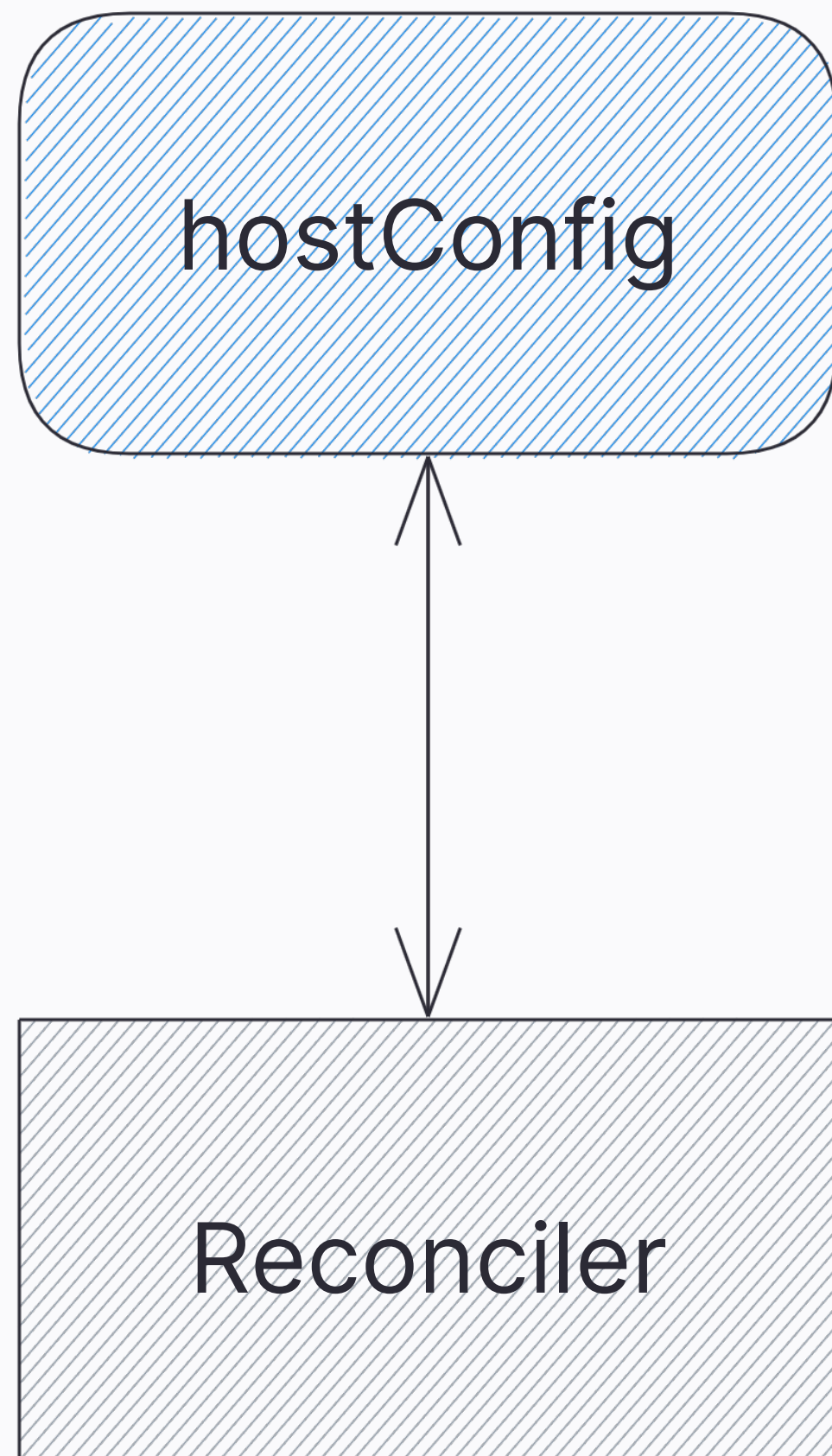
# О чем поговорим?

- React Reconciler
- React → DOM
- Figma & Figma API
- React → Figma
- Итоги

# О ЧЕМ ПОГОВОРИМ?

- **React Reconciler**
- React → DOM
- Figma & Figma API
- React → Figma
- Итоги

```
import React from 'react';  
import ReactDOM from 'react-dom';
```



Набор функций, специфичных для среды отрисовки.

Например, как создать или изменить узел.

Модуль react-reconciler с npm.

Является общим для всех отрисовщиков

# Reconciler

- UI agnostic
- Находит изменения дерева элементов
- Передает найденные изменения отрисовщику



# Renderer

- UI dependent
- Оперирует хост-компонентами
- Атомарно применяет изменения в конкретной среде

# О ЧЕМ ПОГОВОРИМ?

- React Reconciler
- **React → DOM**
- Figma & Figma API
- React → Figma
- Итоги

```
import ReactDOM from 'react-dom';
```

```
    . . .
```

```
ReactDOM.render(<App/>, root);
```

```
import Reconciler from 'react-reconciler';

const hostConfig = {};

const render = (jsx, root) => {
  const reconciler = Reconciler(hostConfig);
  const container = reconciler.createContainer(root, false, false);

  reconciler.updateContainerjsx, container);
};
```

```
import Reconciler from 'react-reconciler';

const hostConfig = {};

const render = (jsx, root) => {
  const reconciler = Reconciler(hostConfig);
  const container = reconciler.createContainer(root, false, false);

  reconciler.updateContainerjsx, container);
};
```

```
import Reconciler from 'react-reconciler';

const hostConfig = {};

const render = (jsx, root) => {
  const reconciler = Reconciler(hostConfig);
  const container = reconciler.createContainer(root, false, false);

  reconciler.updateContainer(jsx, container);
};
```

```
import Reconciler from 'react-reconciler';

const hostConfig = {};

const render = (jsx, root) => {
  const reconciler = Reconciler(hostConfig);
  const container = reconciler.createContainer(root, false, false);

  reconciler.updateContainer(jsx, container);
};
```

```
import Reconciler from 'react-reconciler';

const hostConfig = {};

const render = (jsx, root) => {
  const reconciler = Reconciler(hostConfig);
  const container = reconciler.createContainer(root, false, false);

  reconciler.updateContainer(jsx, container);
};
```



# Режимы работы

# Режимы работы

**supportsMutation: true**

Мутирует существующие ноды:

- appendChild, insertBefore, removeChild
- commitUpdate

Например: React DOM, React Figma, ...

# Режимы работы

**supportsPersistence: true**

Клонировует и заменяет ноды:

- `replaceContainerChildren`, `createContainerChildSet`
- `cloneInstance`

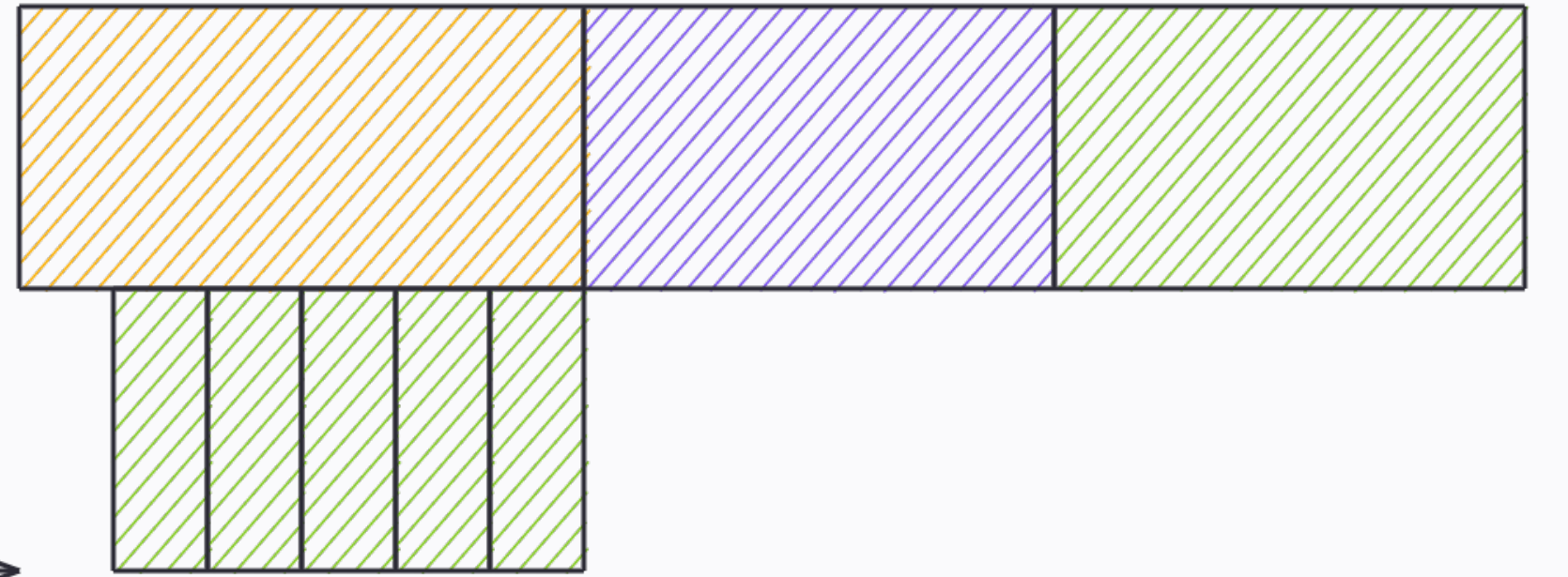
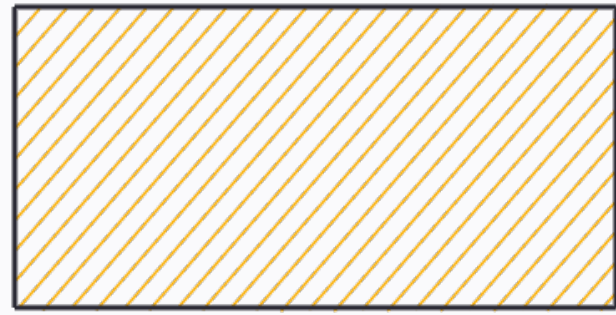
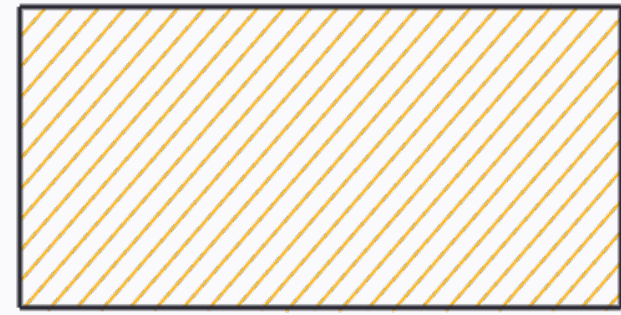
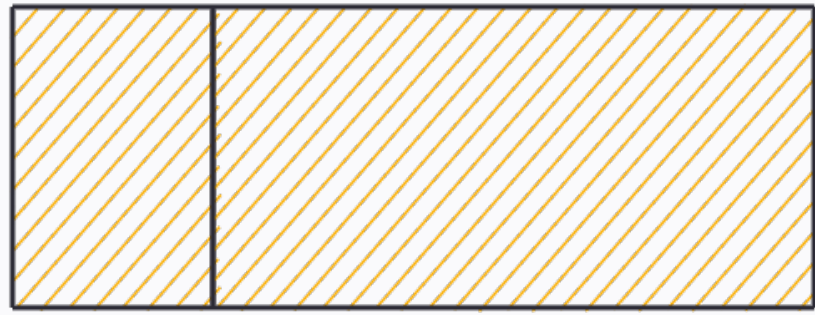
Например: React Native, canvas, console, ...

setState

Diff & Patch

Layout

Paint





# Live code

- Отрисовываем React в DOM
- Компоненты
  - `<div>`, `<p>`, `<img>`
- Атрибуты
  - `className`, `onClick`, `src`

**Первоначальная отрисовка**

```
const InitialRenderExample = () => (  
  <div className="initial-render-example">  
      
    <p>Hello!</p>  
  </div>  
)
```



Hello!

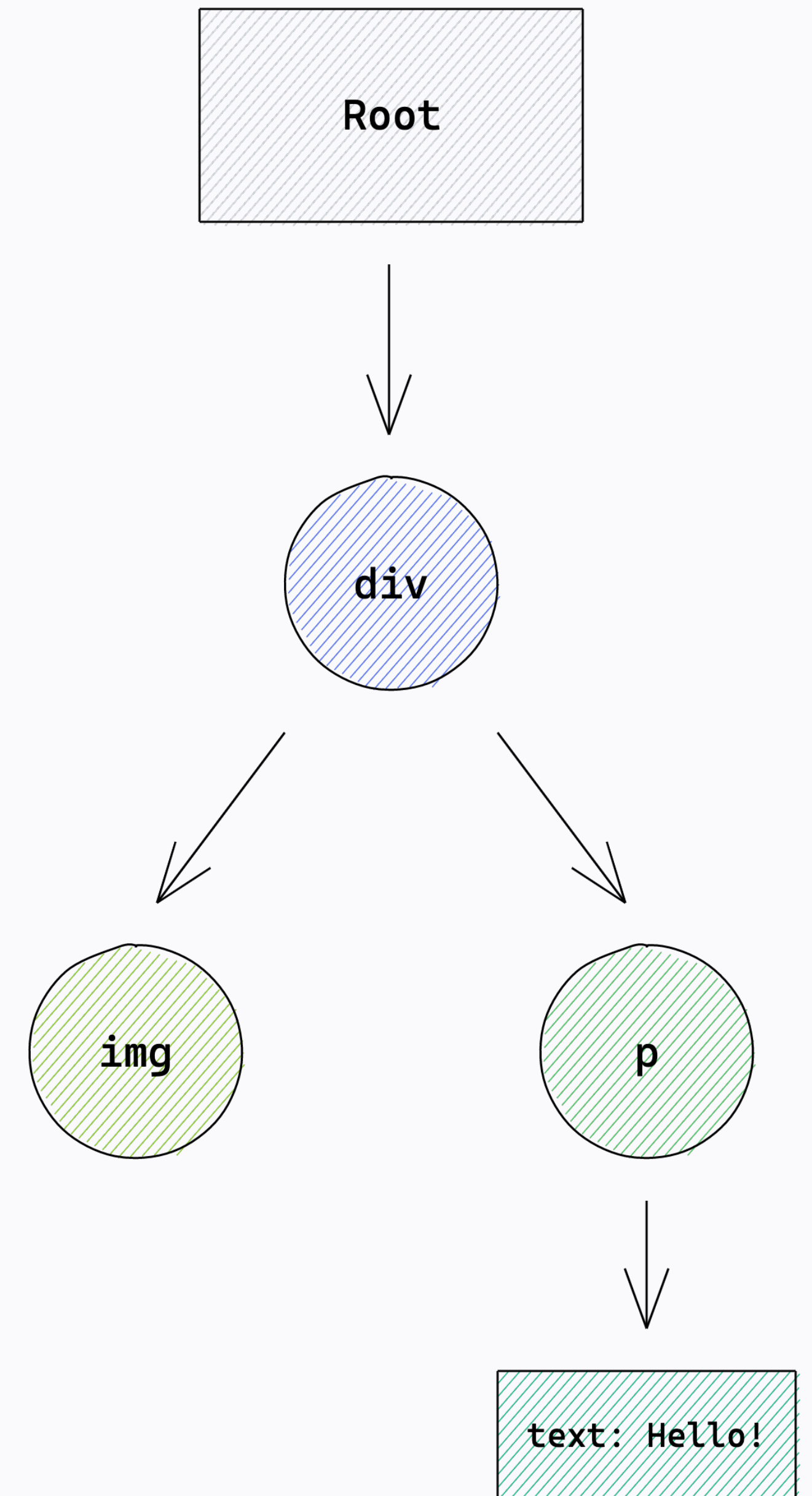
```
const InitialRenderExample = () => (  
  <div className="initial-render-example">  
      
    <p>Hello!</p>  
  </div>  
)
```



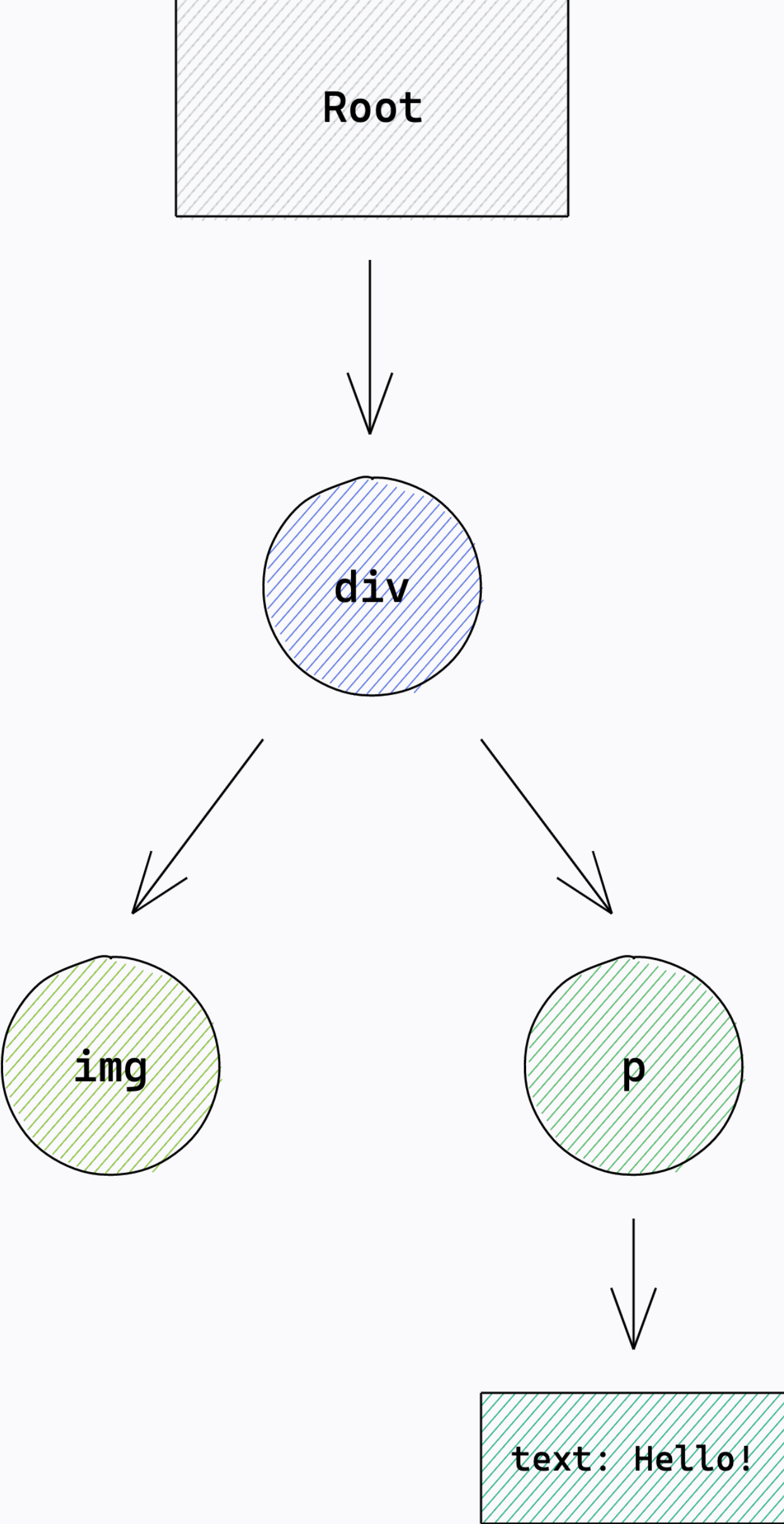
Hello!



```
const InitialRenderExample = () => (  
  <div className="initial-render-example">  
      
    <p>Hello!</p>  
  </div>  
)
```



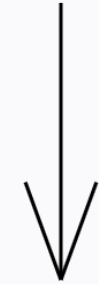
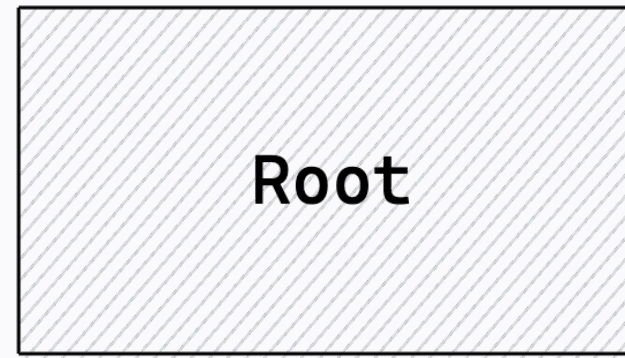
1. getRootHostContext



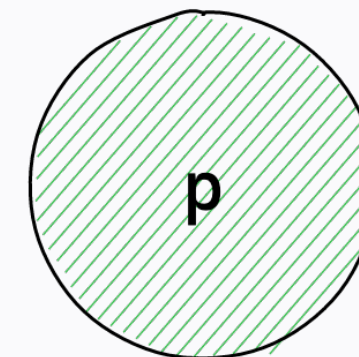
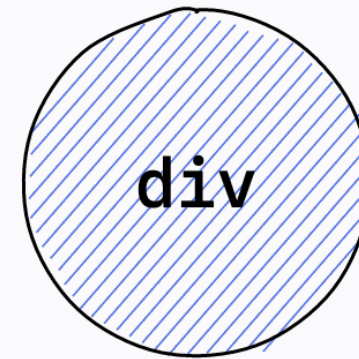
```
getRootHostContext(rootContainerInstance: Container): HostContext
```

Передает первоначальный контекст между вызовами методов

1. `getRootHostContext`

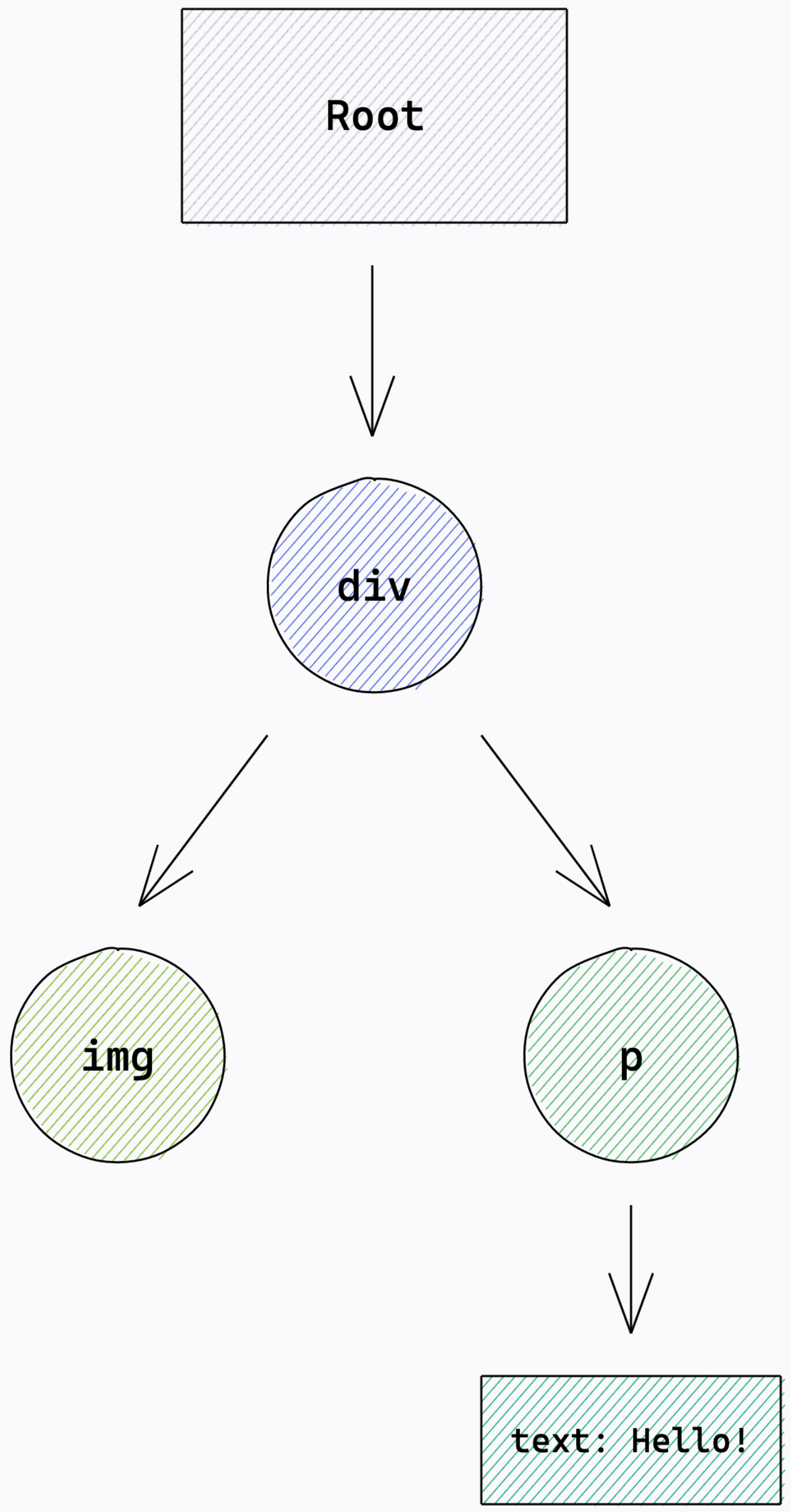


2. `getChildHostContext`



```
getChildHostContext(parentContext: HostContext,  
                    type: string, ...): HostContext
```

Передаёт контекст от родителя к детям



1. `getRootHostContext`

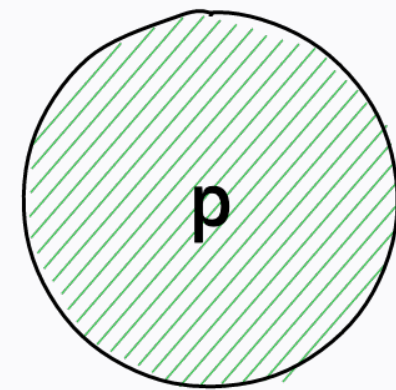
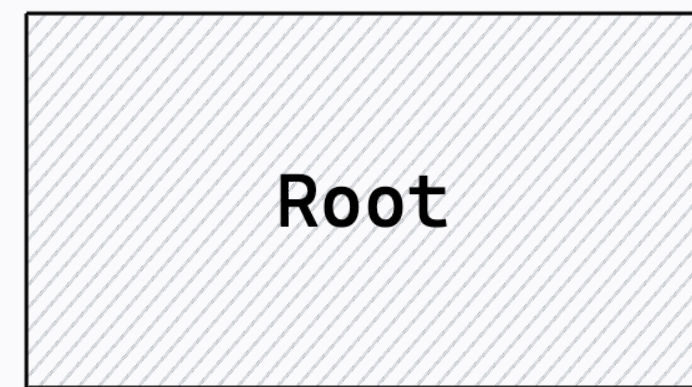
2. `getChildHostContext`

**3. `shouldSetTextContent`**

```
shouldSetTextContent(type: string, props: Props): boolean
```

Нужно ли обрабатывать текст как отдельные ноды

# shouldSetTextContent



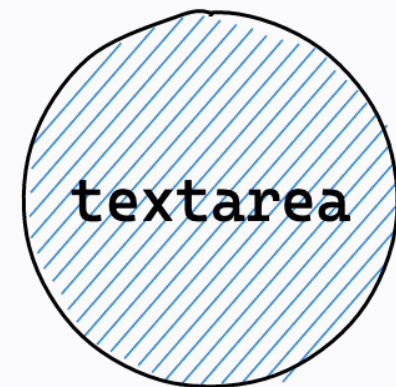
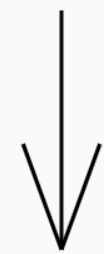
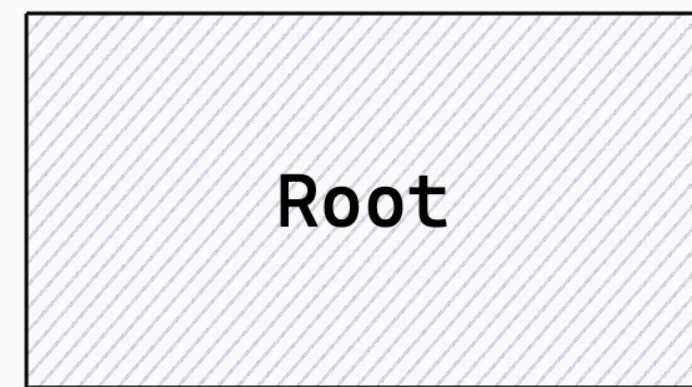
setTextContent: false



```
const paragraph = document.createElement('p');  
const helloText = document.createTextNode('Hello!');  
  
paragraph.appendChild(helloText);
```



