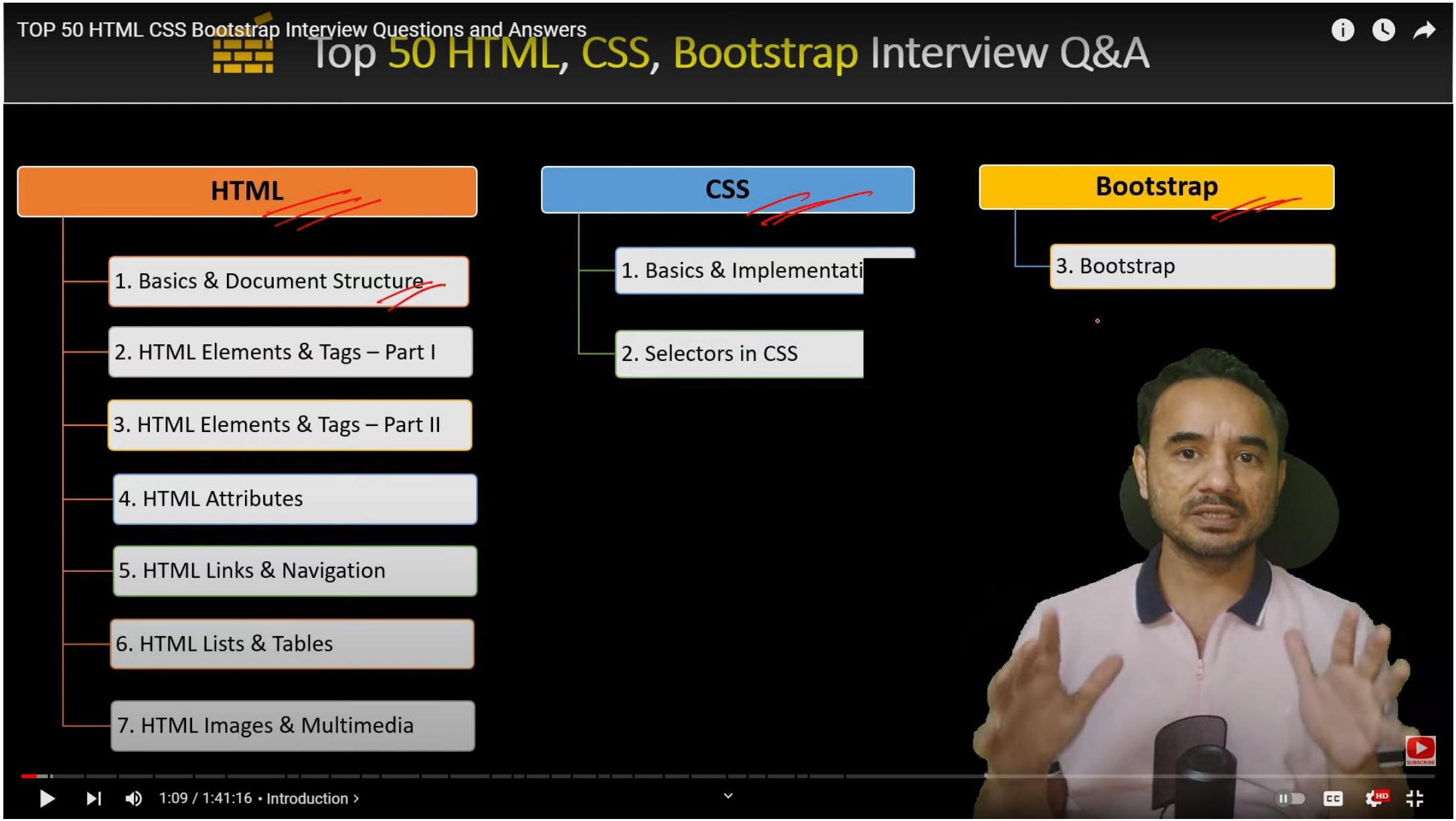
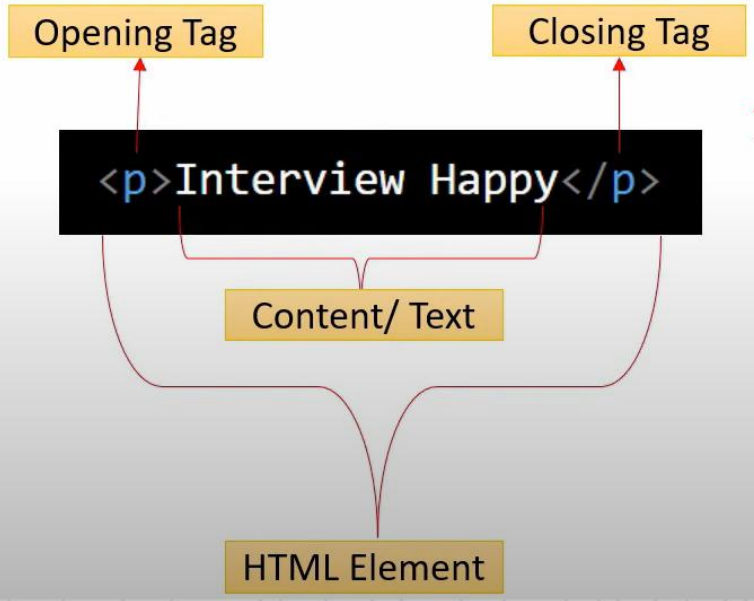

TOP 50 HTML CSS Bootstrap Interview Questions and Answers
Top 50 HTML, CSS, Bootstrap Interview Q&A
HTML
1. Basics & Document Structure
2. HTML Elements & Tags – Part I
3. HTML Elements & Tags – Part II
4. HTML Attributes
5. HTML Links & Navigation
6. HTML Lists & Tables
7. HTML Images & Multimedia
CSS
1. Basics & Implementati
2. Selectors in CSS
Bootstrap
3. Bootstrap
1:09 / 1:41:16 • Introduction ›

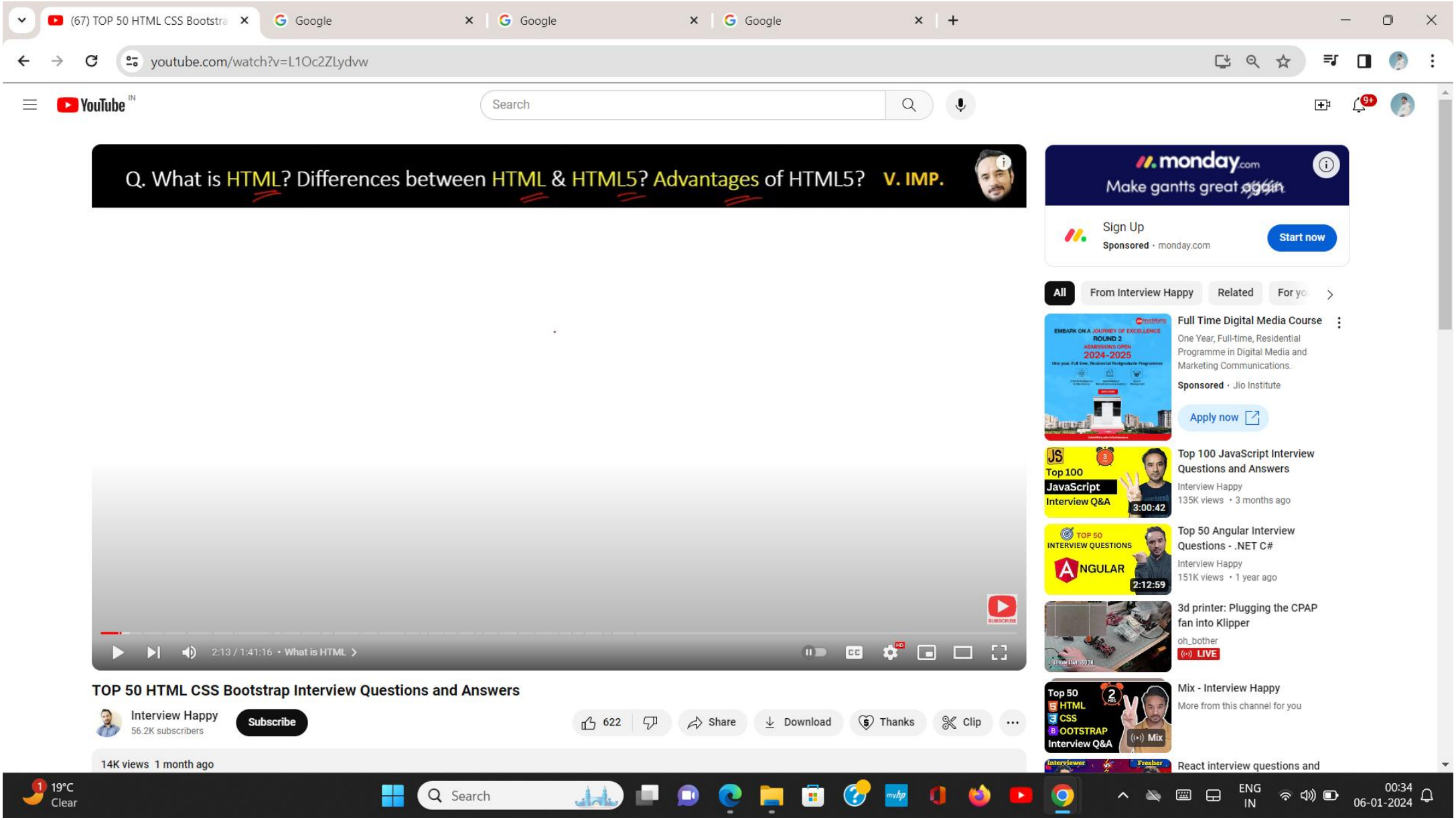
TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are HTML elements? What is the difference between Element & Tag? V. IMP.

- HTML elements are the **building blocks** of web pages.
- Element consists of a start tag, content, and an end tag.
- Tag is a specific part of an element that denotes the beginning or end of that element.




```
<!DOCTYPE html>
<html>
  <head>
    <title>Title Element</title>
  </head>
  <body>
    <h1>Heading Element</h1>
    <ul><!-- List Element -->
      <li>List Item 1</li>
      <li>List Item 2</li>
    </ul>
    <p>Paragraph Element</p>
    <div>Division Element</div>
    <img src="ImageElement.jpg" />
    <a href="~/AnchorElement.html"></a>
  </body>
</html>
```


(67) TOP 50 HTML CSS Bootstra
Google
Google
Google
youtube.com/watch?v=L1Oc2ZLydvw
YouTube IN
Search
Q. What is HTML? Differences between HTML & HTML5? Advantages of HTML5? V. IMP.
2:13 / 1:41:16 • What is HTML
TOP 50 HTML CSS Bootstrap Interview Questions and Answers
Interview Happy
56.2K subscribers
Subscribe
622
Share
Download
Thanks
Clip
14K views 1 month ago
monday.com
Make gantts great again
Sign Up
Sponsored · monday.com
Start now
All
From Interview Happy
Related
Full Time Digital Media Course
One Year, Full-time, Residential Programme in Digital Media and Marketing Communications.
Sponsored · Jio Institute
Apply now
Top 100 JavaScript Interview Questions and Answers
Interview Happy
135K views • 3 months ago
3:00:42
Top 50 Angular Interview Questions - .NET C#
Interview Happy
151K views • 1 year ago
2:12:59
3d printer: Plugging the CPAP fan into Klipper
oh_bother
LIVE
Mix - Interview Happy
More from this channel for you
React interview questions and
19°C
Clear
Search
ENG
IN
00:34
06-01-2024

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is HTML? Differences between HTML & HTML5? Advantages of HTML5? V. IMP.

- HTML(HyperText Markup Language) is the standard **markup language** used to create web pages.
- Markup language meaning a language which define the **structure of a document using elements** like headings, paragraphs, links, lists, and more.
- **HTML is not a programming language**, it's a markup language like XML.

XML

Hyper

Text

Markup

Language

Structure

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML</title>
  </head>
  <body>
    <a href="https://abc.com">Link</a>
    <h1>Heading Text</h1>
    <p>Para Text</p>
  </body>
</html>
```

Programming 1990

JS, Java

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is HTML? Differences between HTML & HTML5? Advantages of HTML5? V. IMP.

- HTML5 is the fifth and **latest version** of HTML.

Old HTML/ HTML4

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Old HTML Example</title>
    <meta http-equiv="Content-Type"
    content="text/html; charset=UTF-8" />
  </head>
  <body>
    <h1>Tag</h1>
    <p>Paragraph</p>
    <a href="https://abc.com">Link</a>
  </body>
</html>
```

HTML/ HTML5

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>HTML5 Example</title>
  </head>
  <body>
    <header>
      <h1>Tag</h1>
    </header>
    <section>
      <p>Paragraph</p>
      <a href="https://abc.com">Link</a>
    </section>
  </body>
</html>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is HTML? Differences between HTML & HTML5? Advantages of HTML5? V. IMP.

5 Advantages of HTML5:

1. New Semantic Elements: <header>, <nav>, <article>, <section>, <aside>, <footer>
2. Form Input Types: <input type="date">, <input type="email">
3. Audio and Video Support: <audio>, <video>
4. Mobile Compatibility
5. Simpler Code

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the difference between HTML and XHTML?

- XHTML(eXtensible HyperText Markup Language) is a markup language that follows the **rules of XML** to define the structure of web pages.

XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>XHTML Example</title>
  </head>
  <body>
    <h1>XHTML</h1>
    <img src="image.jpg" />
  </body>
</html>
```

HTML/ HTML5

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML</title>
  </head>
  <body>
    <h1>HTML</h1>
    <img src="image.jpg">
  </body>
</html>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the difference between HTML and XHTML?

HTML/ HTML5	XHTML
1. HTML has more **lenient syntax** rules. Closing tag is not mandatory for some elements. <img src="image.jpg" alt="Description">.	XHTML follows **stricter syntax** rules. All tags must be properly nested and closed.
2. HTML is **not case-sensitive**. <DIV>, <P> will work.	XHTML is case-sensitive. , <p> will work.
3. HTML is **widely supported** by all browsers and web platforms.	XHTML has limited support by browsers.

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the role of DOCTYPE in HTML? V. IMP.

- DOCTYPE(Document Type) declaration specifies the **version** of HTML.
- DOCTYPE tells the browser which version of HTML it is and how to interpret the code.

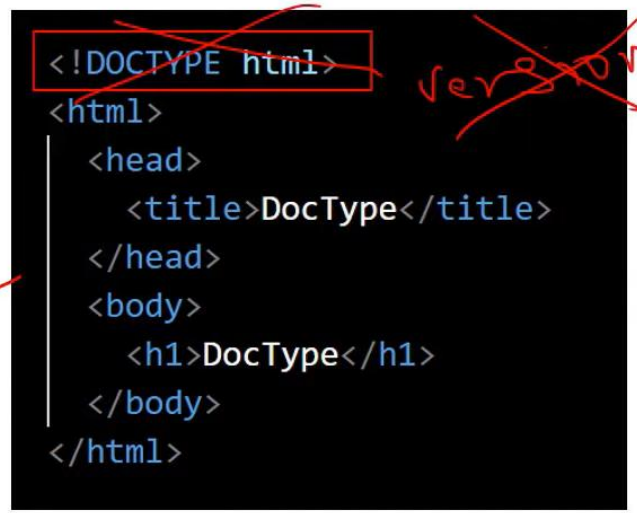
```
<!DOCTYPE html>
<html>
  <head>
    <title>DocType</title>
  </head>
  <body>
    <h1>DocType</h1>
  </body>
</html>
```

```
<!-- Different kind of Doctypes -->
<!-- HTML 4.01 Strict -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

<!-- XHTML 1.0 Strict: -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- HTML5 (Preferred for modern web development): -->
<!DOCTYPE html>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What if you remove <!DOCTYPE html> from your HTML?

- Then browsers can still render the page, but they will not be able to validate the version of HTML, therefore it may lead to some **compatibility issues** with SEO or debugging.



TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. Difference between Head & Body in HTML? Where to place JS link reference? V. IMP.

- The <head> element is where you place **meta-information** (information about the document). For example, <title>, <meta>, <link>, <script>, <style> are normally kept under head element.

- The <body> element is where you place the **actual content** of your HTML web page. For example, , <h1>, <p>, <img>, <a> are normally kept under body element.

