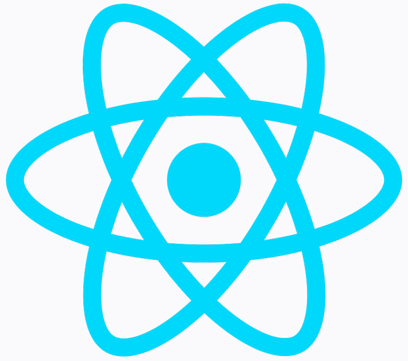


React Reconciler

Как написать собственный рендерер

HolyJS Moscow 2020

Ярослав Лосев

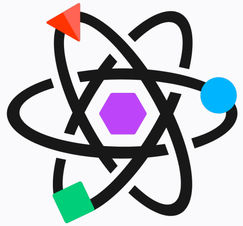
Redis, креативное агентство

LosYear

github.com/losyear

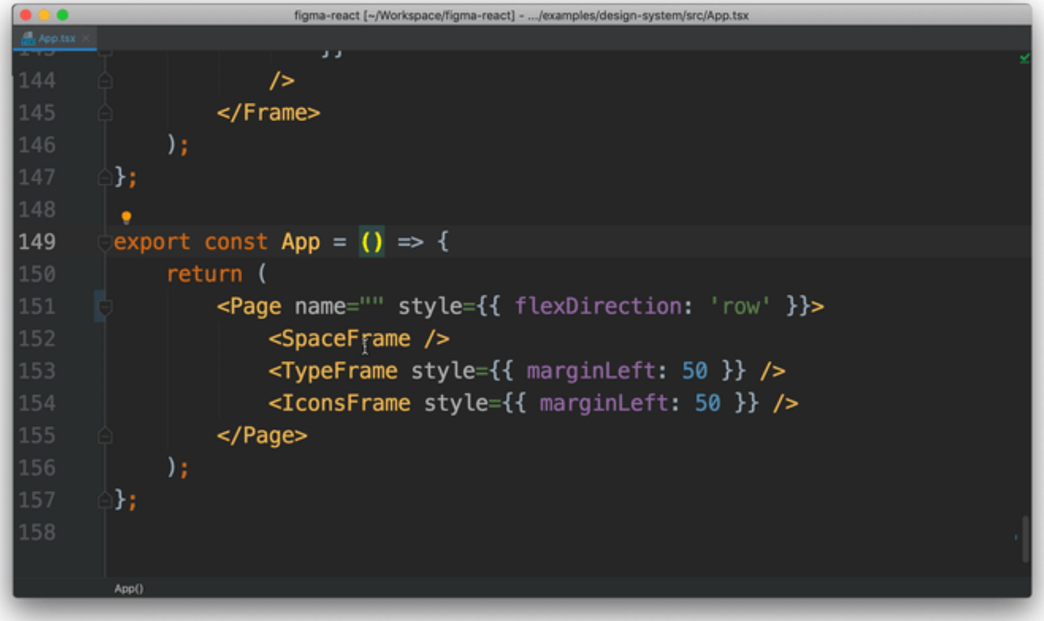
twitter.com/losyear

React Figma



react-figma.dev

github.com/react-figma



@ilialesik

@losyear

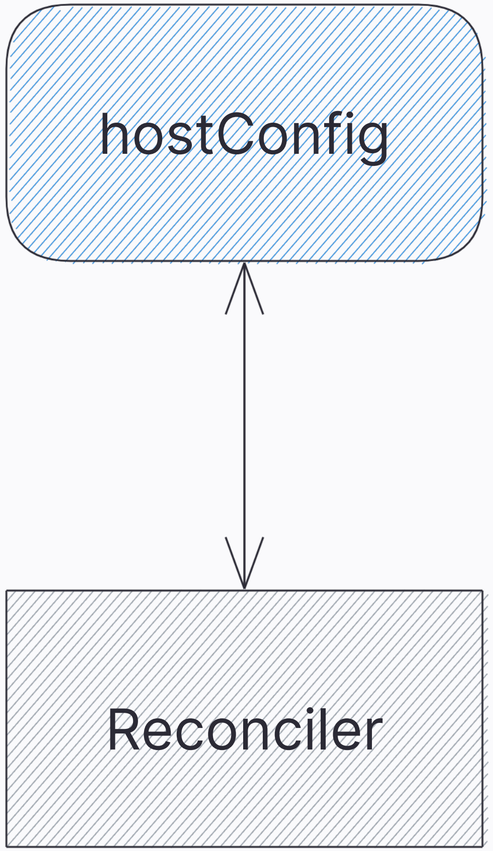
О чем поговорим?

- 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.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);
};
```

```
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, ...

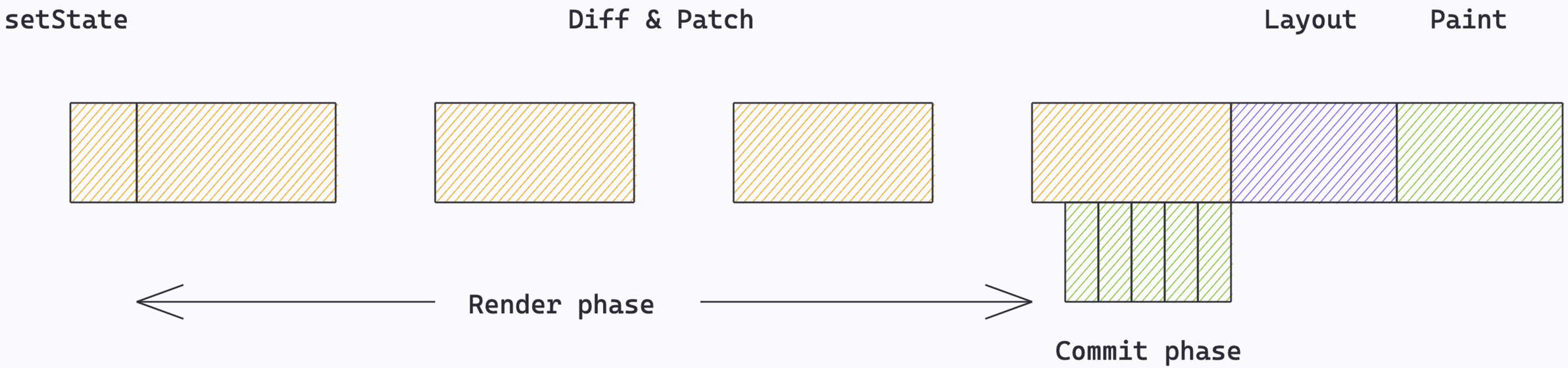
Режимы работы

supportsPersistence: true

Клонирует и заменяет ноды:

- replaceContainerChildren, createContainerChildSet
- cloneInstance

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


setState
Diff & Patch
Layout
Paint
Render phase
Commit phase

Live code

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

Первоначальная отрисовка

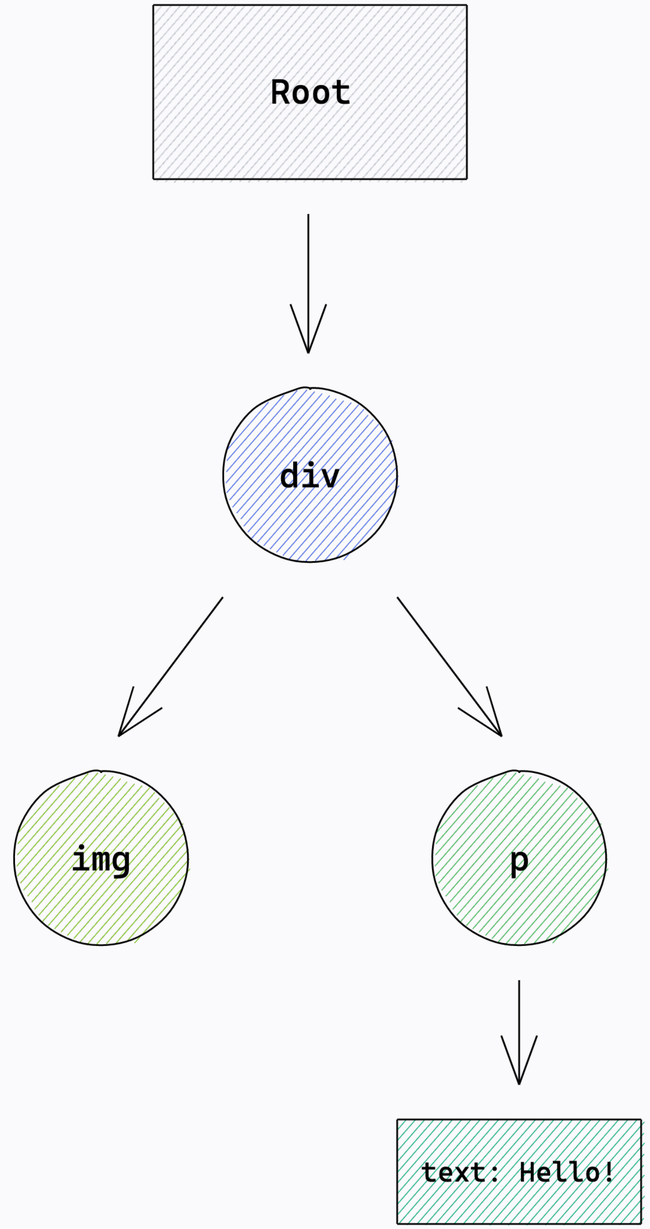
```
const InitialRenderExample = () => (
    <div className="initial-render-example">
        <img src="/public/1.png" />
        <p>Hello!</p>
    </div>
);
```

Hello!

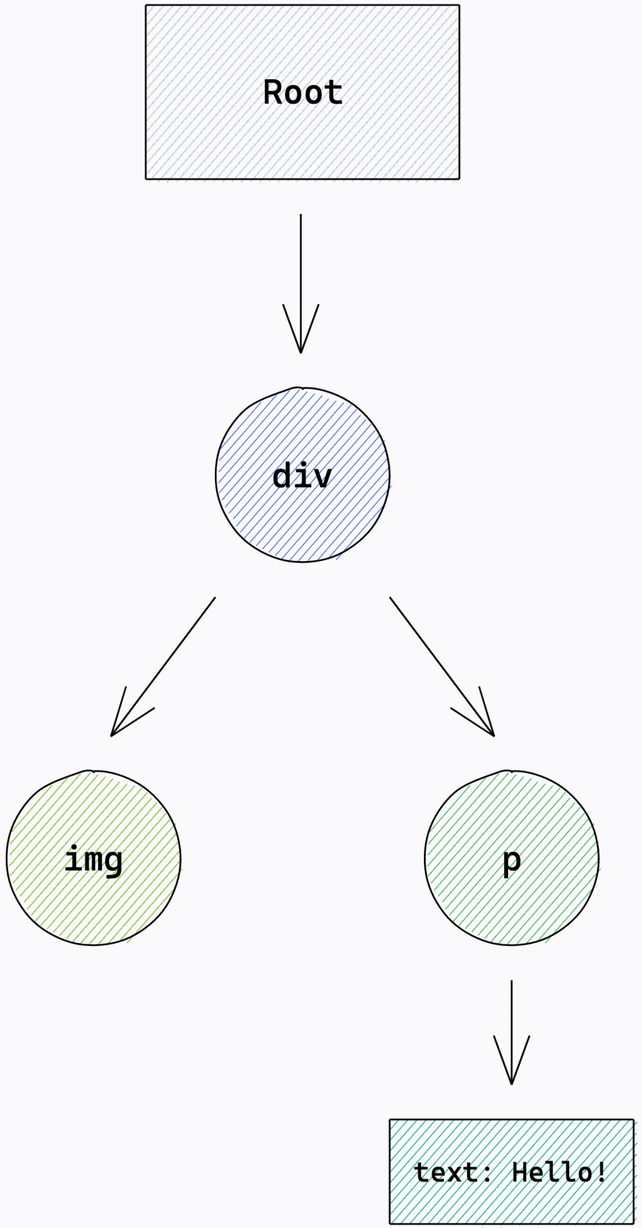
```jsx
const InitialRenderExample = () => (
    <div className="initial-render-example">
        <img src="/public/1.png" />
        <p>Hello!</p>
    </div>
);
```

Hello!

```jsx
const InitialRenderExample = () => (
    <div className="initial-render-example">
        <img src="/public/1.png" />
        <p>Hello!</p>
    </div>
);
```