# shouldSetTextContent

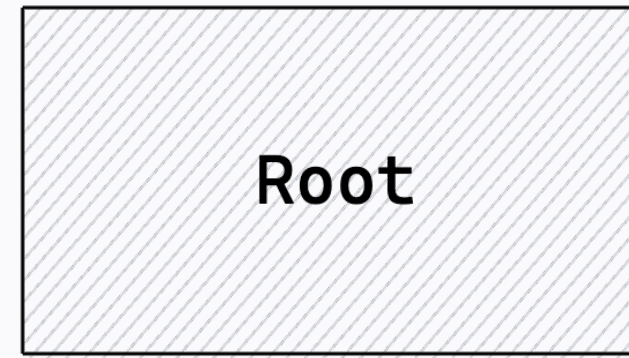


setTextContent: true



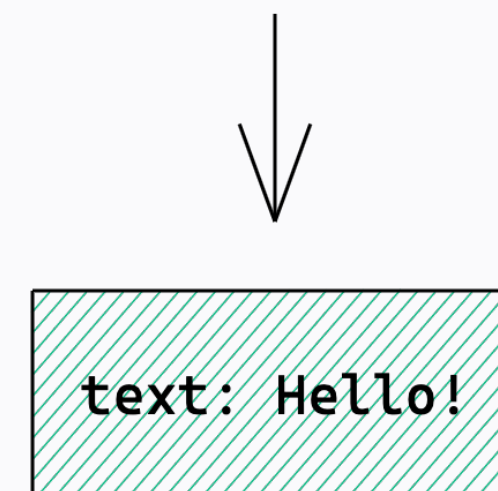
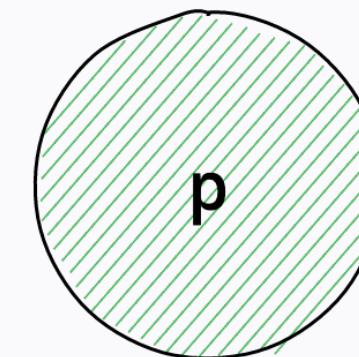
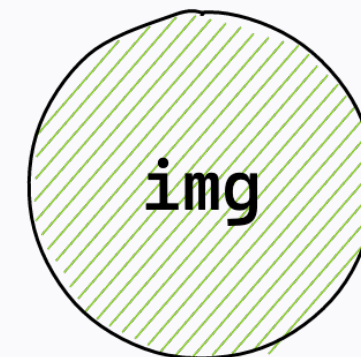
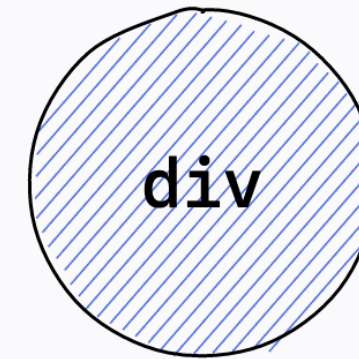
```
const textarea = document.createElement('textarea');  
textarea.value = "Hello!";
```

1. `getRootHostContext`



2. `getChildHostContext`

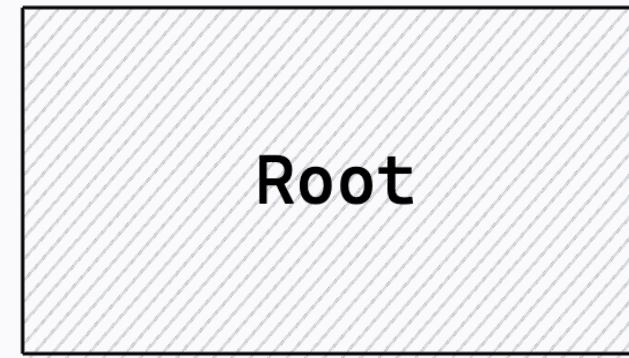
3. `shouldSetTextContent`



**4. `getChildHostContext`**

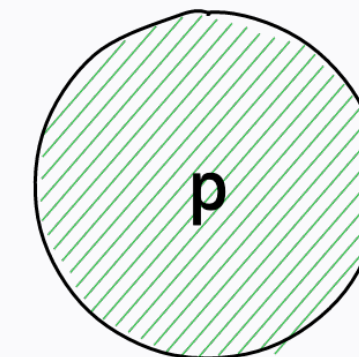
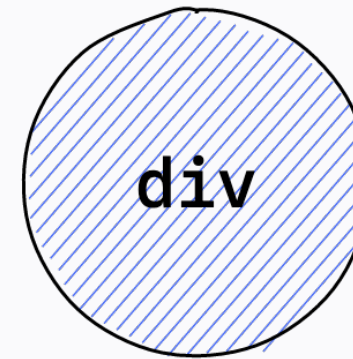
**5. `shouldSetTextContent`**

1. `getRootHostContext`



2. `getChildHostContext`

3. `shouldSetTextContent`



4. `getChildHostContext`

5. `shouldSetTextContent`

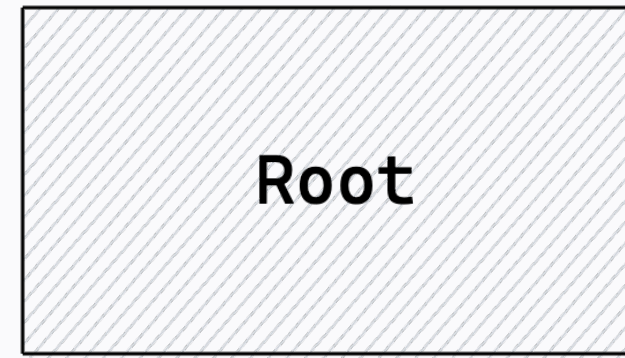
**6. `createElement`**



```
createInstance(type: string, props: Props, ...): Instance
```

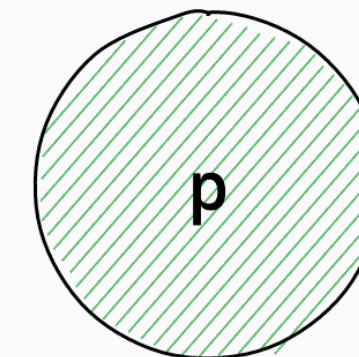
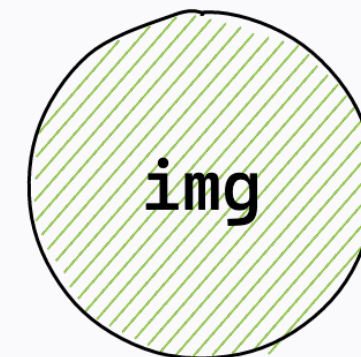
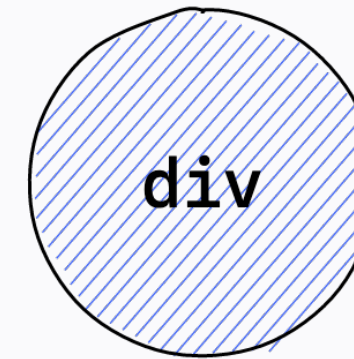
Создает представление хост-компонента в среде

1. `getRootHostContext`



2. `getChildHostContext`

3. `shouldSetTextContent`

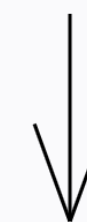


4. `getChildHostContext`

5. `shouldSetTextContent`

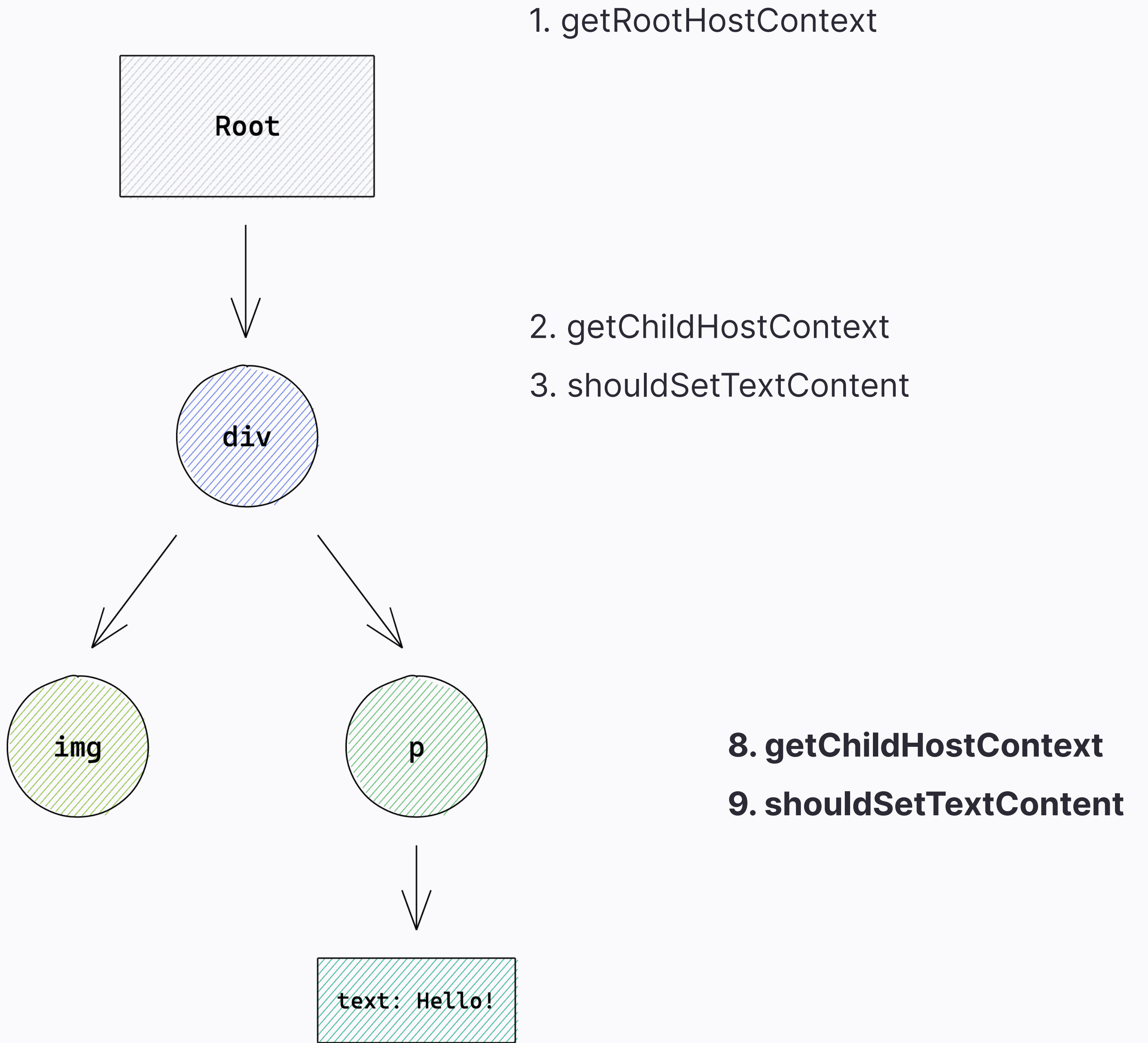
6. `createElement`

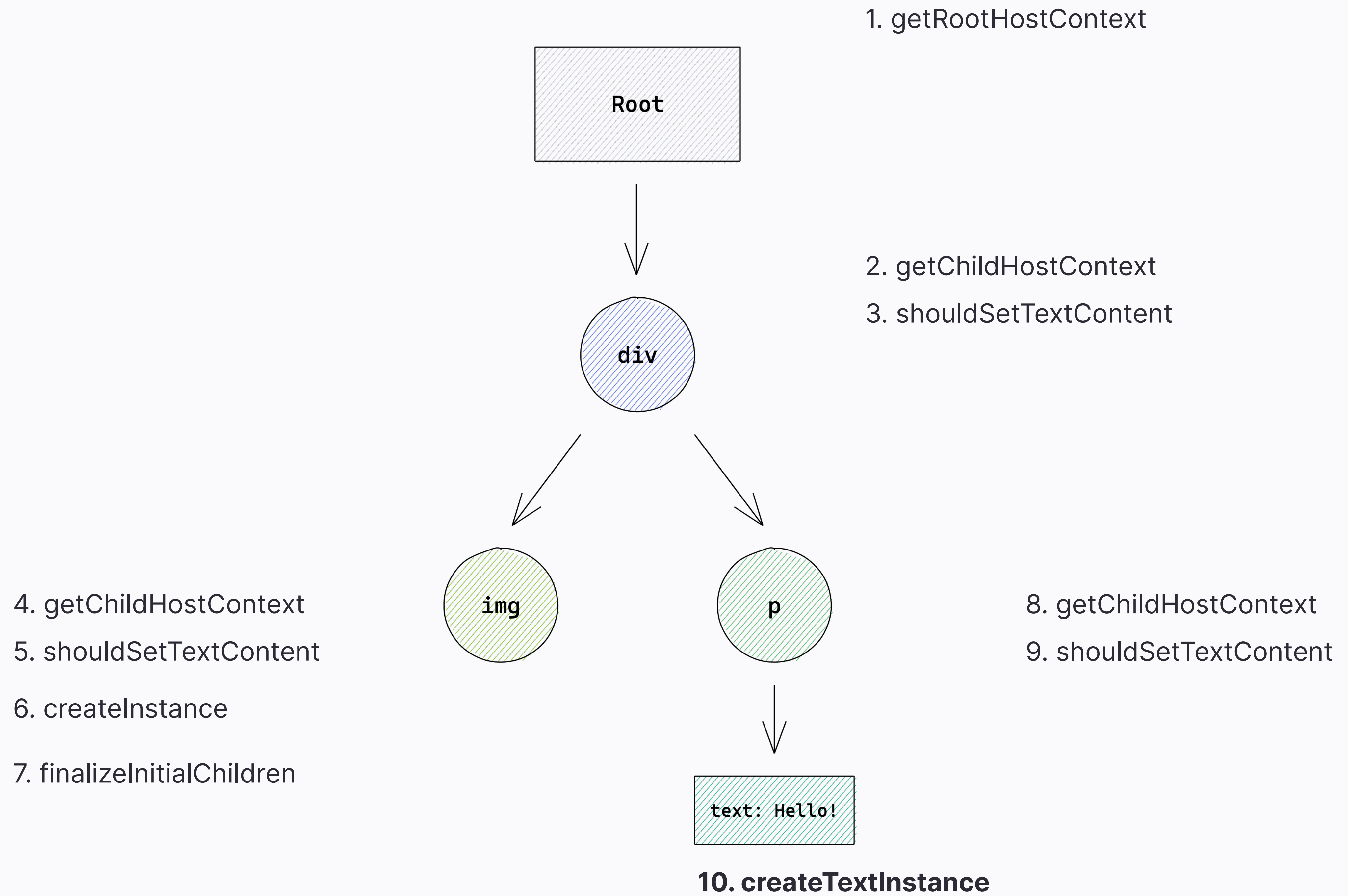
**7. `finalizeInitialChildren`**



```
finalizeInitialChildren(parent: Instance,  
  type: string, props: Props): boolean
```

Совершает дополнительные действия после создания узла

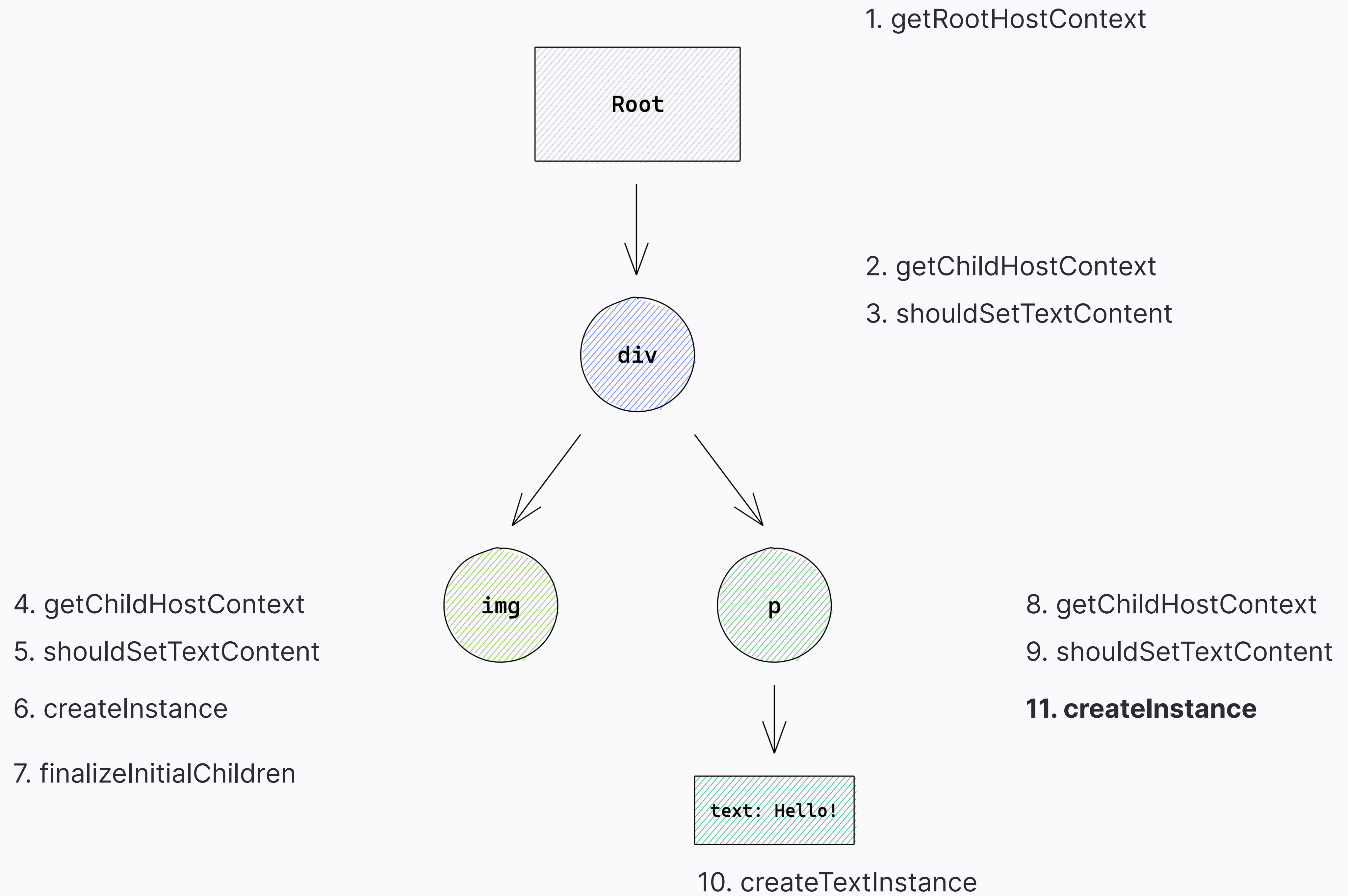


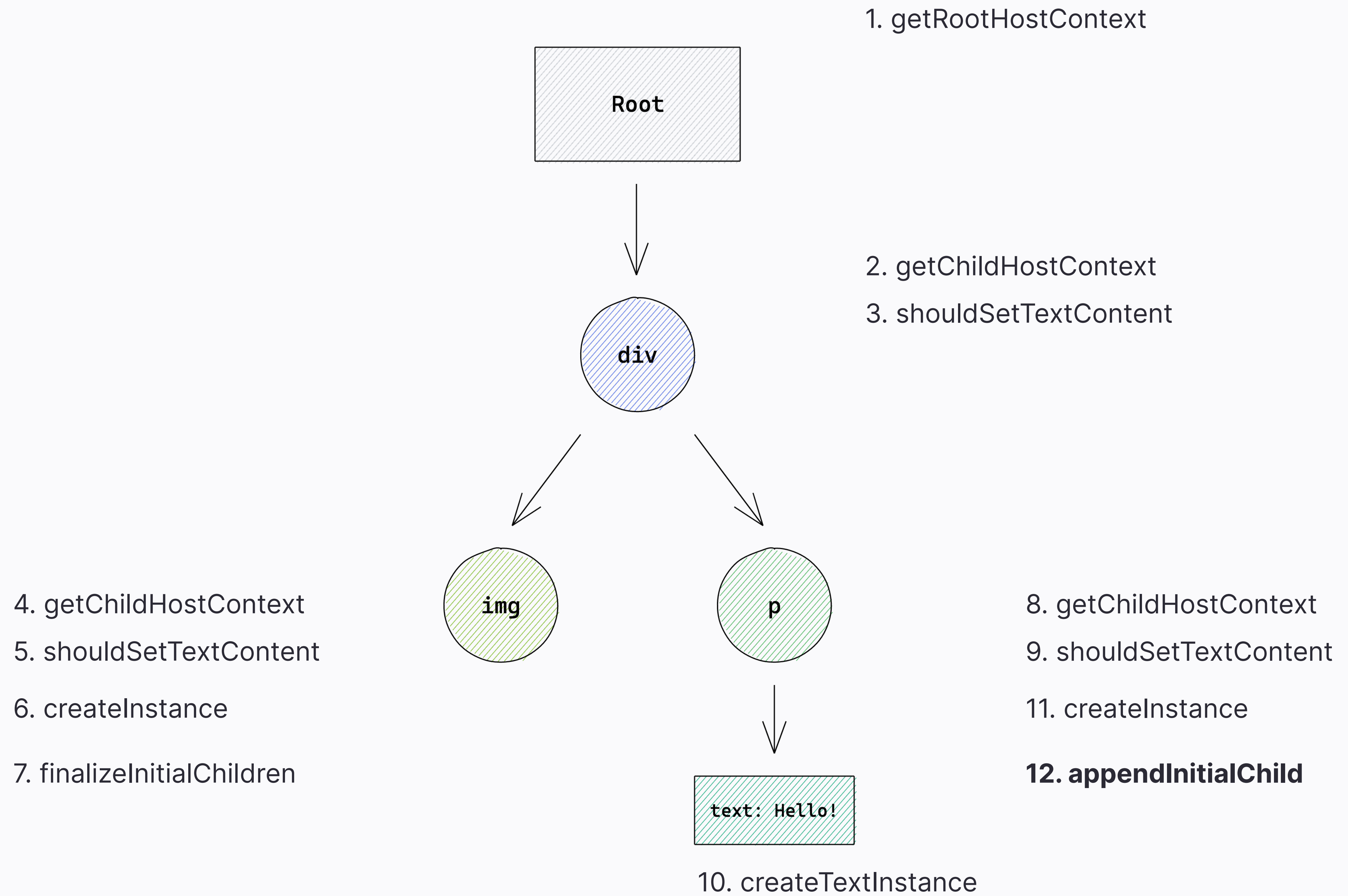




```
createTextInstance(text: string, ...): TextInstance
```

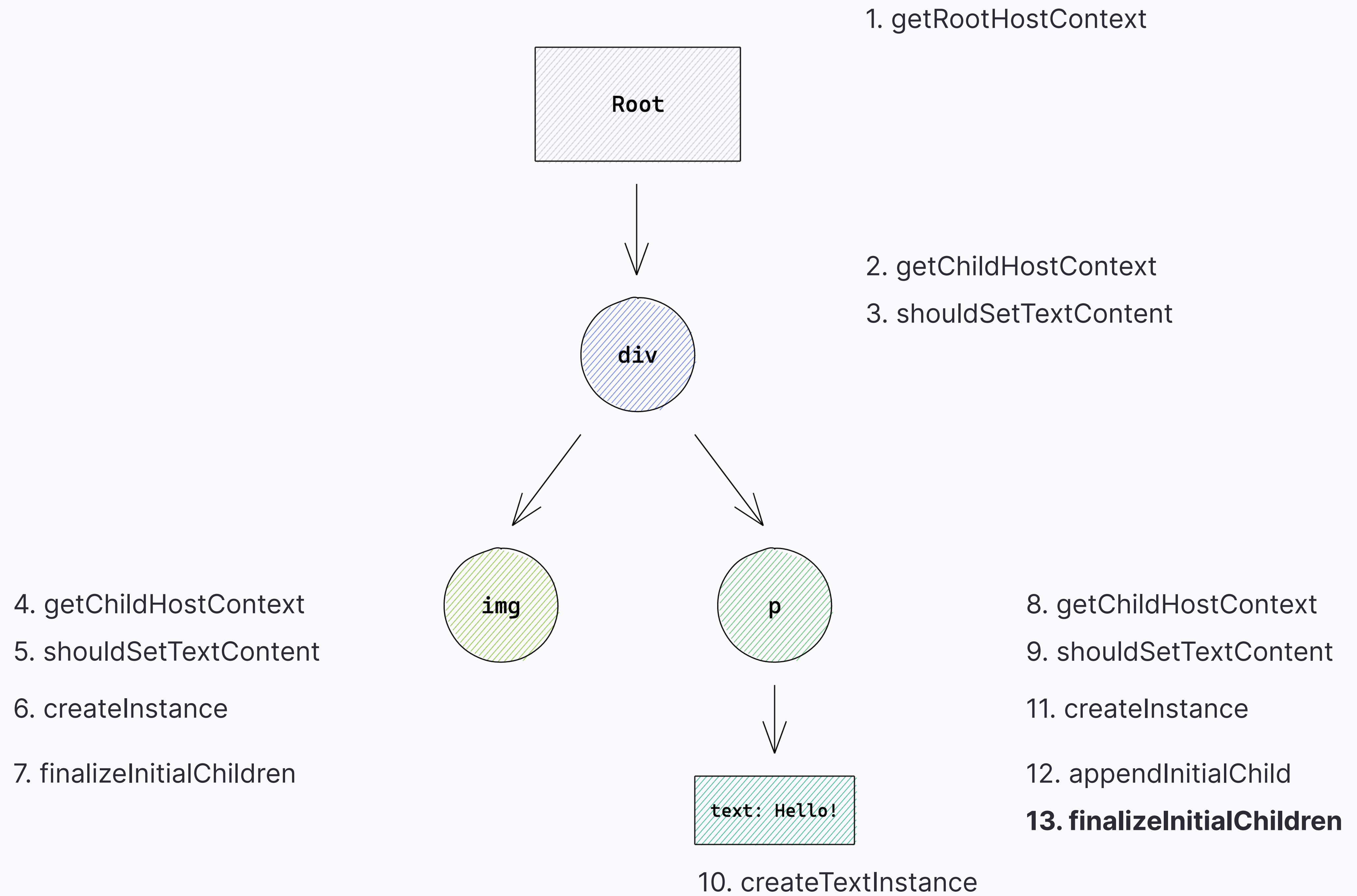
Создает представление текстового листа

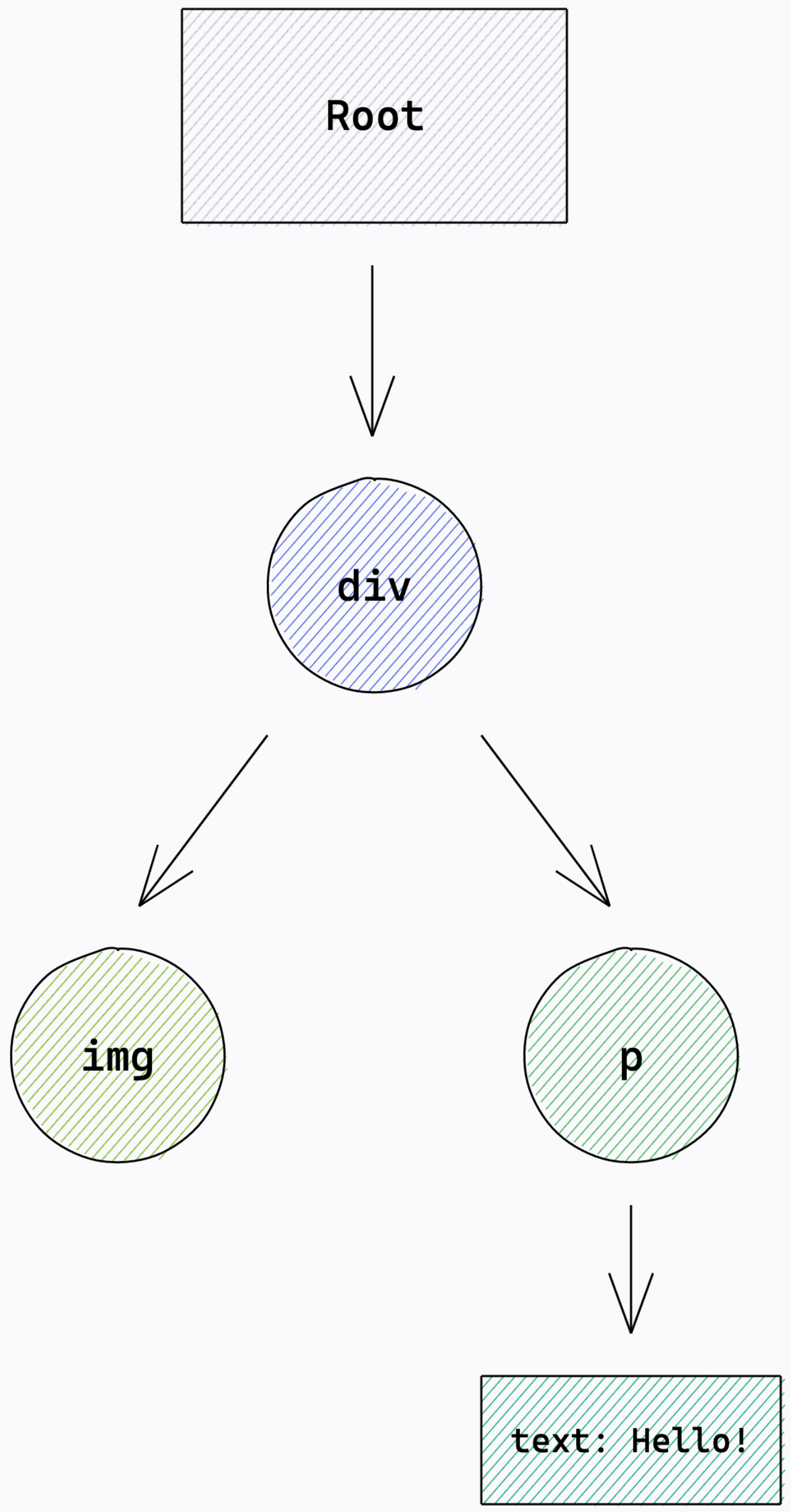




```
appendInitialChild(  
    parent: Instance,  
    child: Instance | TextInstance  
): void
```