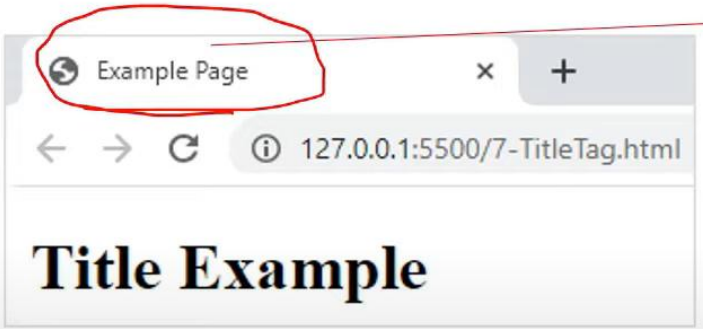
- Head will load before the body, therefore, if you are manipulating HTML elements in your JS functions, then place the JS link at the end of the body section because until then all the elements will be loaded. Else, place it inside the head tag.

```
<!DOCTYPE html>
<html>
<head>
    <title>Sample Page</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="styles.css">
    <script src="script.js"></script>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>This is a sample webpage.</p>
    <img src="image.jpg">
</body>
</html>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is Title Tag in HTML? What are the 4 advantages of Title tag? V. IMP.

- The <title> tag in HTML is used to define the **title** of a web page.



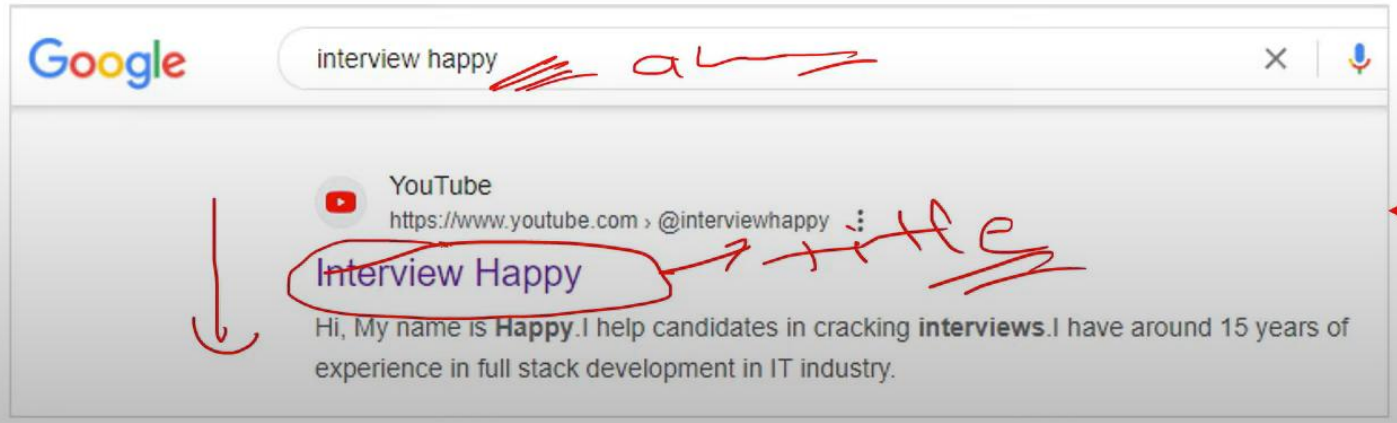

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Example Page</title>
  </head>
  <body>
    <h1>Title Example</h1>
  </body>
</html>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is Title Tag in HTML? What are the 4 advantages of Title tag? V. IMP.

❖ Advantages of title tag:

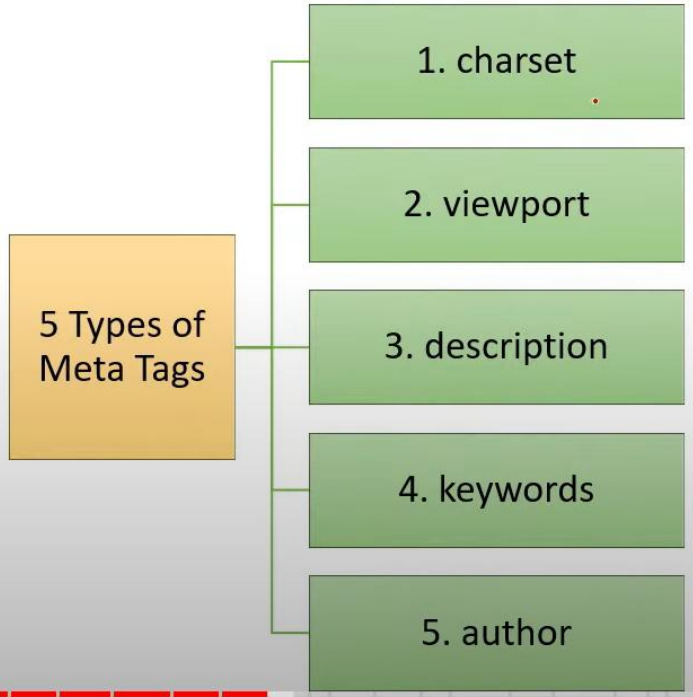
1. **Window Title:** <title> element text is displayed in the tab of the browser.
2. **SEO:** Search engines use the title as the main heading for search results.
3. **Bookmarks/Favorite:** Title tag is used as the default name for the bookmark.
4. **Social Media Sharing:** Title tag is used as the default title in the shared post.



TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are Meta Tags? What are the 5 types of meta tags?

- Meta tags in HTML are elements used to provide metadata or additional information about a web page.




```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Example Page</title>
  </head>
  <body>
    <h1>Title Example</h1>
  </body>
</html>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are Meta Tags? What are the 5 types of meta tags?

1. Character Encoding:
This helps browsers to interpret the different characters in the document. The charset="UTF-8" attribute ensures that characters from **different languages** will be displayed correctly in the browser.

2. Responsive Design:
viewport meta tag is crucial for creating mobile-friendly, **responsive web designs** for various screen sizes.

3. Description for SEO:
This meta tag provides description of the content of the page. Search engines may use this for search results.

4. Keywords for SEO:
This meta tag used to be important for search engines, but it's now of less importance.

5. Author Info:
This meta tag can be used to specify the author of the document.

```
<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Interview Happy</title>
    <meta name="description" content="Technical interview preparation" />
    <meta name="keywords" content=".NET, Angular, React, JS, HTML" />
    <meta name="author" content="Happy Rawat" />
  </head>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

2. HTML Elements & Tags – Part I

Q. What are HTML Elements? What is the difference between Element & Tag?

Q. What are the roles and uses of the element in HTML?

Q. What is the difference between and element?

Q. What is the role of <p>, <a>,
, <hr>, , <img>, <input> & <button> elements?

Q. What is the role of header, main, section, footer & address elements in HTML?

Q. What is the role of Heading tags in HTML? How does it impact SEO?

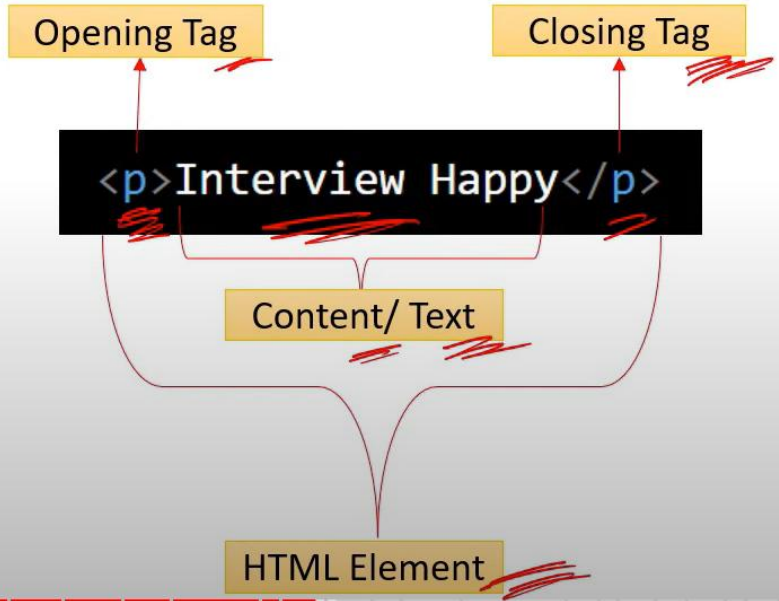
Q. What is the difference between <section> & <article> elements?

19:13 / 1:41:16 • Elements in HTML ›

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are HTML elements? What is the difference between Element & Tag? V. IMP.

- HTML elements are the **building blocks** of web pages.
- Element consists of a start tag, content, and an end tag.
- Tag is a specific part of an element that denotes the beginning or end of that element.




```
<!DOCTYPE html>
<html>
  <head>
    <title>Title Element</title>
  </head>
  <body>
    <h1>Heading Element</h1>
    <ul><!-- List Element -->
      <li>List Item 1</li>
      <li>List Item 2</li>
    </ul>
    <p>Paragraph Element</p>
    <div>Division Element</div>
    <img src="ImageElement.jpg" />
    <a href="~/AnchorElement.html"></a>
  </body>
</html>
```

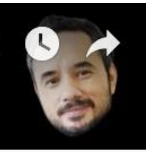
Q. What are HTML elements? What is the difference between Element & Tag? V. IMP.

Top 10 most common used HTML elements

1. **Paragraph Element**: Defines a block of text.<p>.
2. **Heading Elements**: <h1> <h2><h3><h4><h5><h6>.
3. **Header and Footer Elements**: <header> and <footer>.
4. **Sectioning Elements**: <section>, <article>, <nav>, <aside>, <main>.
5. **Image Element**: Embeds images into a webpage.<img> (self-closing).
6. **List Elements**: <ul> (unordered list), <ol> (ordered list)), <li> (list item).
7. **Anchor Element (Link)**: Creates hyperlinks to other pages or resources.<a>.
8. **Division Element**: groups together content for applying styles or scripting. .
9. **Table Elements**: <table>, <tr> (table row), <th> (table header) and <td> (table data).
10. **Form Elements**: Used for creating forms.<form>, <input>, <button>, <select>, <textarea>, etc.

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are the roles and uses of the element in HTML? V. IMP.



- The (division) element in HTML is a **container** that is used to **group and structure** the content on a webpage.

- **Top 3 uses of element:**

1. Grouping & structuring Content:

- It allows you to group together related elements.

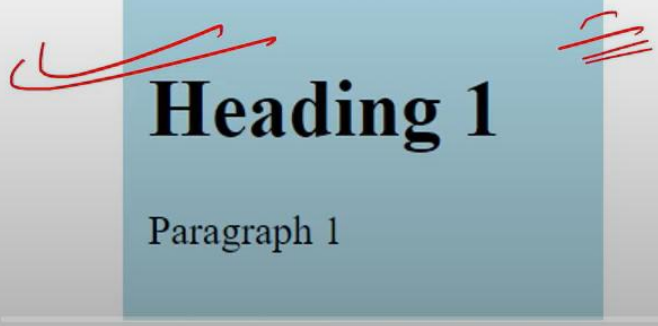
2. Styling and Layout:

- elements are used to apply common styles or css to grouped elements.

3. Scripting:

- JavaScript and other scripting languages can target elements to manipulate their content or behavior.

```
<body>
    <div class="container">

        <h1>Heading 1</h1>
        <p>Paragraph 1</p>

    </div>
</body>
```



TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the difference between and element?

- The (division) element in HTML is a **container** that is used to **group and structure** the content on a webpage.
- The element in HTML is an **inline container** used to apply styles or scripting to a specific section of text or content.
- is a block-level element and is an inline element.

```
<body>
  <div>
    <h1>Main Heading</h1>
    <p>Interview <span style="color: blue">Happy</span></p>
  </div>
</body>
```

inline

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the role of <p>, <a>,
, <hr>, , <img>, <input> & <button> elements?

1. <p>

- Defines a paragraph of text.

This is a paragraph.

```
<p>This is a paragraph.</p>
```

2. <a>

- Creates a hyperlink. The href attribute specifies the url and content is the text of hyperlink.

link.

```
<a href="https://abc.com">link</a>.
```

3.

- Inserts a line break.

first line.
second line.

```
<p>first line.<br />second line.</p>
```

4. <hr>

- Creates a horizontal line break.

```
<hr />
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the role of <p>, <a>,
, <hr>, , <img>, <input> & <button> elements?

1.

- Emphasizes text displayed in italics.

inline *italics text.*

```
<p>inline <em>italics text.</em></p>
```

2. <img>

- Embeds an image. The src attribute specifies the image file, and alt provides alternative text.

Image

```
<img src="abc.jpg" alt="Image" />
<br /><br />
```

3. <input>

- Creates an input field within a form.

abc

```
<input type="text" id="user" />
<br /><br />
```

4. <button>

- Creates a clickable button.

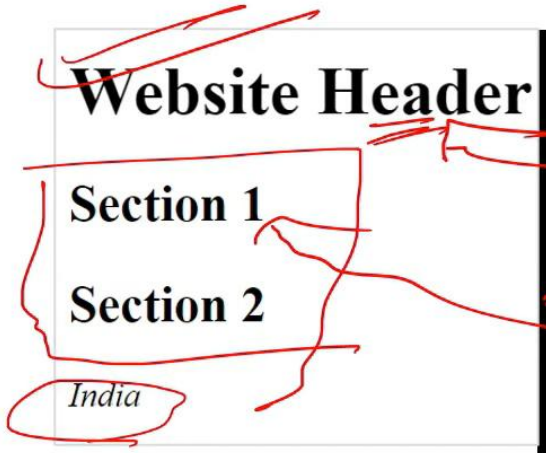
Click

```
<button type="button">Click</button>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the role of header, main, section, footer & address elements in HTML?

- <header>, <main>, <section> and <footer> are semantic elements, used to define the layout and **structure of a webpage** in a meaningful and organized way.




```
<body>
  <header>
    <h1>Website Header</h1>
  </header>
  <main>
    <section id="section1">
      <h2>Section 1</h2>
    </section>
    <section id="section2">
      <h2>Section 2</h2>
    </section>
  </main>
  <footer>
    <address>India</address>
  </footer>
</body>
```

<header>	Contains the header content of website.
<main>	Contains the main content of the document.
<footer>	contains the group related content.
<address>	provides contact information.

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the role of Heading tags in HTML? How does it impact SEO?

- Heading tags in HTML are used to define the **headings of sections** within a webpage.

Electronics Mobiles Samsung Mobiles Samsung Galaxy S Series Samsung Galaxy S20 Samsung Galaxy S20 Ultra

- Top 3 uses of heading tags:

1. Organization & Readability: Heading tags organize the content in better way, making it easier for users to understand the organization of the page.
2. SEO (Search Engine Optimization): Search engines use headings to understand the importance of content.
3. Styling and Layout: For example, same CSS can be applied to all h1 tags to show their importance. That will help in creating consistent layouts.

Electronics

Mobiles

Samsung Mobiles

Samsung Galaxy S Series

Samsung Galaxy S20

Samsung Galaxy S20 Ultra

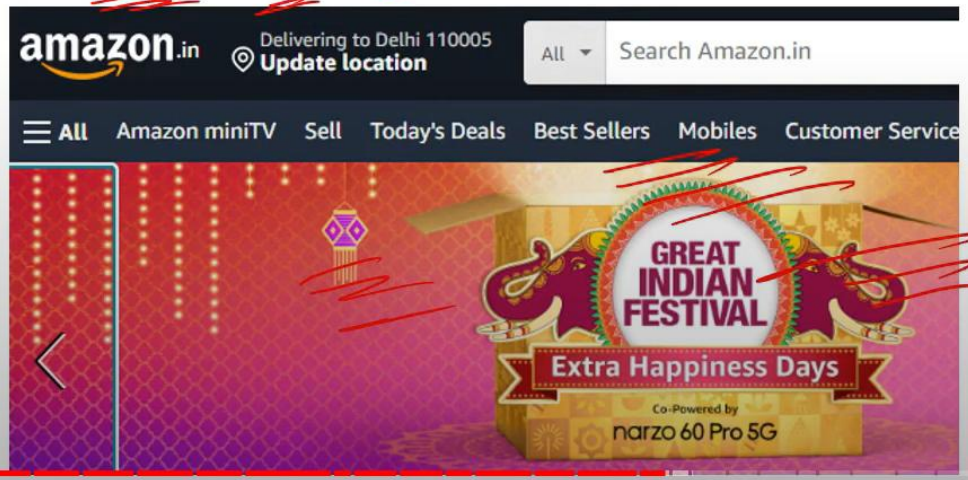
```
<body>
  <h1>Electronics</h1>

  <h2>Mobiles</h2>

  <h3>Samsung Mobiles</h3>

  <h4>Samsung Galaxy S Series</h4>

  <h5>Samsung Galaxy S20</h5>

  <h6>Samsung Galaxy S20 Ultra</h6>
</body>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the difference between <section> & <article> elements?