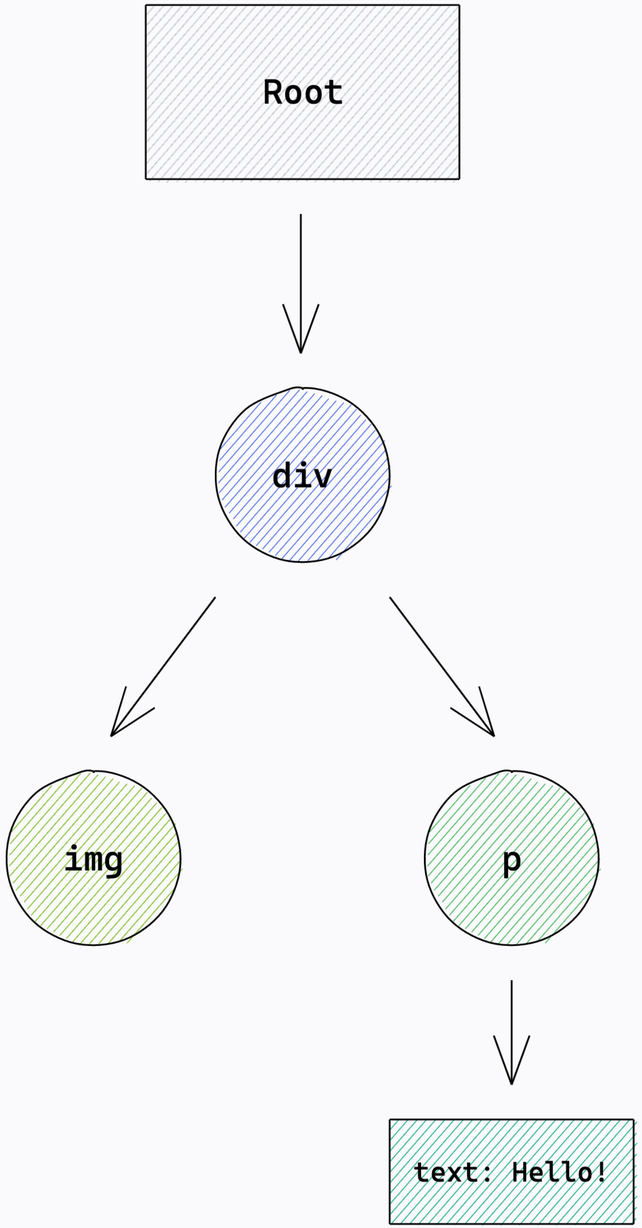
1. getRootHostContext



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

Передает первоначальный контекст между вызовами методов

1. getRootHostContext

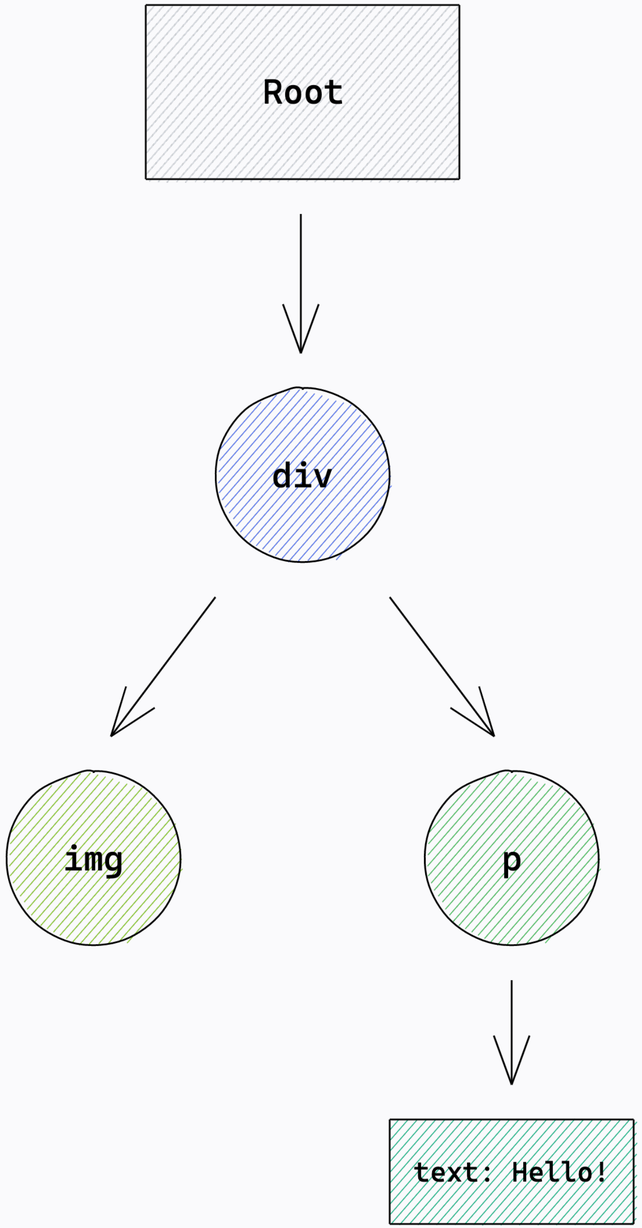


2. getChildHostContext

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

Передает контекст от родителя к детям

1. getRootHostContext



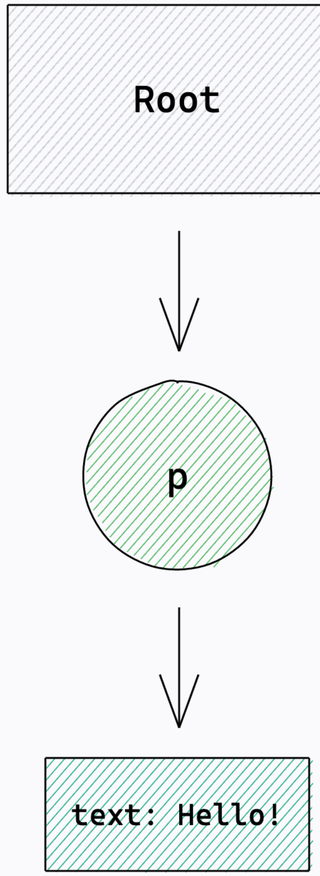
2. getChildHostContext

3. shouldSetTextContent

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

Нужно ли обрабатывать текст как отдельные ноды

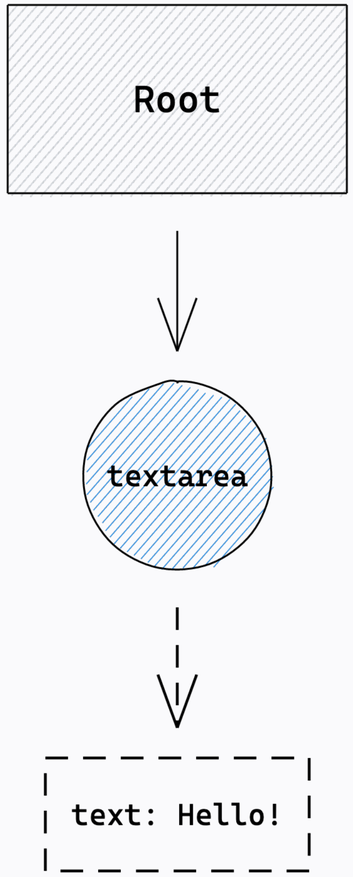
shouldSetTextContent



setTextContent: false

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

paragraph.appendChild(helloText);
```

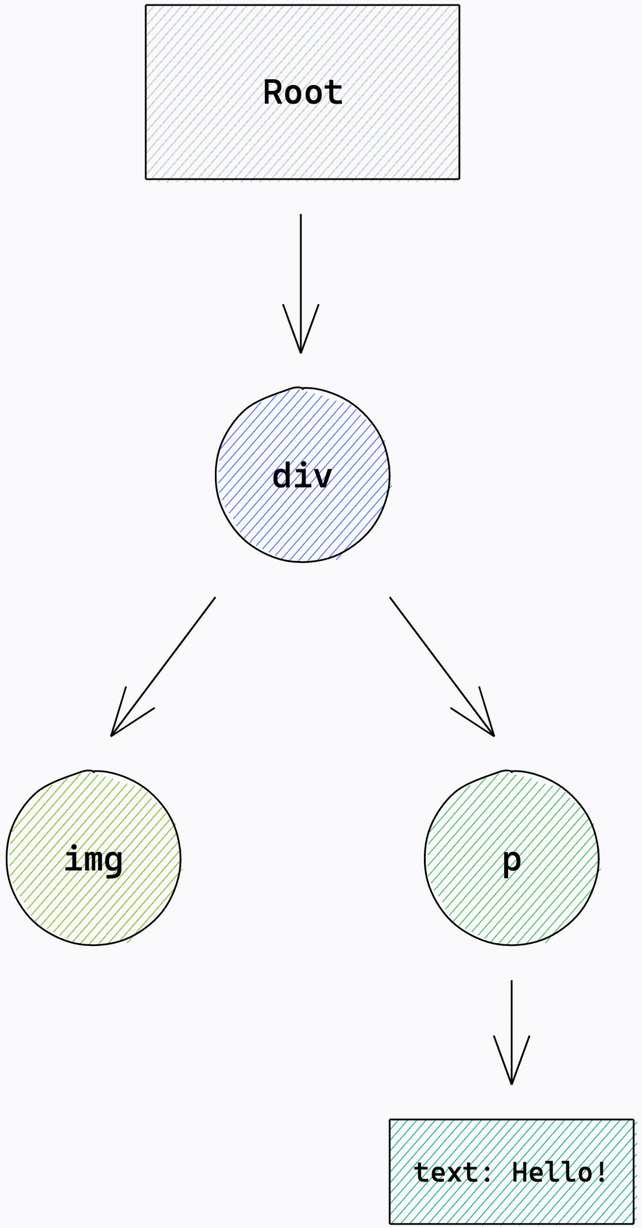
shouldSetTextContent



setTextContent: true

```
const textarea = document.createElement('textarea');