Добавляет вершину родителю при первоначальной отрисовке





1. `getRootHostContext`

2. `getChildHostContext`

3. `shouldSetTextContent`

**14. `createElement`**

**15, 16. `appendChild`**

**17. `finalizeInitialChildren`**

4. `getChildHostContext`

5. `shouldSetTextContent`

6. `createElement`

7. `finalizeInitialChildren`

8. `getChildHostContext`

9. `shouldSetTextContent`

11. `createElement`

12. `appendChild`

13. `finalizeInitialChildren`

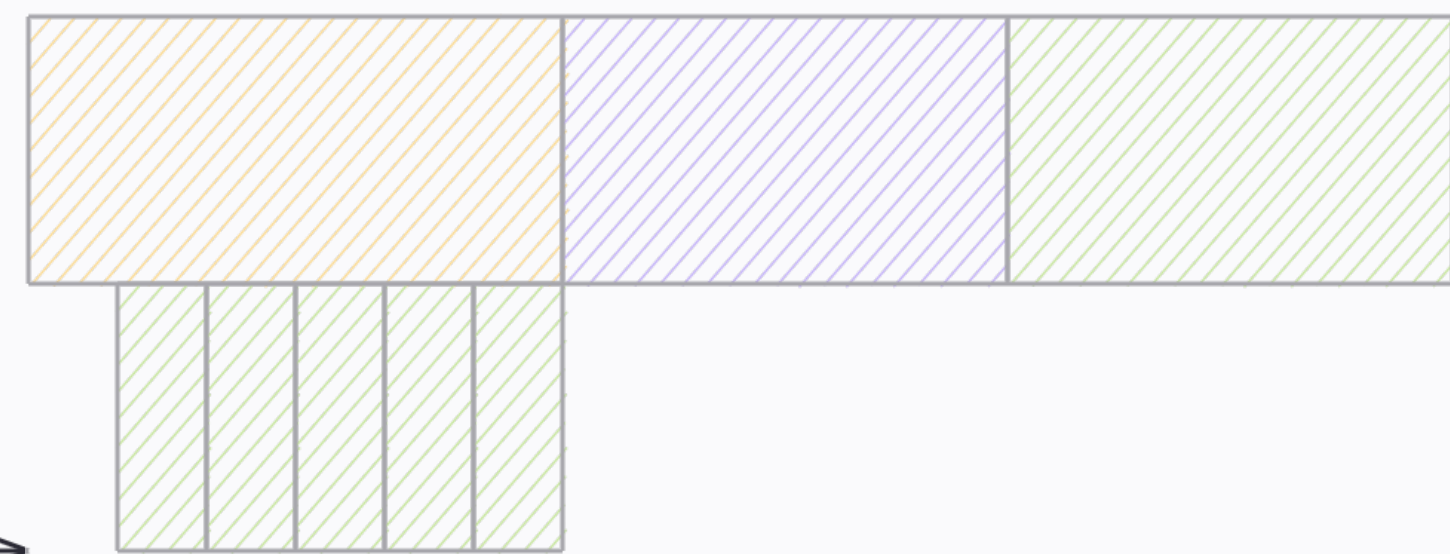
10. `createTextInstance`

setState

Diff & Patch

Layout

Paint



Render phase



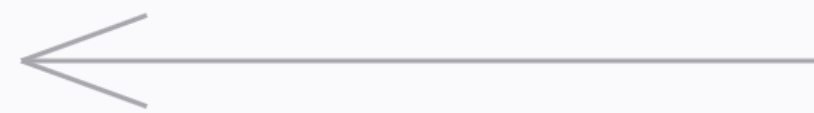
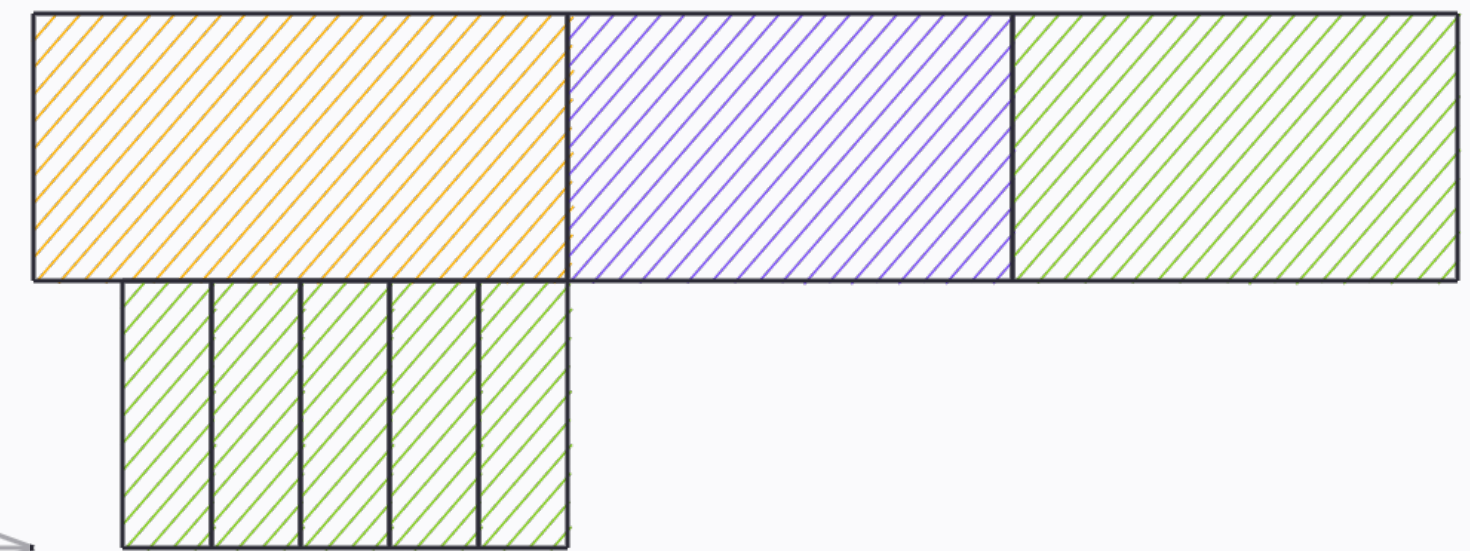
Commit phase

setState

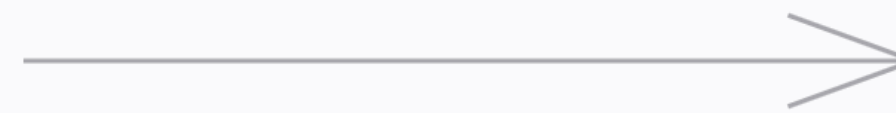
Diff & Patch

Layout

Paint



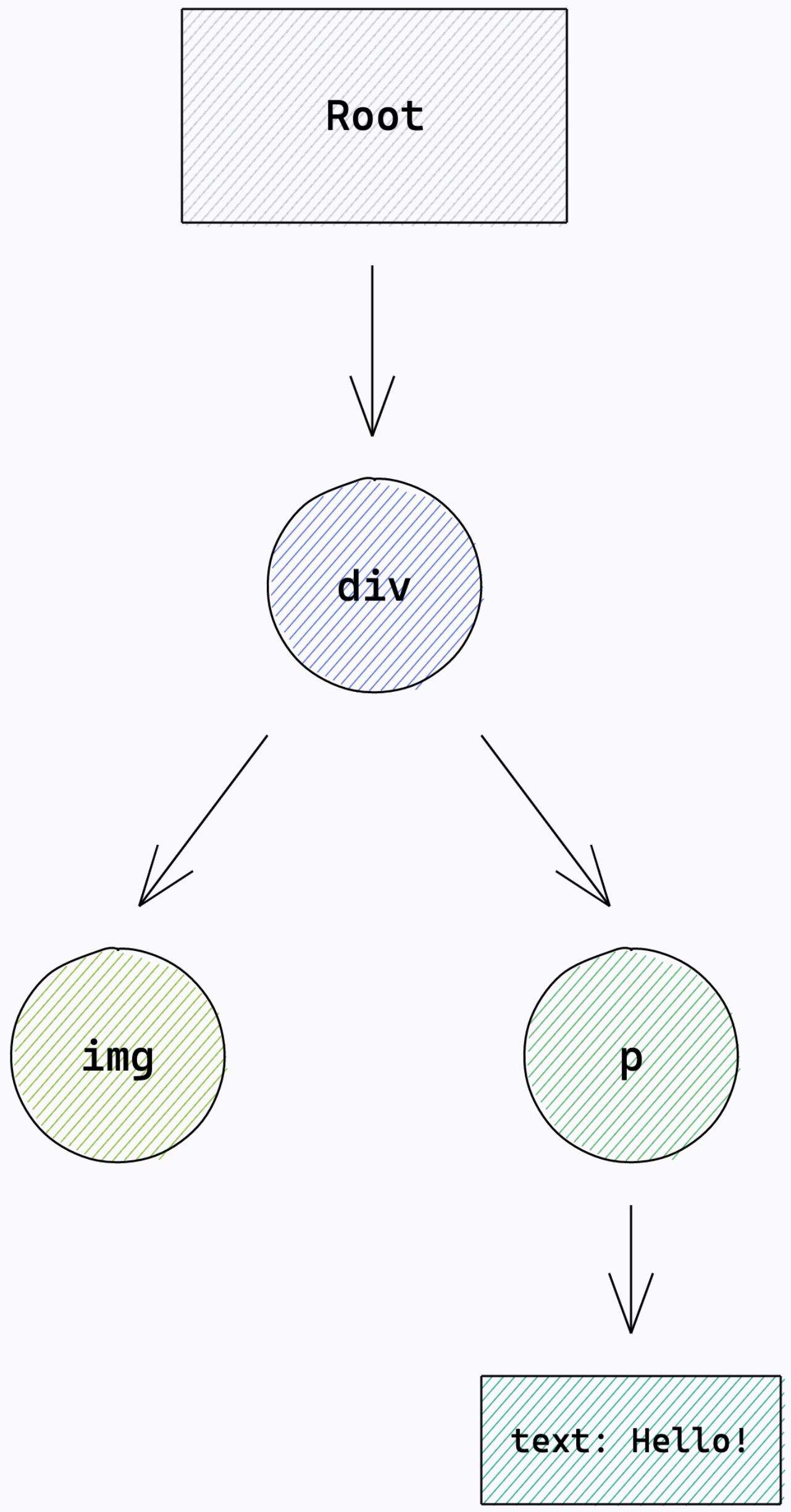
Render phase



Commit phase



- 4. getChildHostContext
- 5. shouldSetTextContent
- 6. createInstance
- 7. finalizeInitialChildren



- 10. createTextInstance

- 1. getRootHostContext

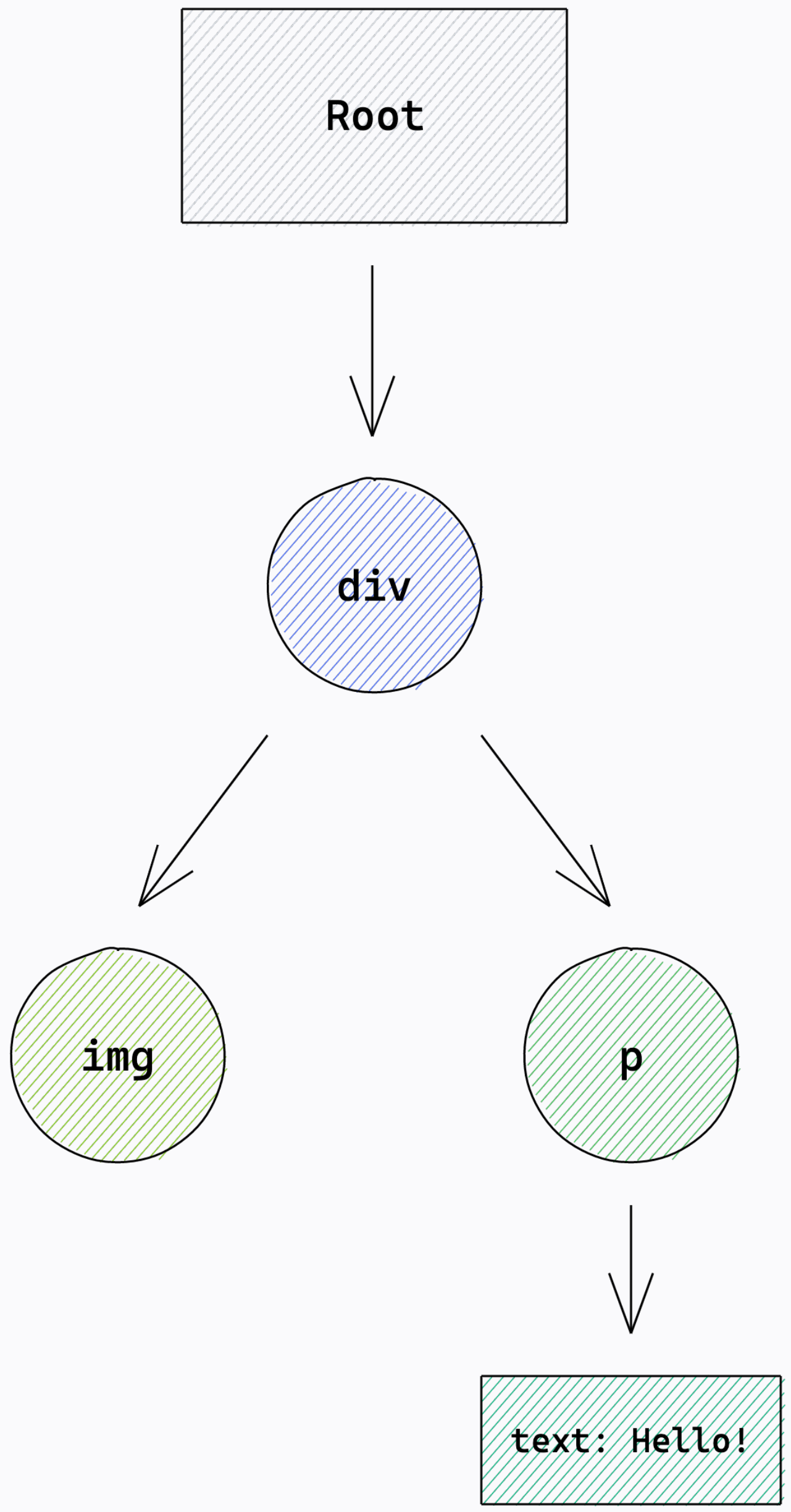
**18. prepareForCommit**

- 2. getChildHostContext
- 3. shouldSetTextContent
- 14. createInstance
- 15, 16. appendInitialChild
- 17. finalizeInitialChildren

- 8. getChildHostContext
- 9. shouldSetTextContent
- 11. createInstance
- 12. appendInitialChild
- 13. finalizeInitialChildren

```
prepareForCommit(containerInfo: Container): void
```

Осуществляет подготовку к внесению изменений



- 4. getChildHostContext
- 5. shouldSetTextContent
- 6. createInstance
- 7. finalizeInitialChildren

- 1. getRootHostContext
- 18. prepareForCommit
- 19. appendChildToContainer**

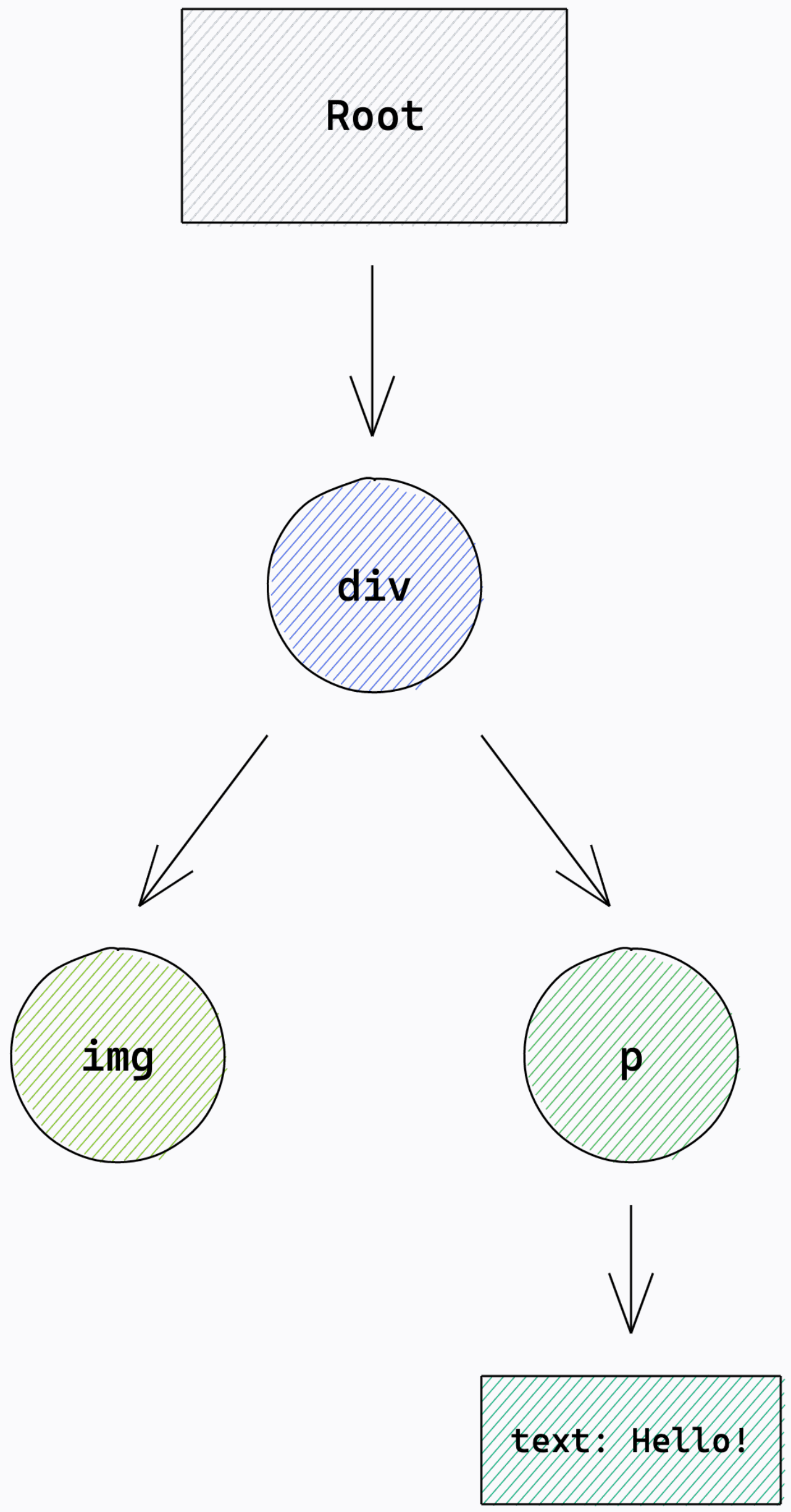
- 2. getChildHostContext
- 3. shouldSetTextContent
- 14. createInstance
- 15, 16. appendInitialChild
- 17. finalizeInitialChildren

- 8. getChildHostContext
- 9. shouldSetTextContent
- 11. createInstance
- 12. appendInitialChild
- 13. finalizeInitialChildren

- 10. createTextInstance

```
appendChildToContainer(  
  container: Container,  
  child: Instance | TextInstance  
): void
```

Добавляет ребенка в представление



- 4. getChildHostContext
- 5. shouldSetTextContent
- 6. createInstance
- 7. finalizeInitialChildren

- 1. getRootHostContext
- 18. prepareForCommit
- 19. appendChildToContainer
- 20. resetAfterCommit**

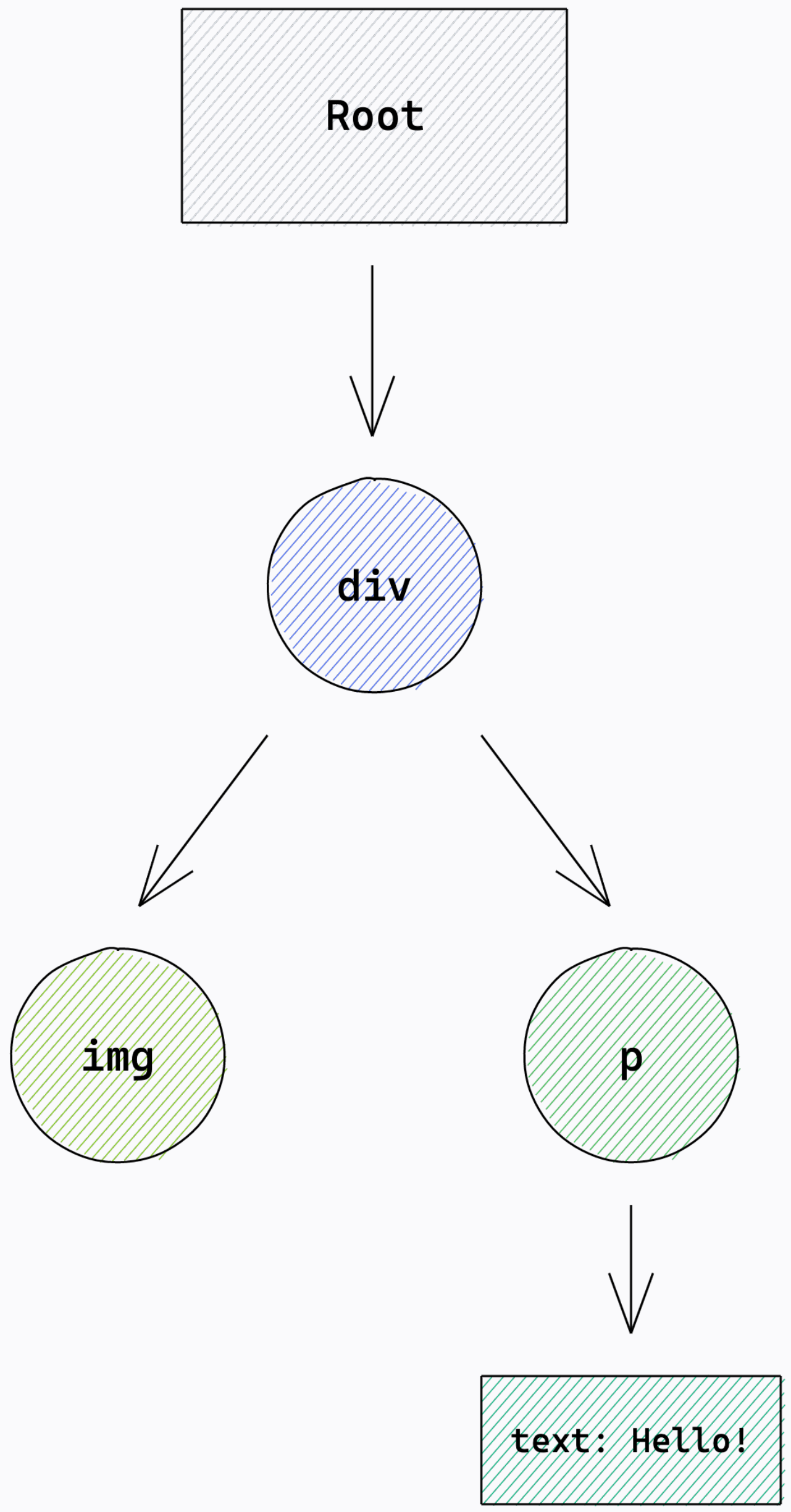
- 2. getChildHostContext
- 3. shouldSetTextContent
- 14. createInstance
- 15, 16. appendInitialChild
- 17. finalizeInitialChildren

- 8. getChildHostContext
- 9. shouldSetTextContent
- 11. createInstance
- 12. appendInitialChild
- 13. finalizeInitialChildren

- 10. createTextInstance

```
resetAfterCommit(containerInfo: Container): void
```

Осуществляет дополнительные действия после внесения изменений



- 4. getChildHostContext
- 5. shouldSetTextContent
- 6. createInstance
- 7. finalizeInitialChildren**
- 21. commitMount**

- 1. getRootHostContext
- 18. prepareForCommit
- 19. appendChildToContainer
- 20. resetAfterCommit