- The <section> element is a generic container used to **group related content together**.
- The <article> element represents a self-contained or **independent piece of content** with a title and content. For example: a blog post, a news article, a forum post, a comment, etc.

- Article example: offer




```
<body>
  <section>
    <h2>Mobile Section</h2>
    <p>Samsung, Nokia etc</p>
  </section>
  <section>
    <h2>Books Section</h2>
    <p>Technology, Philosphy etc</p>
  </section>

  <article>
    <h1>Festival Sale</h1>
    <p>10% Discount on all items</p>
    <footer>
      <p>Terms and conditions apply</p>
    </footer>
  </article>
</body>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

3. HTML Elements & Tags – Part II

Q. What are Root, Parent, Child & Nested elements?

Q. What are Empty Elements?

Q. What are Semantic Elements in HTML?

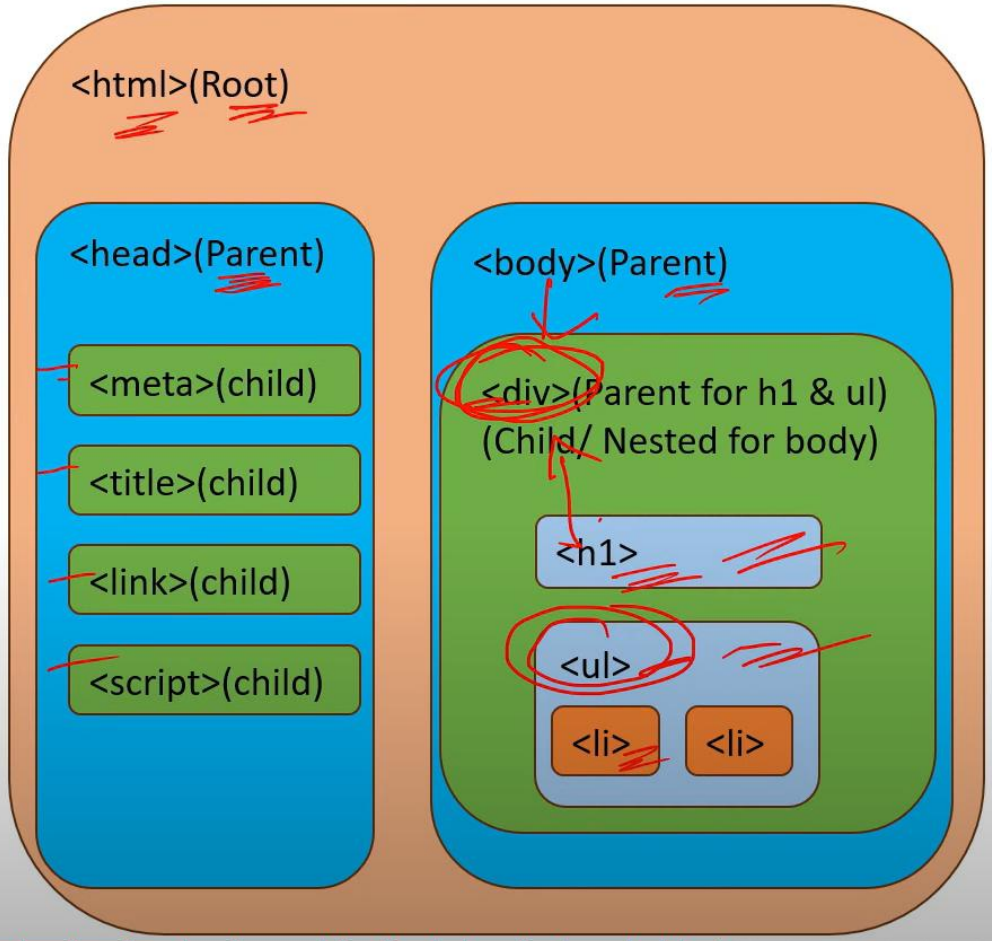
Q. Can HTML tags be written in Uppercase?

Q. What are the 3 differences between Block-Level & Inline Elements?

Q. List all Block-Level & Inline Elements in HTML.

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are Root, Parent, Child & Nested elements? V. IMP.




```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Nested Elements</title>
    <link rel="stylesheet" href="styles.css" />
    <script src="script.js"></script>
  </head>
  <body>
    <div>
      <h1>Inside container</h1>
      <ul>
        <li>Item 1</li>
        <li>Item 2</li>
      </ul>
    </div>
  </body>
</html>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are Root, Parent, Child & Nested elements? V. IMP.

1. The **root element** is the highest-level element in the hierarchy of an HTML document. In HTML5, the root element is <html>.

2. A **parent element** is an element that contains other child elements.

3. A **child or nested element** is an element that is contained within a parent element. Child and nested elements are same.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Nested Elements</title>
    <link rel="stylesheet" href="styles.css" />
    <script src="script.js"></script>
  </head>
  <body>
    <div>
      <h1>Inside container</h1>
      <ul>
        <li>Item 1</li>
        <li>Item 2</li>
      </ul>
    </div>
  </body>
</html>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are Empty Elements?

- An empty element in HTML is an element that doesn't need content between opening and closing tags.
- Empty elements are also called a **self-closing** or void elements.
- Empty elements in HTML: <img>, <input>,
, <hr>, <meta>, <link>, <area>, <base>, <col>, <embed>.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Empty Elements Example</title>
    <link rel="stylesheet" href="style.css" />
    <meta charset="UTF-8" />
  </head>
  <body>
    <img src="test.jpg" alt="Example" />
    <input type="text" />
    <br />
  </body>
</html>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are Semantic Elements in HTML? Is div a semantic element? V. IMP.

- Semantic elements in HTML are elements that provide **meaning to the content** they contain.
- is not a semantic element, because div is a general-purpose structural element. It doesn't give any meaning to the content.

Top 5 Semantic elements

1. <header>
2. <main>
3. <section>
4. <footer>
5. <address>

```
<body>
  <header>
    <h1>Website Header</h1>
  </header>
  <main>
    <section id="section1">
      <h2>Section 1</h2>
    </section>
  </main>
  <aside>
    <h2>Aside Content</h2>
  </aside>
  <footer>
    <address>India</address>
  </footer>
</body>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are Semantic Elements in HTML? Is div a semantic element? V. IMP.

❖ **10 more semantic elements:**

1. **<progress>** - Displays the progress of a task.
2. **<nav>** - Contains navigation links for a webpage.
3. **<time>** - Represents a specific period in time or a date.
4. **<mark>** - Highlights parts of the text for reference or emphasis.
5. **<summary>** - Provides a summary, caption, or legend for a <details> element.
6. **<meter>** - Represents a scalar measurement within a known range (like a gauge).
7. **<figure>** - Contains content like images, illustrations, diagrams, etc., along with a caption.
8. **<details>** - Represents additional information or controls that can be toggled open or closed.
9. **<aside>** - Contains content that is related to the main content but can be considered separate.
10. **<article>** - Represents a self-contained composition in a document, such as a blog post or a news article.

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. Can HTML tags be written in Uppercase?

- **Yes**, HTML tags not case-sensitive therefore can be written in uppercase, lowercase, or a combination of both.
- But it is not recommended as per standards.

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
    <TITLE>ALL UPPERCASE HTML</TITLE>
  </HEAD>
  <BODY>
    <H1>HELLO, THIS IS AN H1 HEADING</H1>
    <P>THIS IS A PARAGRAPH.</P>
    <A HREF="https://www.abc.com">THIS IS A LINK</A>
  </BODY>
</HTML>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are the 3 differences between Block-Level and Inline Elements? V. IMP.

1. Block-level elements create "blocks" of content.
   , <p>, <h1>, <ul>, <li>, <table>, <form> etc.
2. By default, Block-level elements start on a new line.
3. You can set both width and height for block-level elements.

1. Inline elements length depends on their content length.
   , <a>, , , <img>, <input>,
 etc.
2. Inline elements do not start on a new line.
3. You can't set width and height for inline elements.

```
<!DOCTYPE html>
<html>
  <head>
    <title>BlockLevel Elements</title>
  </head>
  <body>
    <div>block-level element</div>
    <p>block-level element</p>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Inline Elements</title>
  </head>
  <body>
    <p>My name is<strong> Happy</strong></p>
    <p>Click <a href="https://abc.com">here</a></p>
  </body>
</html>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. List all Block-Level & Inline elements in HTML.

❖ Top 20 block-level elements in HTML:

1. **<h1>, <h2>, <h3>, <h4>, <h5>, <h6>:** Heading elements from highest (h1) to lowest (h6) level.
2. **<header>:** Represents the introductory content or a group of navigational links.
3. ** :** The generic container element used for grouping and applying styles.
4. **<footer>:** Represents the footer of a section or document.
5. **<nav>:** Defines a section with navigation links.
6. **<section>:** Defines a section of a document.
7. **<ol>:** Creates an ordered (numbered) list.
8. **<p>:** Represents a paragraph of text.
9. **<li>:** Represents a list item in a list.
10. **<ul>:** Creates an unordered list.

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. List all Block-Level & Inline elements in HTML.

❖ Top 20 block-level elements in HTML(Continued...):

11. **<article>:** Represents a self-contained piece of content, such as a blog post or article.
12. **<aside>:** Represents content related to the main content but is considered separate.
13. **<blockquote>:** Represents a section that is quoted from another source.
14. **<figure>:** Used for encapsulating media and its caption.
15. **<form>:** Used for creating forms to collect user input.
16. **<hr>:** Creates a horizontal rule (a thematic break).
17. **<figcaption>:** Represents a caption for a <figure>.
18. **<table>:** Defines structure of a table.
19. **<tr>:** Represents a table row.
20. **<td>:** Represents a table cell.

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. List all Block-Level & Inline elements in HTML.

Top 20 inline elements in HTML:

1. **:** A generic container that doesn't have any specific semantic meaning.
2. **:** Indicates strong importance, often displayed as bold text.
3. **:** A generic container used for applying styles or scripting.
4. **:** Emphasizes on text, often displayed as italicized text.
5. **<small>:** Reduces the text size, often used for fine print.
6. **<a>:** Creates hyperlinks to other pages or resources.
7. **<label>:** Provides a label for an input element.
8. **<input>:** Used to create form input fields.
9. **<code>:** Represents a snippet of code.
10. **<cite>:** Specifies the title of a work.

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. List all Block-Level & Inline elements in HTML.

❖ Top 20 inline elements in HTML(Continued...):

11. **
:** Inserts a line break, forcing content after it onto a new line.
12. **<select>:** Creates a dropdown menu for selecting options.
13. **<time>:** Represents a specific period in time or a date.
14. **<small>:** Reduces the text size, used for fine print.
15. **<abbr>:** Defines an abbreviation or acronym.
16. **<del>:** Indicates deleted or removed text.
17. **<q>:** Defines a short inline quotation.
18. **<sup>:** Renders text as superscript.
19. **<sub>:** Renders text as subscript.
20. **<img>:** Embeds an image.

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

4. HTML Attributes

Q. What are HTML Attributes? What are the Types of HTML attributes?

Q. What are the Id, Style & Class attributes of an element? When to use what?

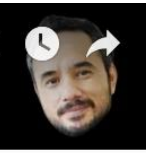
Q. What will happen if two elements have same ids?

Q. How to specify Multiple Classes for single element? What is the style precedence?

Q. What are Data Attributes in HTML?

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are HTML Attributes? What are the Types of HTML attributes? V. IMP.



- HTML attributes provide **additional information** about HTML elements.

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are HTML Attributes? What are the Types of HTML attributes? V. IMP.

❖ **Types of attributes:**

1. Common Attributes (Global Attributes)

- Example: class, id, style, data-*
- They are common and applicable for all the elements.

2. Specific Attributes

- Example: href, src, alt, width, height, target, rel, type, value, name, placeholder, disabled, readonly, checked, selected
- These are specific to the elements.

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are the Id, Style & Class attributes of an element? When to use what? V. IMP.

- **id** attribute is used to **uniquely identify** an element on a page.
  - The primary purpose of the id attribute is to allow JavaScript and CSS to target and manipulate specific elements.
- **style** attribute allows you to apply **inline styles** directly to an individual element.
- **class** attribute is used to group together multiple elements that **share common styles**.
  - classes are recommended for large website not inline styles.

```
<html>
  <head>
    <title>Id, Style and Class</title>
    <style>
      .highlighted {
        color: red;
      }
    </style>
  </head>
  <body>
    <h1 id="uniqueId">Main heading</h1>

    <p style="color: blue">Para 1</p>

    <p class="highlighted">Para 2</p>
    <p class="highlighted">Para 3</p>
  </body>
</html>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What will happen if two elements have same ids?

- May be in browser there is no direct impact, but it will be considered as **invalid HTML.**
- Invalid HTML means it can lead to **unexpected behavior** in your webpage. For example, problems can occur in css styling and JavaScript interactions.

```
<html lang="en">
  <head>
    <title>uplicate IDs</title>
  </head>
  <body>
    <div id="uniqueId1">First Element</div>
    <p id="uniqueId1">Second Element</p>
  </body>
</html>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. How to specify Multiple Classes for single element? What is the style precedence?

- To specify multiple classes for an element in HTML, you can simply add a **space-separated list of class names** within the class attribute.
- In case of multiple classes with same style attribute, style defined in the **last class** will take precedence.

```
<html>
  <head>
    <title>Multiple Classes Example</title>
    <style>
      .class1 {
        color: red;
      }
      .class2 {
        font-size: 20px;
        color: green;
      }
    </style>
  </head>
  <body>
    <div class="class1">single class</div>
    <div class="class1 class2">multiple classes</div>
  </body>
</html>
```

single class

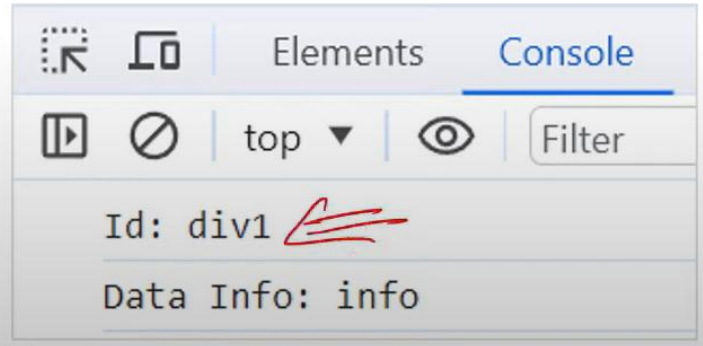
multiple classes

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are Data Attributes in HTML? V. IMP.

- Data attributes in HTML provide a way to add **custom data attributes** to add additional informational in elements.
- Data attributes are specified using the **data-** prefix.
- Data attributes can be accessed by dataset property in JS.

```
<body>
  <div id="div1" data-info="info">Data Attributes</div>

  <script>
    const element = document.querySelector("div");
    const id = element.id;
    const info = element.dataset.info;
    const dataid = element.dataset.id;

    console.log(`Id: ${id}`);
    console.log(`Data Info: ${info}`);
  </script>
</body>
```



TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Press Esc to exit full screen

5. HTML Links & Navigation

Q. What are the 5 Types of Links in HTML?

Q. What is the difference between Absolute & Relative URLs?

Q. What is the purpose of the <nav> element in HTML?

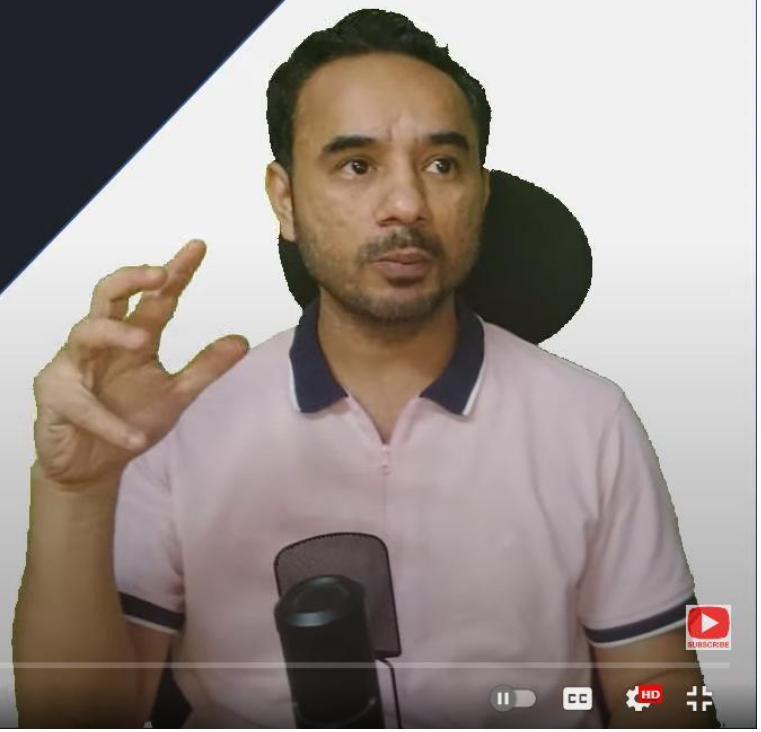
Q. What is a Fragment Identifier in a URL?