textarea.value = "Hello!";
```

1. getRootHostContext



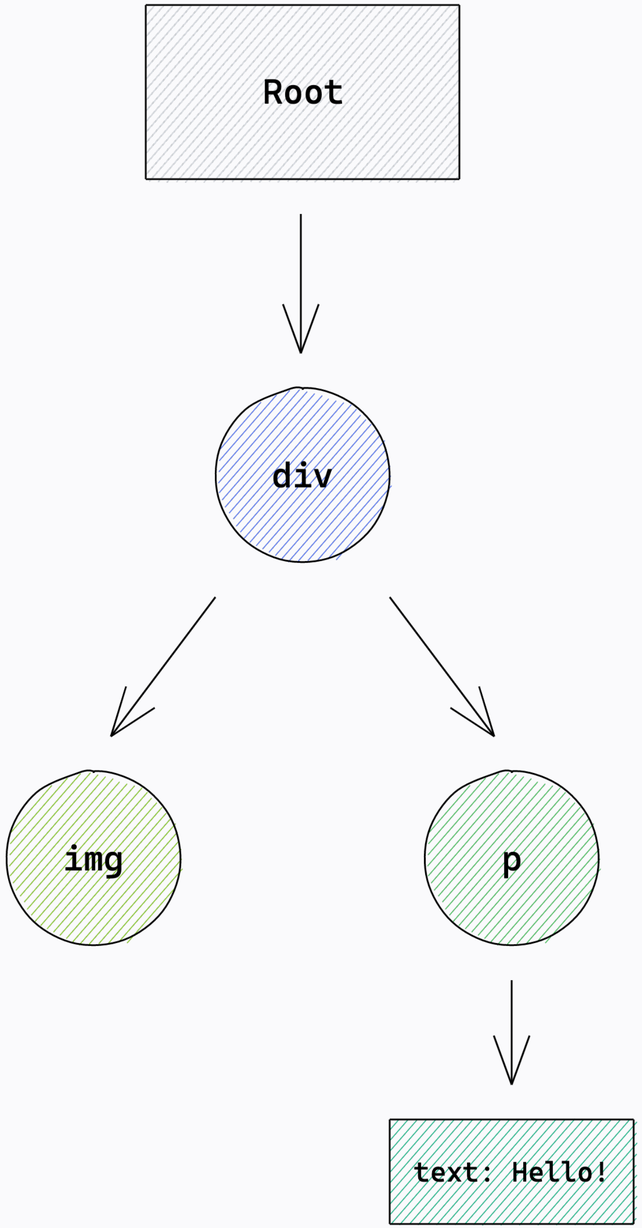
2. getChildHostContext

3. shouldSetTextContent

4. getChildHostContext

5. shouldSetTextContent

1. getRootHostContext



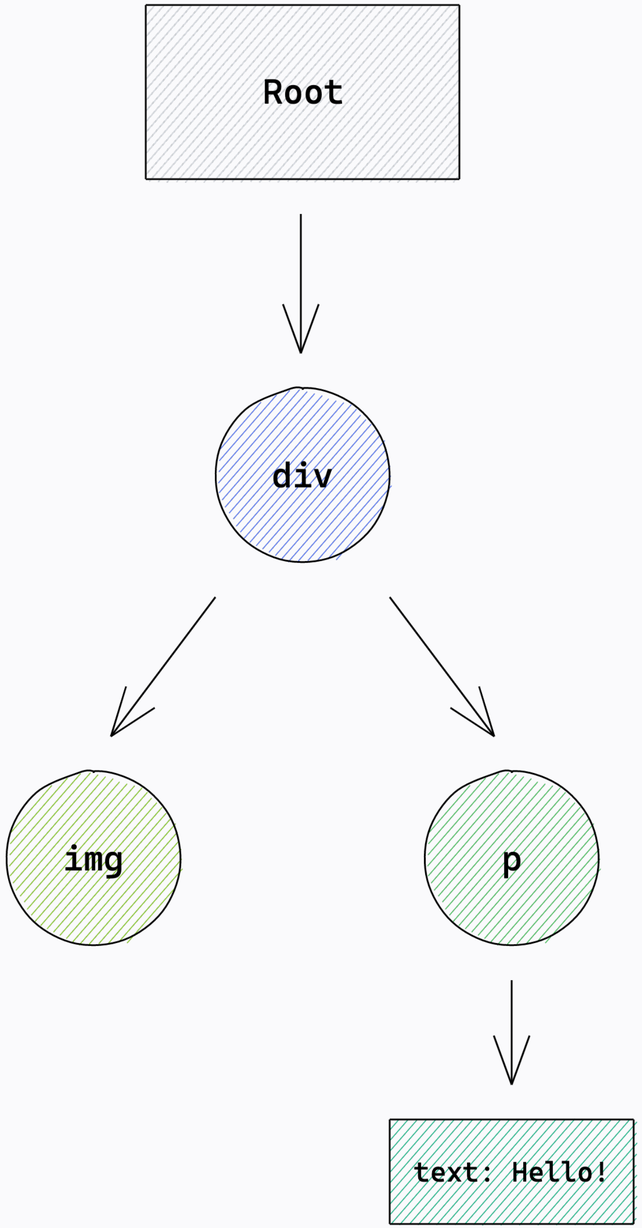
2. getChildHostContext
3. shouldSetTextContent

4. getChildHostContext
5. shouldSetTextContent
6. **createInstance**

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

Создает представление хост-компонента в среде

1. getRootHostContext



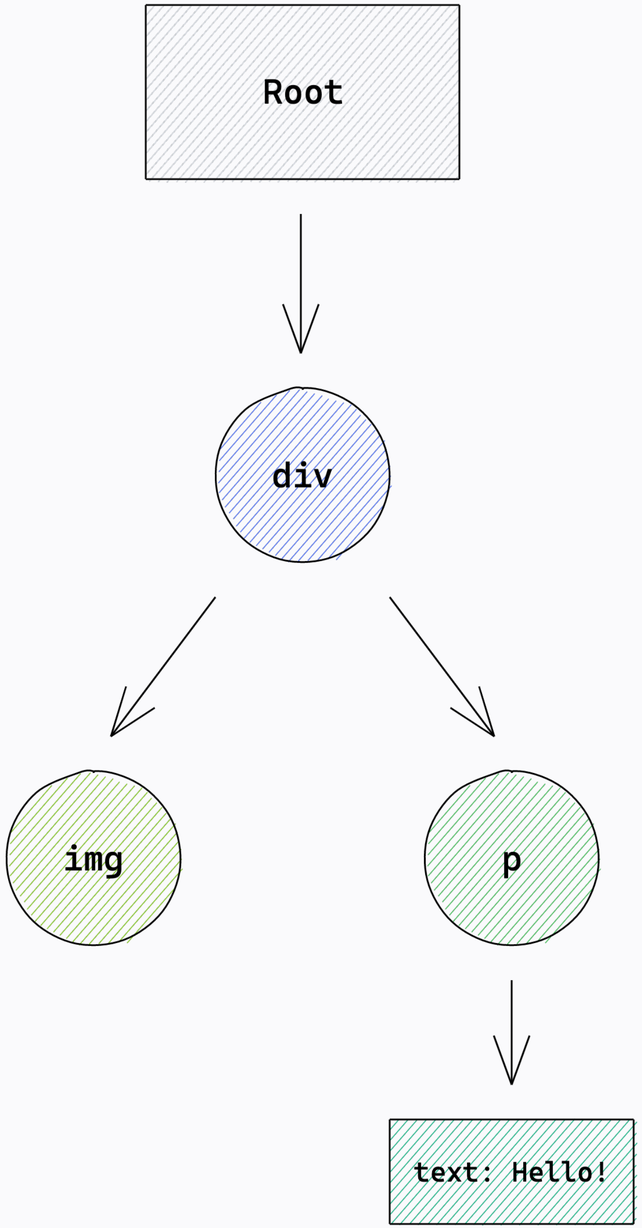
2. getChildHostContext
3. shouldSetTextContent

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

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

Совершает дополнительные действия после создания узла

1. getRootHostContext

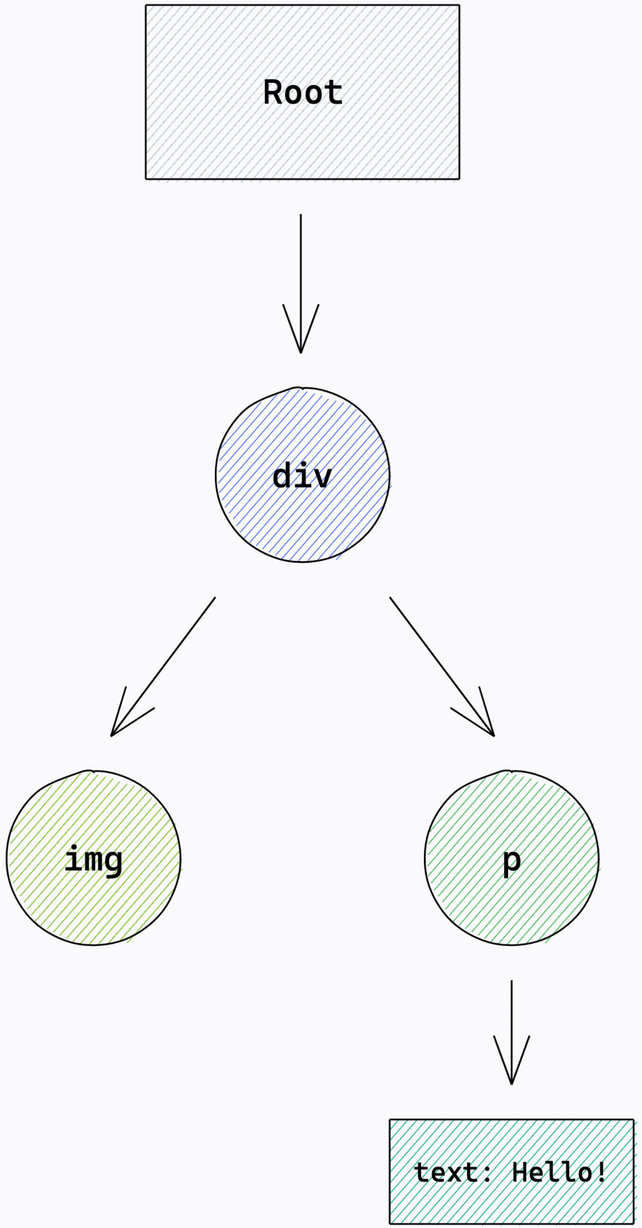


2. getChildHostContext
3. shouldSetTextContent

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

8. getChildHostContext
9. shouldSetTextContent

1. getRootHostContext



2. getChildHostContext

3. shouldSetTextContent

4. getChildHostContext

5. shouldSetTextContent

6. createInstance

7. finalizeInitialChildren

8. getChildHostContext

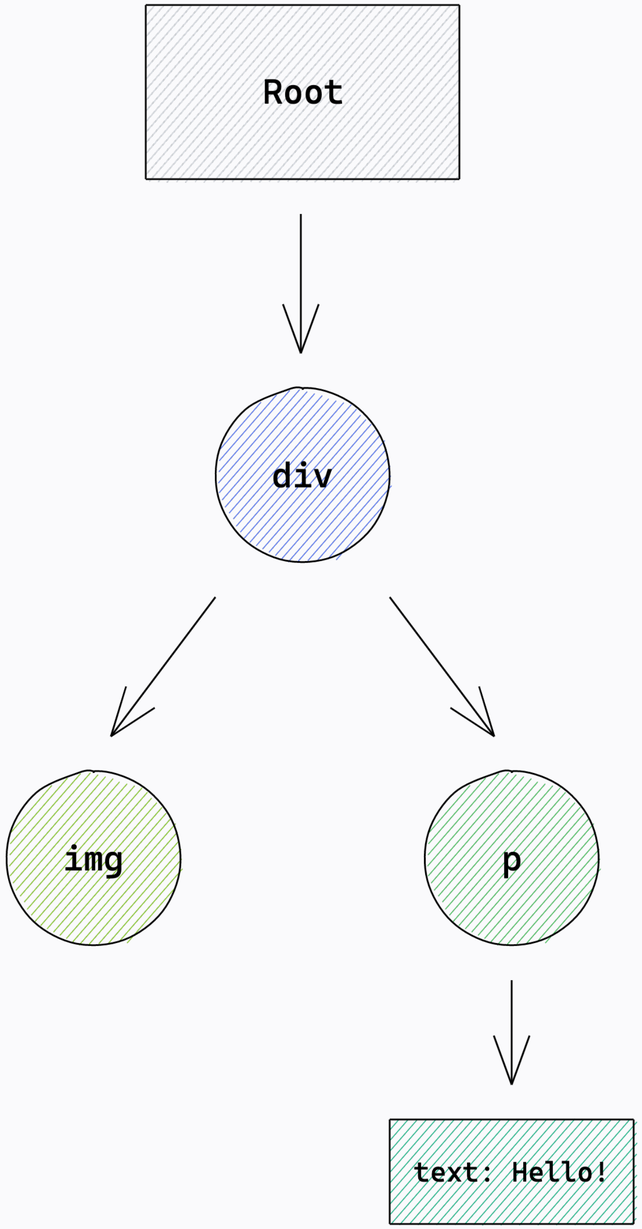
9. shouldSetTextContent

10. createTextInstance

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

Создает представление текстового листа

1. getRootHostContext



2. getChildHostContext

3. shouldSetTextContent

4. getChildHostContext

5. shouldSetTextContent

6. createInstance

7. finalizeInitialChildren

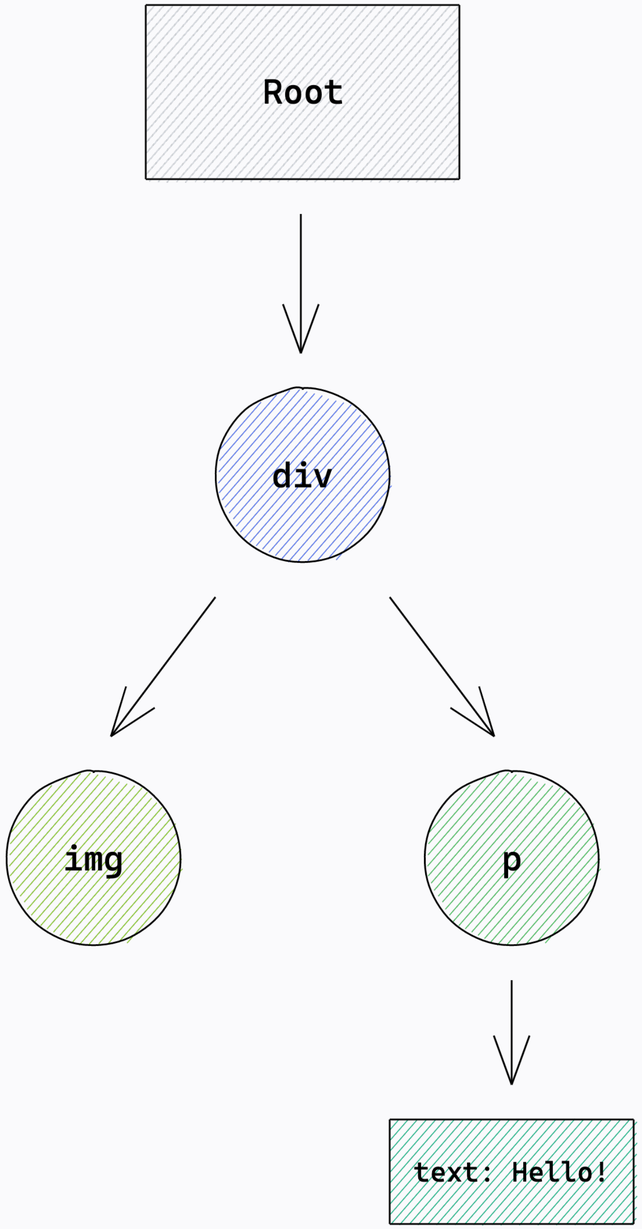
8. getChildHostContext

9. shouldSetTextContent

11. createInstance

10. createTextInstance

1. getRootHostContext



2. getChildHostContext

3. shouldSetTextContent

4. getChildHostContext

5. shouldSetTextContent

6. createInstance

7. finalizeInitialChildren

8. getChildHostContext

9. shouldSetTextContent

10. createTextInstance

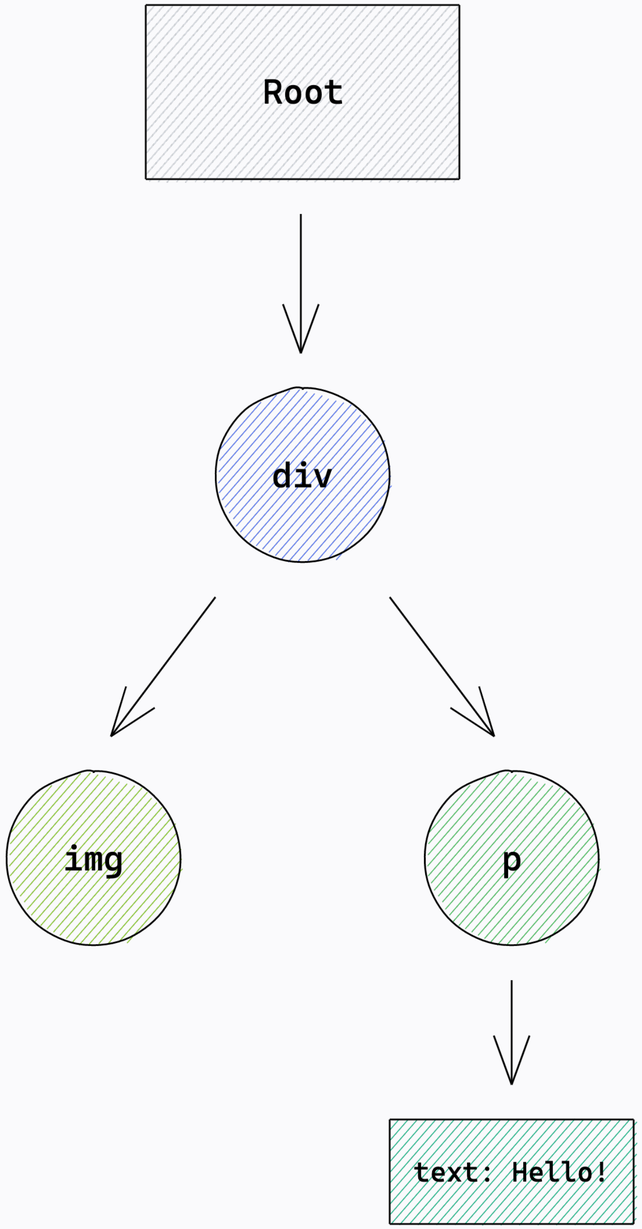
11. createInstance

12. appendInitialChild

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

Добавляет вершину родителю при первоначальной отрисовке

1. getRootHostContext



2. getChildHostContext