- 2. getChildHostContext
- 3. shouldSetTextContent
- 14. createInstance

- 15, 16. appendInitialChild
- 17. finalizeInitialChildren**
- 23. commitMount**

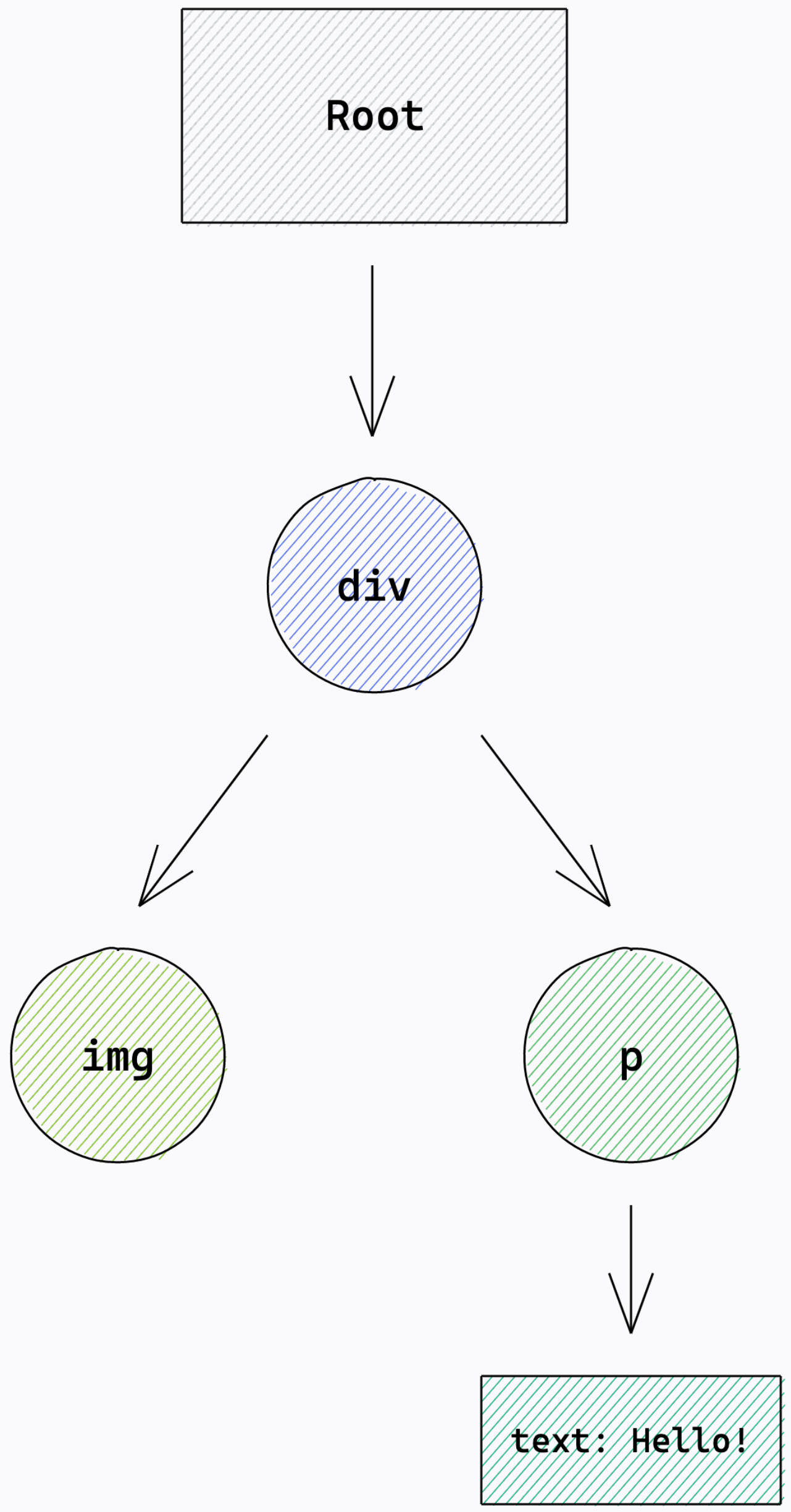
- 8. getChildHostContext
- 9. shouldSetTextContent
- 11. createInstance
- 12. appendInitialChild
- 13. finalizeInitialChildren**
- 22. commitMount**

10. createTextInstance

```
commitMount(instance: Container, type: string, ...): void
```

Дополнительная обработка узлов после их отрисовки





- 4. getChildHostContext
- 5. shouldSetTextContent
- 6. createInstance
- 7. finalizeInitialChildren
- 21. commitMount

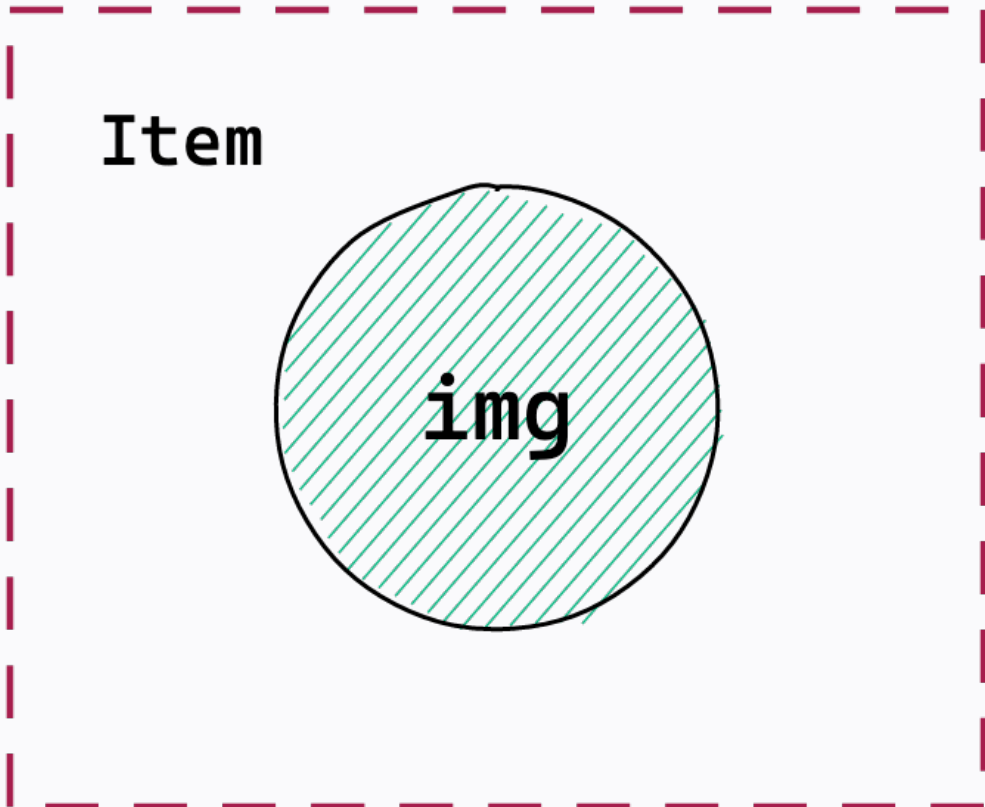
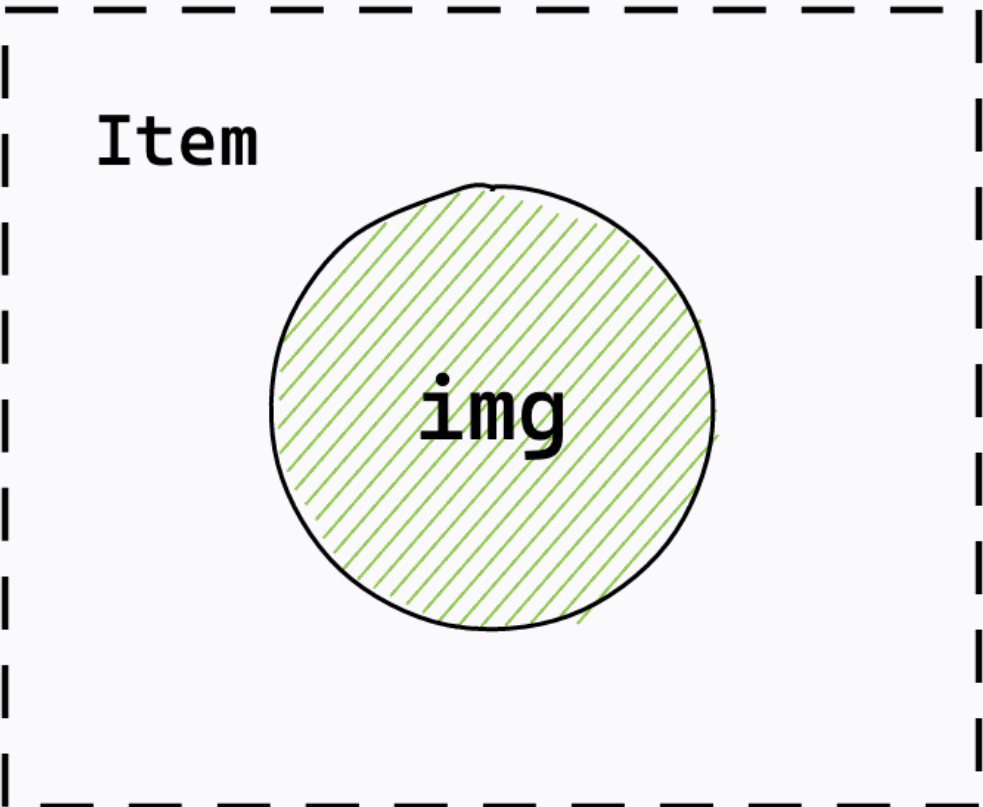
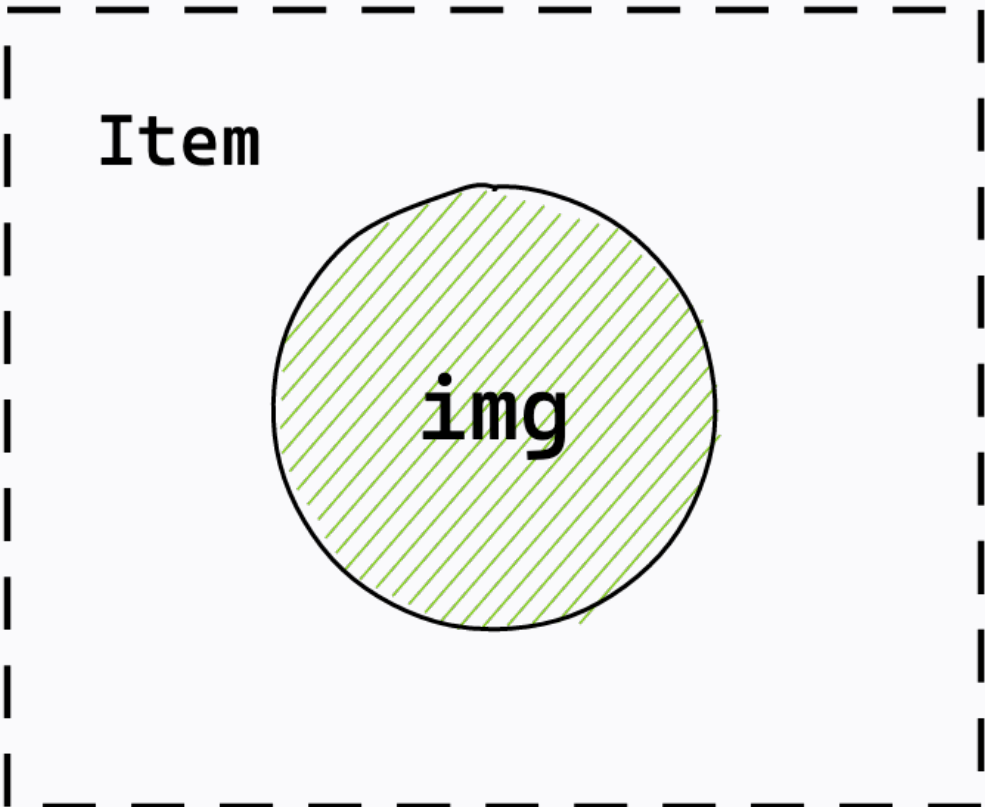
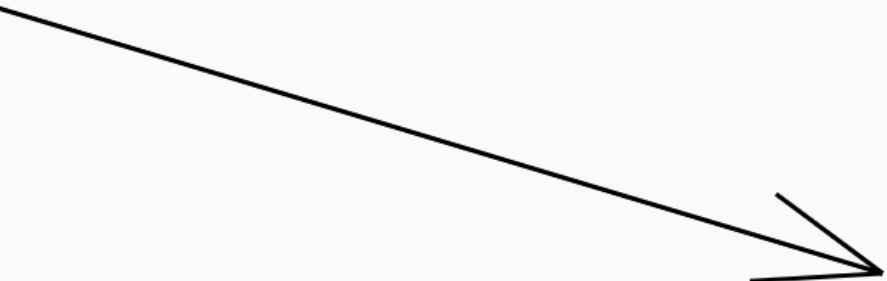
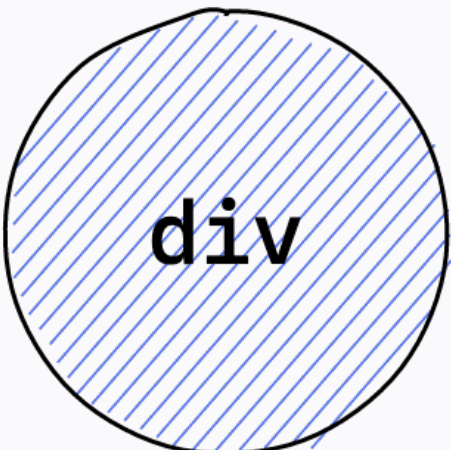
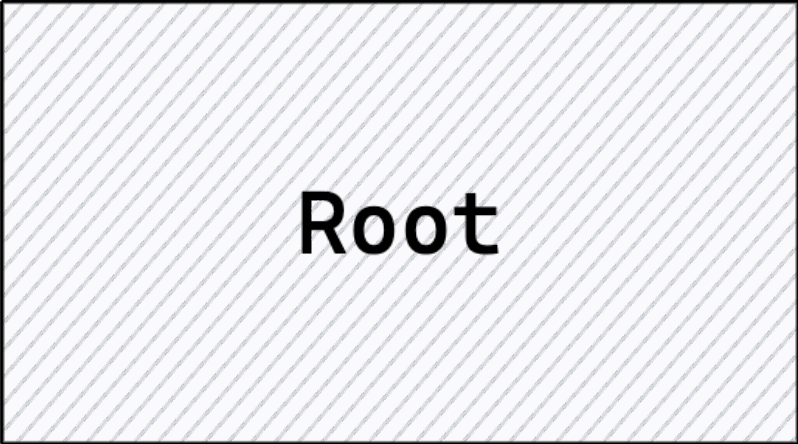
- 1. getRootHostContext
- 18. prepareForCommit
- 19. appendChildToContainer
- 20. resetAfterCommit
- 2. getChildHostContext
- 3. shouldSetTextContent
- 14. createInstance
- 15, 16. appendInitialChild
- 17. finalizeInitialChildren
- 23. commitMount

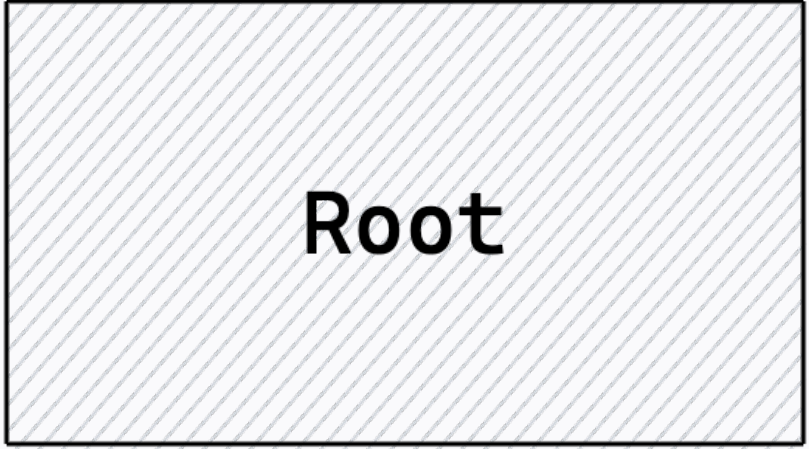
- 8. getChildHostContext
- 9. shouldSetTextContent
- 11. createInstance
- 12. appendInitialChild
- 13. finalizeInitialChildren
- 22. commitMount

10. createTextInstance

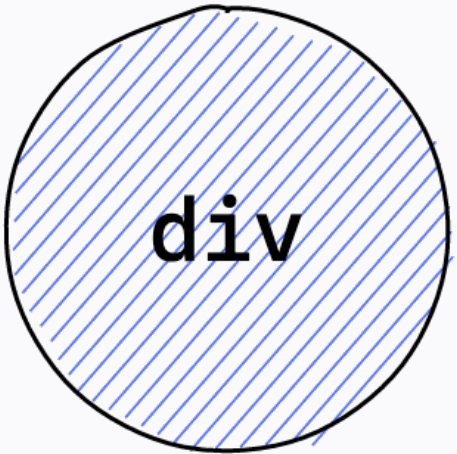
**Отрисовка изменений**

**изменение пропсов**

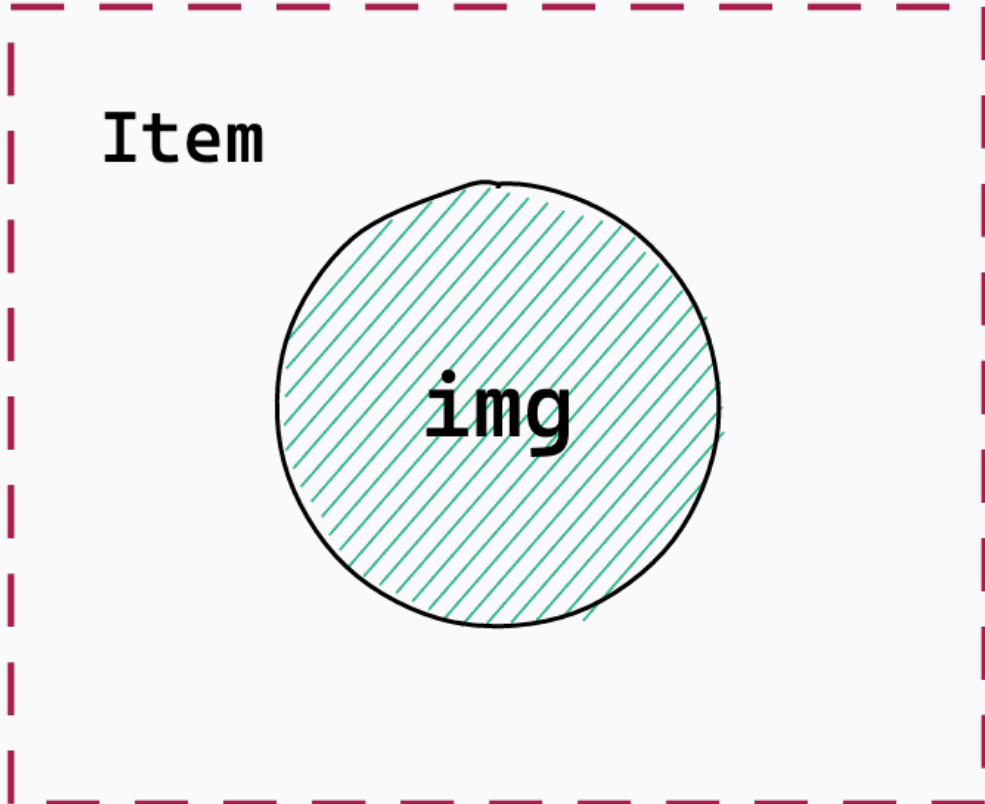
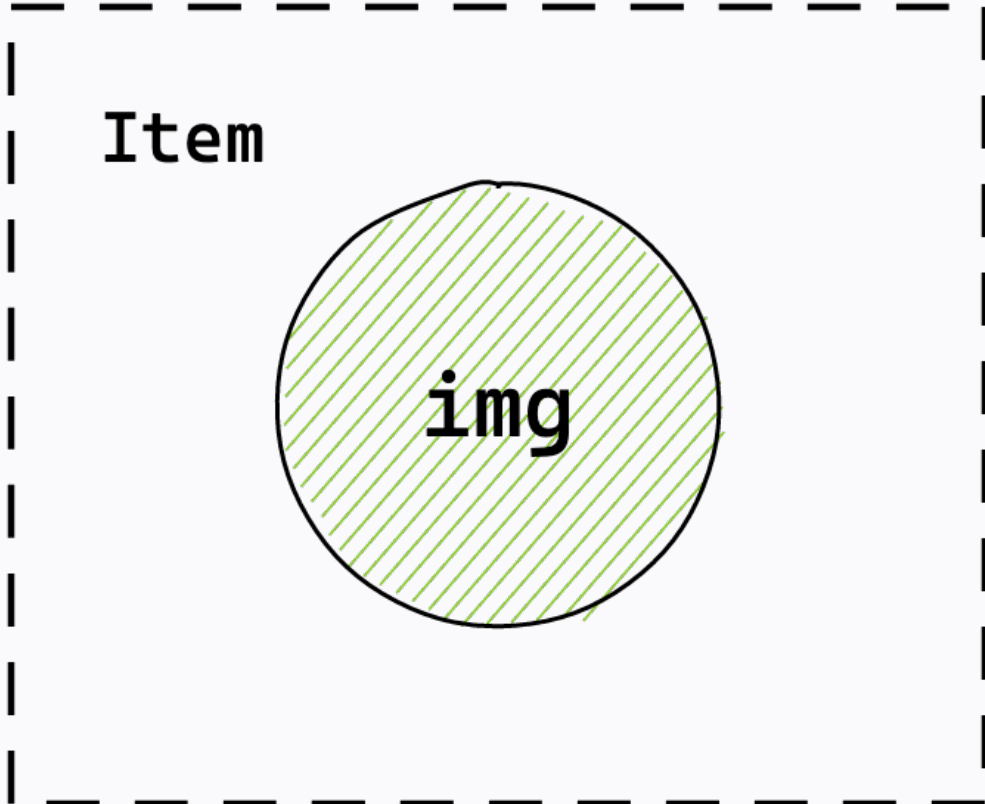
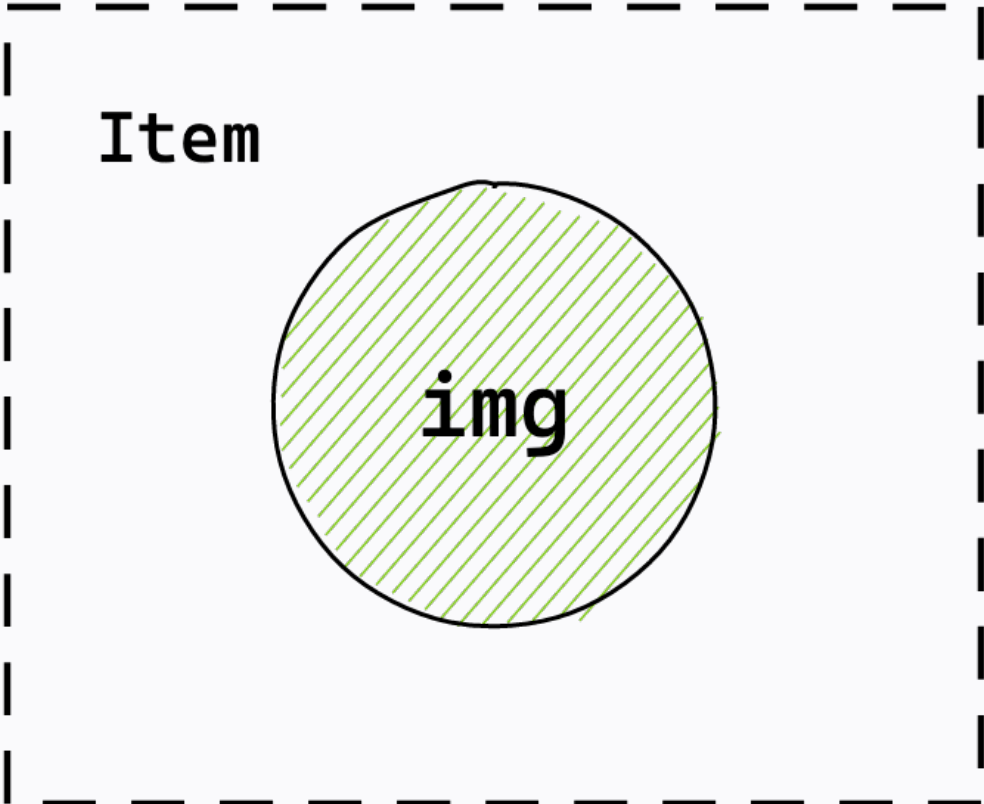
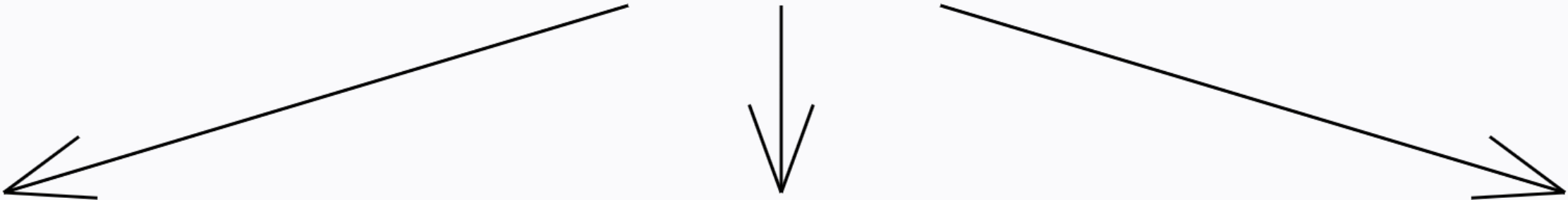