Q. What is the purpose of the <base> element in HTML.?

Q. How do you add an external stylesheet in your HTML?

Q. How do you open a link in a new tab?

Q. How do you create an Email Link?

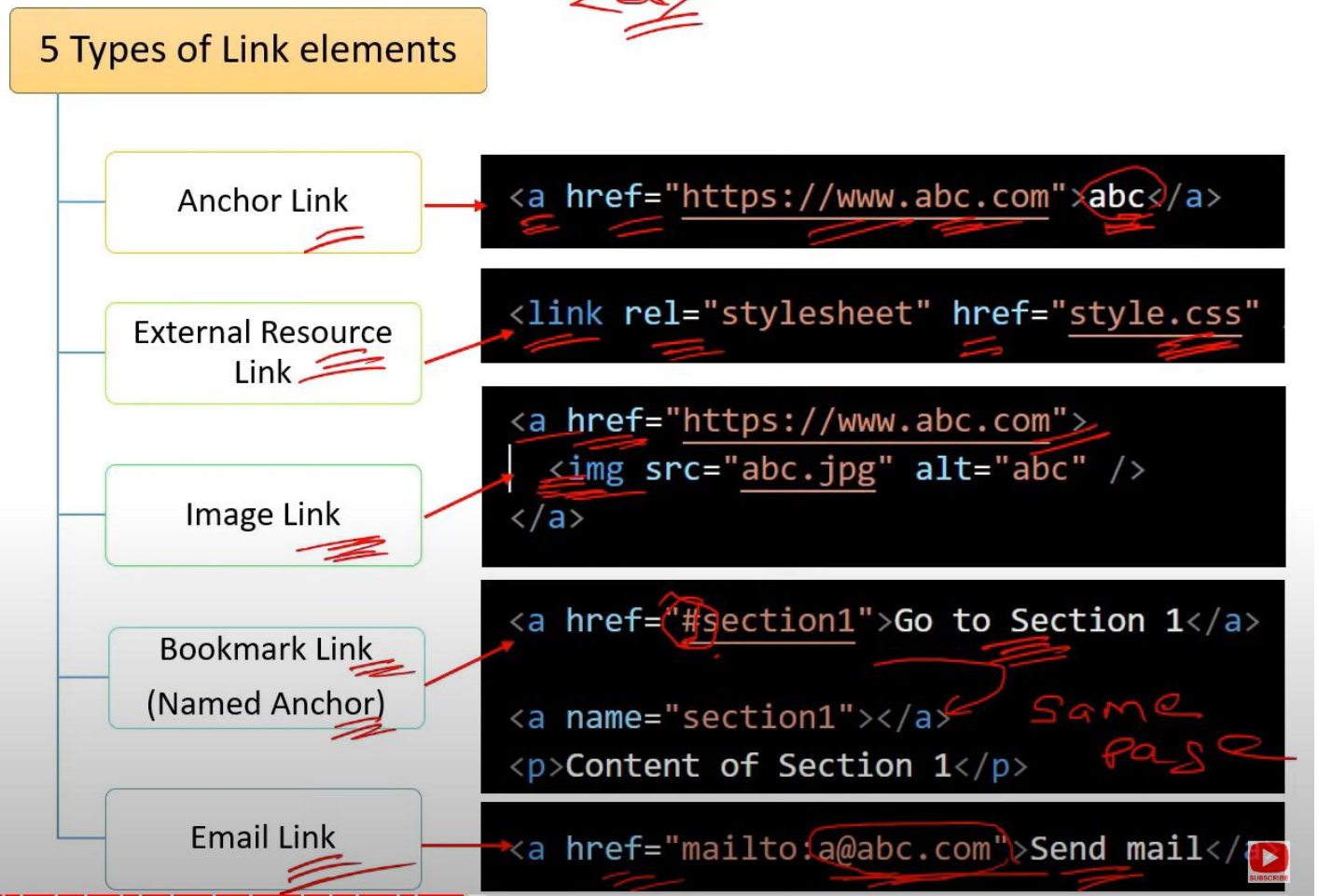


53:58 / 1:41:16 • HTML Links and Navigation ›

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are the 5 Types of Links in HTML? V. IMP.

1. Anchor Link (<a>) used for navigating from one webpage to another.
2. External Resource Link (<link>) used in the <head> section to connect to an external resource like a stylesheet or an icon.
3. Image Link (<img>) used to create a clickable image that leads to another webpage.
4. Bookmark Link (Named Anchor) points to a specific location within a webpage using a named anchor.
5. Email Link creates a clickable link that opens the user's default email client with a pre-filled email address.



TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the difference between Absolute and Relative URLs? V. IMP.

❖ Absolute URLs:

1. Absolute URLs provide the **complete web address** of a resource.
2. Absolute URLs are typically used to link to resources on **different websites**.

```
<h2>Absolute URLs</h2>
<ul>
  <li><a href="http://www.abc.com">HTTP URL</a></li>
  <li><a href="ftp://ftp.abc.com/doc.pdf">FTP URL</a></li>
  <li><a href="mailto:info@abc.com">Mailto URL</a></li>
</ul>
```

❖ Relative URLs:

1. Relative URLs specify the location of a resource in **relation** to the current document. Full url is not required.
2. They are used when linking to resources within the **same website**.

```
<!-- example/index.html -->
<h2>Relative URLs</h2>
<ul>
  <li><a href="page.html">Same Directory</a></li>
  <li><a href="sub/page.html">Subdirectory</a></li>
  <li><a href="../page.html">Parent Directory</a></li>
  <li><img src="images/image.jpg" alt="Image" /></li>
  <li><link rel="stylesheet" href="styles/styles.css"/></li>
  <li><script src="scripts/script.js"></script></li>
</ul>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the purpose of the <nav> element in HTML?

- <nav> element in HTML is used to define a section of a web page that contains **navigation links**.
- <nav> element provide clear navigation structure and keep navigation separate from other body content which is good from SEO perspective.

```
<body>
  <header>
    <h1>Website Header</h1>
  </header>
  <nav>
    <ul>
      <li><a href="home.html">Home</a></li>
      <li><a href="about.html">About</a></li>
      <li><a href="services.html">Services</a></li>
      <li><a href="contact.html">Contact</a></li>
    </ul>
  </nav>
</body>
```

- Home
- About
- Services
- Contact

1:01:36 / 1:41:16 • Absolute vs Relative URLs >

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the purpose of the <nav> element in HTML?

- <nav> element in HTML is used to define a section of a web page that contains **navigation links**.
- <nav> element provide clear navigation structure and keep navigation separate from other body content which is good from SEO perspective.

```
<body>
  <header>
    <h1>Website Header</h1>
  </header>
  <nav>
    <ul>
      <li><a href="home.html">Home</a></li>
      <li><a href="about.html">About</a></li>
      <li><a href="services.html">Services</a></li>
      <li><a href="contact.html">Contact</a></li>
    </ul>
  </nav>
</body>
```

- Home
- About
- Services
- Contact

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is a Fragment Identifier in a URL? V. IMP.

- A fragment identifier is used to navigate to a **specific section** of the same webpage.
- Fragment identifier is preceded by a **# (hash)** symbol.

Go to Section 1

Go to Section 2

Section 1

Content for Section 1.

Section 2

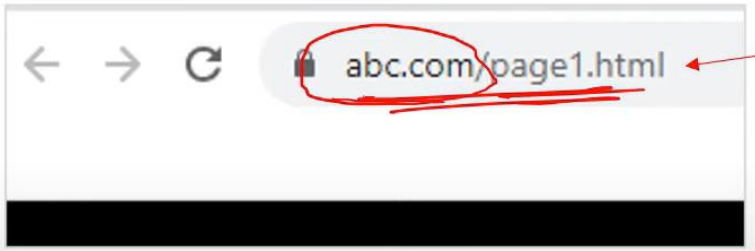
```
<body>
  <a href="#section1">Go to Section 1</a>
  <a href="#section2">Go to Section 2</a>

  <section id="section1">
    <h2>Section 1</h2>
    <p>Content for Section 1.</p>
    <div style="height: 500px">Test</div>
  </section>

  <section id="section2">
    <h2>Section 2</h2>
    <p>Content for Section 2.</p>
  </section>
</body>
```

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What is the purpose of the <base> element in HTML.?

- The <base> element in HTML is used to specify a **base URL** for relative URLs within a document.
- The <base> element is typically placed within the **<head>** section.

```
<html lang="en">
  <head>
    <title>Base Element Example</title>
    <base href="https://www.abc.com/" />
  </head>
  <body>
    <a href="page1.html">Link to Page 1</a>
    <a href="page2.html">Link to Page 2</a>
  </body>
</html>
```



TOP 50 HTML CSS Bootstrap Interview Questions and Answers

6. HTML Lists & Tables

Q. What are the different Types of Lists in HTML?

Q. What is a Nested List in HTML?

Q. What are table, tr, th, td elements? What are table advantages & disadvantages?

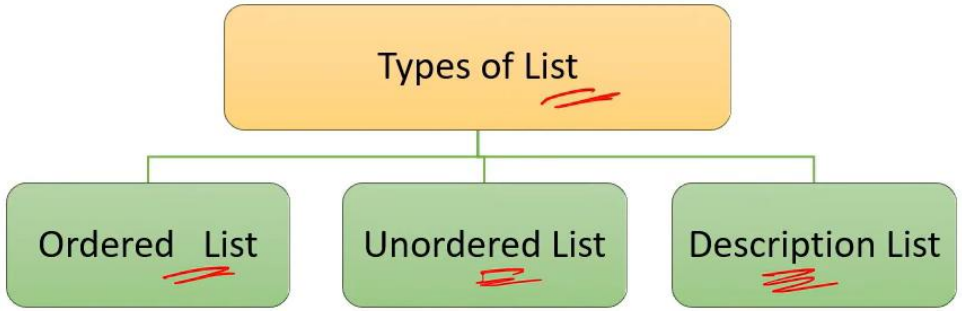
Q. What is the colspan attribute in HTML?

Q. What is the best way to add a border to a table, column and cell?

1:04:29 / 1:41:16 • Absolute vs Relative URLs ›

TOP 50 HTML CSS Bootstrap Interview Questions and Answers

Q. What are the different Types of Lists in HTML? V. IMP.



1. An ordered list is a list where the items are **numbered or ordered** in a specific sequence.
2. An unordered list is a list where the items are **marked with bullets**.
3. A description list is used to display a list of terms along with their **corresponding descriptions** or definitions.

Ordered List:

1. Item 1
2. Item 2

```
<h2>Ordered List:</h2>
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
</ol>
```

Unordered List:

- Item A
- Item B

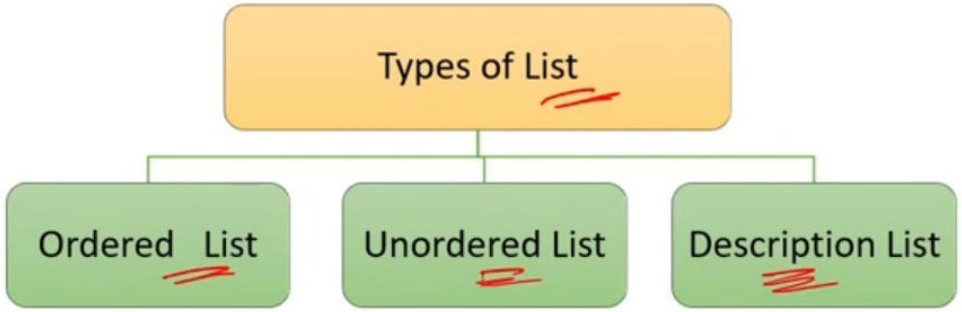
```
<h2>Unordered List:</h2>
<ul>
  <li>Item A</li>
  <li>Item B</li>
</ul>
```

Description List:

Term 1
Description 1
Term 2
Description 2

```
<h2>Description List:</h2>
<dl>
  <dt>Term 1</dt>
  <dd>Description 1</dd>
  <dt>Term 2</dt>
  <dd>Description 2</dd>
</dl>
```

Q. What are the different Types of Lists in HTML? V. IMP.



1. An ordered list is a list where the items are **numbered or ordered** in a specific sequence.
2. An unordered list is a list where the items are **marked with bullets**.
3. A description list is used to display a list of terms along with their **corresponding descriptions** or definitions.

Ordered List:

1. Item 1
2. Item 2

```
<h2>Ordered List:</h2>
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
</ol>
```

Unordered List:

- Item A
- Item B

```
<h2>Unordered List:</h2>
<ul>
  <li>Item A</li>
  <li>Item B</li>
</ul>
```

Description List:

Term 1
Description 1
Term 2
Description 2

```
<h2>Description List:</h2>
<dl>
  <dt>Term 1</dt>
  <dd>Description 1</dd>
  <dt>Term 2</dt>
  <dd>Description 2</dd>
</dl>
```

Q. What is a Nested List in HTML?

- A nested list in HTML is a list that is placed within another list item.

Technologies

- Backend
  - Java
  - .NET
  - Node
- Frontend
  - HTML
  - JS
  - React

```
<h1>Technologies</h1>
<ul>
  <li>
    Backend
    <ul>
      <li>Java</li>
      <li>.NET</li>
      <li>Node</li>
    </ul>
  </li>
  <li>
    Frontend
    <ul>
      <li>HTML</li>
      <li>JS</li>
      <li>React</li>
    </ul>
  </li>
</ul>
```

Q. What are table, tr, th, td elements? What are table advantages & disadvantages? V. IMP.

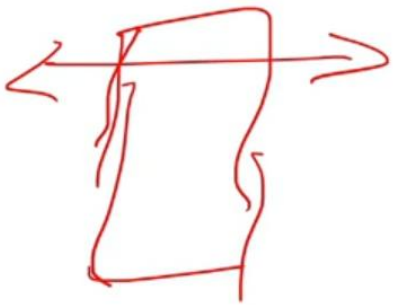
❖ Table elements:

1. <table> is the container for the entire table.
2. <tr>(table row) is used to define a row in the table.
3. <th>(table header) is used to represent the column headers.
4. <td>(table data) is used to represent the regular cells in a table.

Tables are a powerful tool for styling and displaying structured data.

Tables multiple column's structure is not good for mobile devices(not responsive).

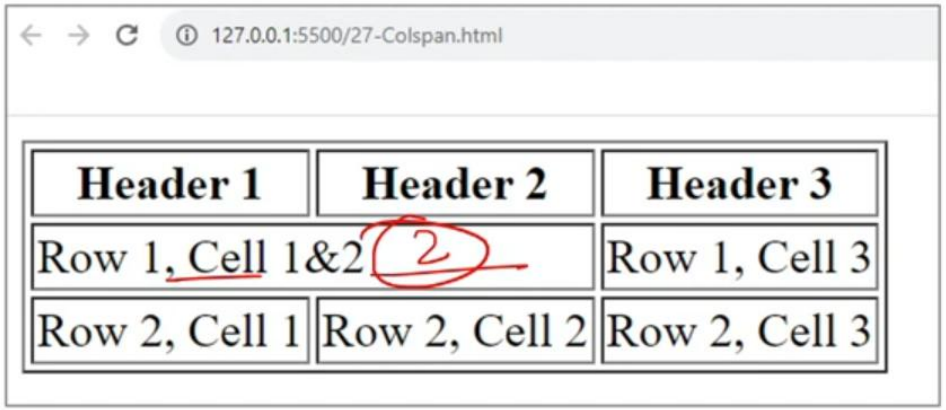
Header 1	Header 2
Row 1, Cell 1	Row 1, Cell 2
Row 2, Cell 1	Row 2, Cell 2



```
<body>
  <table border="1">
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
    </tr>
    <tr>
      <td>Row 1, Cell 1</td>
      <td>Row 1, Cell 2</td>
    </tr>
    <tr>
      <td>Row 2, Cell 1</td>
      <td>Row 2, Cell 2</td>
    </tr>

  </table>
</body>
```

Q. What is the colspan attribute in HTML?

- The colspan attribute is used to **merge multiple cells** horizontally into a single cell.
- colspan attribute is applicable to **<th>** and **<td>** only.