3. shouldSetTextContent

4. getChildHostContext

5. shouldSetTextContent

6. createInstance

7. finalizeInitialChildren

8. getChildHostContext

9. shouldSetTextContent

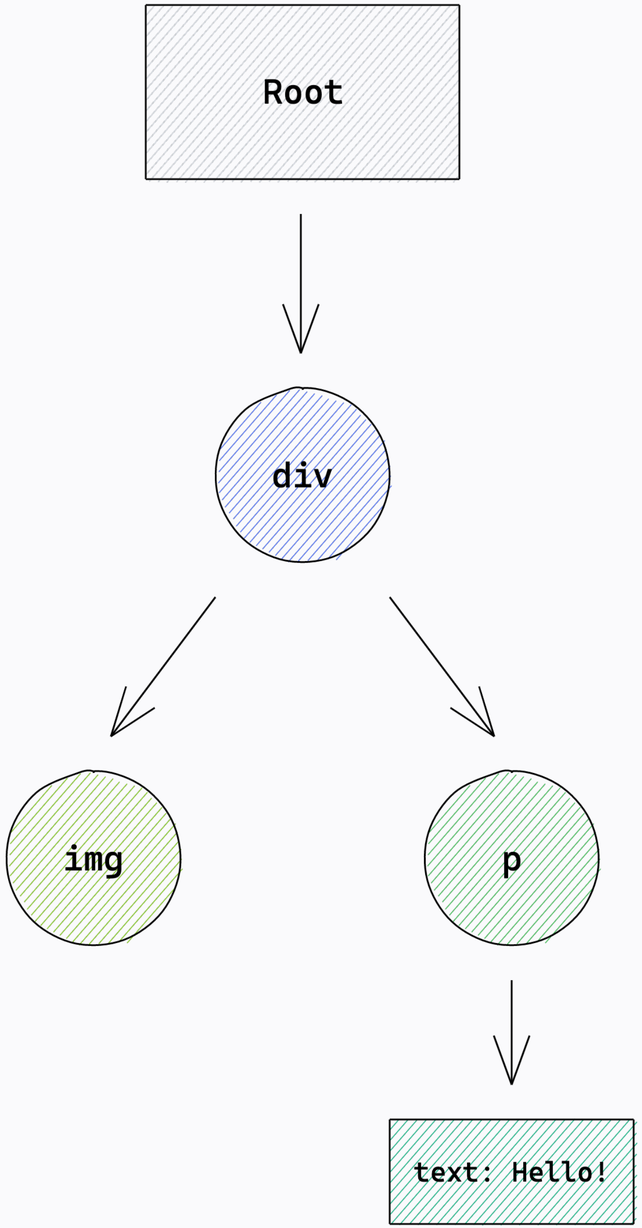
10. createTextInstance

11. createInstance

12. appendInitialChild

13. finalizeInitialChildren

1. getRootHostContext



2. getChildHostContext

3. shouldSetTextContent

14. createInstance

15, 16. appendInitialChild

17. finalizeInitialChildren

4. getChildHostContext

5. shouldSetTextContent

6. createInstance

7. finalizeInitialChildren

8. getChildHostContext

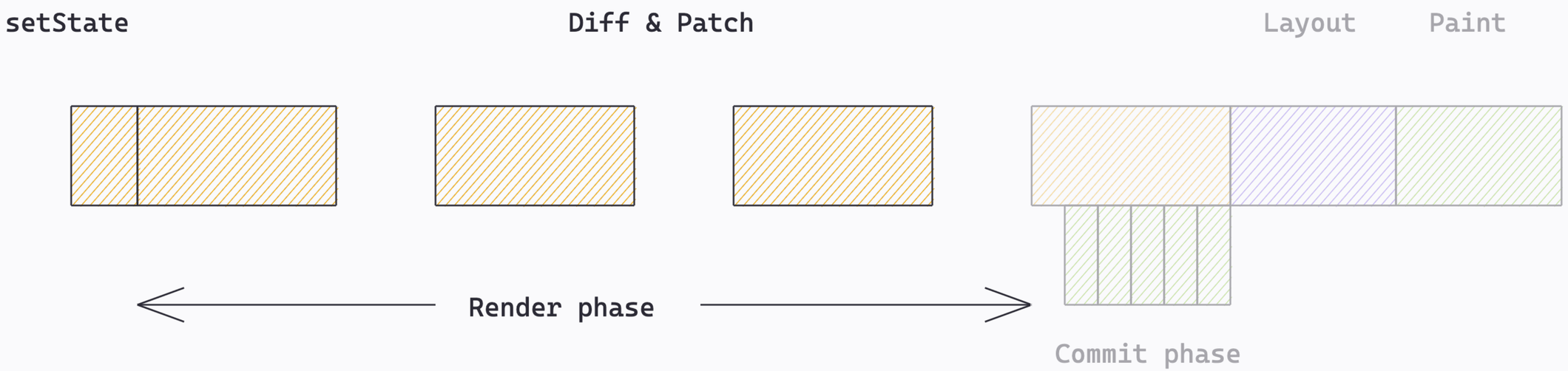
9. shouldSetTextContent

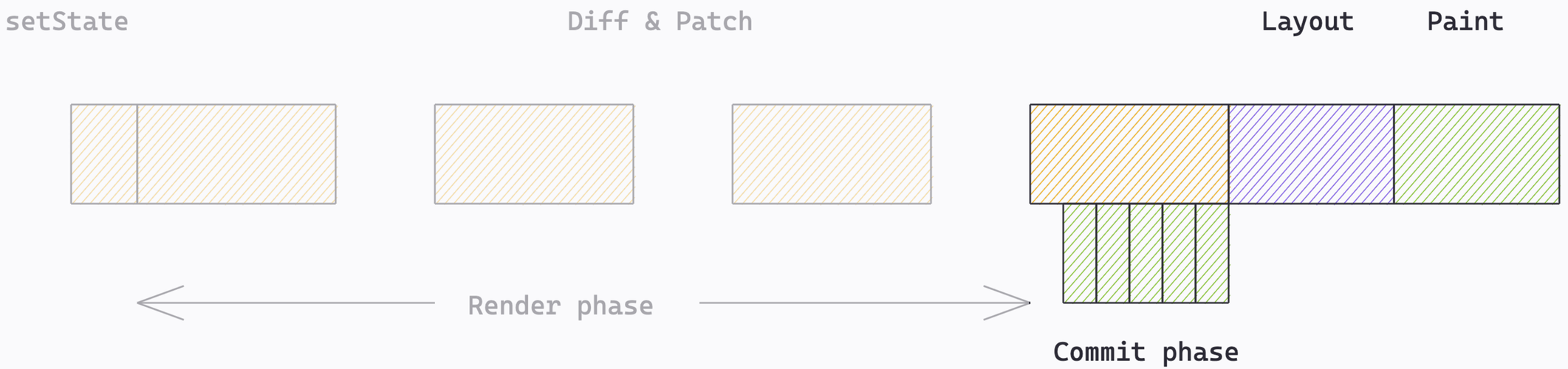
11. createInstance

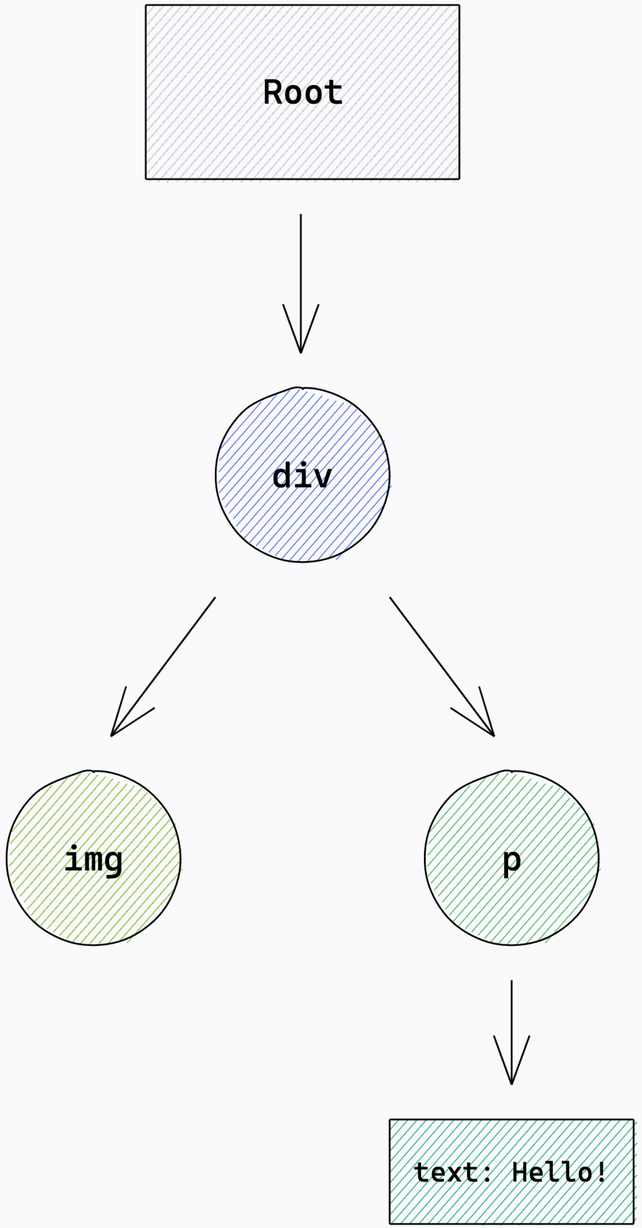
12. appendInitialChild

13. finalizeInitialChildren

10. createTextInstance


setState
Diff & Patch
Layout
Paint
Render phase
Commit phase


setState
Diff & Patch
Layout
Paint
Render phase
Commit phase



1. getRootHostContext

18. prepareForCommit

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

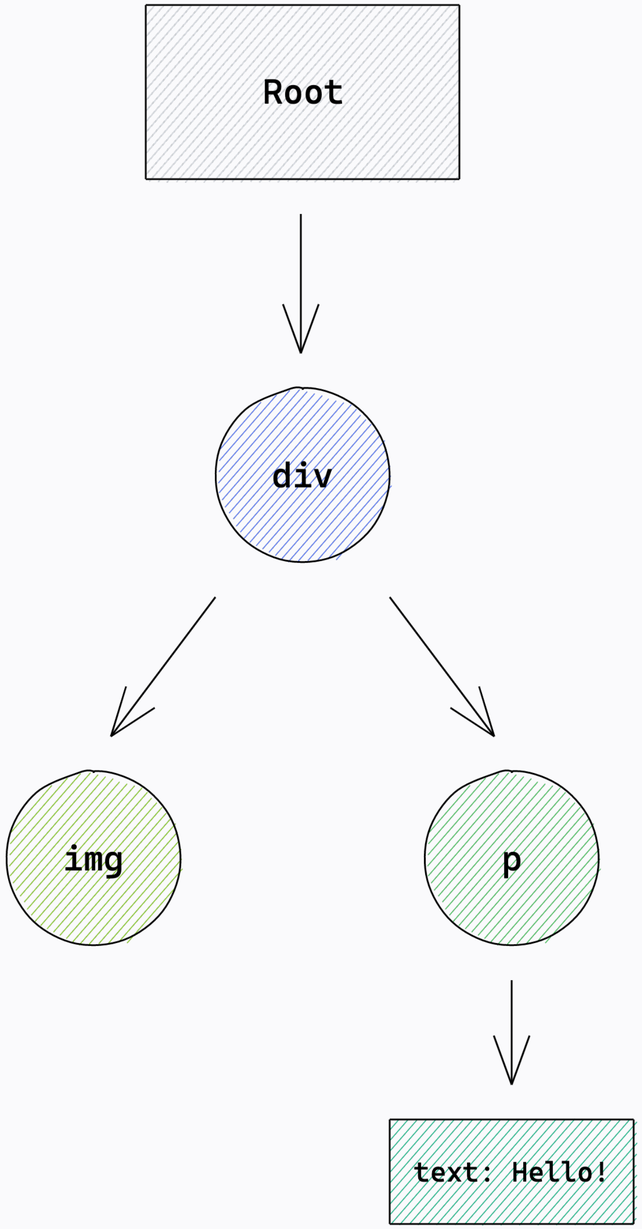
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren

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

10. createTextInstance

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

Осуществляет подготовку к внесению изменений



1. getRootHostContext

18. prepareForCommit

19. appendChildToContainer

2. getChildHostContext

3. shouldSetTextContent

14. createInstance

15, 16. appendInitialChild

17. finalizeInitialChildren

4. getChildHostContext

5. shouldSetTextContent

6. createInstance

7. finalizeInitialChildren

8. getChildHostContext

9. shouldSetTextContent

11. createInstance

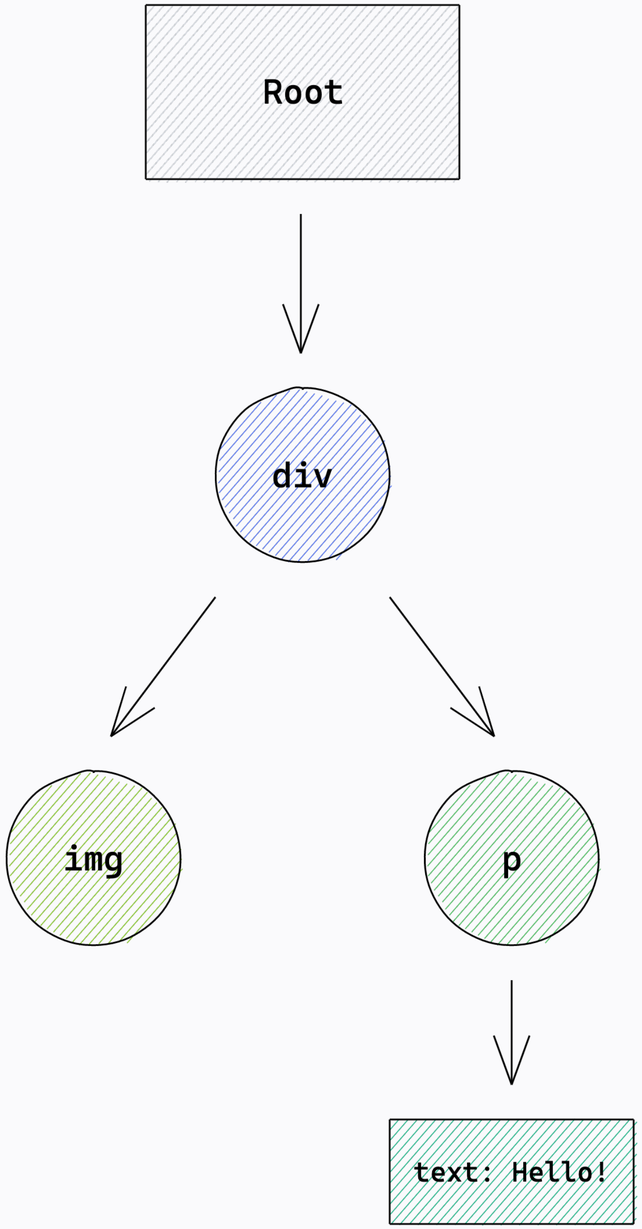
12. appendInitialChild

13. finalizeInitialChildren

10. createTextInstance

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

Добавляет ребенка в представление



1. getRootHostContext

18. prepareForCommit

19. appendChildToContainer

20. resetAfterCommit

2. getChildHostContext

3. shouldSetTextContent

14. createInstance

15, 16. appendInitialChild

17. finalizeInitialChildren

4. getChildHostContext

5. shouldSetTextContent