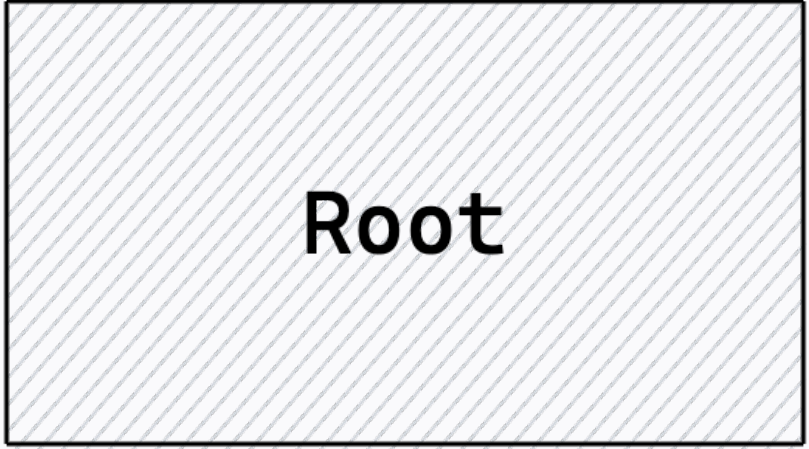
1. **getRootHostContext**



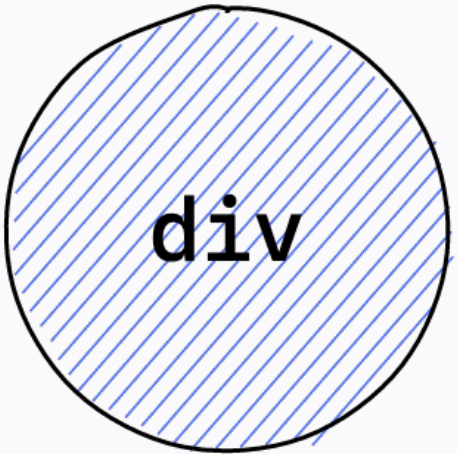
2. **getChildHostContext**



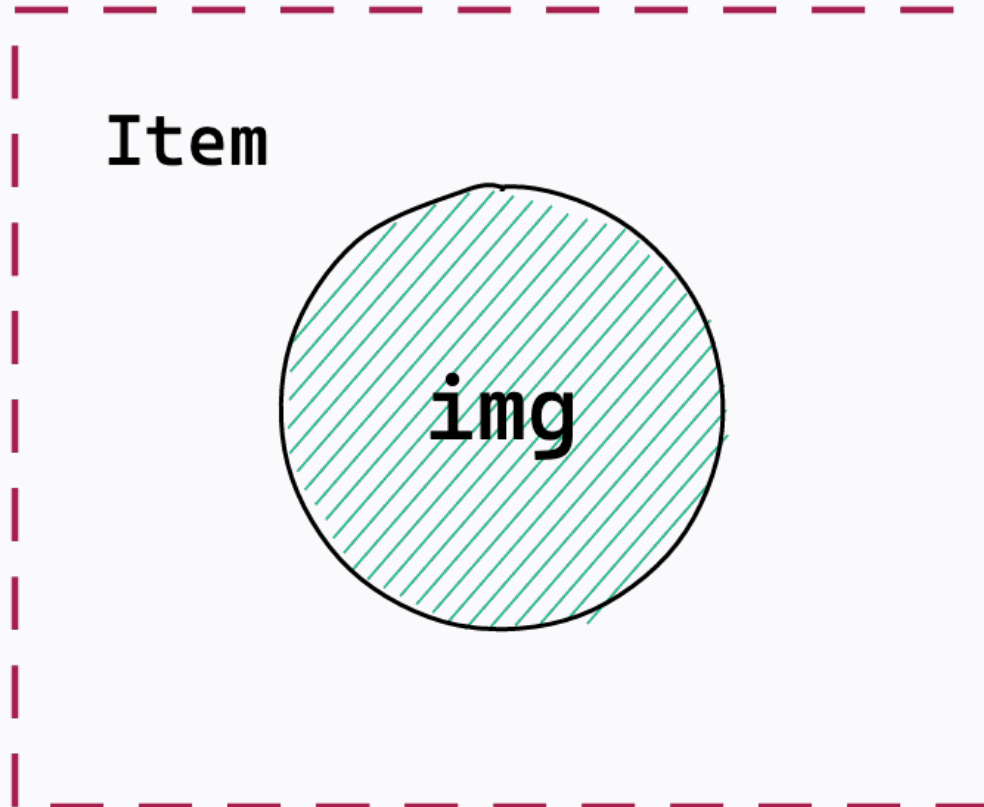
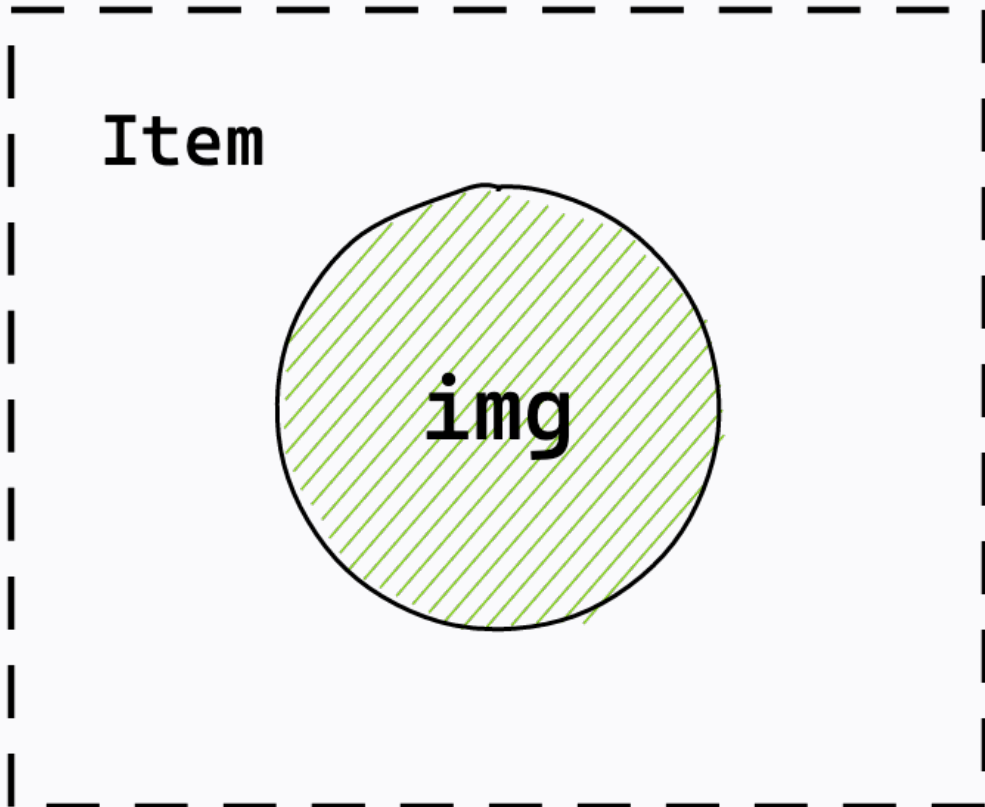
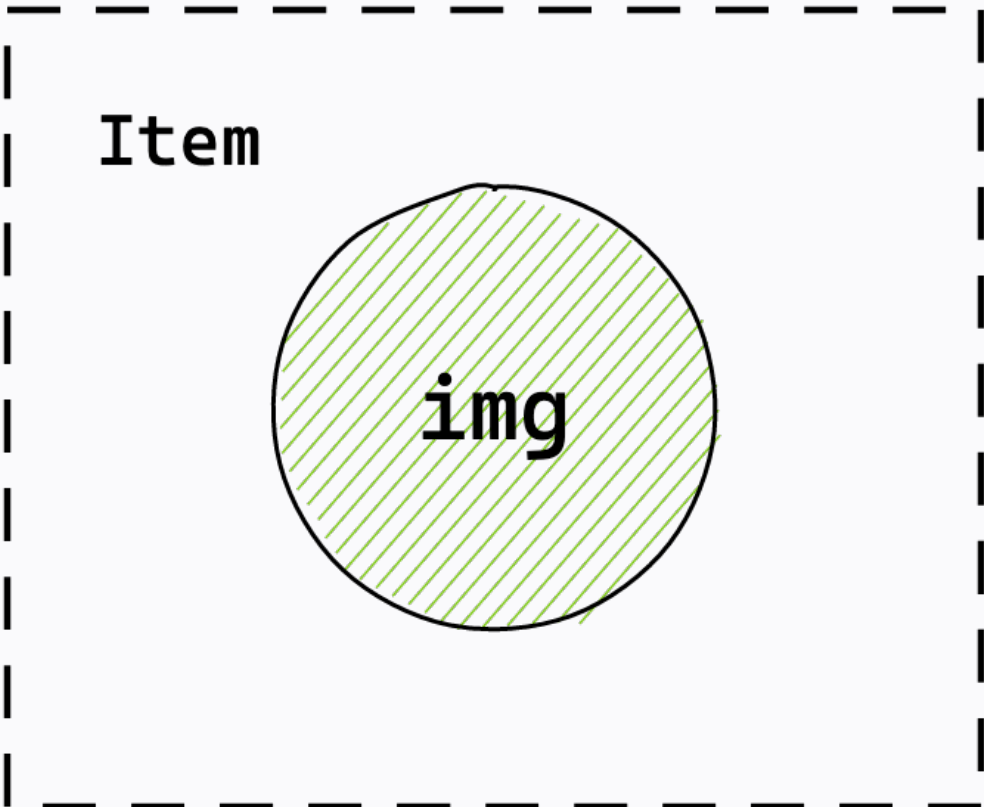
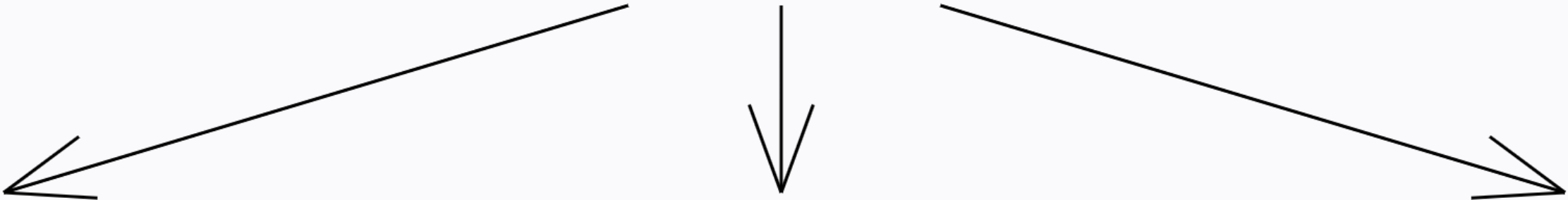
3. **getChildHostContext**  
4. **shouldSetTextContent**



1. `getRootHostContext`



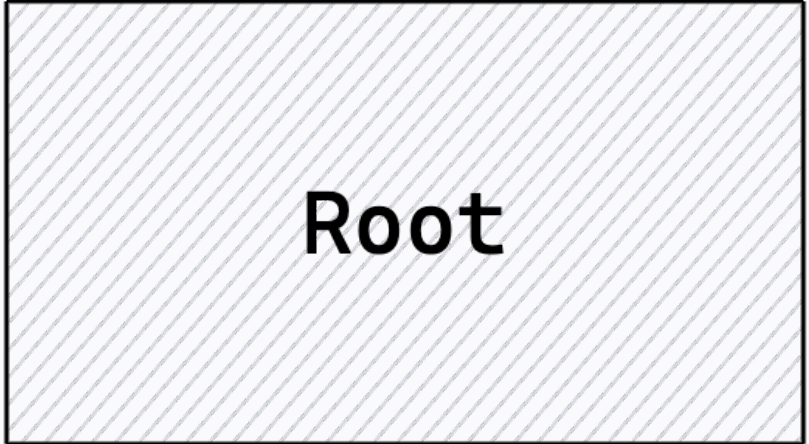
2. `getChildHostContext`



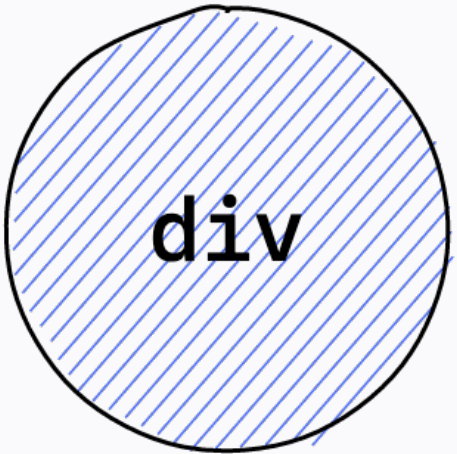
- 3. `getChildHostContext`
- 4. `shouldSetTextContent`
- 5. `prepareUpdate`**

```
prepareUpdate(  
    instance: Instance,  
    type: string,  
    oldProps: Props,  
    newProps: Props,  
    ...  
): null | UpdatePayload;
```

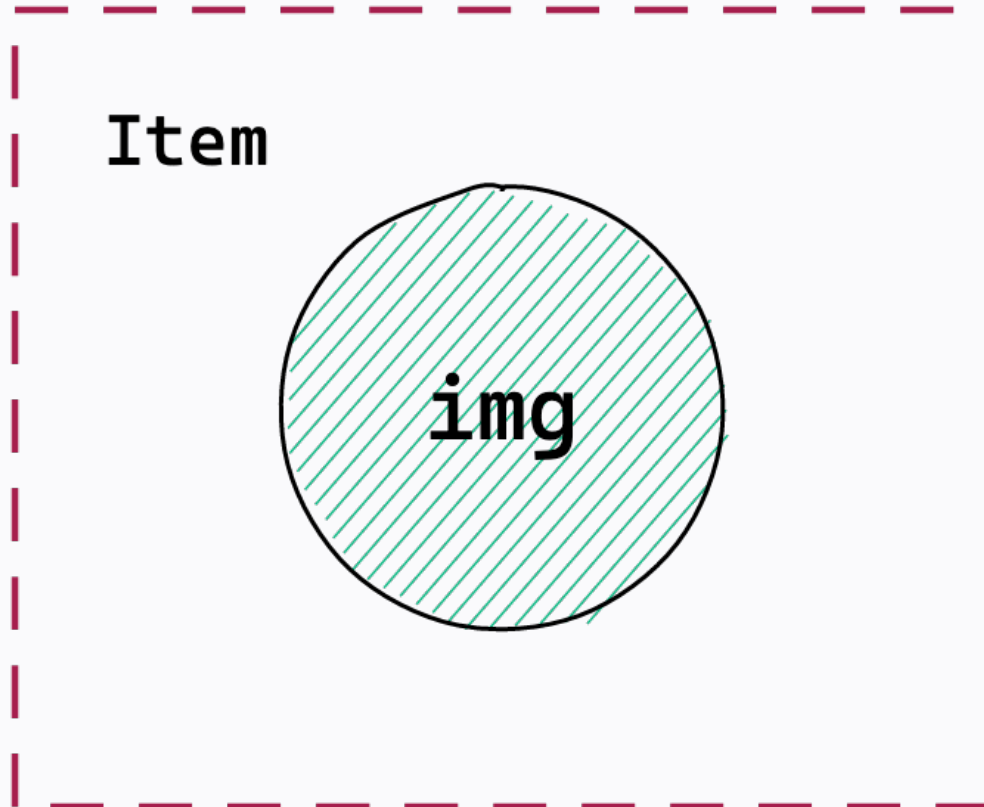
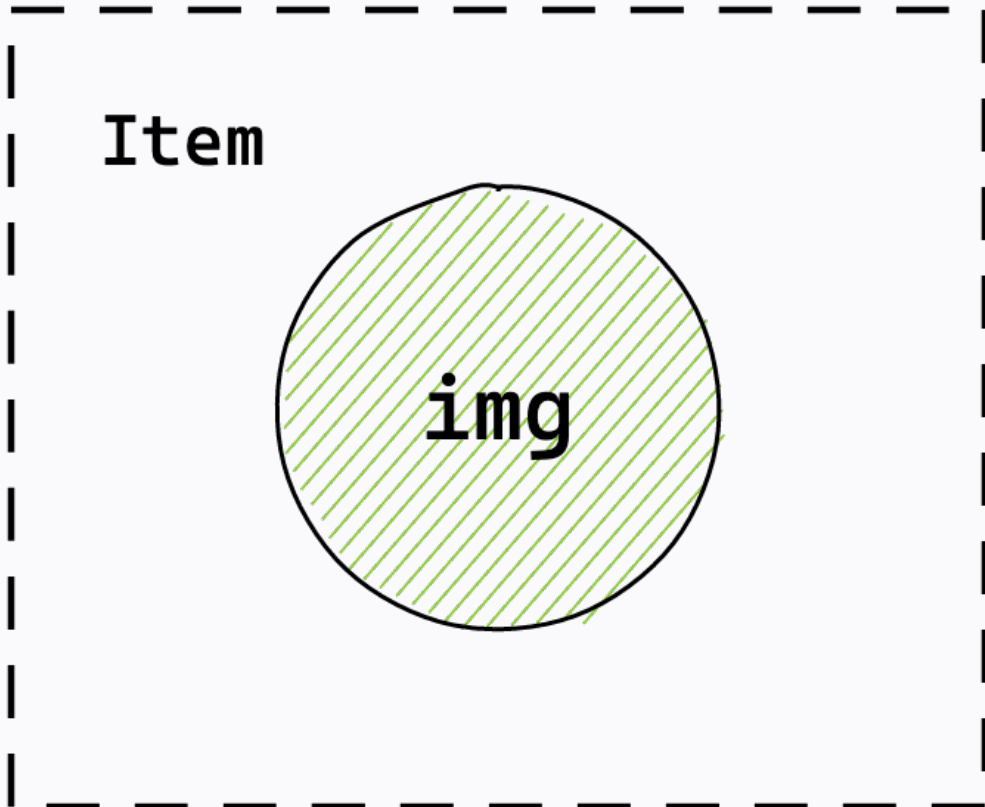
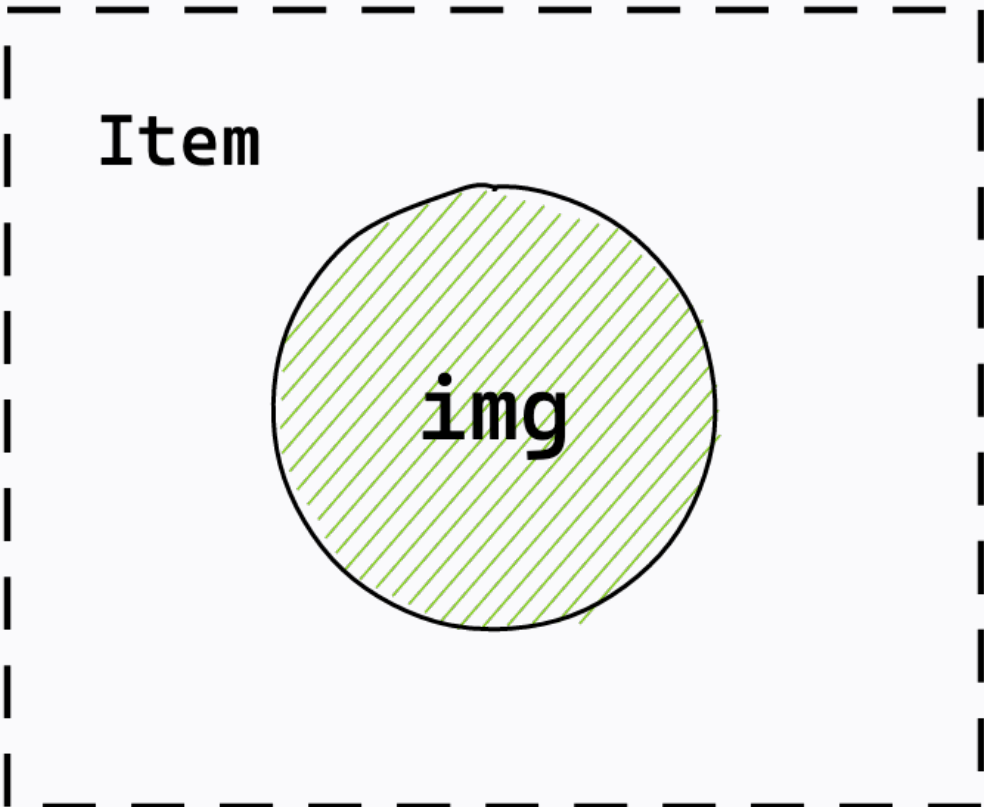
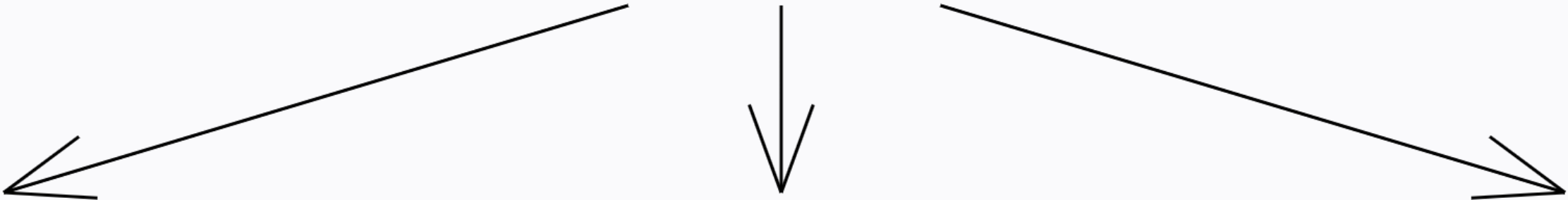
Проверяет есть ли изменения и находит их



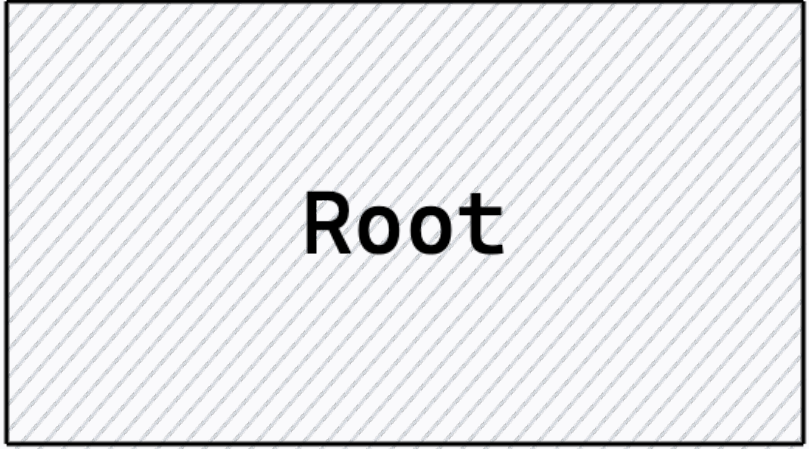
- 1. getRootHostContext
- 6. prepareForCommit**



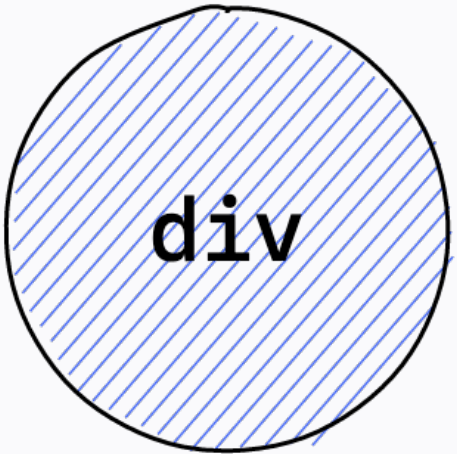
- 2. getChildHostContext



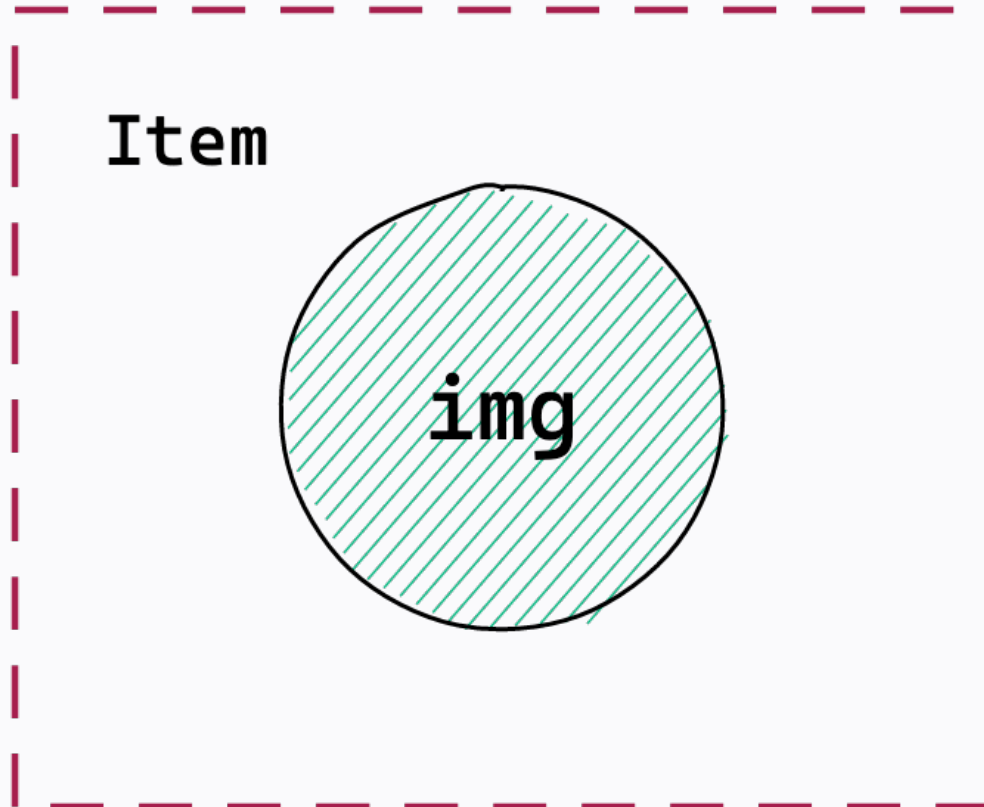
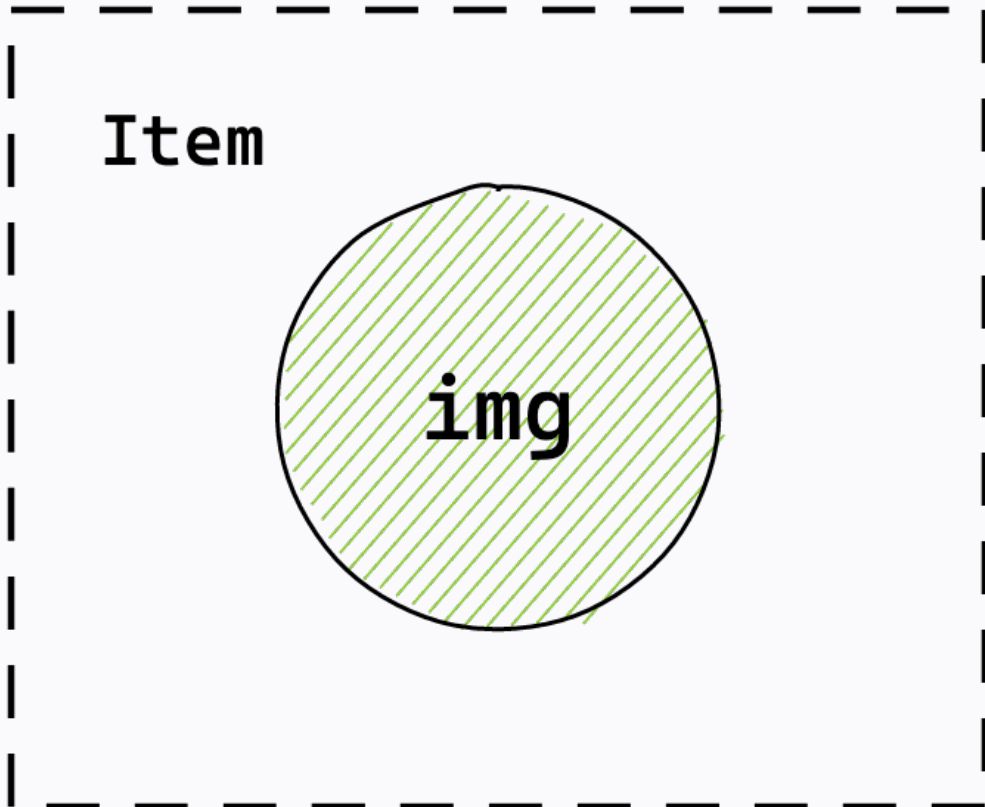
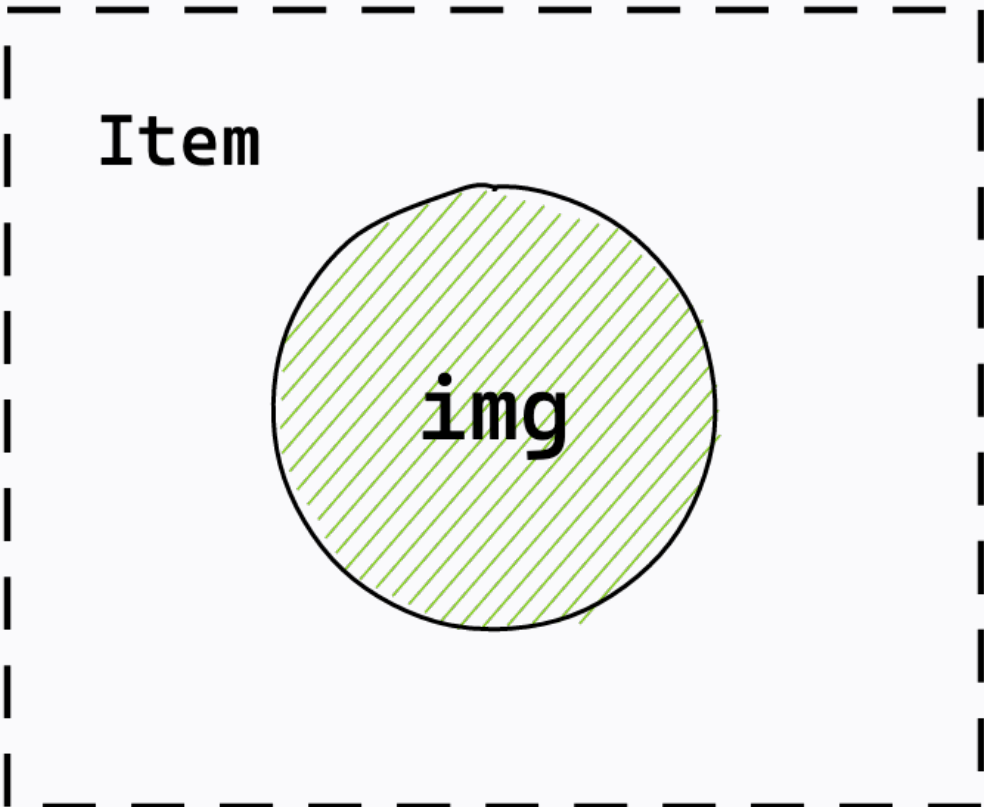
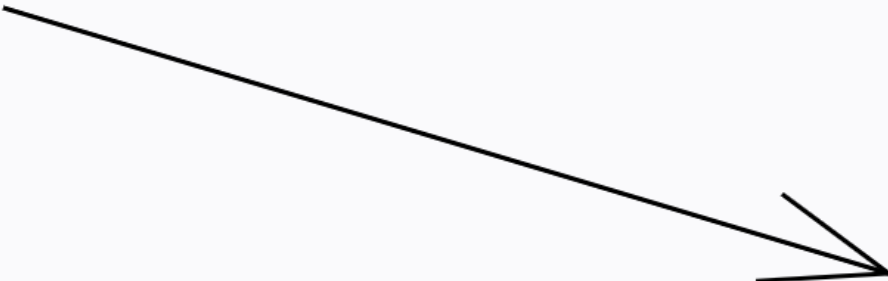
- 3. getChildHostContext
- 4. shouldSetTextContent
- 5. prepareUpdate