```
<body>
  <table border="1">
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
      <th>Header 3</th>
    </tr>
    <tr>
      <td colspan="2">Row 1, Cell 1&2</td>
      <td>Row 1, Cell 3</td>
    </tr>
    <tr>
      <td>Row 2, Cell 1</td>
      <td>Row 2, Cell 2</td>
      <td>Row 2, Cell 3</td>
    </tr>
  </table>
</body>
```

Q. What is the best way to add a border to a table, column and cell?

- Setting the common style for table, tr, th and td, then all the multiple tables in your webpage will follow consistent same style and format.

Header 1	Header 2
Row 1, Cell 1	Row 1, Cell 2
Row 2, Cell 1	Row 2, Cell 2

```
<head>
  <title>Styled Table</title>
  <style>
    table {
      border: 2px solid #000;
      width: 25%;
    }

    th,
    td {
      border: 1px solid #000;
      padding: 8px;
      text-align: center;
    }
  </style>
</head>
```

```
<body>
  <table>
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
    </tr>
    <tr>
      <td>Row 1, Cell 1</td>
      <td>Row 1, Cell 2</td>
    </tr>
    <tr>
      <td>Row 2, Cell 1</td>
      <td>Row 2, Cell 2</td>
    </tr>
  </table>
</body>
```

1. CSS Basics & Implementation

Q. What is CSS? What are the 5 ways to Implement CSS in HTML?

Q. What is Inline Style in CSS? When to use it in real applications?

Q. What is Internal Stylesheet in CSS? When to use it in real applications?

Q. What is External Stylesheet in CSS? When to use it in real applications?

Q. What are the 5 advantages of external stylesheet?

Q. How do you Include CSS in a webpage or HTML?

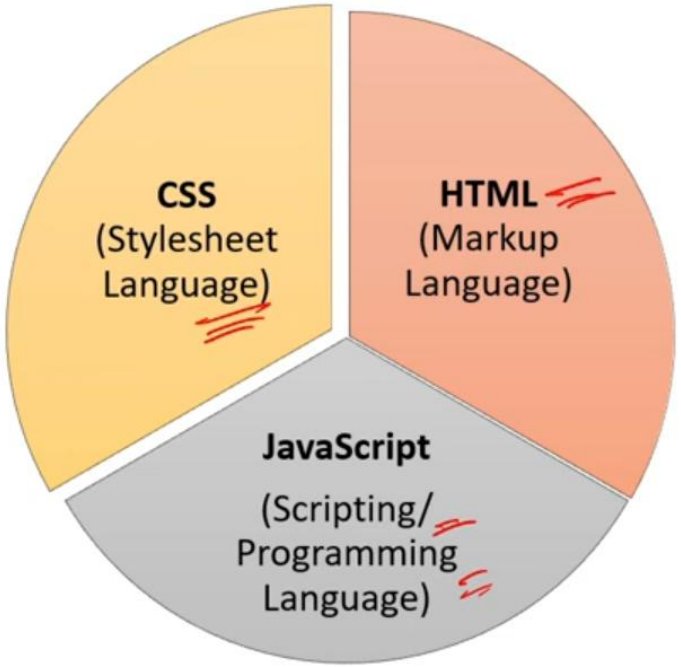
Q. How to implement CSS using @import rule?

Q. What is CSS Preprocessors? What is SCSS?

Q. What are the 3 Types of CSS Preprocessors?

Q. What is CSS? What are the 5 ways to Implement CSS in HTML? V. IMP.

- CSS(Cascading Style Sheets) is a **stylesheet language** used to control the **presentation** of web pages.



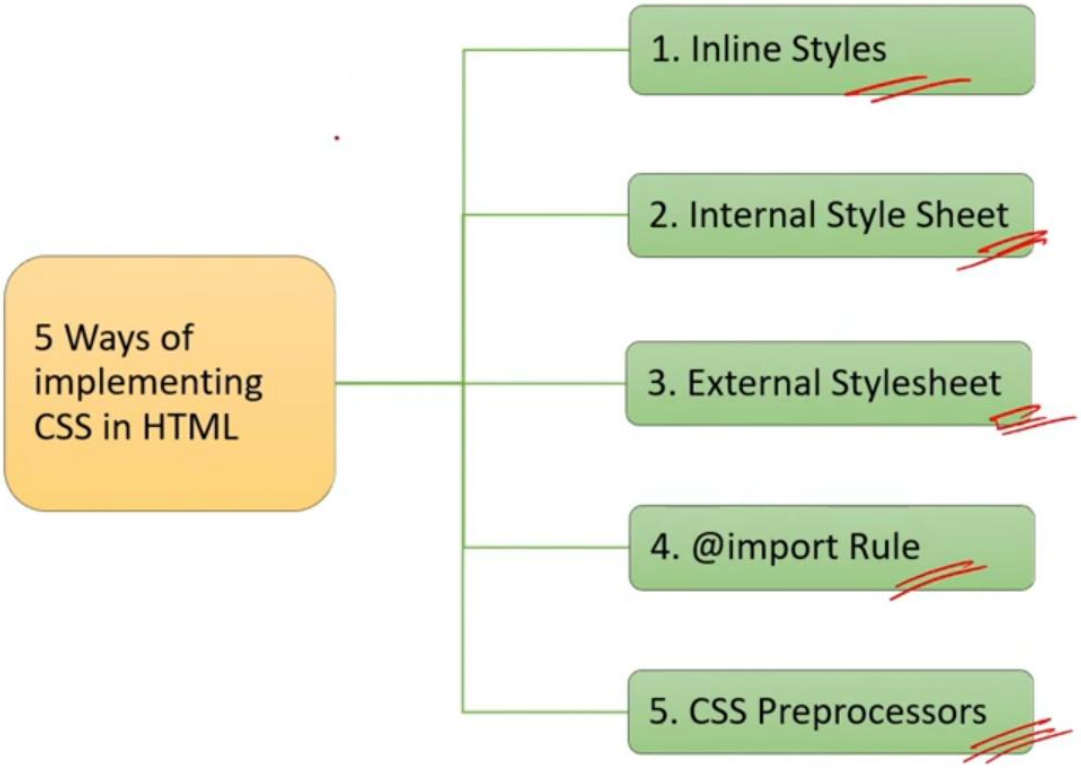

```html
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Inline CSS Example</title>
  </head>
  <body>
    <div style="background-color: #f0f0f0;">
      <h1 style="color: blue">Interview</h1>
      <p style="font-size: 30px">Happy</p>
    </div>
  </body>
</html>
```

Interview

Happy

CSS

Q. What is CSS? What are the 5 ways to Implement CSS in HTML? V. IMP.



Q. What is Inline Style in CSS? When to use it in real applications?

- Inline Styles apply styles directly to individual HTML elements using the **style attribute**.
- This method is suitable for applying styles to a **single element**.

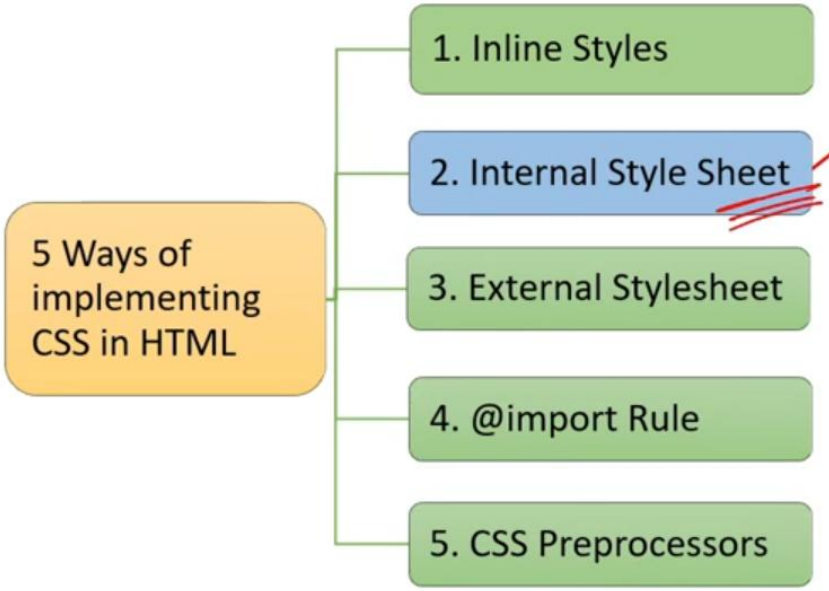
5 Ways of implementing CSS in HTML

1. Inline Styles
2. Internal Style Sheet
3. External Stylesheet
4. @import Rule
5. CSS Preprocessors

```html
<html>
  <head> </head>
  <body>
    <h1 style="text-align: center">Inline Styles</h1>
    <p style="color: green">Paragraph</p>
  </body>
</html>
```

Q. What is Internal Stylesheet in CSS? When to use it in real applications?

- Internal Style Sheets can be implemented by adding the **<style> element** in the <head> section of HTML.
- This method is suitable for **smaller projects.**




```
<html>
  <head>
    <style>
      div {
        text-align: left;
      }
      .para {
        color: green;
      }
    </style>
  </head>
  <body>
    <div>Internal Style Sheet</div>
    <p class="para">Paragraph</p>
  </body>
</html>
```

Q. What is External Stylesheet in CSS? When to use it in real applications? V. IMP.

- In External Stylesheet, a **separate CSS file** is created(.css extension) and link it in the HTML file using the **<link> element**.
- This is most common and **recommended**.

5 Ways of implementing CSS in HTML

1. Inline Styles
2. Internal Style Sheet
3. External Stylesheet
4. @import Rule
5. CSS Preprocessors

2-CssTypes3.css

```css
h1 {
    text-align: center;
}
p {
    color: green;
}
```

```html
<html>
  <head>
    <link rel="stylesheet" href="2-CssTypes3.css">
  </head>
  <body>
    <h1>Internal Style Sheet</h1>
    <p>Paragraph</p>
  </body>
</html>
```

Q. What are the 5 advantages of External Stylesheet? V. IMP.

5 Advantages of CSS:

1. **Separation** of content(HTML) and presentation(Style).
2. **Reusability** of CSS classes in multiple elements.
3. Keeps things organized and **structured**.
4. Adapts to Different **Devices**.
5. Improves Website **Speed.**

HTML

1-Css2.css > ...

```css
.container {
    background-color: #f0f0f0;
    padding: 20px;
}
.heading {
    color: #333;
}
.paragraph {
    font-family: Arial, sans-serif;
}
```

HTML

Q. How do you Include CSS in a webpage or HTML?

- 1. External Stylesheet: By using the **<link> element** in the <head> section of the HTML document.

```
<head>
  <title>Sass Example</title>
  <link rel="stylesheet" href="4-PreProcessor.css" />
</head>
```

- 2. Internal Style Sheet: By using the **<style> element** in the <head> section.

```
<head>
  <style>
    h1 {
      text-align: center;
    }
  </style>
</head>
```

- 3. Inline Styles: Apply styles directly to individual HTML elements using the **style attribute**.

```
<body>
  <h1 style="text-align: center">Inline Styles</h1>
  <p style="color: green">Paragraph</p>
</body>
```

Q. How to implement CSS using @import rule?

- @import rule is to include **external CSS** file to main CSS file.
- This method is less commonly used due to potential **performance issues**.

```css
/* 3-Main.css */
@import url("3-External.css");

p {
  font-size: 18px;
}
```

```css
/* 3-External.css */
h1 {
  color: blue;
}
```

5 Ways of implementing CSS in HTML

1. Inline Styles
2. Internal Style Sheet
3. External Stylesheet
4. @import Rule
5. CSS Preprocessors

```html
<html>
  <head>
    <link rel="stylesheet" href="3-Main.css" />
  </head>
  <body>
    <h1>Imported Styles</h1>
    <p>Paragraph</p>
  </body>
</html>
```

Q. What is CSS Preprocessors? What is SCSS? V. IMP.

- CSS preprocessors are **scripting languages** that provide additional syntax and features that are not available in traditional CSS. For example, variables, loops, and conditional statements.
- SCSS is a type of CSS preprocessor.

4-PreProcessor.scss

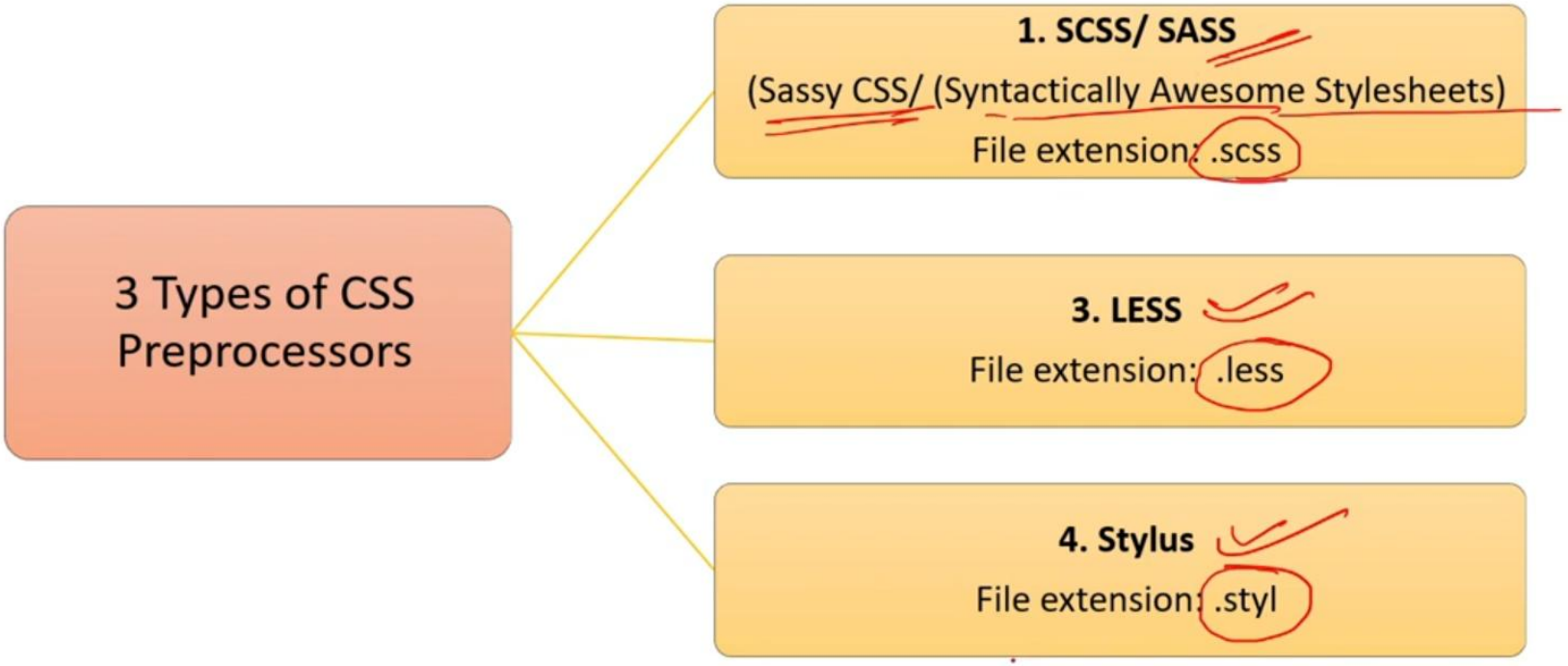
```scss
$primary-color: blue;
$base-font-size: 16px;

.heading {
    font-size: $base-font-size * 1.2;
    color: $primary-color;
}
```

4-PreProcessor.css