6. createInstance

7. finalizeInitialChildren

8. getChildHostContext

9. shouldSetTextContent

11. createInstance

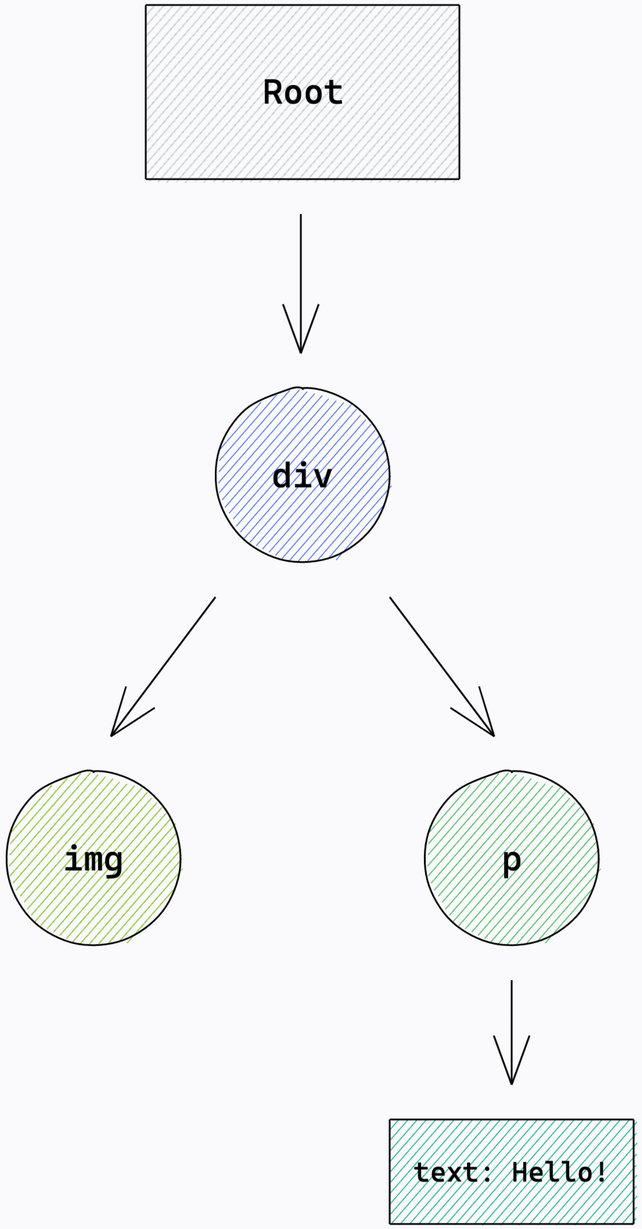
12. appendInitialChild

13. finalizeInitialChildren

10. createTextInstance

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

Осуществляет дополнительные действия после внесения изменений



1. getRootHostContext

18. prepareForCommit

19. appendChildToContainer

20. resetAfterCommit

2. getChildHostContext

3. shouldSetTextContent

14. createInstance

15, 16. appendInitialChild

17. finalizeInitialChildren

23. commitMount

4. getChildHostContext

5. shouldSetTextContent

6. createInstance

7. finalizeInitialChildren

21. commitMount

8. getChildHostContext

9. shouldSetTextContent

11. createInstance

12. appendInitialChild

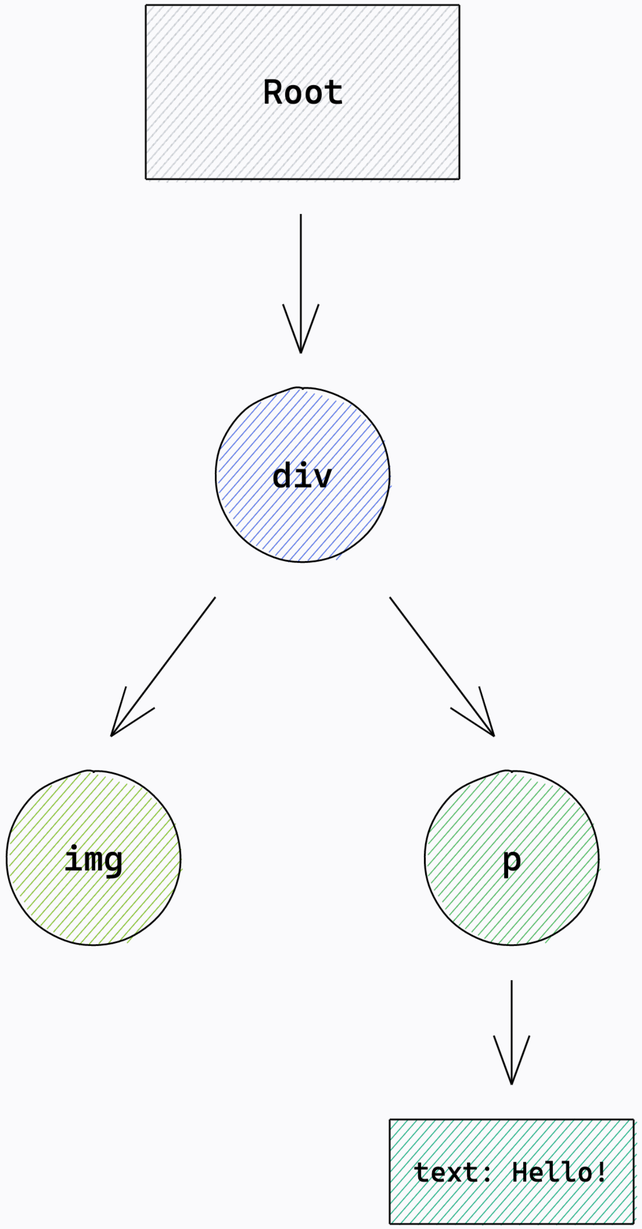
13. finalizeInitialChildren

22. commitMount

10. createTextInstance

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

Дополнительная обработка узлов после их отрисовки



1. getRootHostContext

18. prepareForCommit

19. appendChildToContainer

20. resetAfterCommit

2. getChildHostContext

3. shouldSetTextContent

14. createInstance

15, 16. appendInitialChild

17. finalizeInitialChildren

23. commitMount

4. getChildHostContext

5. shouldSetTextContent

6. createInstance

7. finalizeInitialChildren

21. commitMount

8. getChildHostContext

9. shouldSetTextContent

11. createInstance

12. appendInitialChild

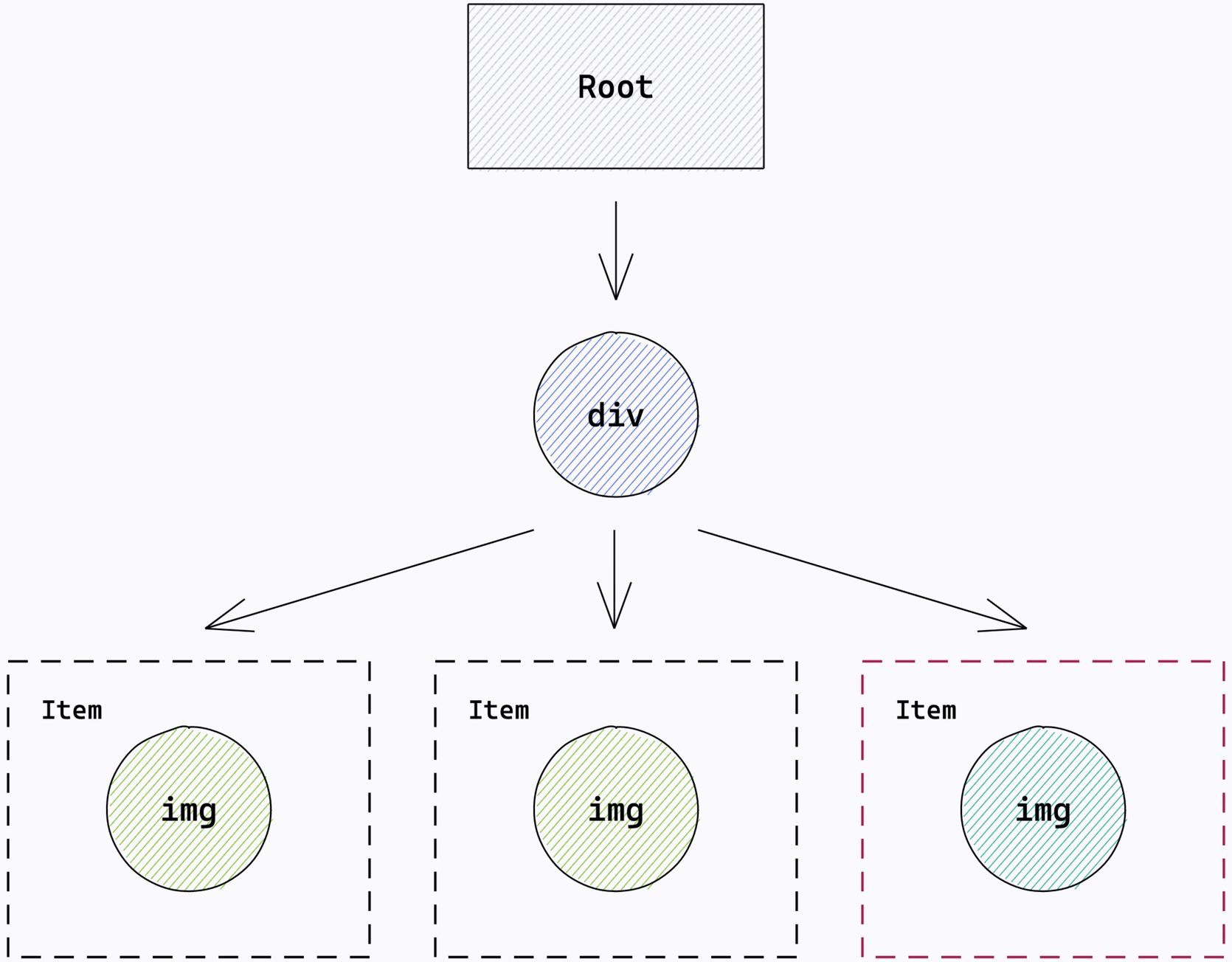
13. finalizeInitialChildren

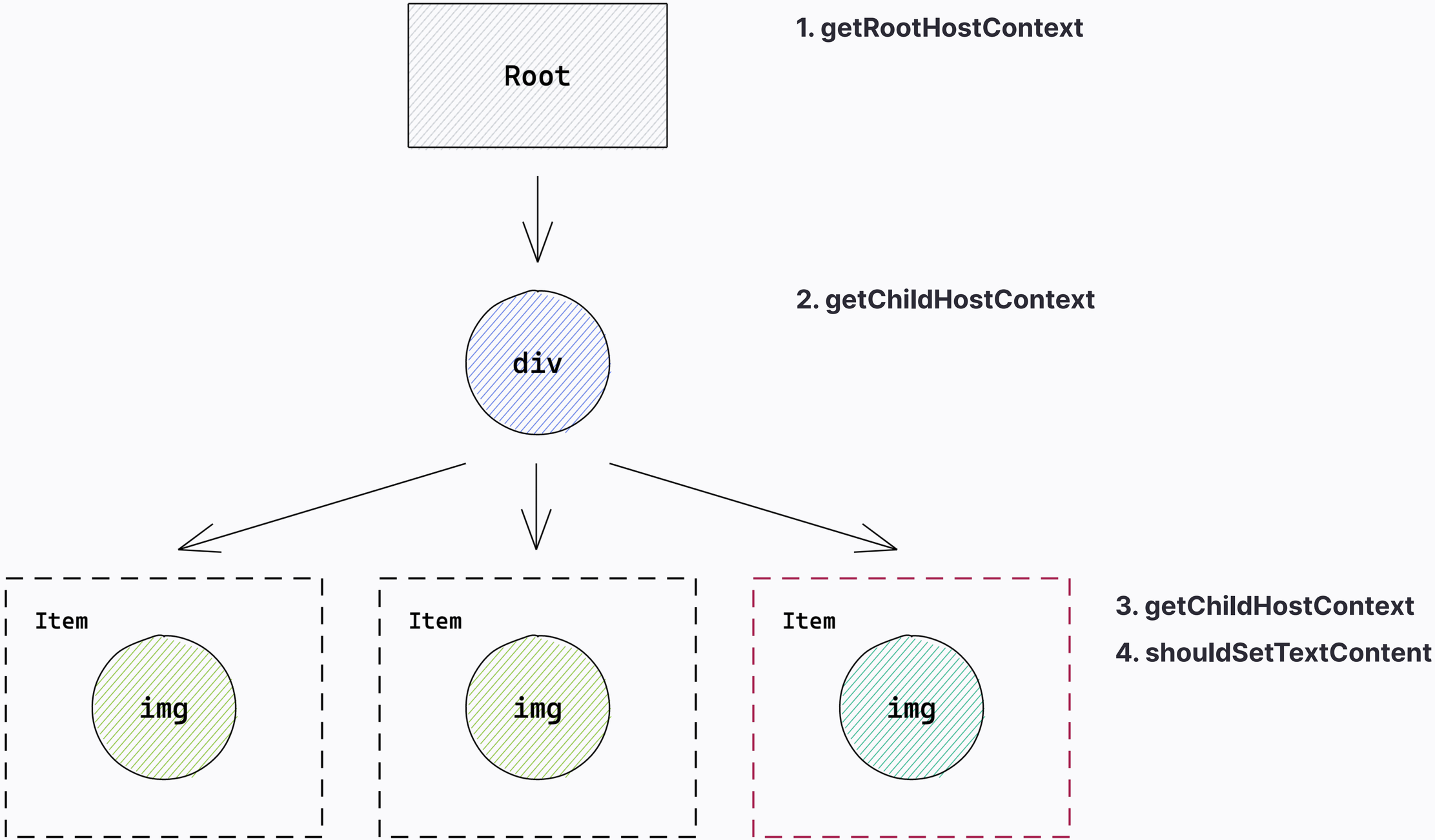
22. commitMount

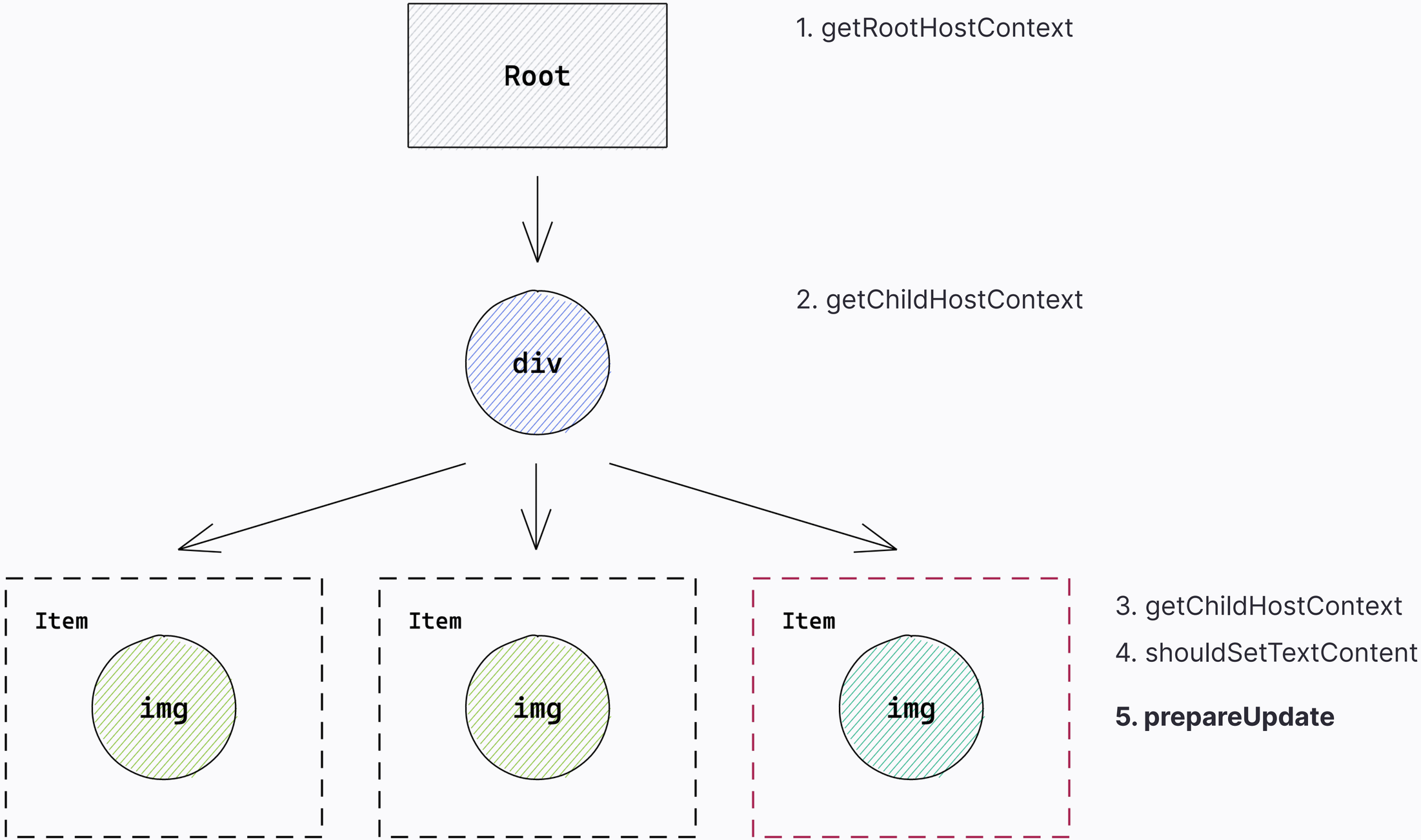
10. createTextInstance

Отрисовка изменений

изменение пропсов

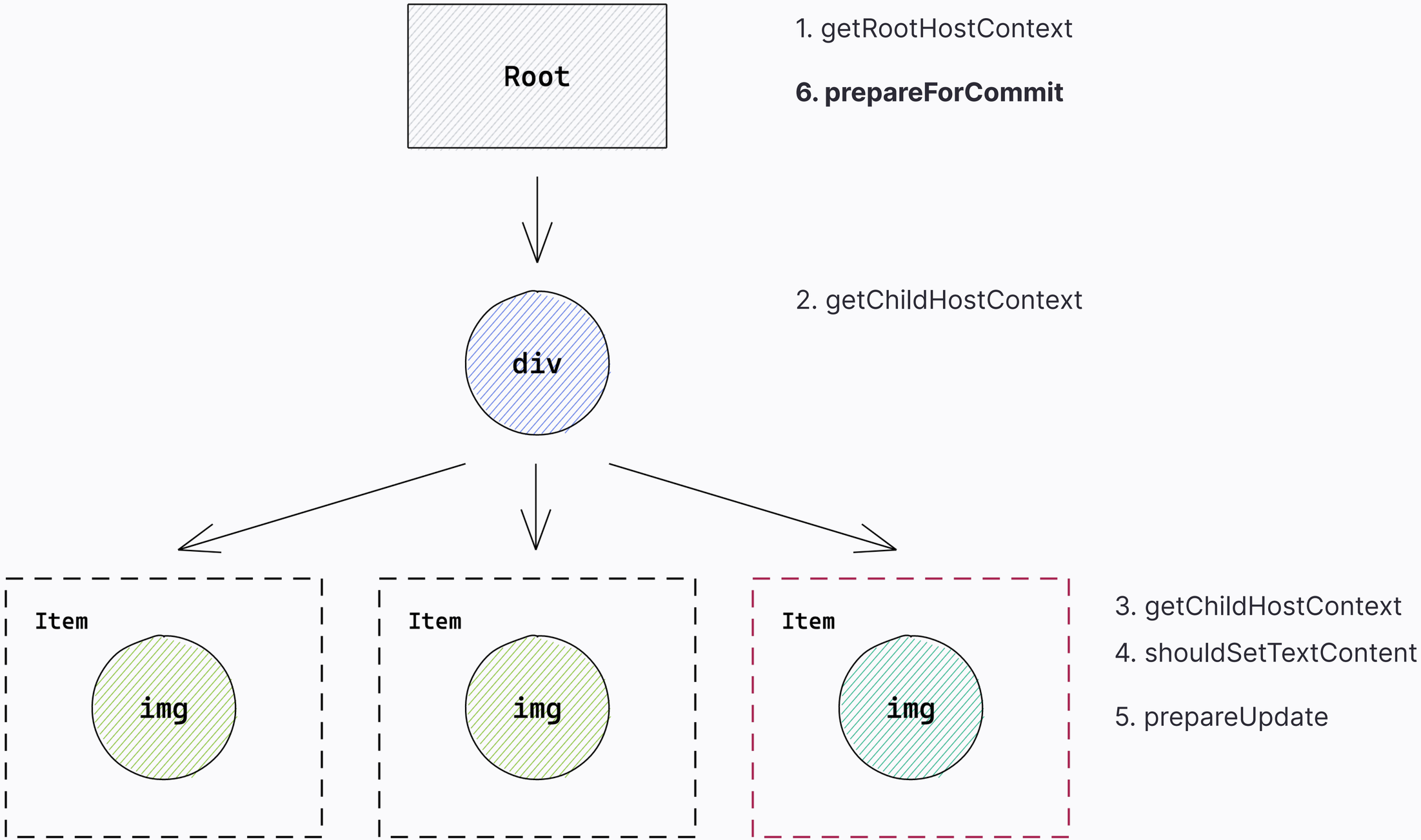

Root
div
Item
img
Item
img
Item
img


Root
1. getRootHostContext
div
2. getChildHostContext
Item
img
Item
img
Item
img
3. getChildHostContext
4. shouldSetTextContent


Root
1. getRootHostContext
div
2. getChildHostContext
Item
img
Item
img
Item
img
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

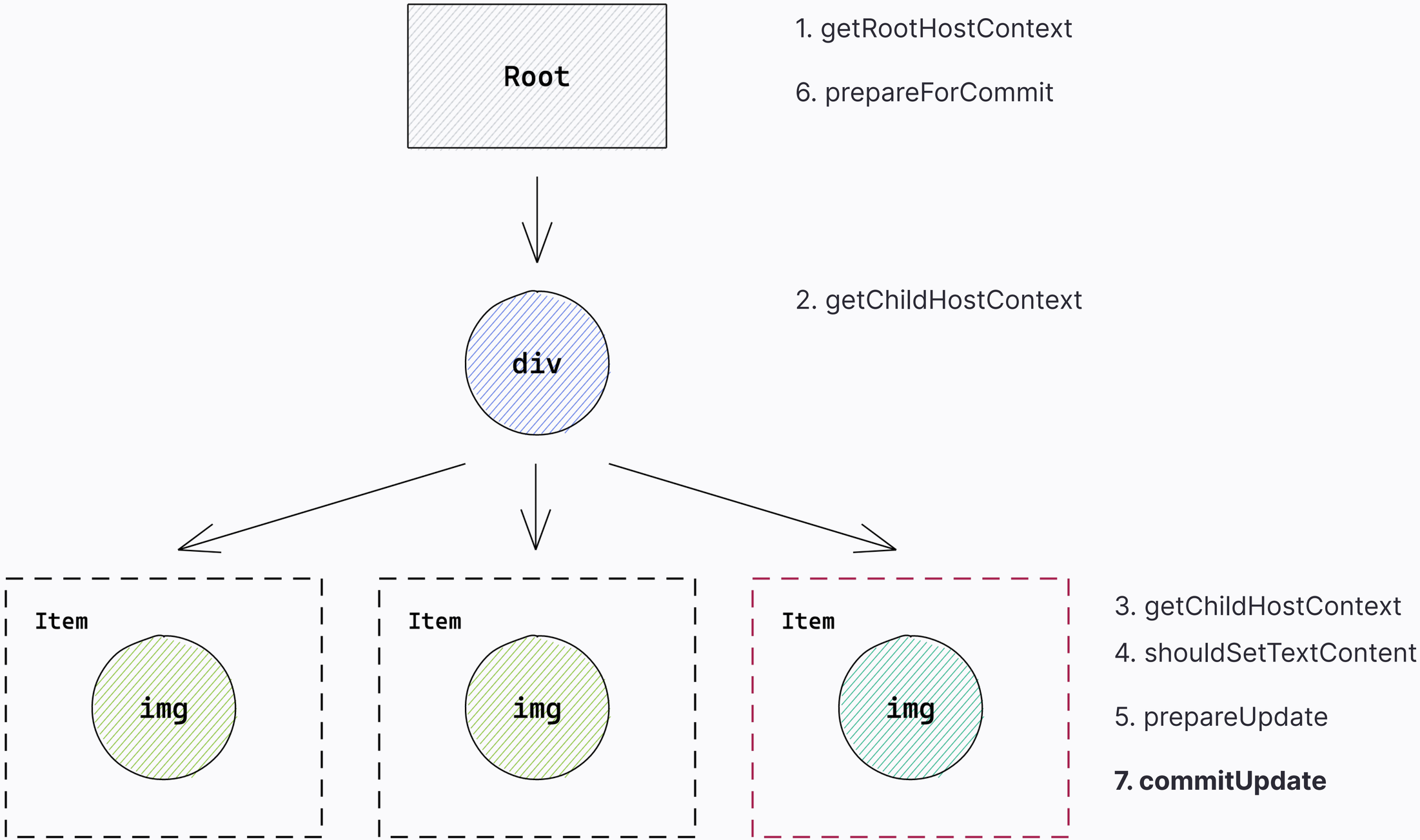
6. prepareForCommit

2. getChildHostContext

3. getChildHostContext