- 1. getRootHostContext
- 6. prepareForCommit



- 2. getChildHostContext

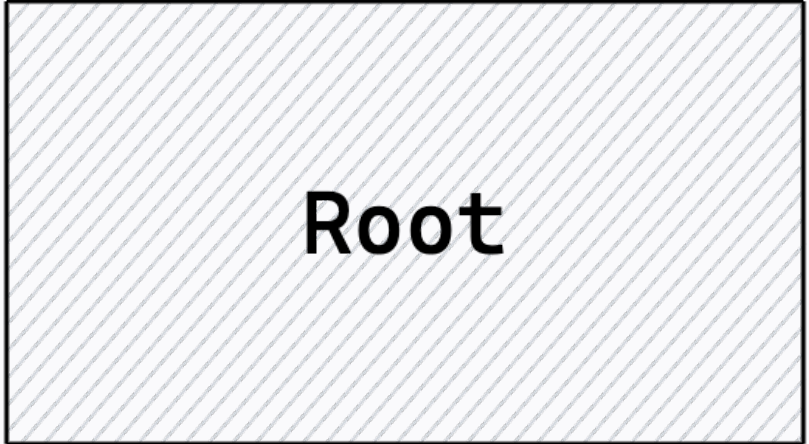


- 3. getChildHostContext
- 4. shouldSetTextContent
- 5. prepareUpdate
- 7. commitUpdate**

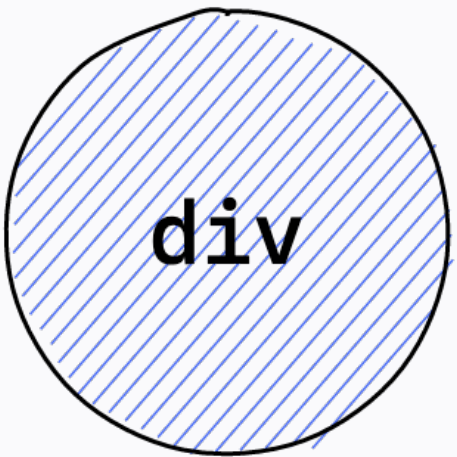


```
commitUpdate(instance: Instance,  
             updatePayload: UpdatePayload,  
             type: string, ...  
): void;
```

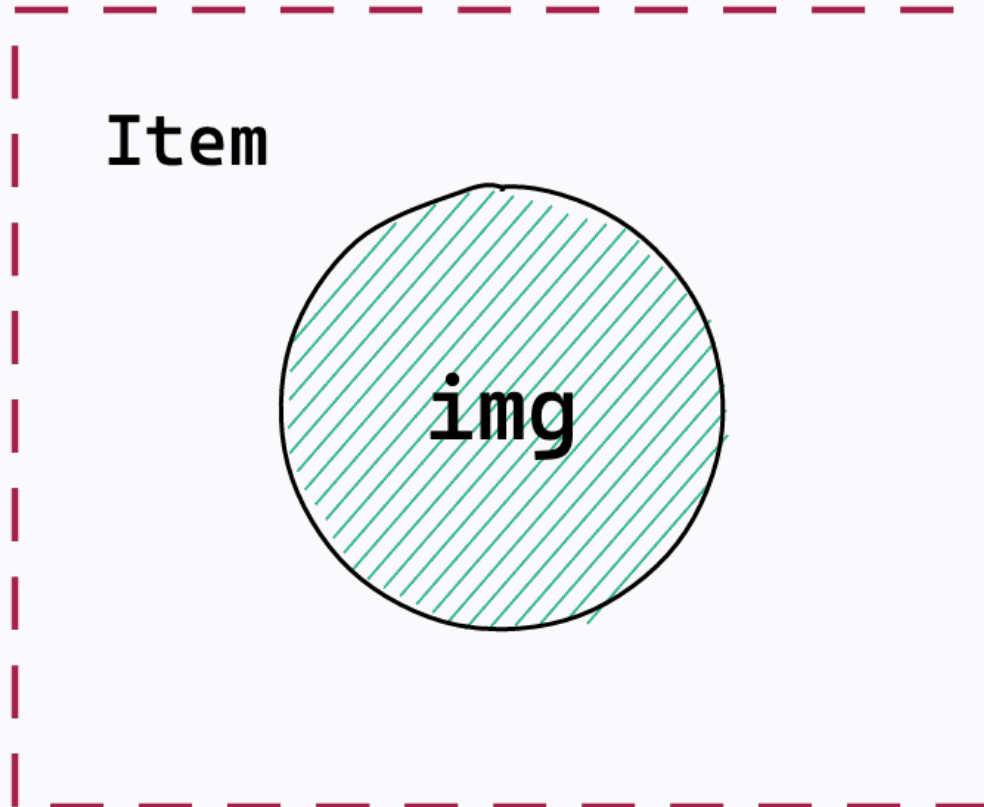
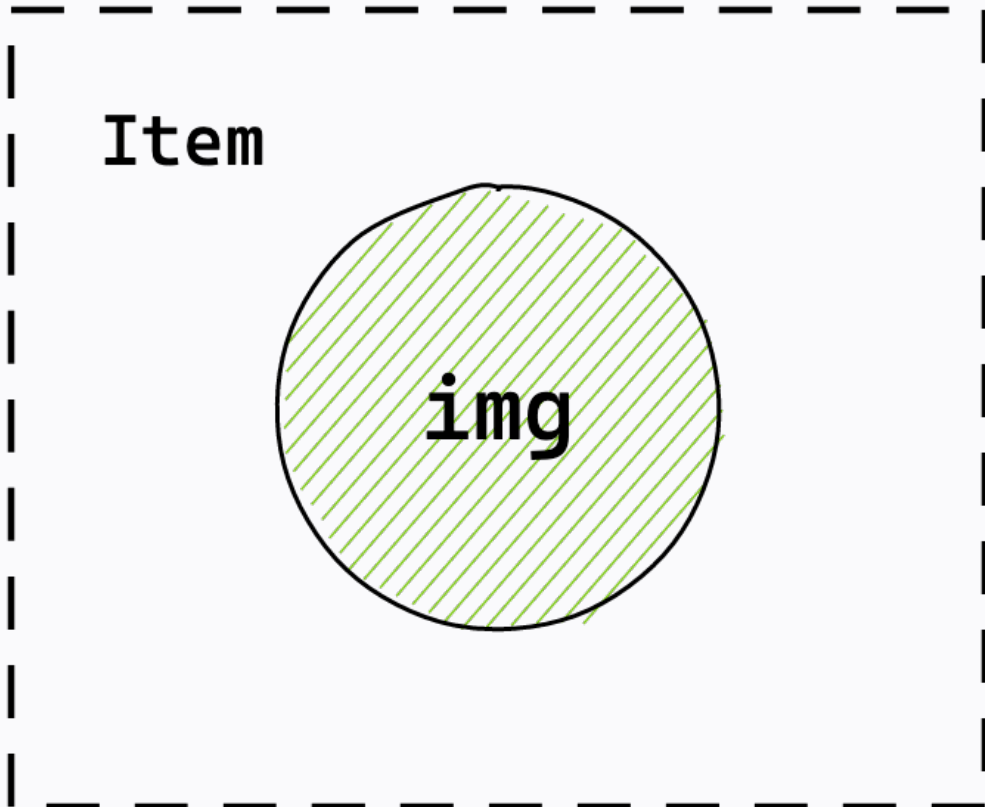
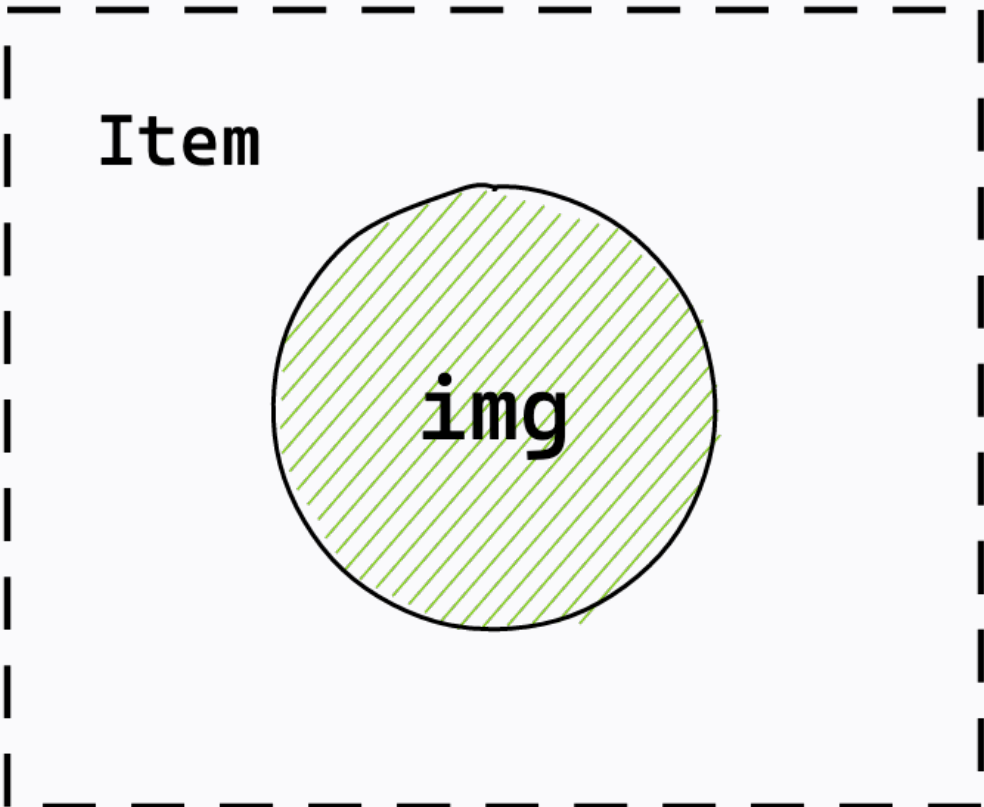
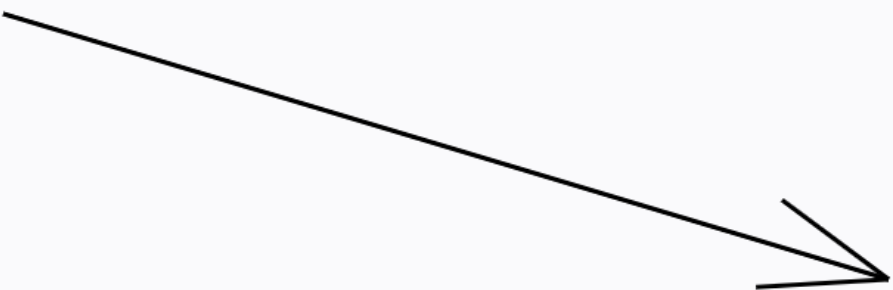
Применяет изменения к элементу



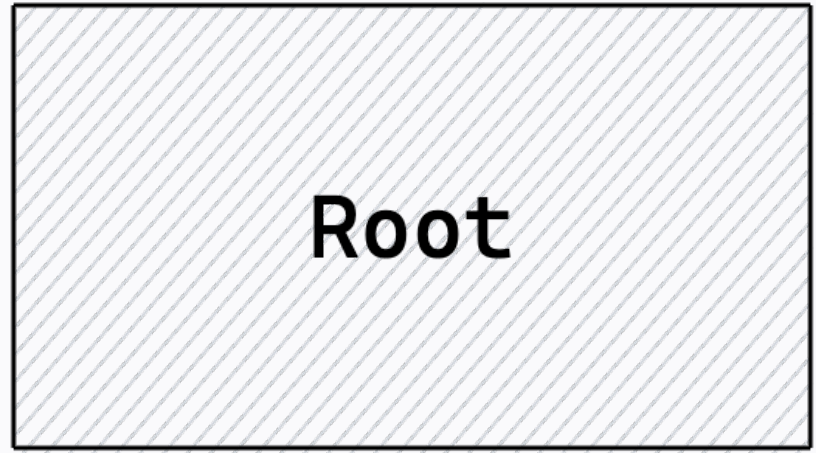
- 1. getRootHostContext
- 6. prepareForCommit
- 8. resetAfterCommit**



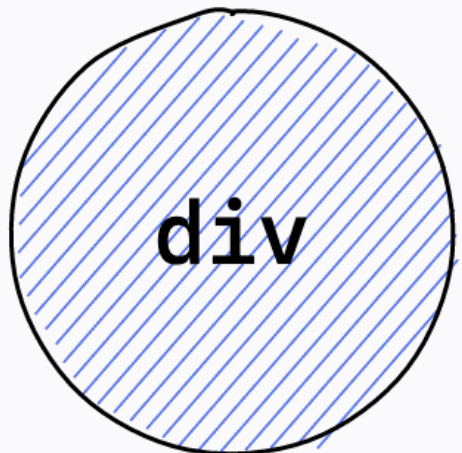
- 2. getChildHostContext



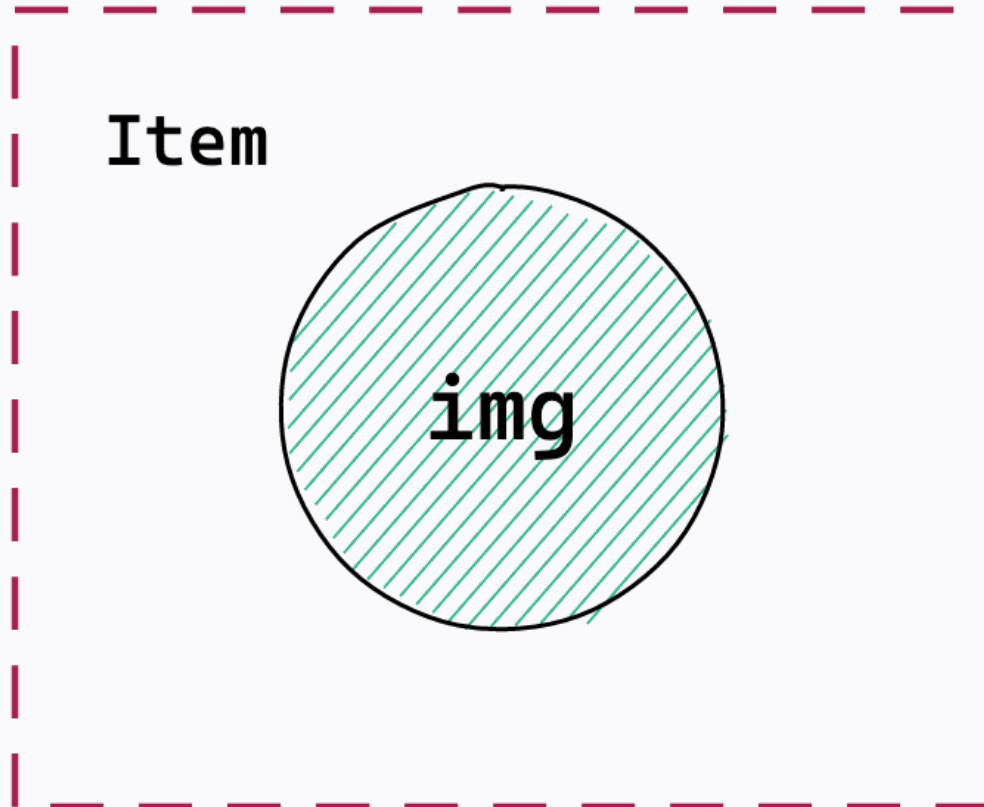
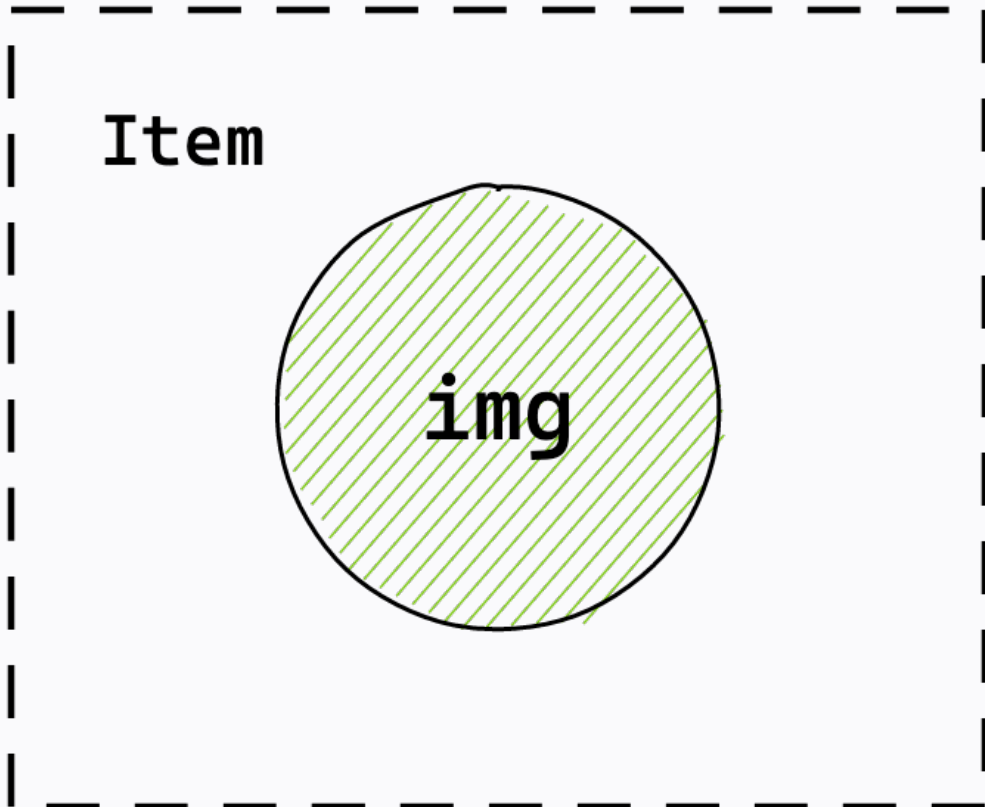
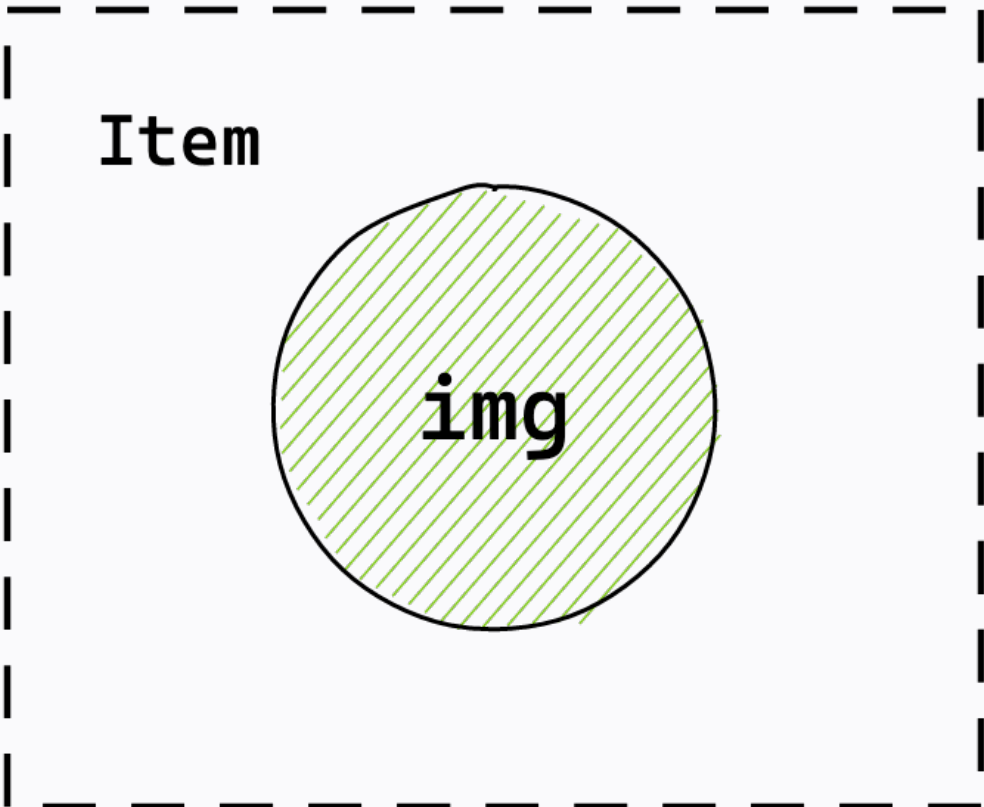
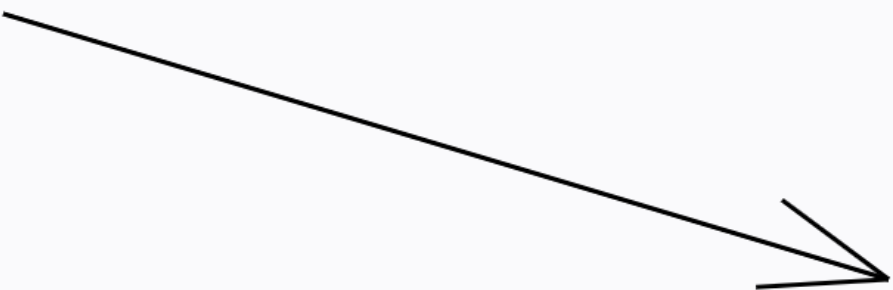
- 3. getChildHostContext
- 4. shouldSetTextContent
- 5. prepareUpdate
- 7. commitUpdate



- 1. getRootHostContext
- 6. prepareForCommit
- 8. resetAfterCommit



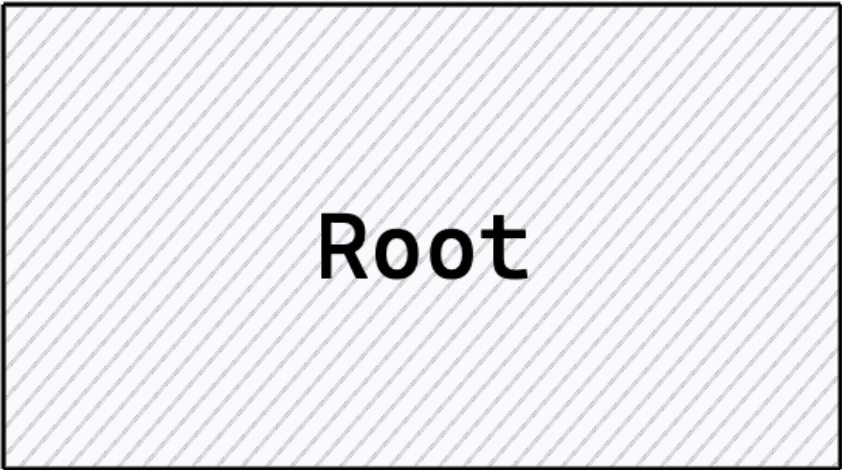
- 2. getChildHostContext



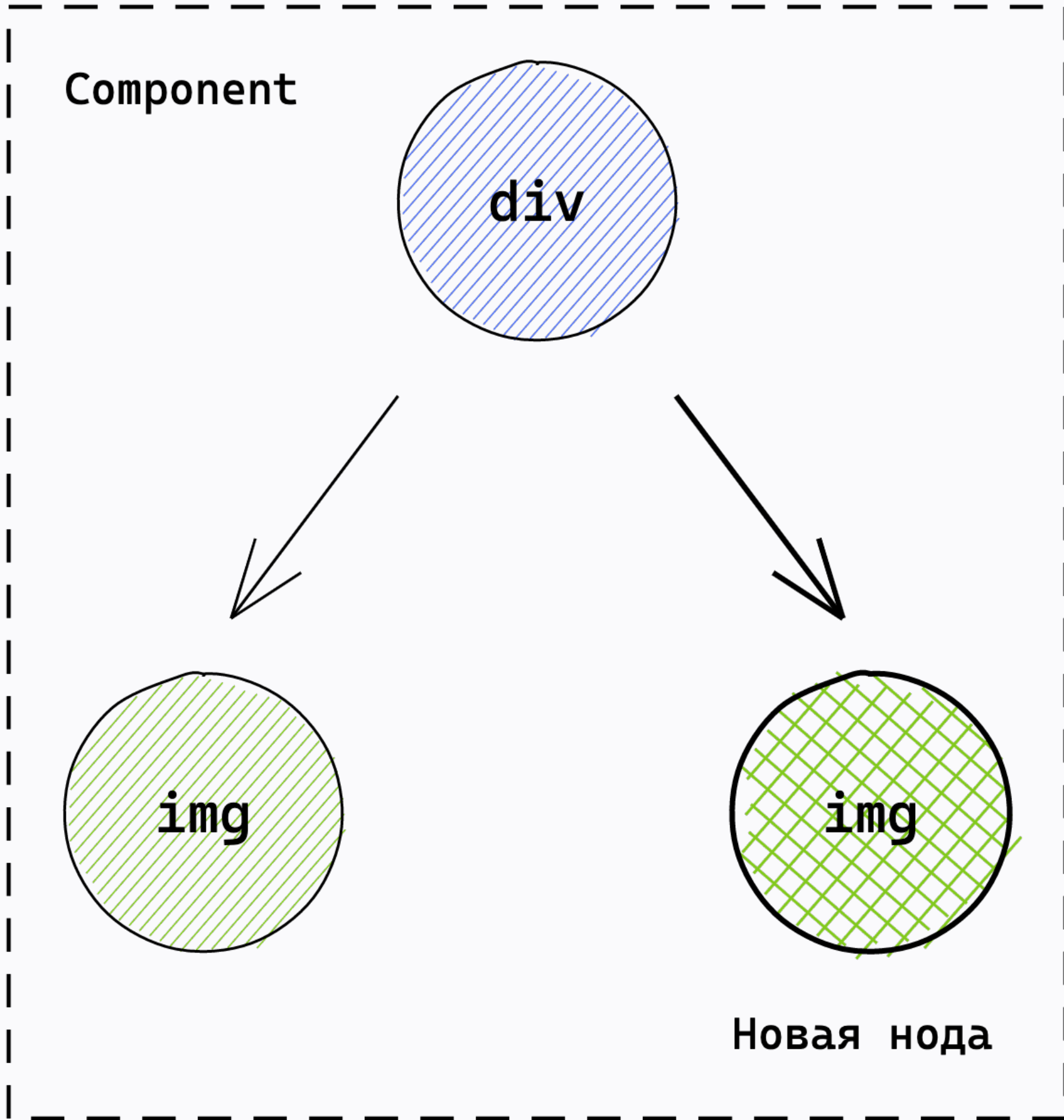
- 3. getChildHostContext
- 4. shouldSetTextContent
- 5. prepareUpdate
- 7. commitUpdate