```css
/* Generated css from scss */

.heading {
  font-size: 19.2px;
  color: blue;
}
```

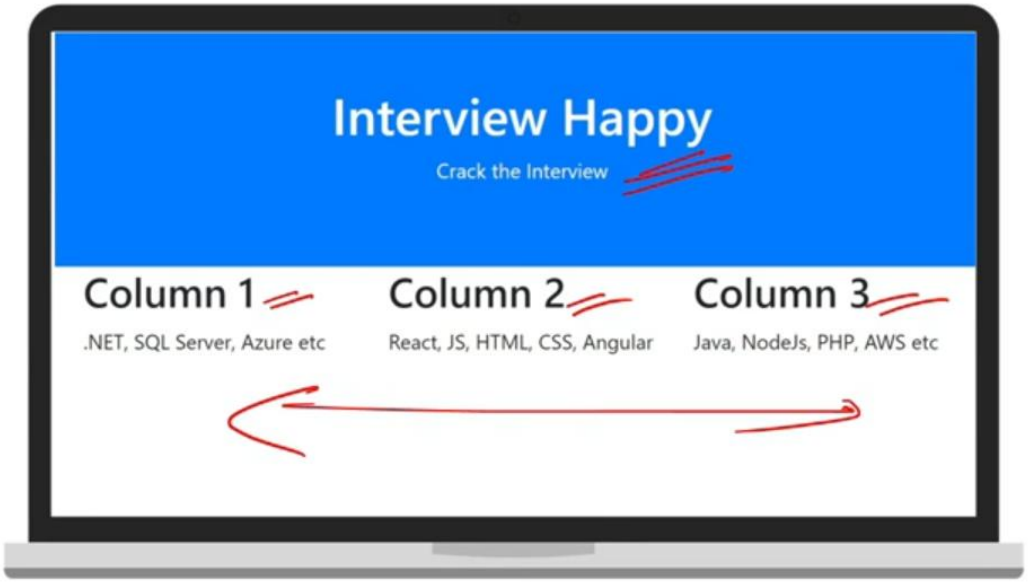
```html
<html>
  <head>
    <title>Sass Example</title>
    <link rel="stylesheet" href="4-PreProcessor.css" />
  </head>
  <body>
    <h1 class="heading">Interview Happy</h1>
    <p class="paragraph">Crack Interviews</p>
  </body>
</html>
```

Q. What are the 3 Types of CSS Preprocessors?

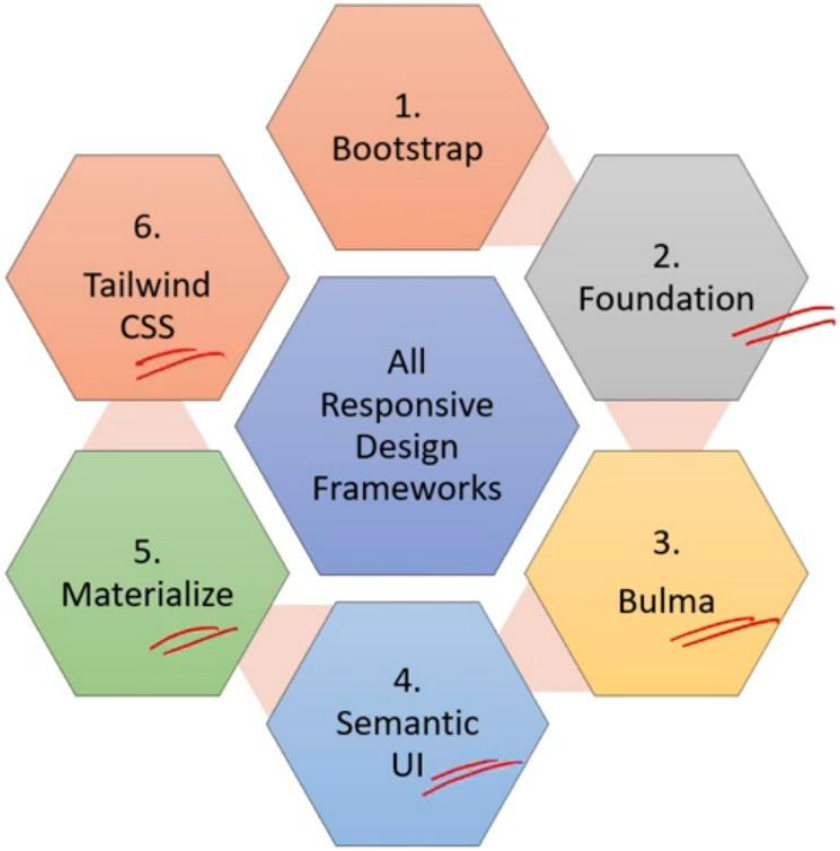


Q. What is Bootstrap? What are the other 5 responsive design frameworks? V. IMP.

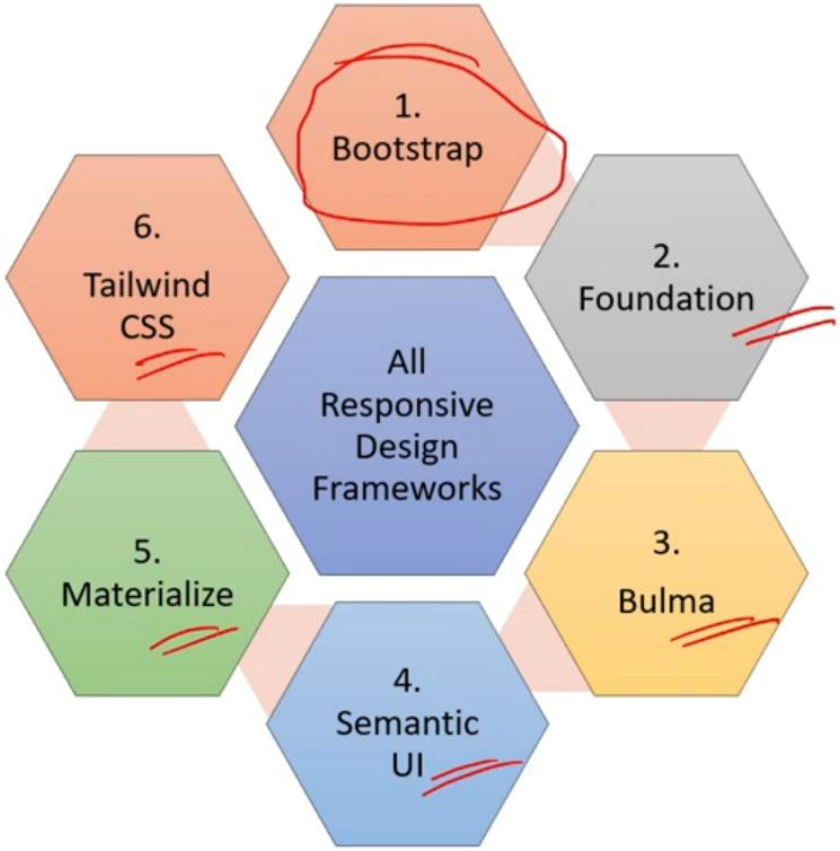
- Bootstrap is a popular open-source front-end framework which provides responsive and mobile-first CSS.



Q. What is Bootstrap? What are the other 5 responsive design frameworks? **V. IMP.**



Q. What is Bootstrap? What are the other 5 responsive design frameworks? **V. IMP.**



Q. What are the 5 Advantages of using Bootstrap?

1. Rapid Development

- Bootstrap provides a wide range of ready-to-use components like navigation bars, buttons, forms, modals, and more. This saves time and effort in writing code from scratch.

2. Responsive Design

- Bootstrap is built with a mobile-first approach, meaning it prioritizes designing for smaller screens first and then scales up for larger screens. This ensures websites with Bootstrap is responsive for every device.

3. Cross-Browser Compatibility

- Bootstrap provides a consistent set of styles and components across different browsers, which helps in reducing cross-browser compatibility issues.

4. Large Community and Support

- This means there are plenty of resources, documentation, tutorials, and forums available for assistance.

5. Accessibility

- Bootstrap incorporates best practices for web accessibility, making it easier to create websites that are usable by people with disabilities.

Q. What are the 2 ways to include Bootstrap framework for your website.

2 ways to include Bootstrap:

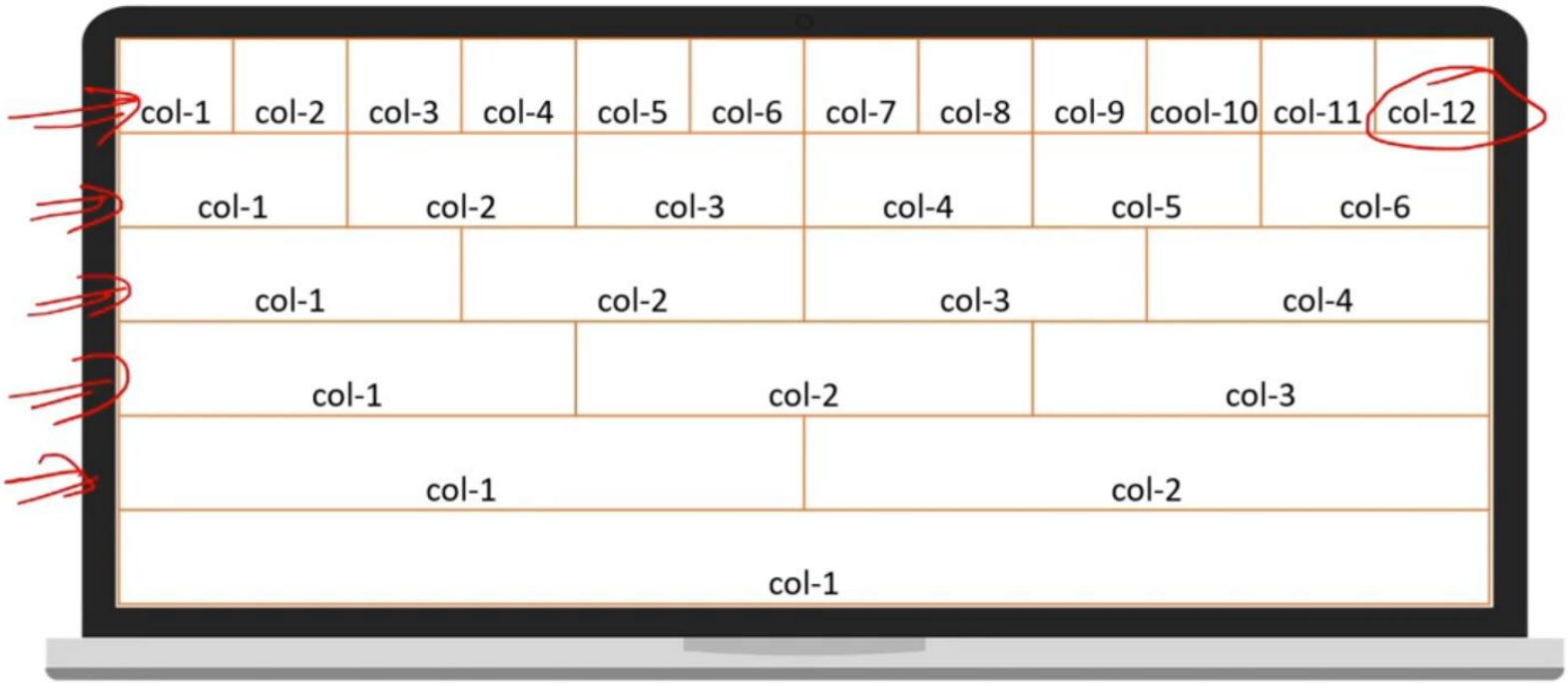
1. Put the link of Bootstrap CDN (Content Delivery Network) in head section.

2. Download the Bootstrap files and add them locally in your HTML document.

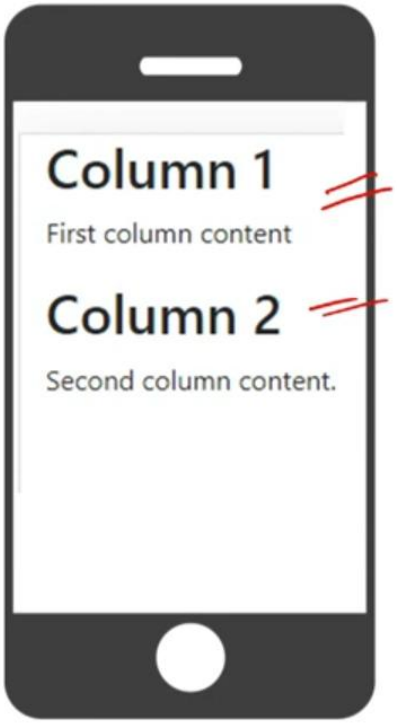
```
<head>
  <title>Bootstrap</title>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <!-- Add Bootstrap CSS via CDN -->
  <link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
  />
</head>
```

Q. Explain the Grid System in Bootstrap? V. IMP.

- Grid system is 12-column layout and is designed to adapt to various screen sizes.



Q. Explain the Grid System in Bootstrap? **V. IMP.**



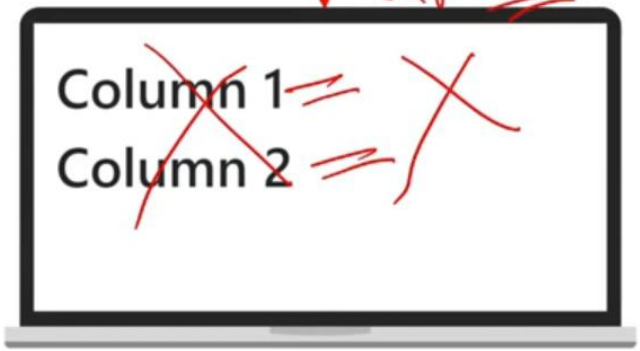

```
<body>
  <div class="container">
    <div class="row">
      <div class="col-md-6">
        <h2>Column 1</h2>
        <p>First column content</p>
      </div>
      <div class="col-md-6">
        <h2>Column 2</h2>
        <p>Second column content.</p>
      </div>
    </div>
  </div>
</body>
```

Q. What is the difference between col-xs, col-sm, col-md, col-lg & col-xl?

1. col-xs (Extra Small): Applies to extra small screens (phones).
2. col-sm (Small): Applies to small screens (tablets).
3. col-md (Medium): Applies to medium screens (laptops).
4. col-lg (Large): Applies to large screens (larger desktops).
5. col-xl (Extra Large): Applies to extra large screens (large TVs).