4. shouldSetTextContent

5. prepareUpdate



1. getRootHostContext

6. prepareForCommit

2. getChildHostContext

3. getChildHostContext

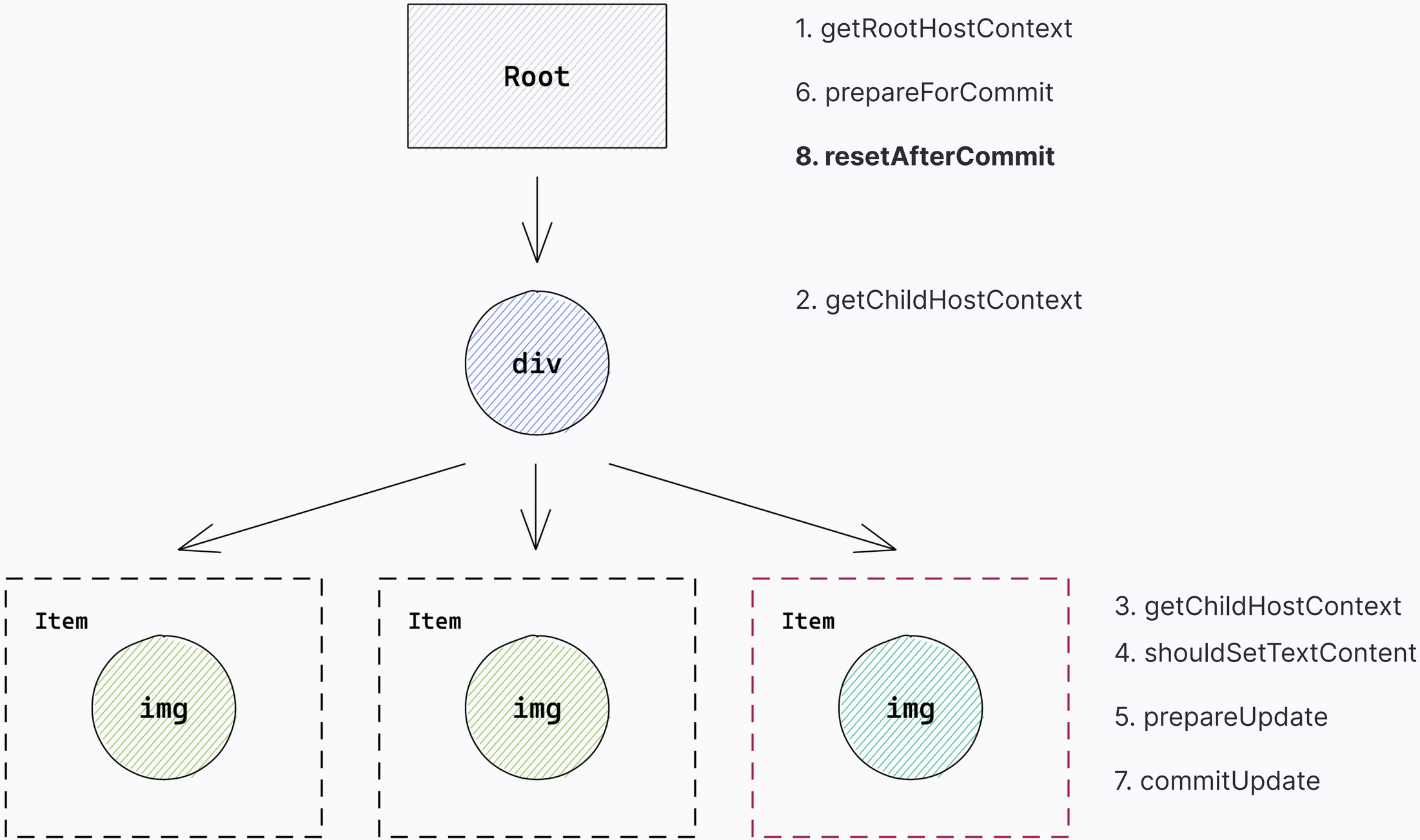
4. shouldSetTextContent

5. prepareUpdate

7. commitUpdate

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

Применяет изменения к элементу



1. getRootHostContext

6. prepareForCommit

8. resetAfterCommit

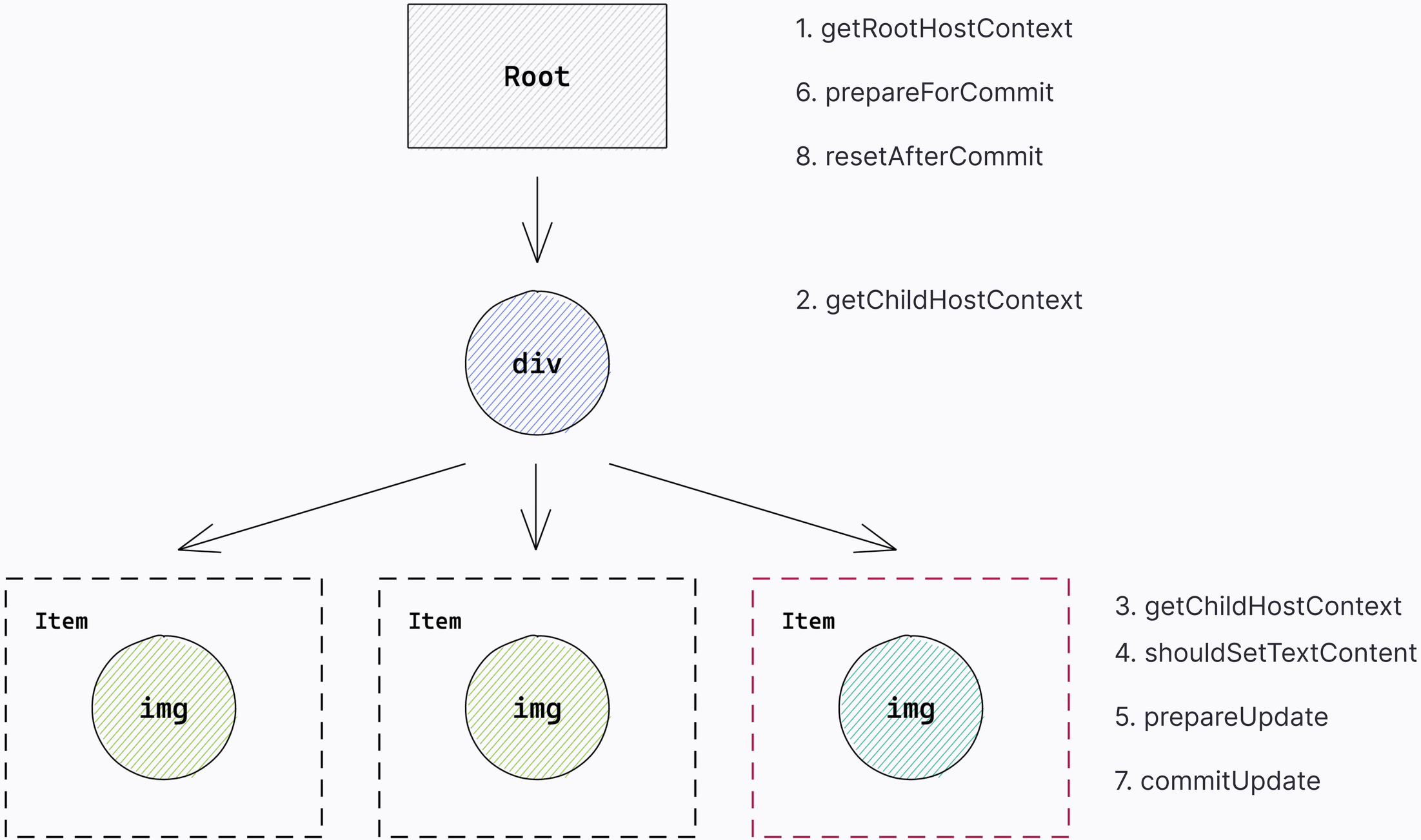
2. getChildHostContext

3. getChildHostContext

4. shouldSetTextContent

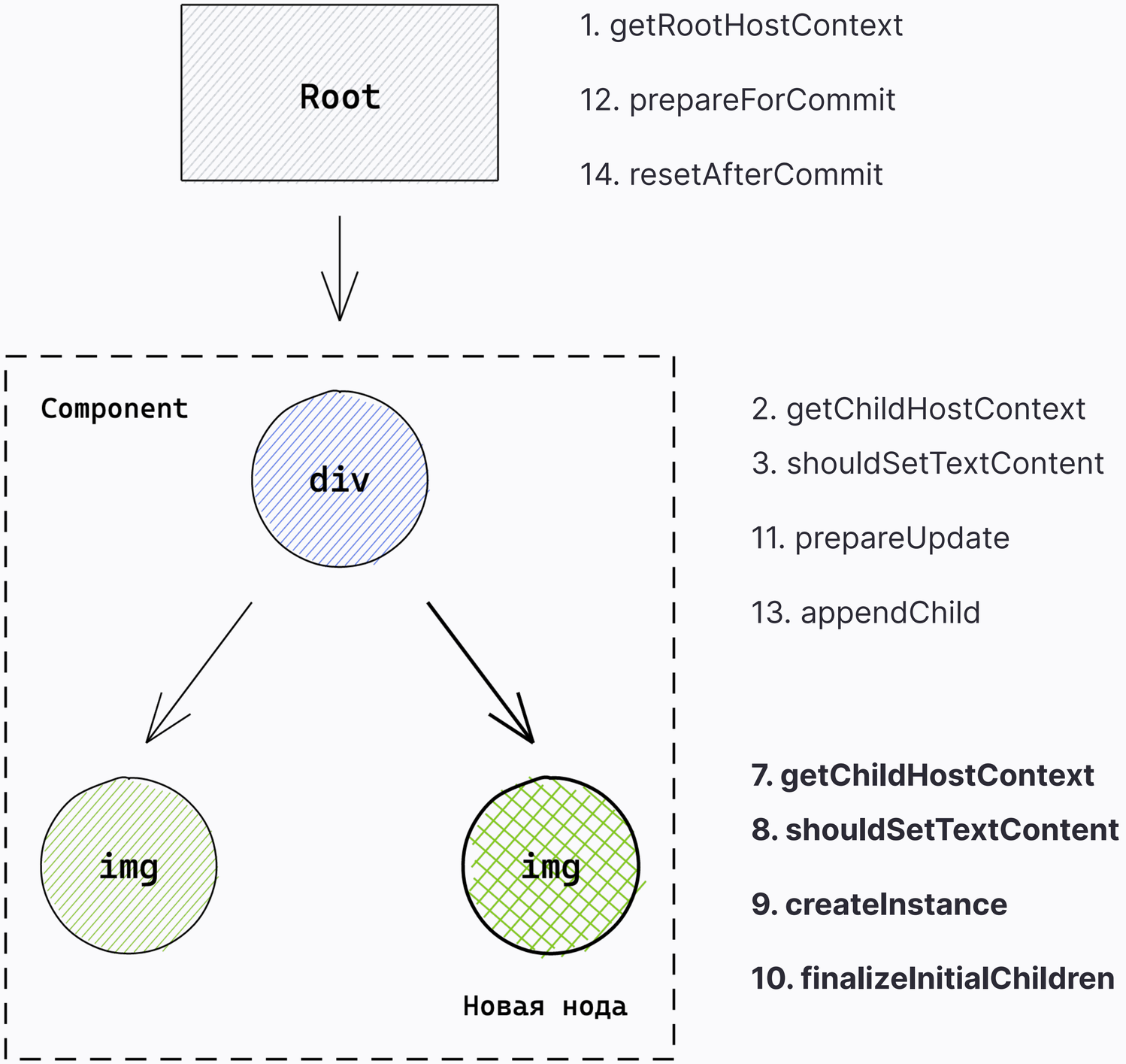
5. prepareUpdate

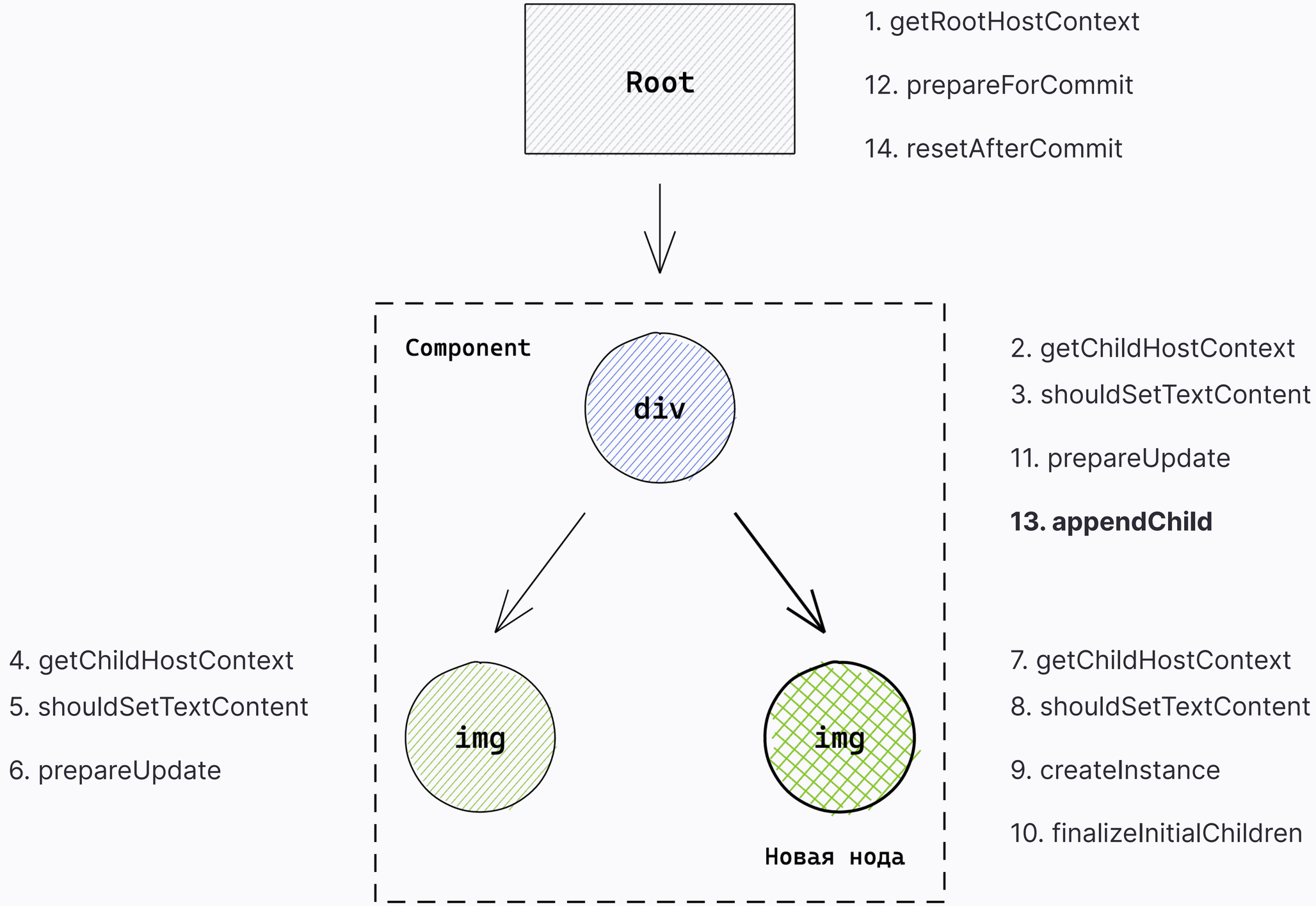
7. commitUpdate


Root
div
Item
img
Item
img
Item
img
1. getRootHostContext
6. prepareForCommit
8. resetAfterCommit
2. getChildHostContext
3. getChildHostContext
4. shouldSetTextContent
5. prepareUpdate
7. commitUpdate

Отрисовка изменений

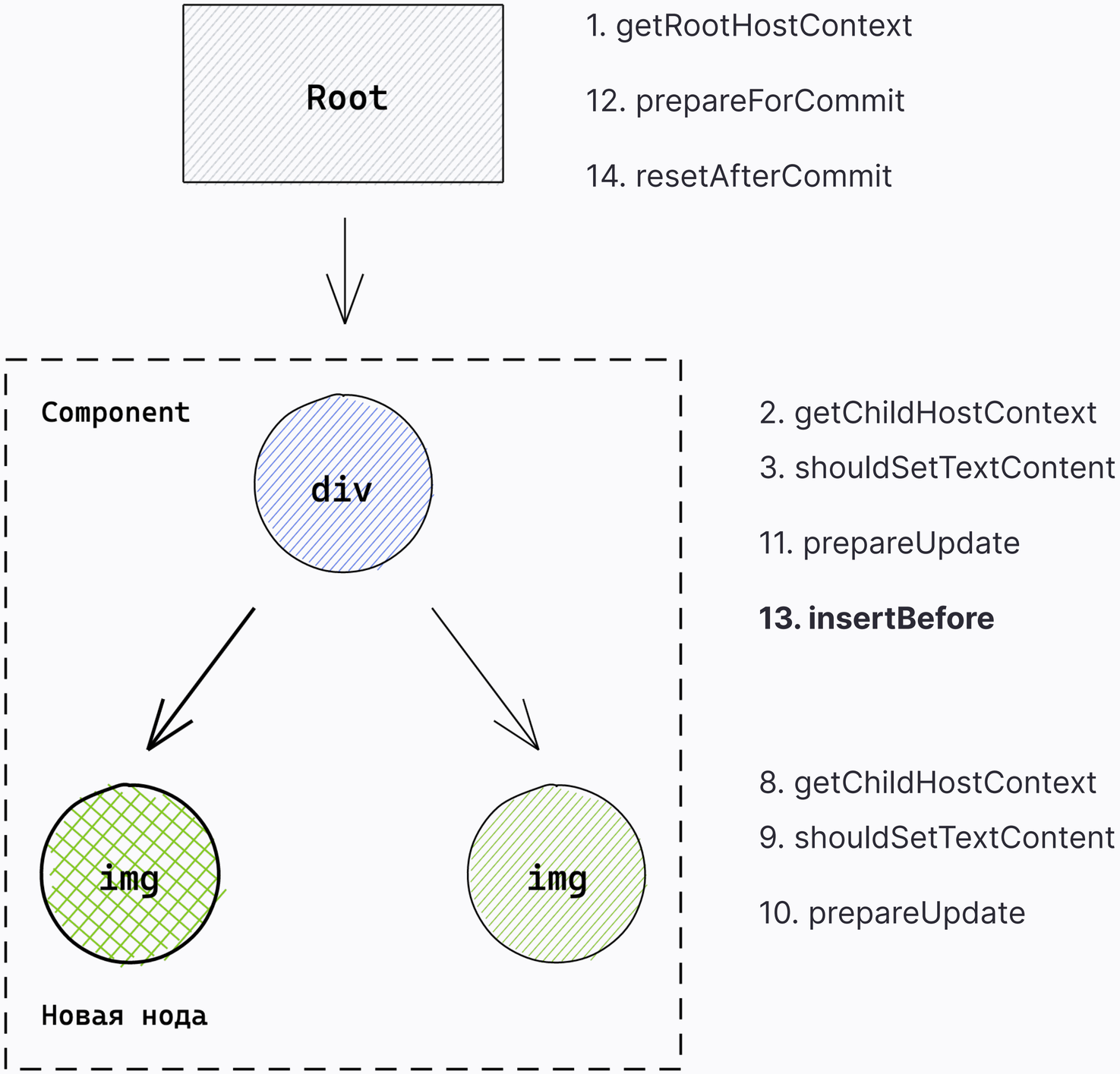
вставка узлов


Root
1. getRootHostContext
12. prepareForCommit
14. resetAfterCommit
Component
div
2. getChildHostContext
3. shouldSetTextContent
11. prepareUpdate
13. appendChild
4. getChildHostContext
5. shouldSetTextContent
6. prepareUpdate
img
img
7. getChildHostContext
8. shouldSetTextContent
9. createInstance
10. finalizeInitialChildren
Новая нода


Root
1. getRootHostContext
12. prepareForCommit
14. resetAfterCommit
Component
div
2. getChildHostContext
3. shouldSetTextContent
11. prepareUpdate
13. appendChild
4. getChildHostContext
5. shouldSetTextContent
6. prepareUpdate
img
img
7. getChildHostContext
8. shouldSetTextContent
9. createInstance
10. finalizeInitialChildren
Новая нода

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

Добавляет ребенка в конец списка детей


Root
1. getRootHostContext
12. prepareForCommit
14. resetAfterCommit
Component
div
2. getChildHostContext
3. shouldSetTextContent
11. prepareUpdate
13. insertBefore
4. getChildHostContext
5. shouldSetTextContent
6. createInstance
7. finalizeInitialChildren
img
img
8. getChildHostContext
9. shouldSetTextContent
10. prepareUpdate
Новая нода

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

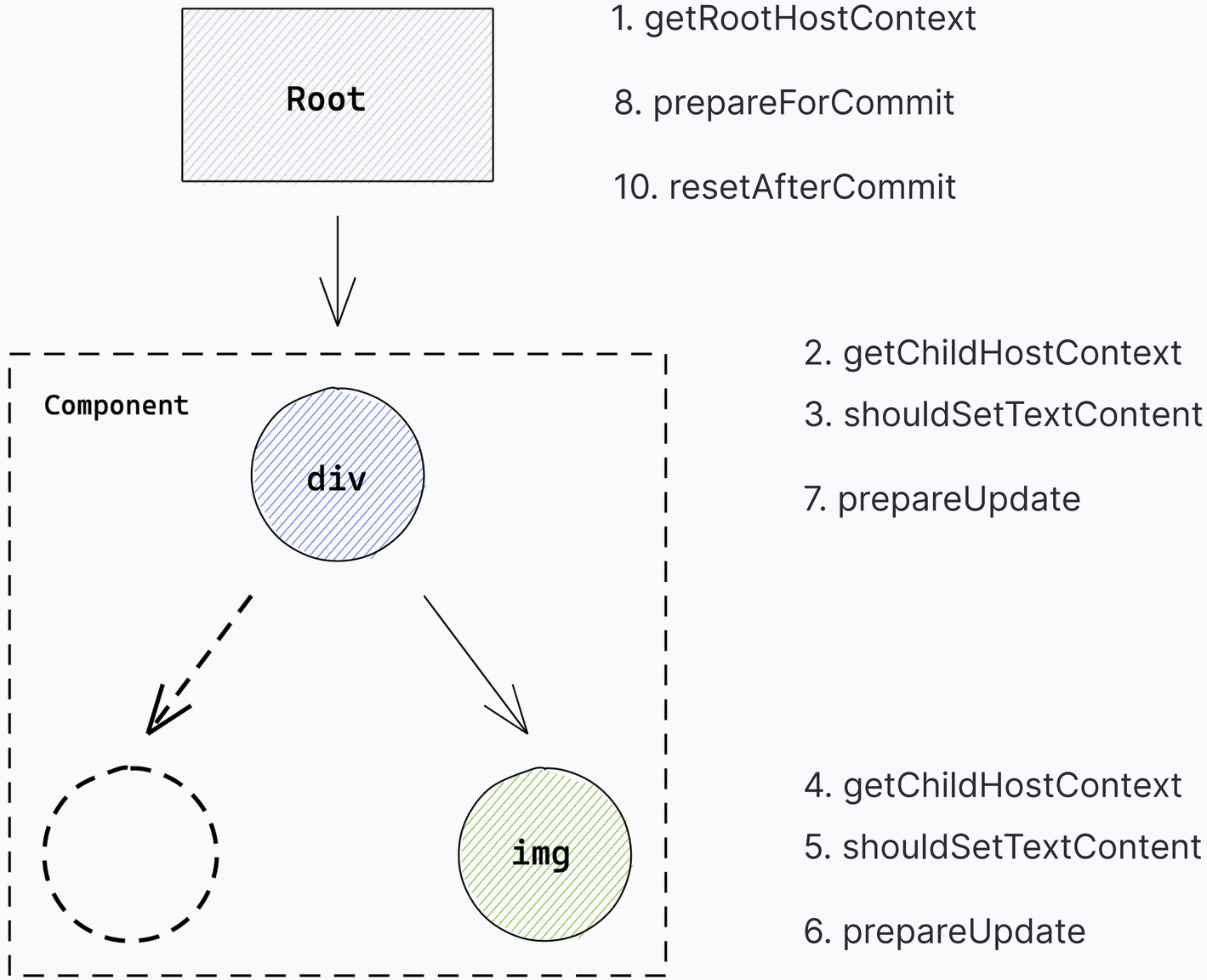
Добавляет ребенка после определенного элемента