**Отрисовка изменений**

**вставка узлов**



- 1. `getRootHostContext`
- 12. `prepareForCommit`
- 14. `resetAfterCommit`

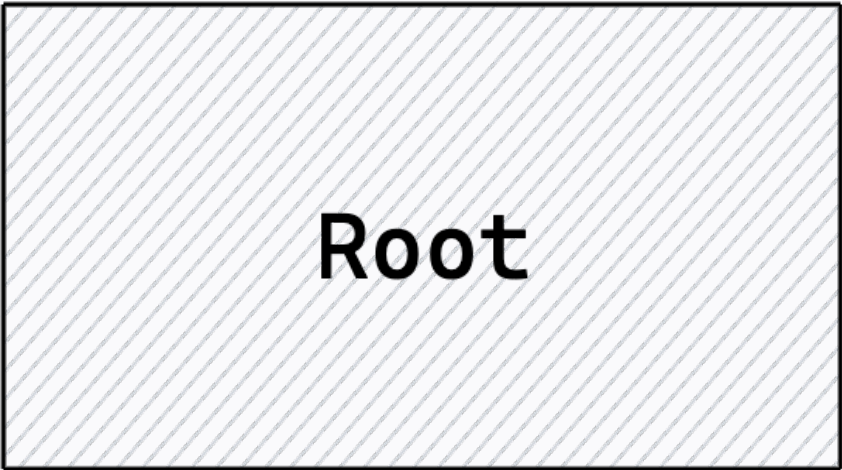


- 4. `getChildHostContext`
- 5. `shouldSetTextContent`
- 6. `prepareUpdate`

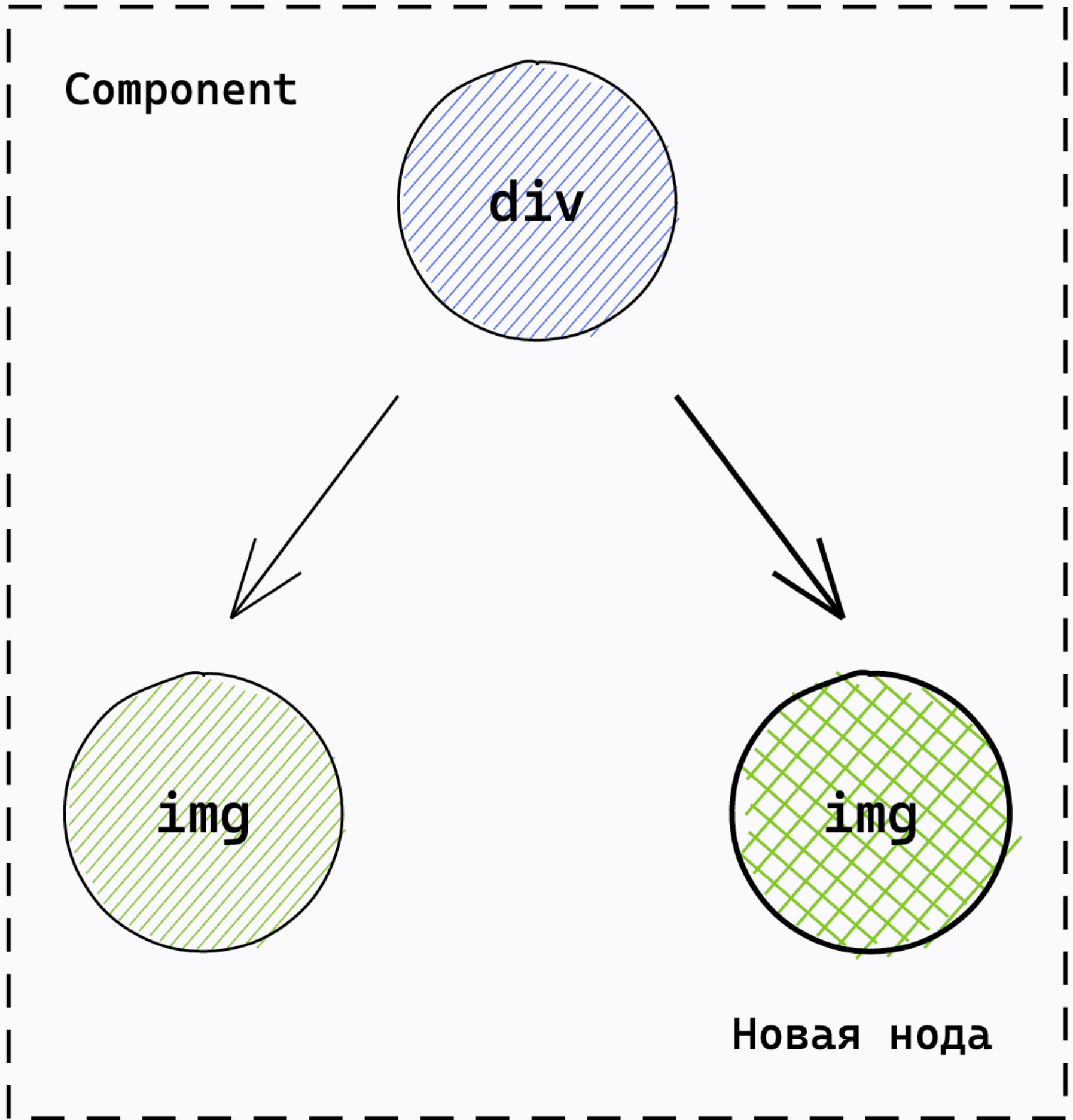
- 2. `getChildHostContext`
- 3. `shouldSetTextContent`
- 11. `prepareUpdate`
- 13. `appendChild`

- 7. `getChildHostContext`**
- 8. `shouldSetTextContent`**
- 9. `createElement`**
- 10. `finalizeInitialChildren`**

Новая нода



- 1. getRootHostContext
- 12. prepareForCommit
- 14. resetAfterCommit

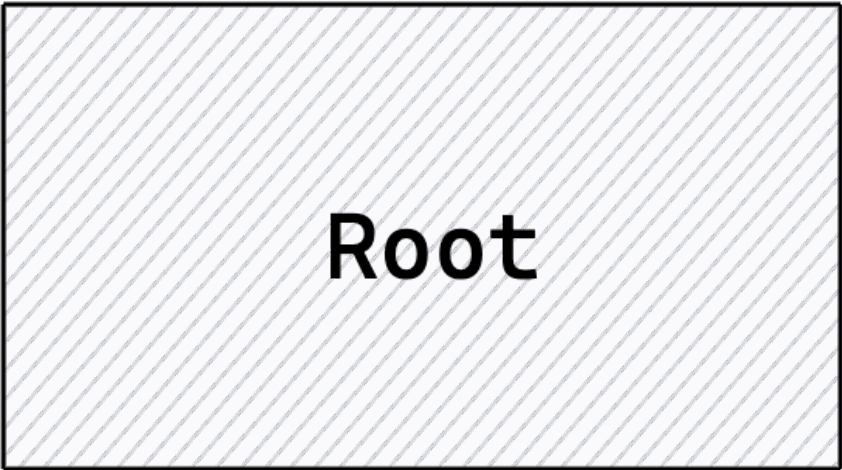


- 4. getChildHostContext
- 5. shouldSetTextContent
- 6. prepareUpdate

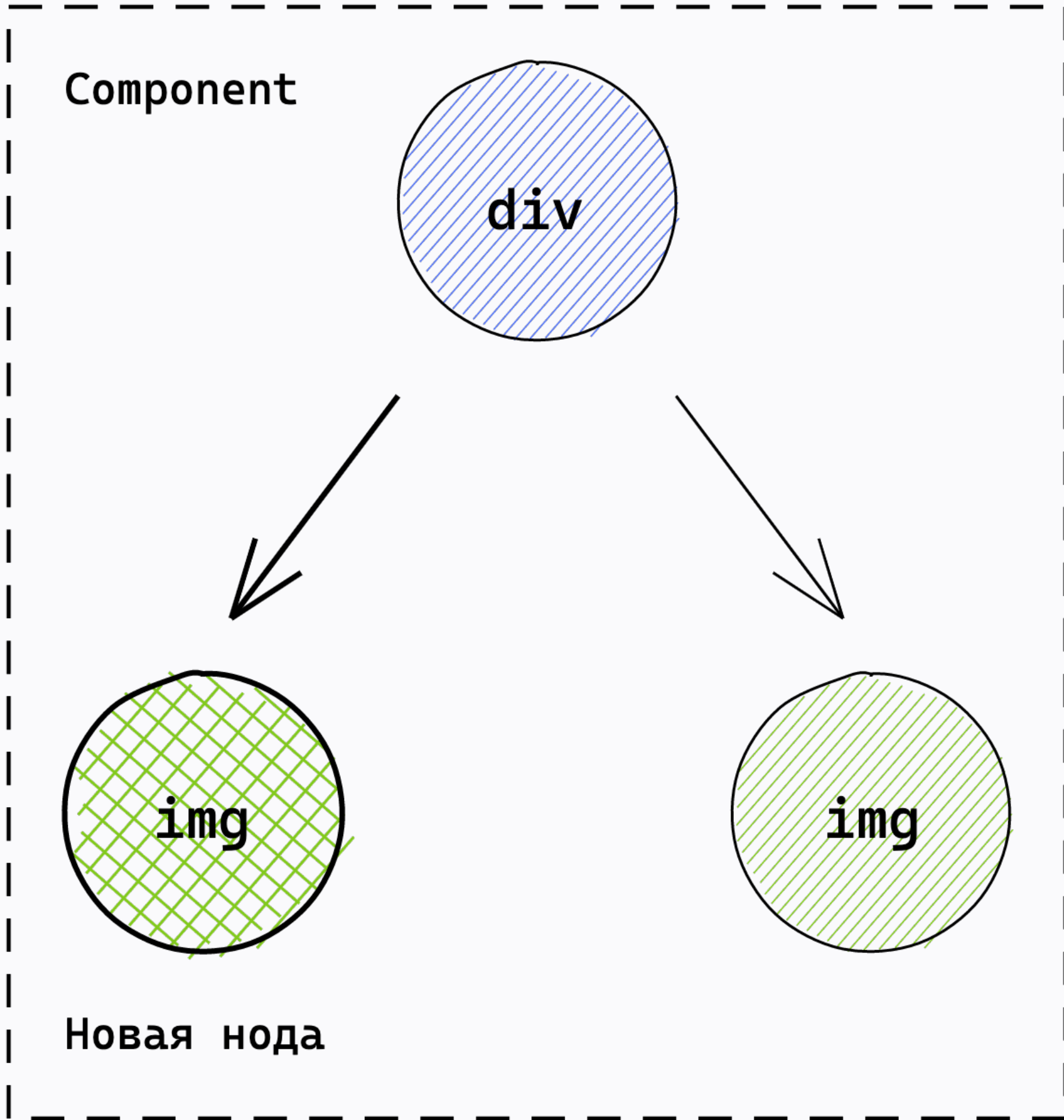
- 2. getChildHostContext
- 3. shouldSetTextContent
- 11. prepareUpdate
- 13. appendChild**
- 7. getChildHostContext
- 8. shouldSetTextContent
- 9. createInstance
- 10. finalizeInitialChildren

```
appendChild(  
    parentInstance: Instance,  
    child: Instance | TextInstance  
): void
```

Добавляет ребенка в конец списка детей



- 1. getRootHostContext
- 12. prepareForCommit
- 14. resetAfterCommit



- 4. getChildHostContext
- 5. shouldSetTextContent
- 6. createInstance
- 7. finalizeInitialChildren

- 2. getChildHostContext
- 3. shouldSetTextContent
- 11. prepareUpdate
- 13. insertBefore**
- 8. getChildHostContext
- 9. shouldSetTextContent
- 10. prepareUpdate



```
insertBefore(  
    parentInstance: Instance,  
    child: Instance | TextInstance,  
    beforeChild: Instance | TextInstance  
): void
```

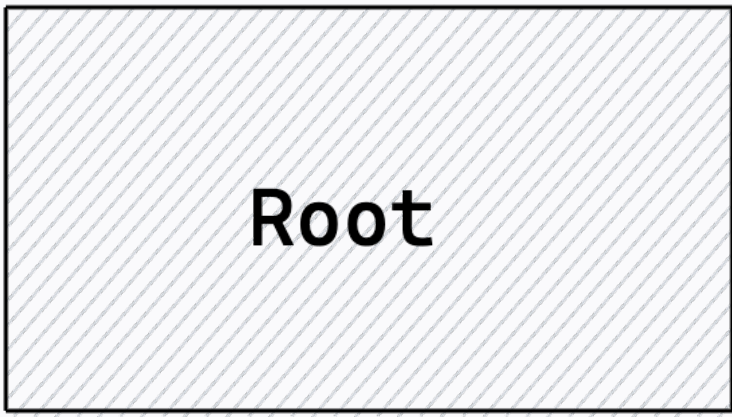
Добавляет ребенка после определенного элемента

```
insertInContainerBefore(  
    container: Container,  
    child: Instance | TextInstance,  
    beforeChild: Instance | TextInstance  
): void
```

Добавляет ребенка после определенного элемента в контейнере

# Отрисовка изменений

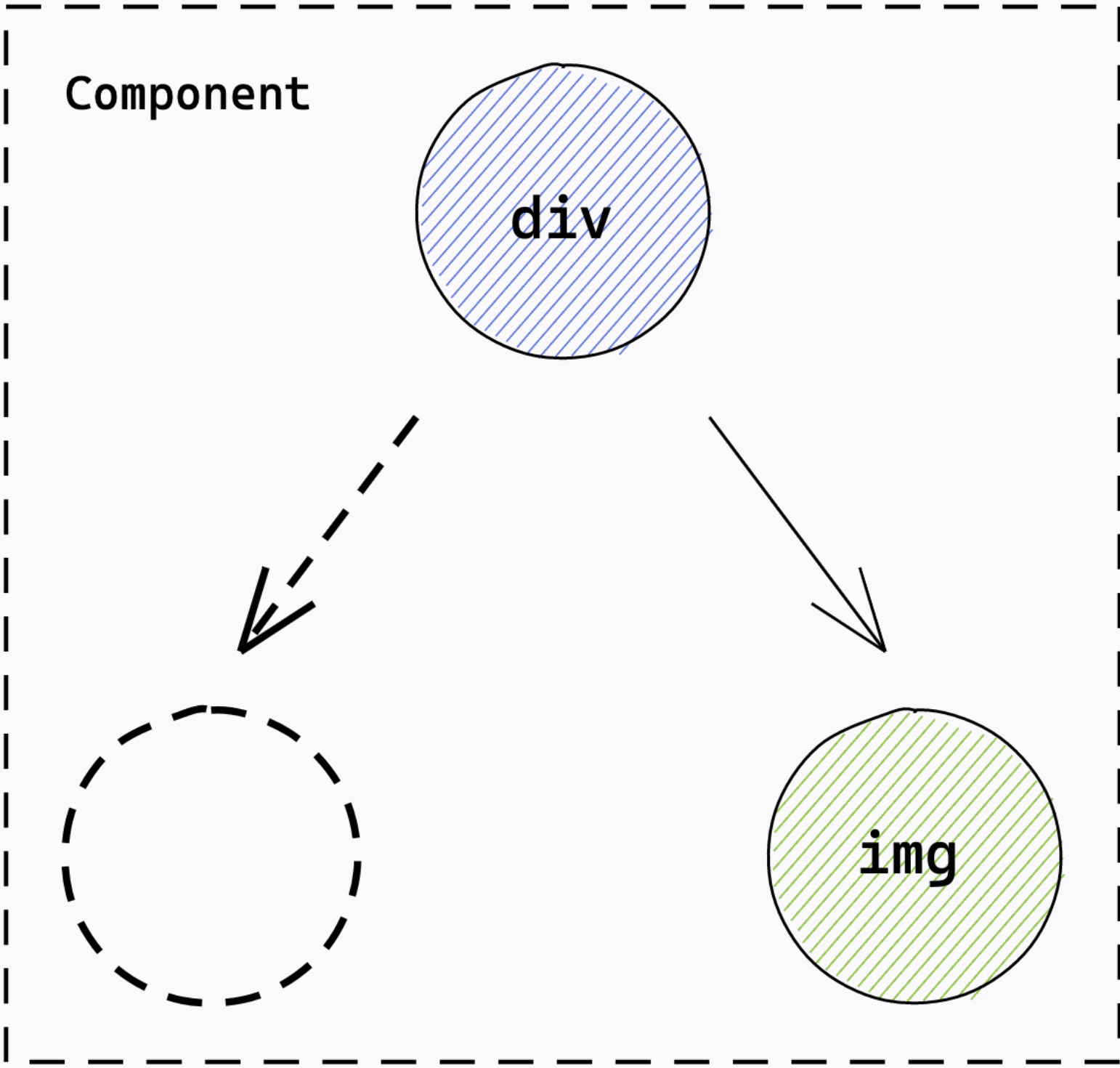
удаление узла



1. getRootHostContext

8. prepareForCommit

10. resetAfterCommit



2. getChildHostContext

3. shouldSetTextContent

7. prepareUpdate

**9. removeChild**

4. getChildHostContext

5. shouldSetTextContent

6. prepareUpdate

```
removeChild(  
    parentInstance: Instance,  
    child: Instance | TextInstance  
): void
```

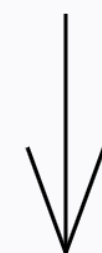
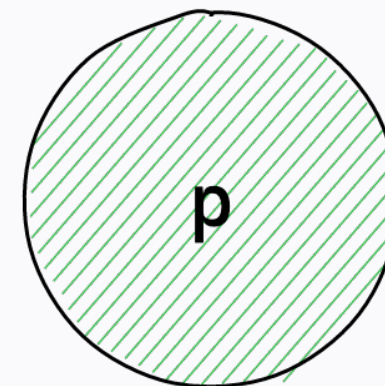
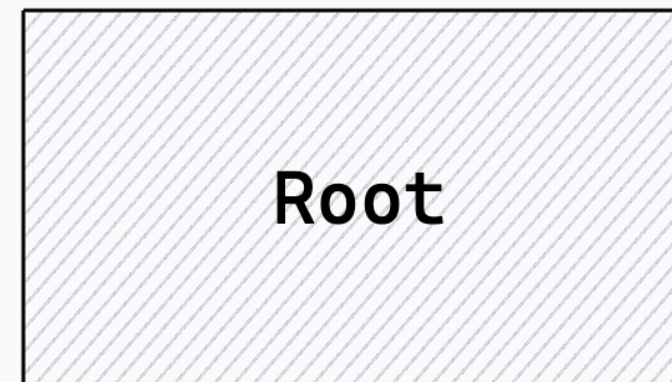
Удаляет поддерево

```
removeChildFromContainer(  
    container: Container,  
    child: Instance | TextInstance  
): void
```

Удаляет поддерево из контейнера

**Отрисовка изменений**

**ТЕКСТОВЫЕ ЛИСТЫ**



1. `getRootHostContext`

5. `prepareForCommit`

7. `resetAfterCommit`

2. `getChildHostContext`

3. `shouldSetTextContent`

4. `prepareUpdate`

**6. `commitTextUpdate`**



```
commitTextUpdate(  
    textInstance: TextInstance,  
    oldText: string,  
    newText: string  
): void
```

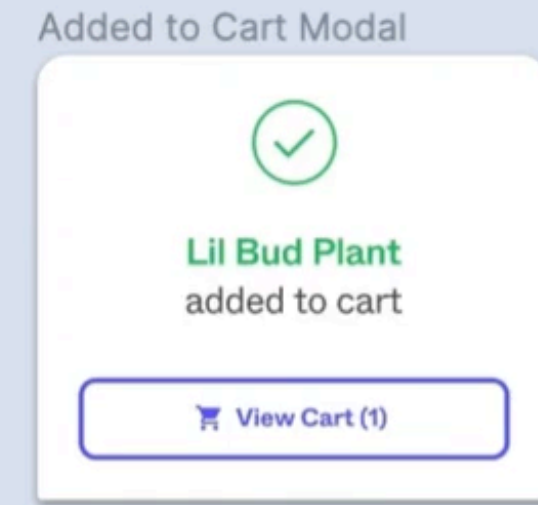
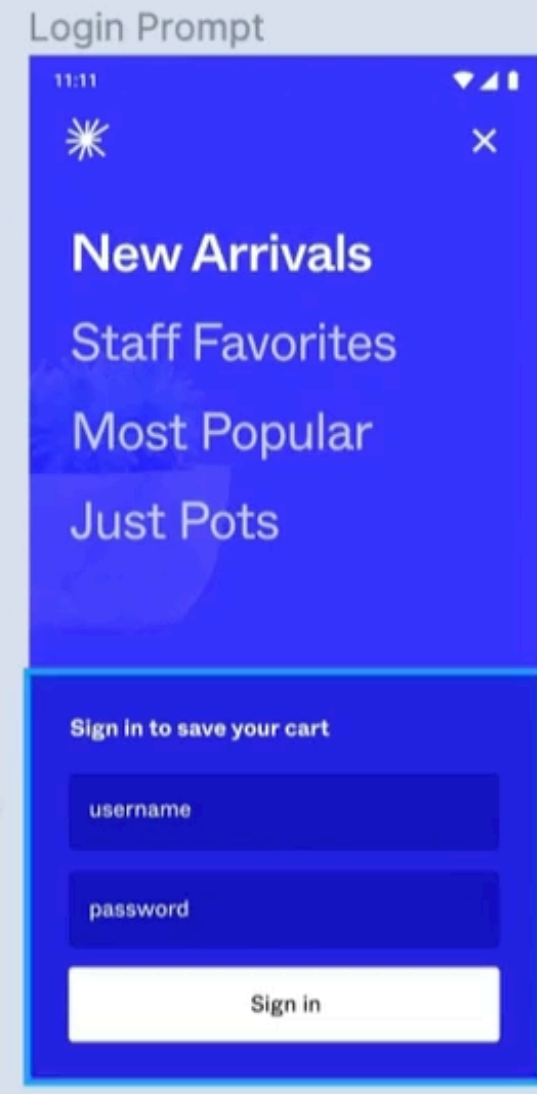
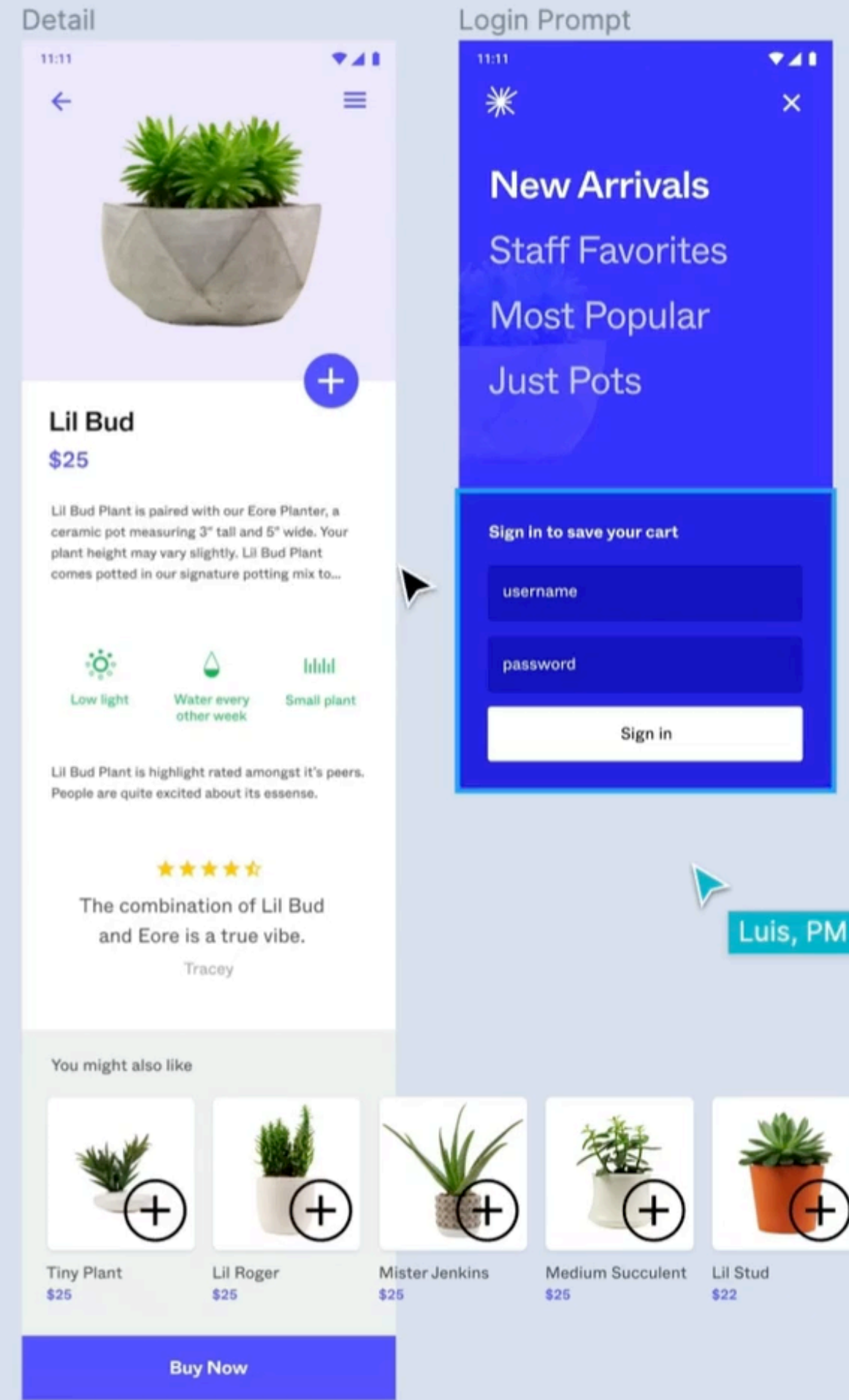
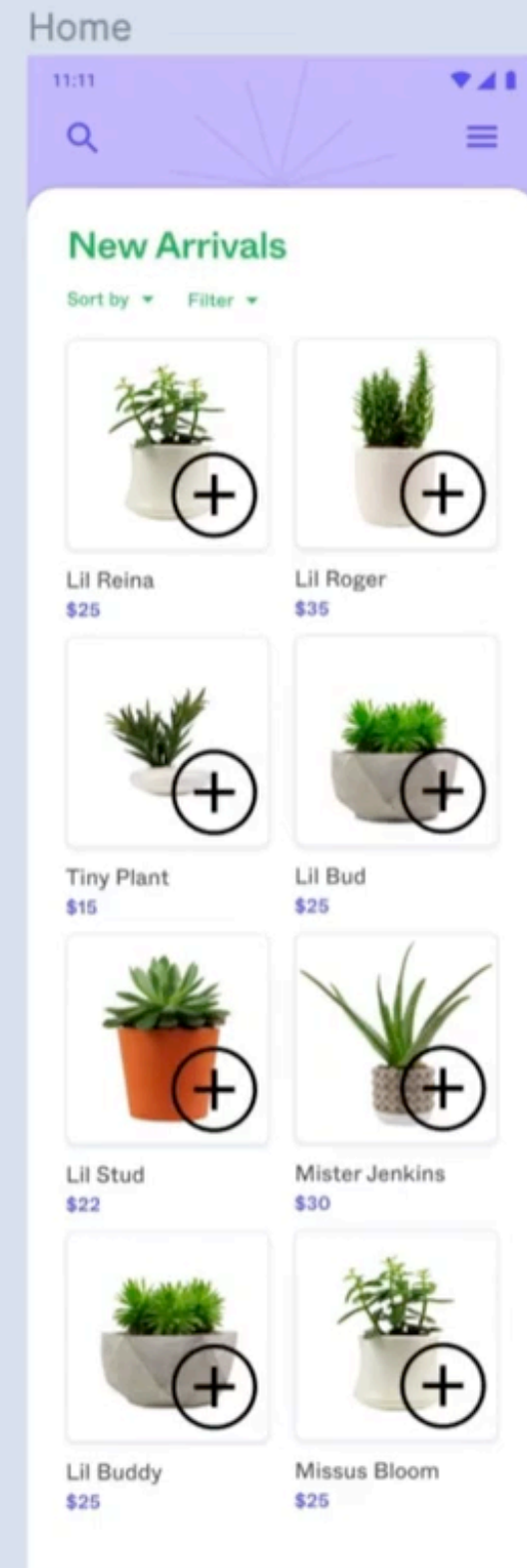
Вносит изменения в текстовый лист

# Что осталось?

- Поддержка гидрации
- Работа в режиме persistence
- Поддержка тестовых селекторов

# О чем поговорим?

- React Reconciler
- React → DOM
- **Figma & Figma API**
- React → Figma
- Итоги



Design Prototype Code

**Background**

D9E1F1 100%

**Local Styles**

Text Styles

- Ag Mobile - Header
- Ag Mobile - Small Text
- Ag Mobile - Body
- Ag Mobile - Body Links

Color Styles

- Lil Green
- Stone Grey
- Lil Coral
- Lil Red
- Lil Yellow
- Lil Lavender
- Lil Blue
- Lil Ultramarine

?