```
<div class="container">
  <div class="row">
    <div class="col-md-6">
      <h2>Column 1</h2>
    </div>
    <div class="col-md-6">
      <h2>Column 2</h2>
    </div>
  </div>
</div>
```


```
<div class="container">
  <div class="row">
    <div class="col-xl-6">
      <h2>Column 1</h2>
    </div>
    <div class="col-xl-6">
      <h2>Column 2</h2>
    </div>
  </div>
</div>
```

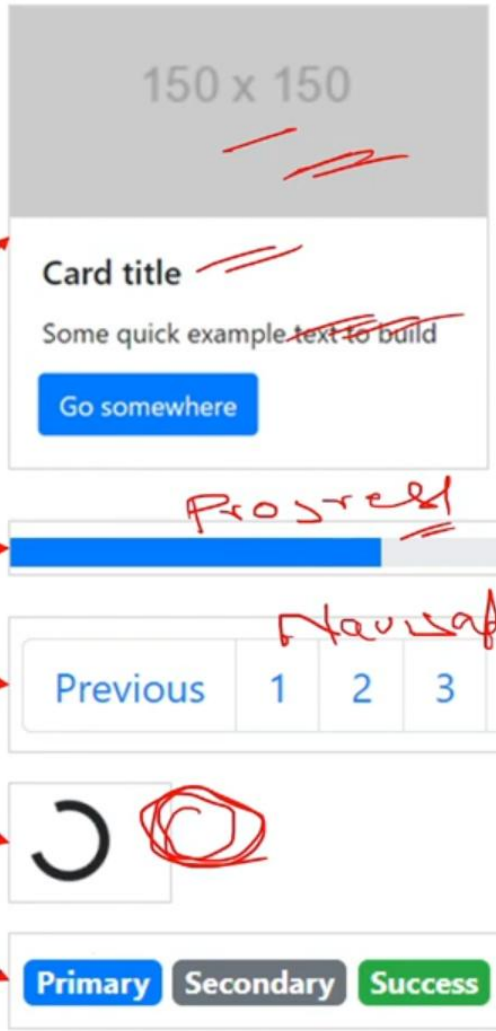


Q. What are Bootstrap Components? What are Top 10 bootstrap components? V. IMP.

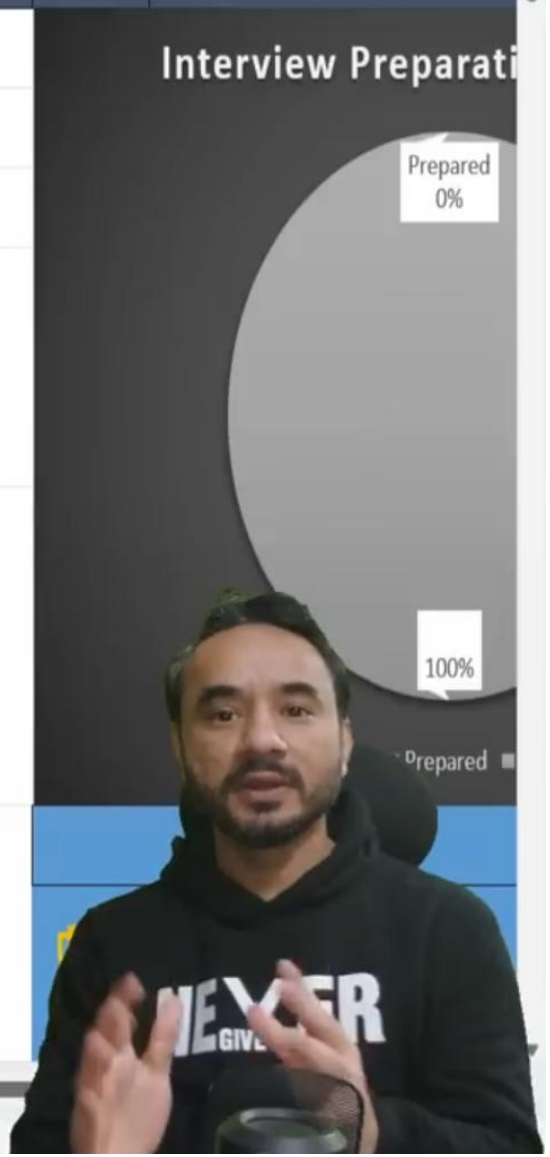
- Bootstrap components are **pre-designed elements** that can be easily integrated into a web project.

Top 10 Bootstrap components

1. Buttons
2. Navbar
3. Forms
4. Carousel
5. Cards
6. Progress Bars
7. Pagination
8. Spinners
9. Badges
10. Modals



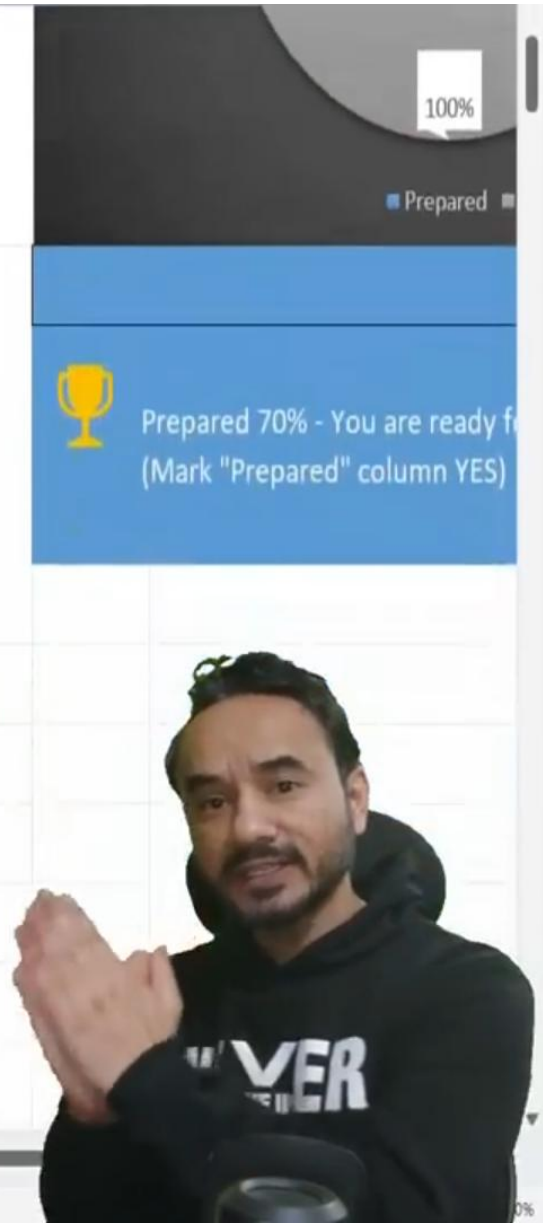
	Topic	Question	Prepared?	Answer Key
2	React-Basics - I	Q. What is **React**? What is the **Role of React** in software development?	NO	View Answer
3	React-Basics - I	Q. What are the **Key Features** of React?	NO	View Answer
4	React-Basics - I	Q. What is **DOM**? What is the difference between **HTML** and **DOM**?	NO	View Answer
5	React-Basics - I	Q. What is **Virtual DOM**? Difference between **DOM** and **Virtual DOM**?	NO	View Answer
6	React-Basics - I	Q. What are **React Components**? What are the main elements of it?	NO	View Answer
7	React-Basics - I	Q. What is **SPA(Single Page Application)**?	NO	View Answer
8	React-Basics - I	Q. What are the **5 Advantages** of React?	NO	View Answer
9	React-Basics - I	Q. What are the **Disadvantages** of React?	NO	View Answer
10	React-Basics - I	Q. What is the role of **JSX** in React? (3 points)	NO	View Answer
11	React-Basics - I	Q. What is the difference between **Declarative** & **Imperative** syntax?	NO	View Answer
12	React-Basics - II	Q. What is **Arrow Function Expression** in JSX?	NO	View Answer
13	React-Basics - II	Q. How to **Setup** React first project?	NO	View Answer
14	React-Basics - II	Q. What are the **Main Files** in a React project?	NO	View Answer



Top 200 Questions

Ready Accessibility: Investigate

9	React-Basics - I	Q. What are the **Disadvantages** of React?	NO	View Answer
10	React-Basics - I	Q. What is the role of **JSX** in React? (3 points)	NO	View Answer
11	React-Basics - I	Q. What is the difference between **Declarative** & **Imperative** syntax?	NO	View Answer
12	React-Basics - II	Q. What is **Arrow Function Expression** in JSX?	NO	View Answer
13	React-Basics - II	Q. How to **Setup** React first project?	NO	View Answer
14	React-Basics - II	Q. What are the **Main Files** in a React project?	NO	View Answer
15	React-Basics - II	Q. How React App **Load** and display the components in browser?	NO	View Answer
16	React-Basics - II	Q. What is the difference between **React** and **Angular**?	NO	View Answer
17	React-Basics - II	Q. What are other **5 JS frameworks** other than React?	NO	View Answer
18	React-Basics - II	Q. Whether React is a **Framework** or a **Library**? What is the difference?	NO	View Answer
19	React-Basics - II	Q. How React provides **Reusability** and **Composition**?	NO	View Answer
20	React-Basics - II	Q. What are **State, Stateless, Stateful and State Management** terms?	NO	View Answer
21	React-Basics - II	Q. What are **Props** n JSX?	NO	View Answer
22	React Project - Main Files & F	Q. What is **NPM**? What is the role of **node_modules** folder?	NO	View Answer



Top 200 Questions

Ready Accessibility: Investigate

22	React Project - Main Files & F	Q. What is **NPM**? What is the role of **node_modules** folder?	NO	View Answer
23	React Project - Main Files & F	Q. What is the role of **public folder** in React?	NO	View Answer
24	React Project - Main Files & F	Q. What is the role of **src folder** in React?	NO	View Answer
25	React Project - Main Files & F	Q. What is the role of **index.html** page in React?	NO	View Answer
26	React Project - Main Files & F	Q. What is the role of **index.js file** and **ReactDOM** in React?	NO	View Answer
27	React Project - Main Files & F	Q. What is the role of **App.js** file in React?	NO	View Answer
28	React Project - Main Files & F	Q. What is the role of **function** and **return** inside App.js?	NO	View Answer
29	React Project - Main Files & F	Q. Can we have a **function without a return** inside App.js?	NO	View Answer
30	React Project - Main Files & F	Q. What is the role of **export default** inside App.js?	NO	View Answer
31	React Project - Main Files & F	Q. Does the **file name** and the **component name** must be same in React?	NO	View Answer
32	JSX	Q. What is the **role of JSX** in React? (3 points)	NO	View Answer
33	JSX	Q. What are the **5 Advantages** of JSX?	NO	View Answer
34	JSX	Q. What is **Babel**?	NO	View Answer
35	JSX	Q. What is the role of **Fragment** in JSX?	NO	View Answer
36	JSX	Q. What is **Spread Operator** in JSX?	NO	View Answer

13R x 1C

Top 200 Questions

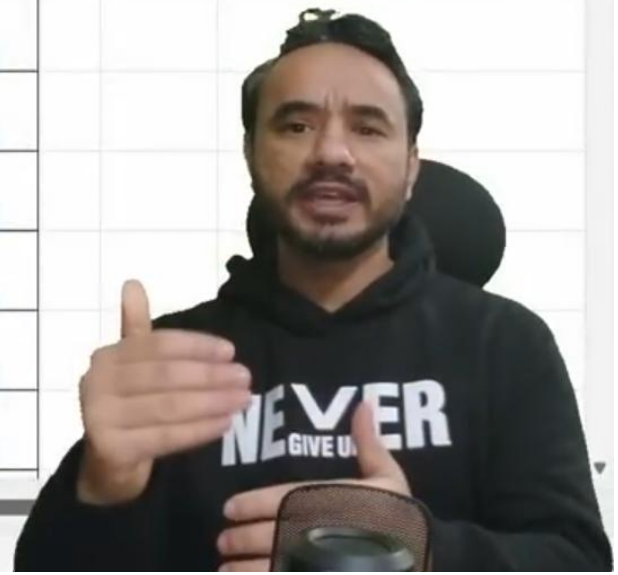
Ready Accessibility: Investigate

39	JSX	Q. Can a browser **read** a JSX File?	NO	View Answer
40	JSX	Q. What is **Transpiler**? What is the difference between **Compiler** & **Transpi**	NO	View Answer
41	JSX	Q. Is it possible to use **JSX without React**?	NO	View Answer
42	Components - Functional/ Cl	Q. What are **React Components**? What are the main elements of it?	NO	View Answer
43	Components - Functional/ Cl	Q. What are the **Types** of React components? What are **Functional Compor**	NO	View Answer
44	Components - Functional/ Cl	Q. How do you **pass data** between functional components in React?	NO	View Answer
45	Components - Functional/ Cl	Q. What is **Prop Drilling** in React?	NO	View Answer
46	Components - Functional/ Cl	Q. Why to **Avoid** Prop Drilling? In how many **ways** can avoid Prop Drilling?	NO	View Answer
47	Components - Functional/ Cl	Q. What are **Class Components** In React?	NO	View Answer
48	Components - Functional/ Cl	Q. How to **pass data between class components** in React?	NO	View Answer
49	Components - Functional/ Cl	Q. What is the role of **this keyword** in class components?	NO	View Answer
50	Components - Functional/ Cl	Q. What are the 5 differences btw **Functional components** and **Class comp**	NO	View Answer
51	Routing	Q. What is **Routing** and **Router** in React?	NO	View Answer
52	Routing	Q. How to **Implement Routing** in React?	NO	View Answer

47	Components - Functional/ Cl	Q. What are **Class Components** In React?	NO	View Answer
48	Components - Functional/ Cl	Q. How to **pass data between class components** in React?	NO	View Answer
49	Components - Functional/ Cl	Q. What is the role of **this keyword** in class components?	NO	View Answer
50	Components - Functional/ Cl	Q. What are the 5 differences btw **Functional components** and **Class comp**	NO	View Answer
51	Routing	Q. What is **Routing** and **Router** in React?	NO	View Answer
52	Routing	Q. How to **Implement Routing** in React?	NO	View Answer
53	Routing	Q. What are the roles of **<Routes>** & **<Route>** component in React Routing?	NO	View Answer
54	Routing	Q. What are **Route Parameters** in React Routing?	NO	View Answer
55	Routing	Q. What is the role of **Switch** Component in React Routing?	NO	View Answer
56	Routing	Q. What is the role of **exact prop** in React Routing?	NO	View Answer
57	Hooks -useState/ useEffect	Q. What are **React Hooks**? What are the **Top** React Hooks?	NO	View Answer
58	Hooks -useState/ useEffect	Q. What are **State, Stateless, Stateful** and **State Management** terms?	NO	View Answer
59	Hooks -useState/ useEffect	Q. What is the role of **useState()** hook and how it works?	NO	View Answer
60	Hooks -useState/ useEffect	Q. What is the role of **useEffect()**. How it works and what is its use?	NO	View Answer

Top 200 Questions

Ready Accessibility: Investigate



Topics Covered

1. React-Basics – I
2. React Basics – II
3. Project Files & Folders
4. JSX
5. Components (Functional/Class)
6. Routing
7. Hooks
8. Components LifeCycle Methods
9. Controlled Components
10. Code Splitting
11. React - Others

1: React-Basics - I

V. IMP.

Q1. What is React? What is the Role of React in software development?

Q2. What are the Key Features of React?

Q3. What is DOM? What is the difference between HTML and DOM?

Q4. What is Virtual DOM? Difference between DOM and Virtual DOM?

Q5. What are React Components? What are the main elements of it?

Q6. What is SPA(Single Page Application)?

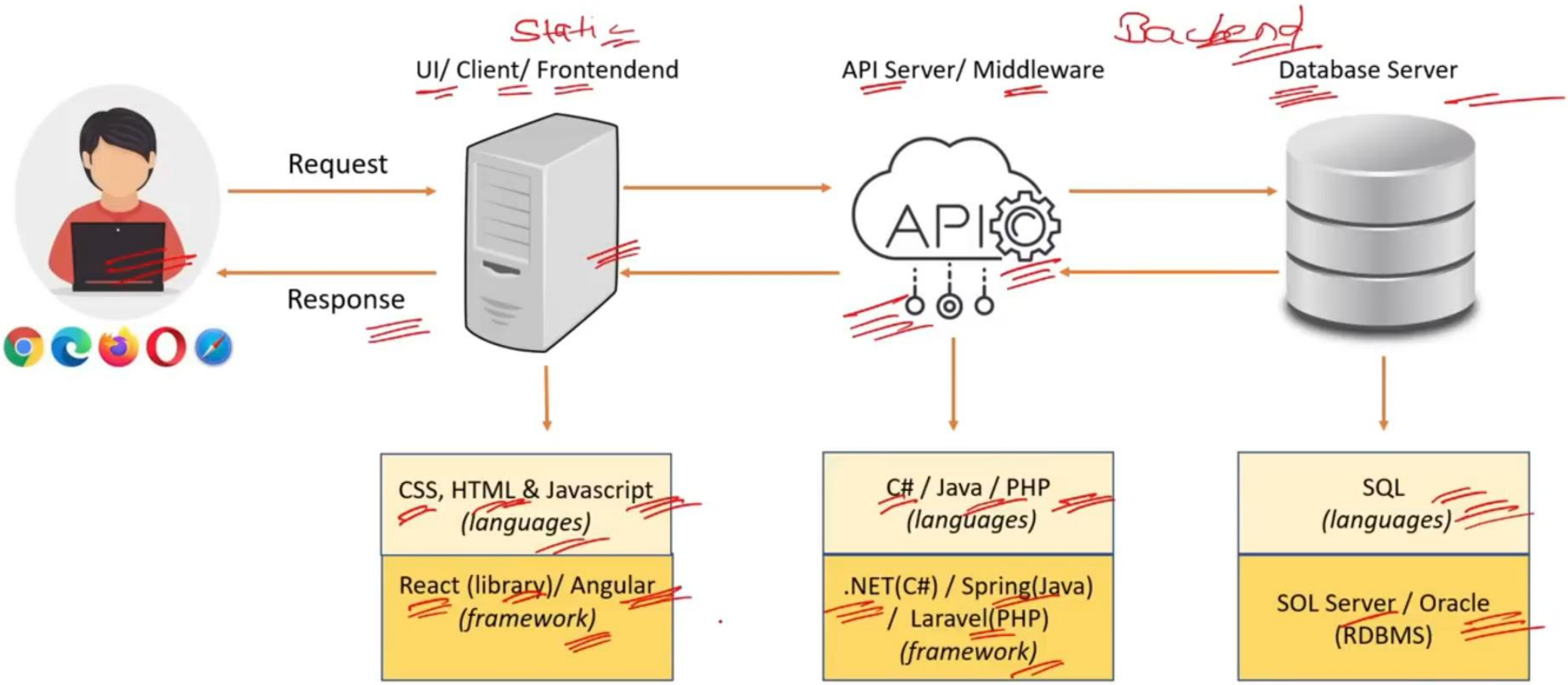
Q7. What are the 5 Advantages of React?

Q8. What are the Disadvantages of React?

Q9. What is the role of JSX in React? (3 points)

Q10. What is the difference between Declarative & Imperative syntax?

Q. What is React? What is the Role of React in software development? V. IMP.

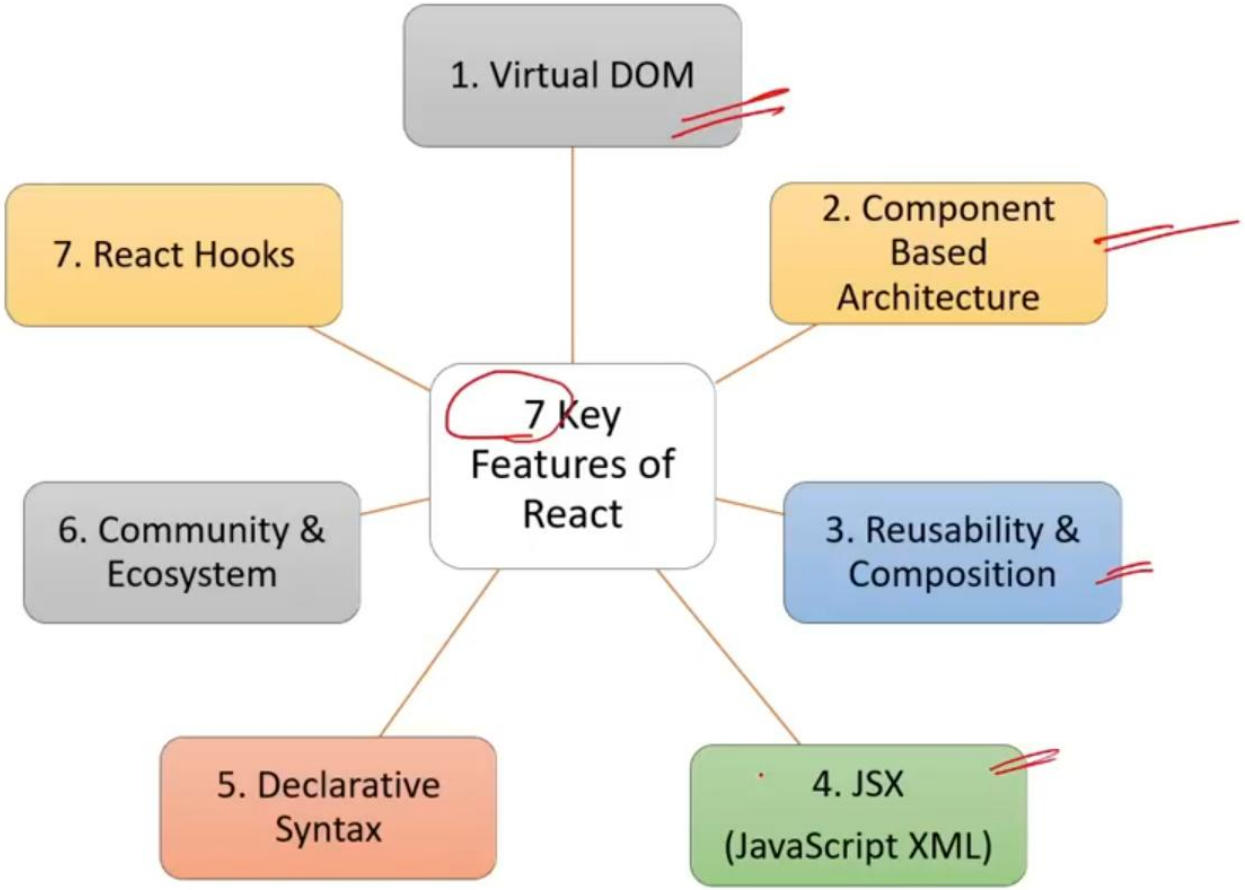


Q. What is React? What is the Role of React in software development? V. IMP.

❖ **3 Main Point about React:**

1. React is an open source Javascript Library.
2. React is used for building user interfaces(UI).
3. React simplifies the creation of SPA by using reusable components.

Q. What are the Key Features of React? V. IMP.

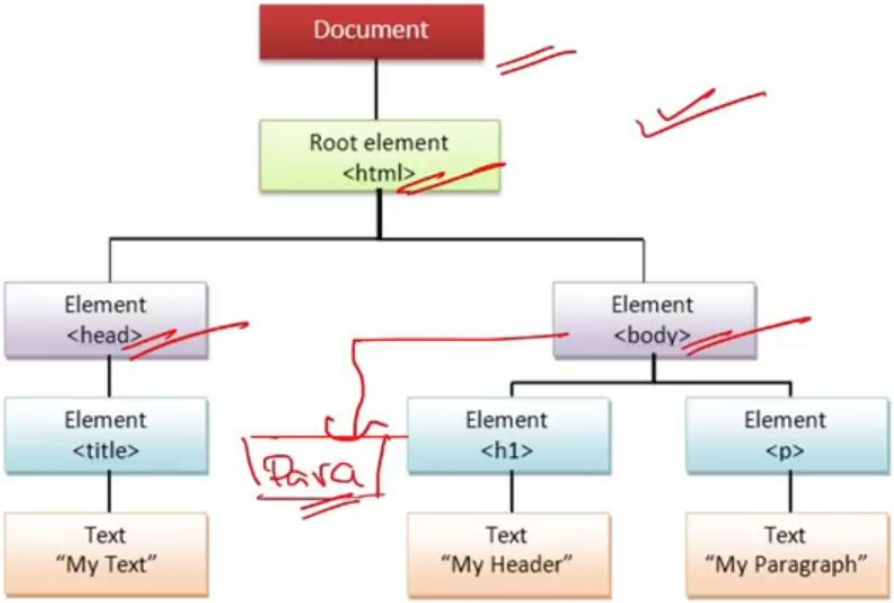


Q. What are the Key Features of React? V. IMP.

1. Virtual DOM:	React utilizes a virtual representation of the DOM, allowing efficient updates by **minimizing direct manipulation of the actual DOM**, resulting in improved performance.
2. Component-Based Architecture:	React structures user interfaces as modular, **reusable components**, promoting a more maintainable and scalable approach to building applications.
3. Reusability & Composition:	React enables the creation of reusable components that can be composed together, fostering a modular and efficient development process.
4. JSX (JavaScript XML):	JSX is a syntax extension for JavaScript used in React, allowing developers to write **HTML-like code** within JavaScript, enhancing readability and maintainability.
5. Declarative Syntax:	React have a declarative programming style(JSX), where developers **focus on "what" the UI should look like** and React handles the "how" behind the scenes. This simplify the code.
6. Community & Ecosystem:	React benefits from a vibrant and **extensive community**, contributing to a rich ecosystem of libraries, tools, and resources, fostering collaborative development and innovation.
7. React Hooks:	Hooks are functions that enable functional components to **manage state and lifecycle features**, providing a more concise and expressive way to handle component logic.

Q. What is DOM? What is the difference between HTML and DOM?

- DOM(Document Object Model) represents the web page as a **tree-like structure** which allows JavaScript to dynamically access and manipulate the content and structure of a web page.

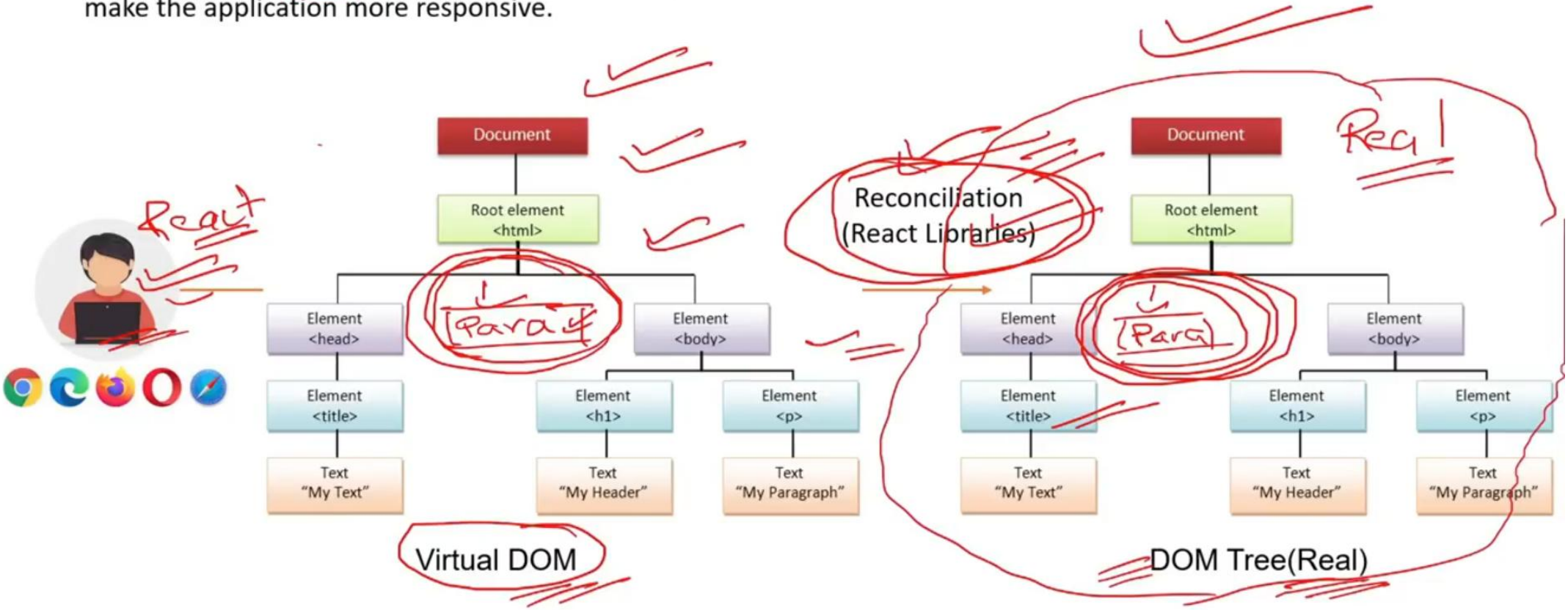


DOM Tree(Real)

Static HTML

Q. What is Virtual DOM? Difference between DOM and Virtual DOM? V. IMP.

- React uses a virtual DOM to efficiently update the UI **without re-render the entire page**, which helps improve performance and make the application more responsive.



Q. What is Virtual DOM? Difference between DOM and Virtual DOM? V. IMP.

DOM	Virtual DOM
1. DOM is actual representation of the webpage.	Virtual DOM is lightweight copy of the DOM.
2. Re-renders the entire page when updates occur.	Re-renders only the changed parts efficiently.
3. Can be slower, especially with frequent updates.	Optimized for faster rendering.
4. Suitable for static websites and simple applications	Ideal for dynamic and complex single-page applications with frequent updates

Q. What are React Components? What are the main elements of it? V. IMP.

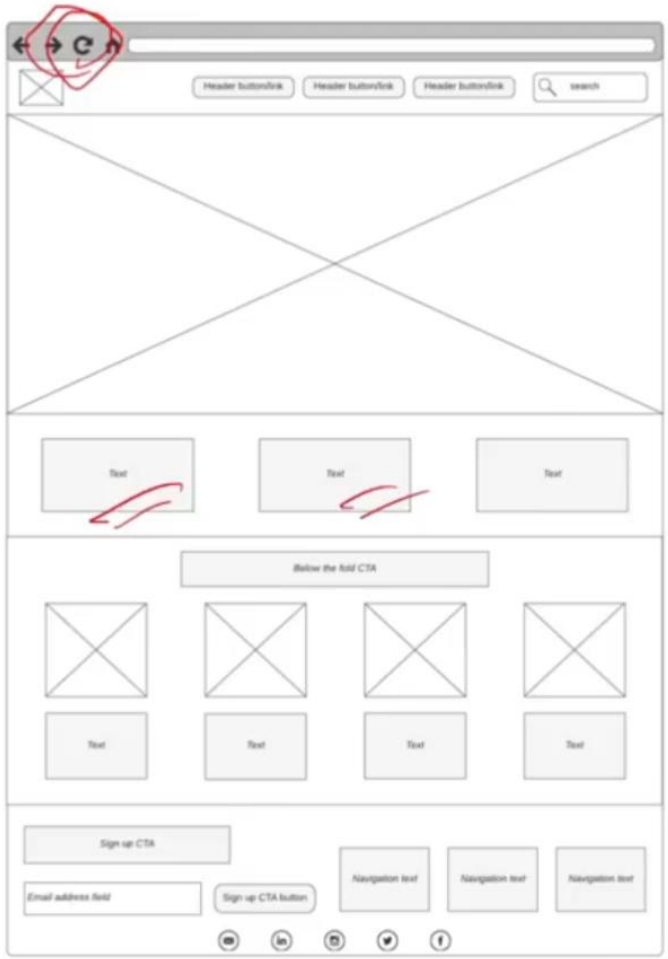
- In React, a component is a **reusable building block** for creating user interfaces.