```
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
```

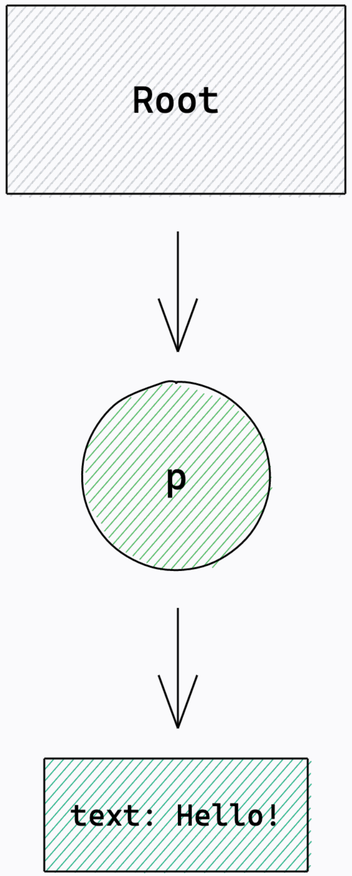
Удаляет поддерево

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

Удаляет поддерево из контейнера

Отрисовка изменений

текстовые листья



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

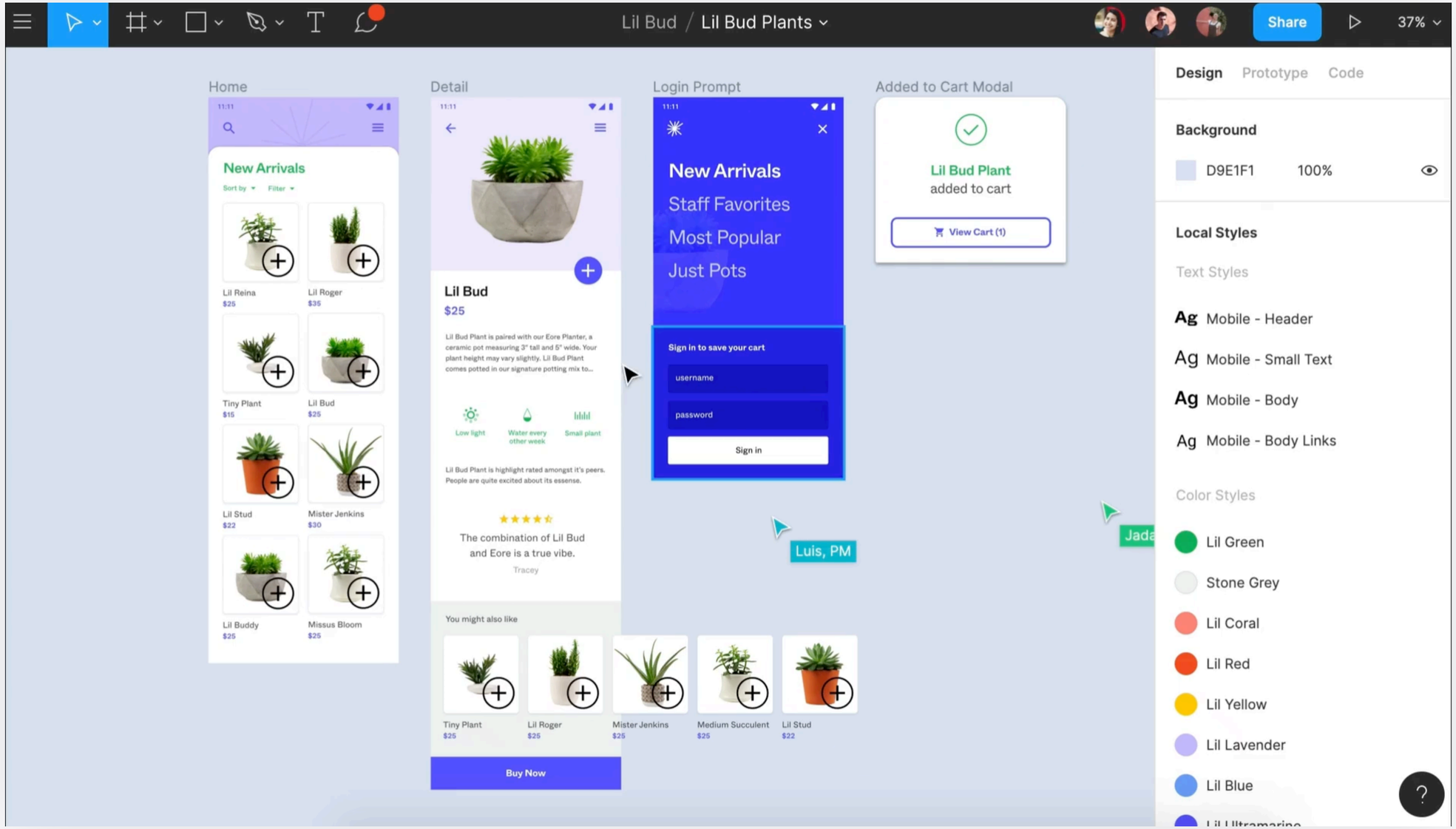
Что осталось?

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

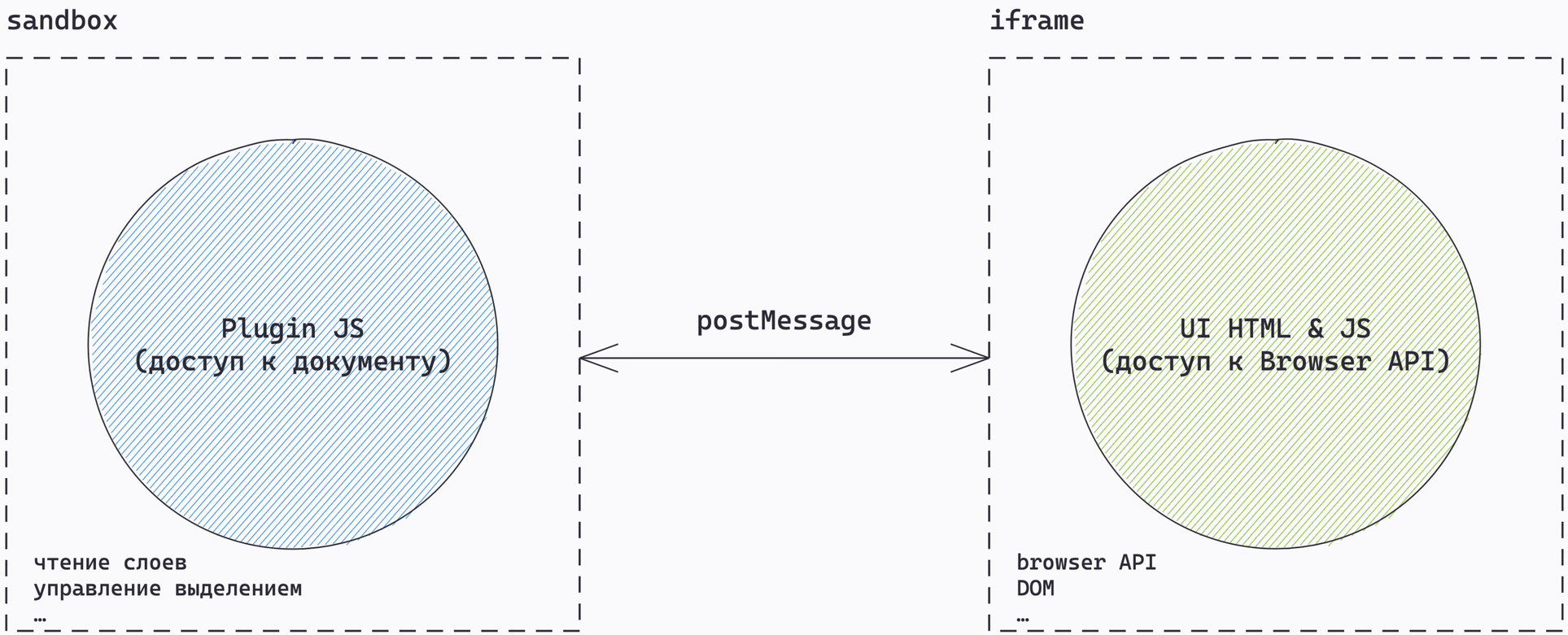
https://github.com/facebook/react/tree/master/packages/react-reconciler

О чем поговорим?

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


Lil Bud / Lil Bud Plants
Share
37%
Home
Detail
Login Prompt
Added to Cart Modal
New Arrivals
Sort by
Filter
Lil Reina
$25
Lil Roger
$35
Tiny Plant
$15
Lil Bud
$25
Lil Stud
$22
Mister Jenkins
$30
Lil Buddy
$25
Missus Bloom
$25
Lil Bud
$25
Lil Bud Plant is paired with our Eore Planter, a ceramic pot measuring 3" tall and 5" wide. Your plant height may vary slightly. Lil Bud Plant comes potted in our signature potting mix to...
Low light
Water every other week
Small plant
Lil Bud Plant is highlight rated amongst it's peers. People are quite excited about its essense.
The combination of Lil Bud and Eore is a true vibe.
Tracey
You might also like
Tiny Plant
$25
Lil Roger
$25
Mister Jenkins
$25
Medium Succulent
$25
Lil Stud
$22
Buy Now
New Arrivals
Staff Favorites
Most Popular
Just Pots
Sign in to save your cart
username
password
Sign in
Luis, PM
Jada
Lil Bud Plant
added to cart
View Cart (1)
Design
Prototype
Code
Background
D9E1F1
100%
Local Styles
Text Styles
Mobile - Header
Mobile - Small Text
Mobile - Body
Mobile - Body Links
Color Styles
Lil Green
Stone Grey
Lil Coral
Lil Red
Lil Yellow
Lil Lavender
Lil Blue
Lil Ultramarine

Figma Plugin API


sandbox
iframe
Plugin JS
(доступ к документу)
postMessage
UI HTML & JS
(доступ к Browser API)
чтение слоев
управление выделением
…
browser API
DOM
…

Figma API demo

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

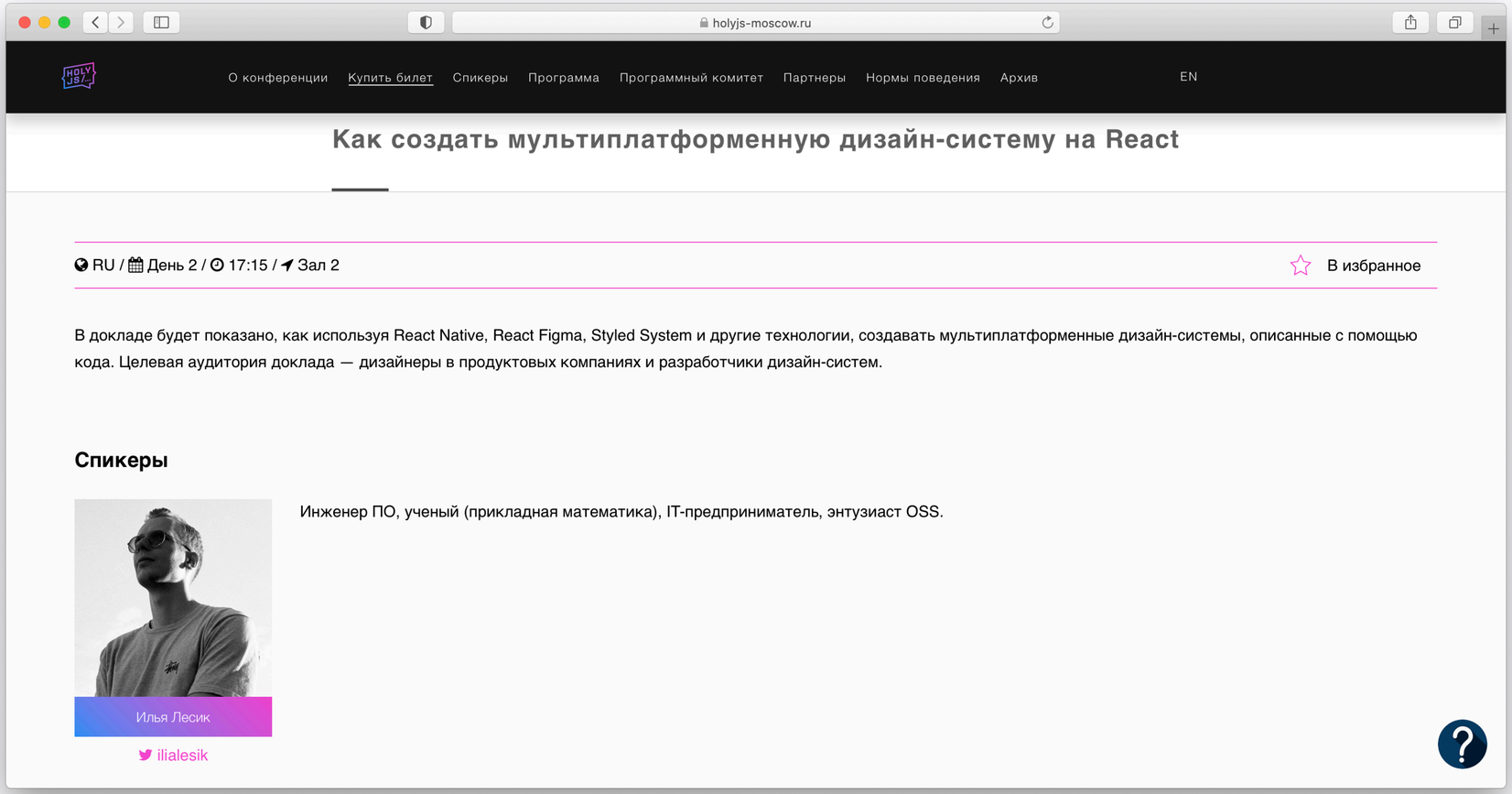
👨‍💻 Live code

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

А можно ли из Figma в React?

- github.com/bernaferrari/FigmaToCode
- overlay-tech.com
- Зар Захаров, Александр Каменяр — Figma to React: доставка дизайна в код

А можно универсально?



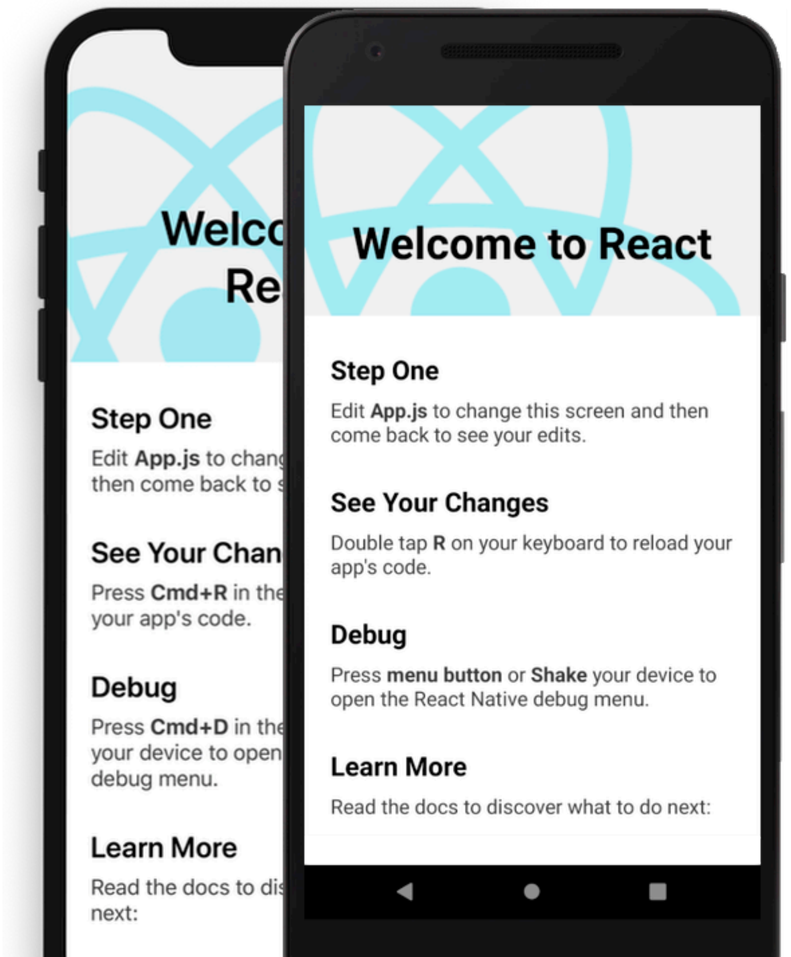
https://holyjs-moscow.ru/2020/msk/talks/5udhzhsrgbzzg3r496hmge/

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>
);
```