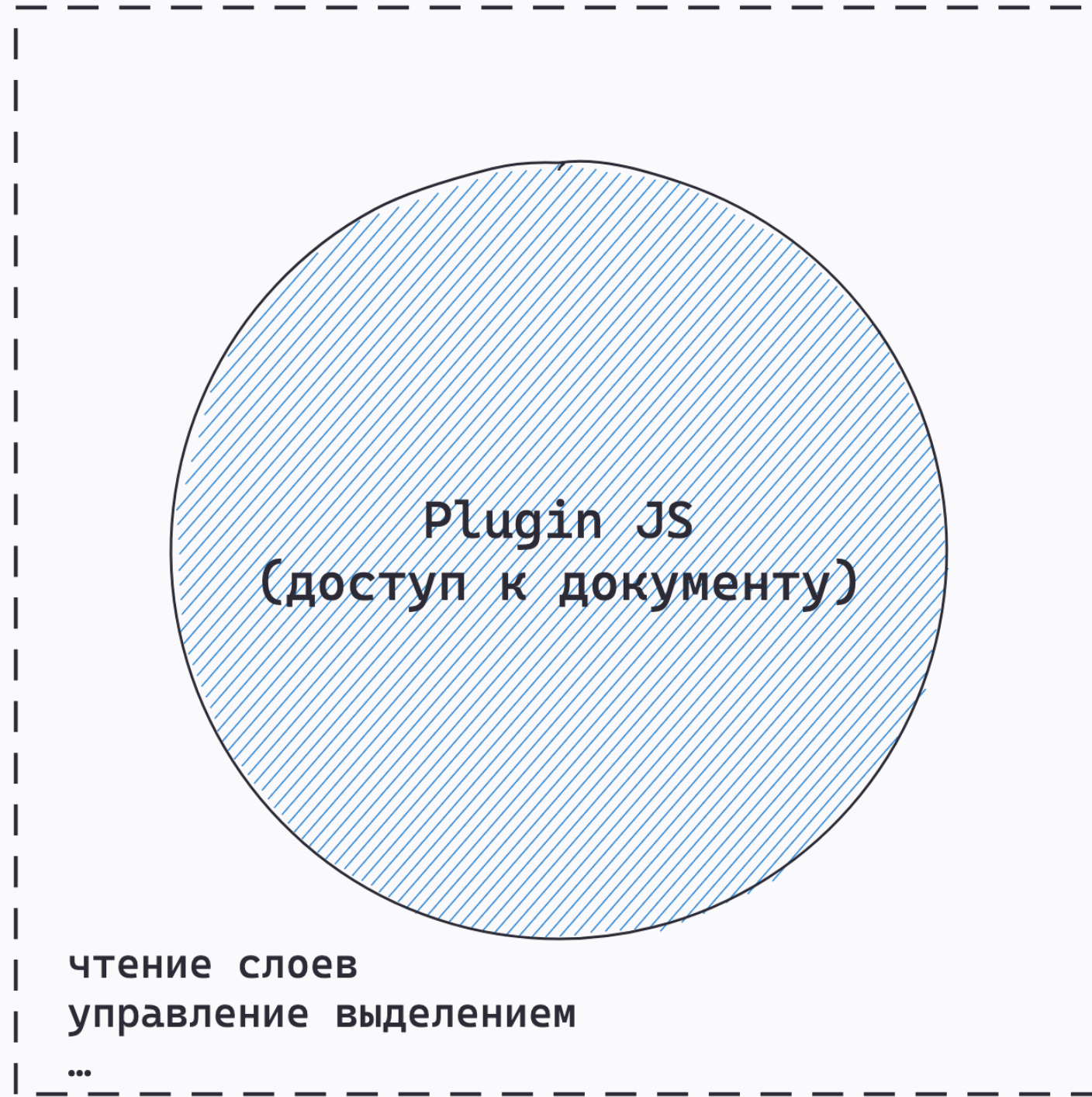
Luis, PM

Jada

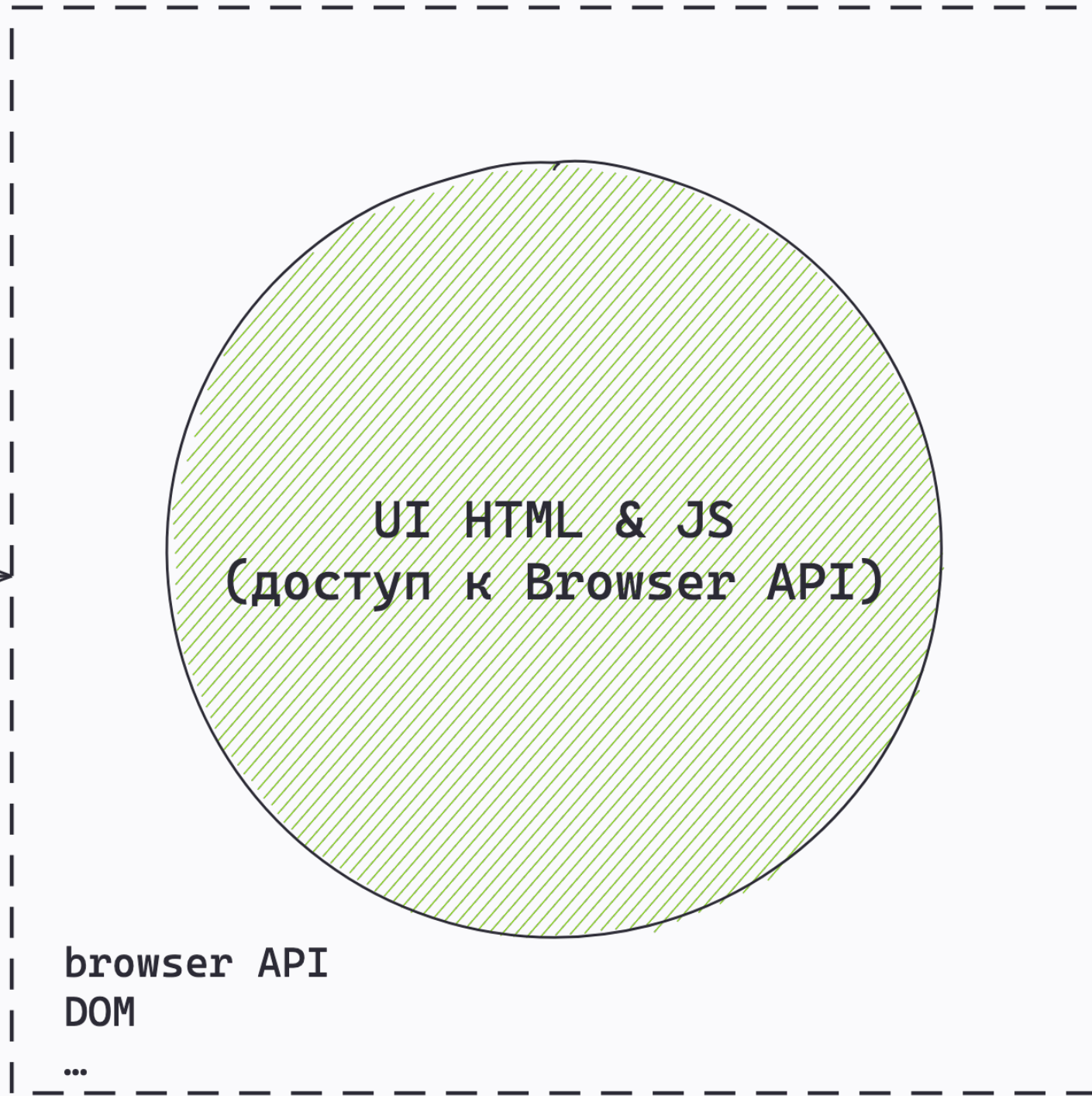


**Figma Plugin API**

sandbox



iframe



# **Figma API demo**

<https://www.figma.com/plugin-docs/intro/>



# Live code

- Отрисовываем React в Figma
- Компоненты
  - `<frame>`, `<text>`, `<rectangle>`
- Атрибуты
  - `x`, `y`, `width`, `height`, `backgroundColor`



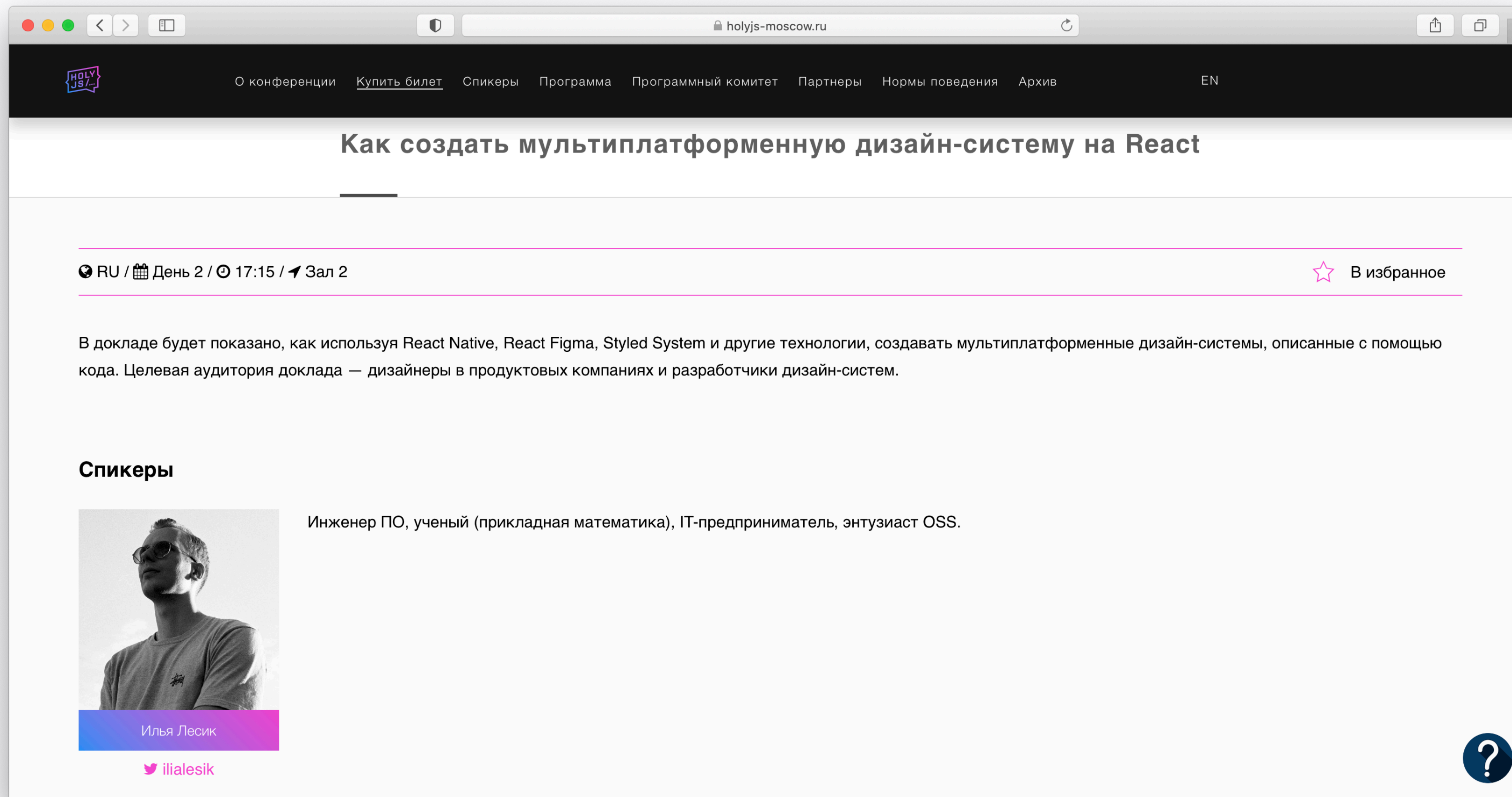
# А можно ли из Figma в React?

- [github.com/bernaferri/FigmaToCode](https://github.com/bernaferri/FigmaToCode)
- [overlay-tech.com](https://overlay-tech.com)
- Зар Захаров, Александр Каменяр — Figma to React: доставка дизайна в код
- [github.com/react-figma/PrimerDemo](https://github.com/react-figma/PrimerDemo)

# А можно ли из Figma в React?

- [github.com/bernafeerrari/FigmaToCode](https://github.com/bernafeerrari/FigmaToCode)
- [overlay-tech.com](https://overlay-tech.com)
- Зар Захаров, Александр Каменяр — Figma to React: доставка дизайна в код

# А можно универсально?



The screenshot shows a web browser window with the URL `holyjs-moscow.ru`. The page features a dark navigation bar with the HolyJS logo and menu items: "О конференции", "Купить билет", "Спикеры", "Программа", "Программный комитет", "Партнеры", "Нормы поведения", "Архив", and "EN". The main heading is "Как создать мультиплатформенную дизайн-систему на React". Below the heading, there is a horizontal bar with location and time information: "RU / 📅 День 2 / 🕒 17:15 / 📍 Зал 2" and a "В избранное" button with a star icon. The main text describes the talk: "В докладе будет показано, как используя React Native, React Figma, Styled System и другие технологии, создавать мультиплатформенные дизайн-системы, описанные с помощью кода. Целевая аудитория доклада — дизайнеры в продуктовых компаниях и разработчики дизайн-систем." Under the "Спикеры" section, there is a profile for Илья Лесик, described as a software engineer, scientist, IT entrepreneur, and OSS enthusiast. A profile picture of Илья Лесик is shown, along with his name and Twitter handle `@ilialesik`. A question mark icon is visible in the bottom right corner of the page.

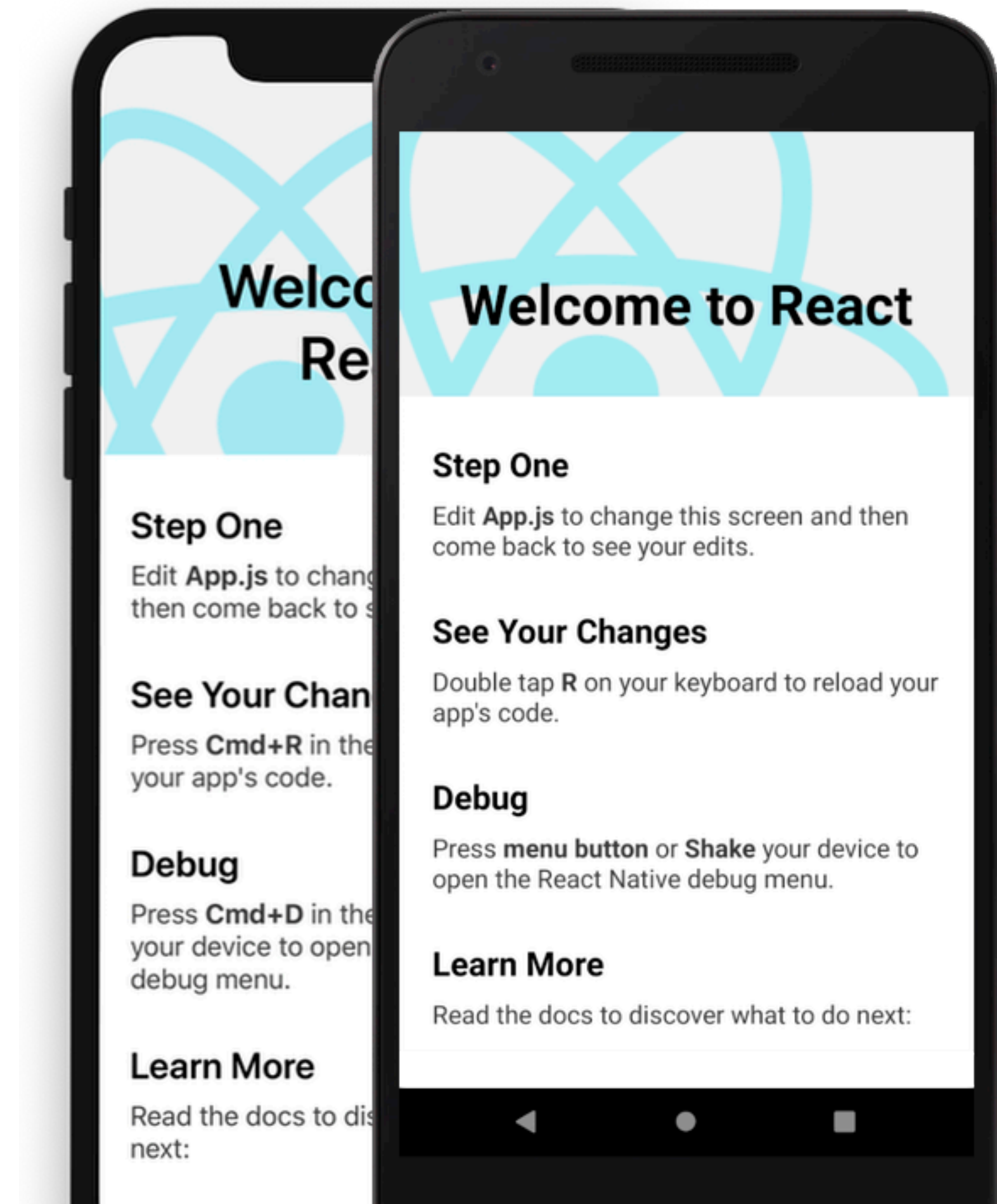
<https://holyjs-moscow.ru/2020/msk/talks/5udhzhshrgbzzg3r496hmge/>

# О чем поговорим?

- React Reconciler
- React → DOM
- Figma & Figma API
- React → Figma
- **Итоги**

# React Native

```
const WelcomeScreen = () => (  
  <View>  
    <Header  
      title="Welcome to React" />  
    <Text style={header}>  
      Step One  
    </Text>  
    <Text>  
      Edit App.js to change...  
    </Text>  
    ...  
  </View>  
);
```




# Ink

```
const Counter = () => {
  const [i, setI] = useState(0);

  useEffect(() => {
    setInterval(() => {
      setI((prev) => prev + 1);
    }, 100);
  }, []);

  return <Color green>
    {i} tests passed
  </Color>;
};
```



```
~/Projects/ink
λ node media/example
3 tests passed
```

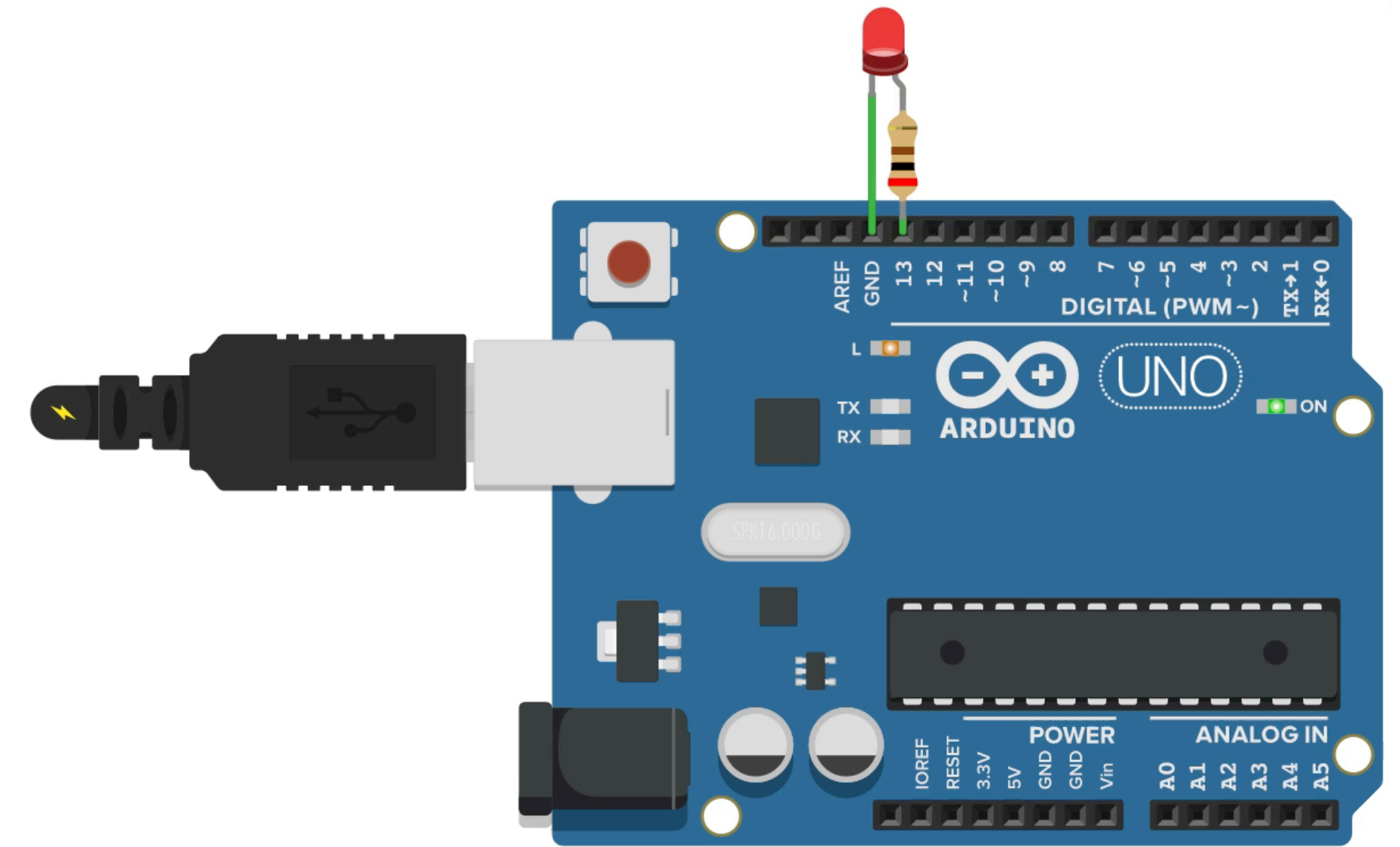
# React Hardware

```
const App = () => {
  const [led, setLed] = useState(false);

  useEffect(() => {
    setInterval(() => {
      setLed((prev) => !prev);
    }, 1000);
  }, []);

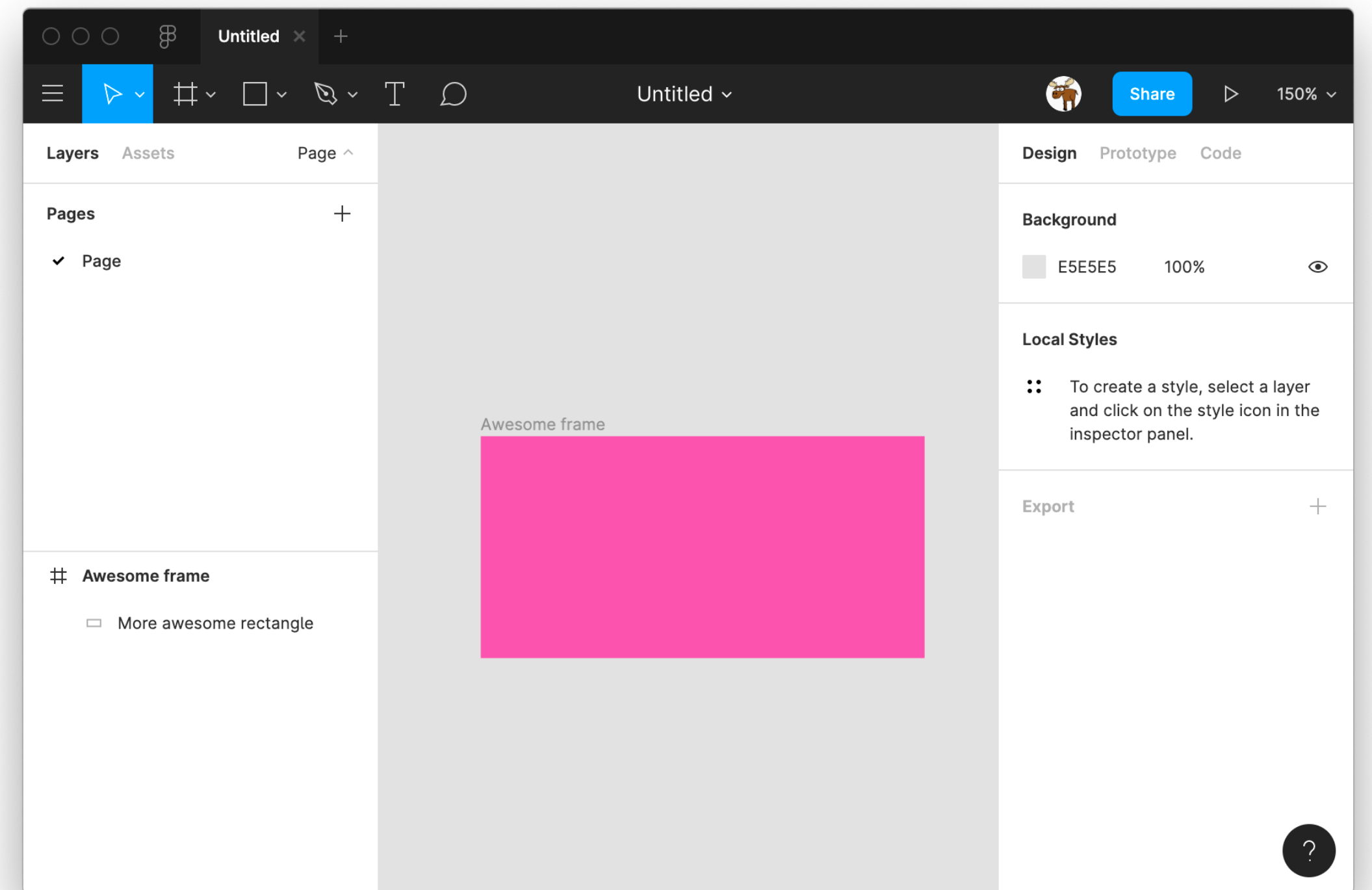
  return <Led
    pin={13}
    value={led ? 255 : 0}
  />;
};
```

```
const PORT = "/dev/tty.usbmodem1411";
ReactHardware.render(<App />, PORT);
```



# React Figma

```
const App = () => (  
  <Frame name="Awesome frame">  
    <Rectangle  
      name="More awesome rectangle"  
      width={200}  
      height={100}  
      backgroundColor="#dd55aa"  
    />  
  </Frame>  
);
```





# React PDF

```
const MyDocument = () => (  
  <Document>  
    <Page style={styles.body}>  
      <Text  
        style={styles.header}  
        fixed>  
        ~ Created with react-pdf ~  
      </Text>  
      ...  
      <Image  
        style={styles.image}  
        src="/images/quijote1.jpg"  
      />  
      ...  
    </Page>  
  </Document>  
) ;
```

~ Created with react-pdf ~

## Don Quijote de la Mancha

Miguel de Cervantes



### Capítulo I: Que trata de la condición y ejercicio del famoso hidalgo D. Quijote de la Mancha

En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero, adarga antigua, rocín flaco y galgo corredor. Una olla de algo más vaca que carnero, salpicón las más noches, duelos y quebrantos los sábados, lentejas los viernes, algún palomino de añadidura los domingos, consumían las tres partes de su hacienda. El resto della concluían sayo de velarte, calzas de velludo para las fiestas con sus pantuflos de lo mismo, los días de entre semana se honraba con su vellori de lo más fino. Tenía en su casa una ama que pasaba de los cuarenta, y una sobrina que no llegaba a los veinte, y un mozo de campo y plaza, que así ensillaba el rocín como tomaba la podadera. Frisaba la edad de nuestro hidalgo con los cincuenta años, era de complexión recia, seco de carnes, enjuto de rostro; gran madrugador y amigo de la caza. Quieren decir que tenía el sobrenombre de Quijada o Quesada (que en esto hay alguna diferencia en los autores que deste caso escriben), aunque por conjeturas verosímiles se deja entender que se llama Quijana; pero esto importa poco a nuestro cuento; basta que en la narración dél no se salga un punto de la verdad

Es, pues, de saber, que este sobredicho hidalgo, los ratos que estaba ocioso (que eran los más del año) se daba a leer libros de caballerías con tanta afición y gusto, que olvidó casi de todo punto el ejercicio de la caza, y aun la administración de su hacienda; y llegó a tanto su curiosidad y desatino en esto, que vendió muchas hanegas de tierra de sembradura, para comprar libros de caballerías en que leer; y así llevó a su casa todos cuantos pudo haber dellos; y de todos ningunos le parecían tan bien como los que compuso el famoso Feliciano de Silva: porque la claridad de su prosa, y aquellas intrincadas razones suyas, le parecían

😊 Всегда последние фишки React

😊 Определяется только средозависимая часть

😊 React DevTools

😊 Hot Reloading

🧠 Дает лучшее понимание React

🔥 Весело

☹️ Мало документации

☹️ “Use it at your own risk”

☹️ Иногда интегрироваться с host-environment сложно



[bit.ly/react-reconciler-holy](https://bit.ly/react-reconciler-holy)

 [github.com/losyear](https://github.com/losyear)

 [LosYear](https://t.me/LosYear)

 [twitter.com/losyear](https://twitter.com/losyear)