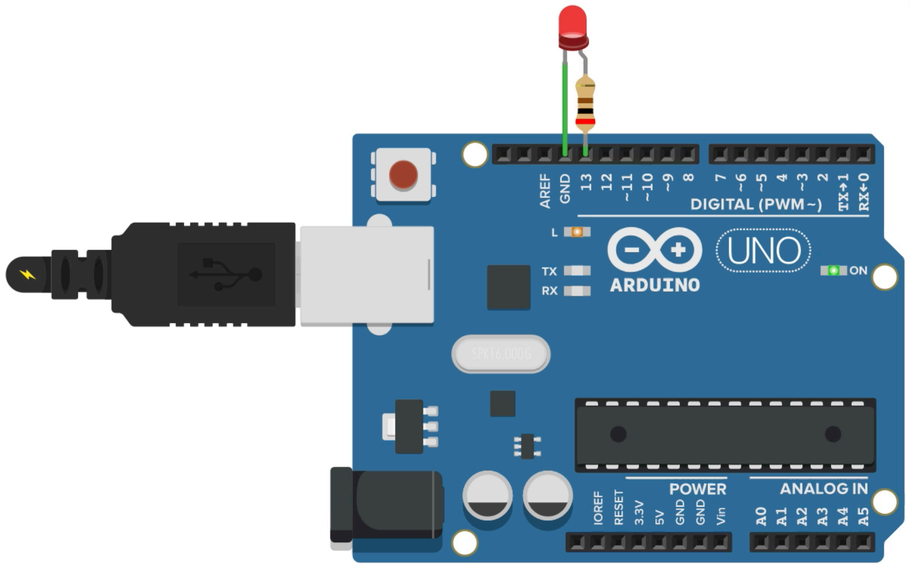


Ink

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

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

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

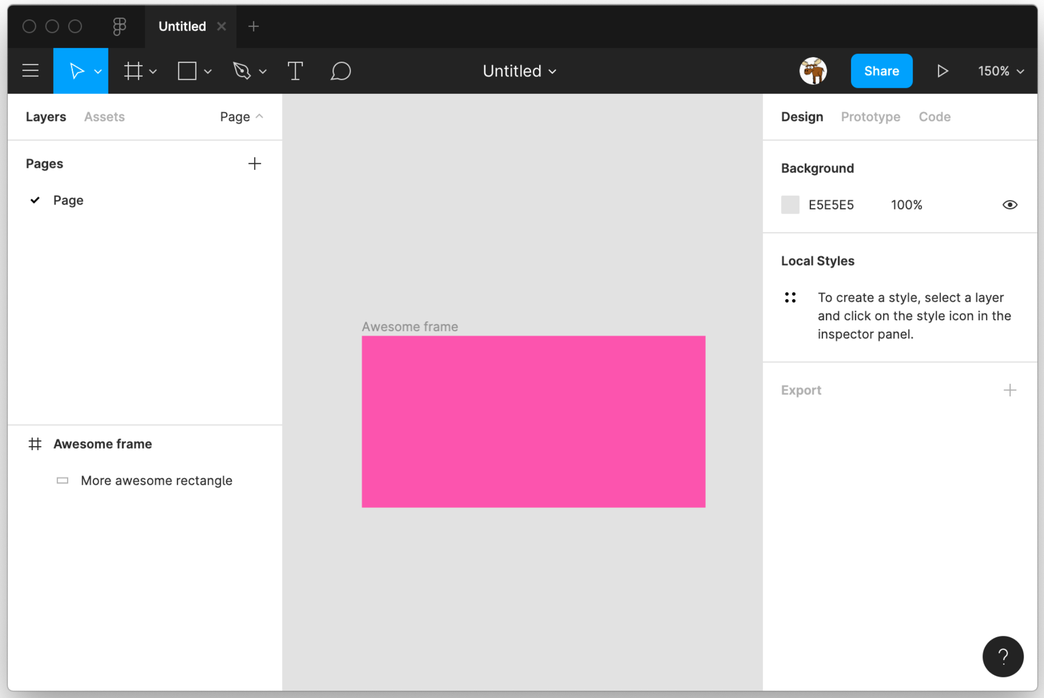
React Hardware

```
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);
```



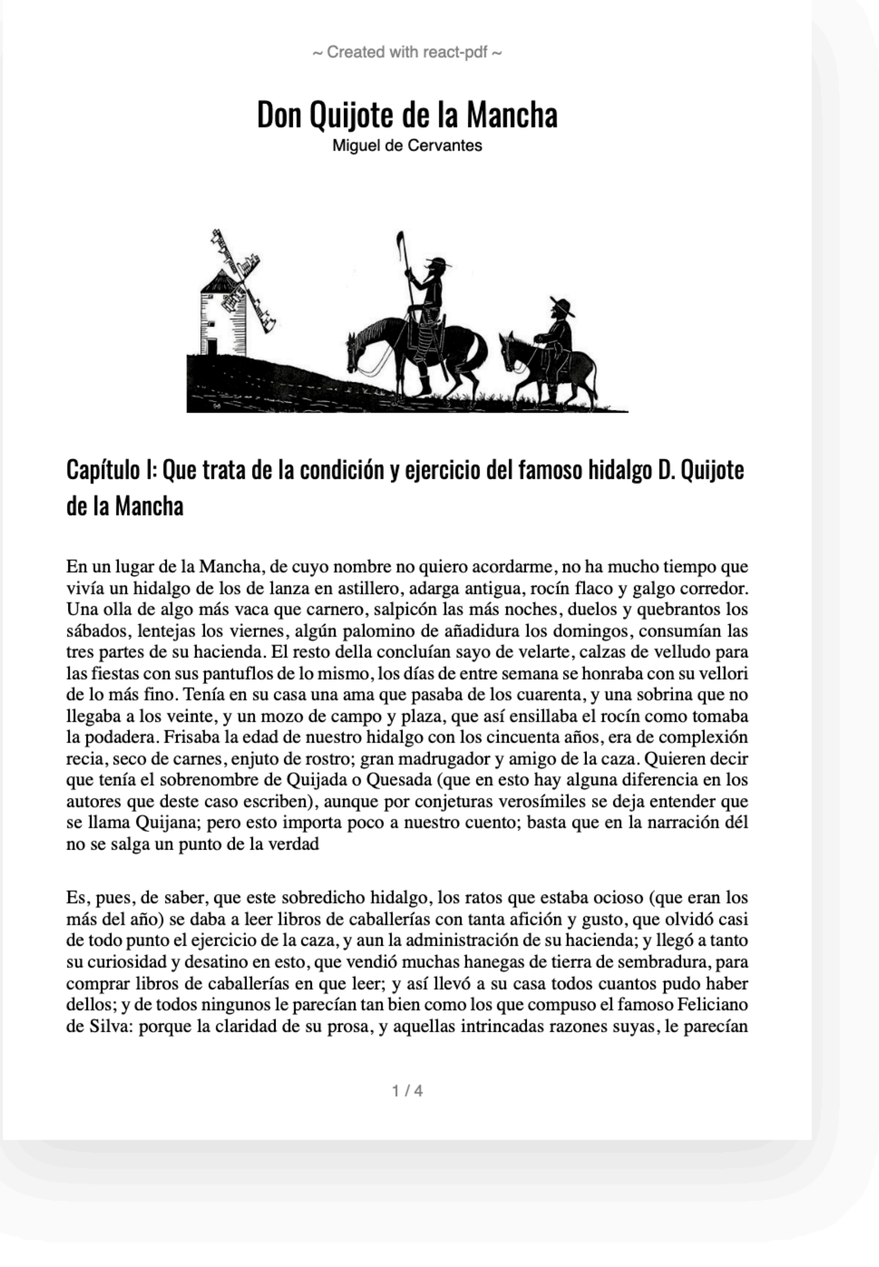
React Figma

```
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>
);
```



🙂 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

github.com/losyear LosYear twitter.com/losyear