```jsx
// 1. Import the React library
import React from "react";

// 2. Define a functional component
function Component() {
  // 3. Return JSX to describe the component's UI
  return (
    <div>
      <h1>I am a React Reusable Component</h1>
    </div>
  );
}

// 4. Export the component to make it available
// for use in other files
export default Component;
```

I am a React Reusable Component

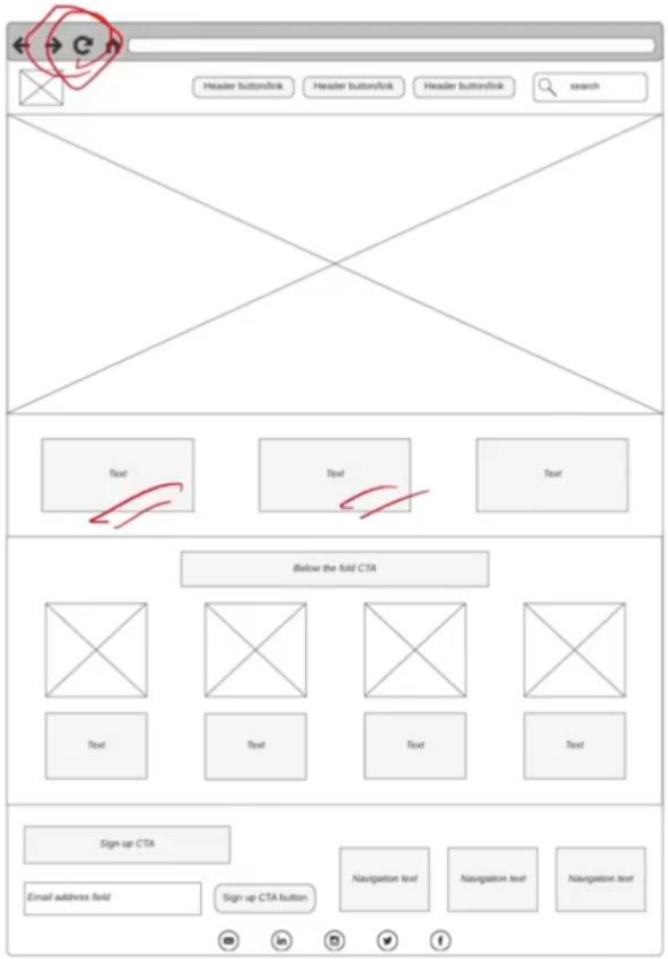
Q. What is SPA(Single Page Application)?

- A Single Page Application (SPA) is a web application that have only one **single web page**.
- Whenever user do some action on the website, then in response content is dynamically updated without refreshing or loading a new page.



Q. What is SPA(Single Page Application)?

- A Single Page Application (SPA) is a web application that have only one **single web page**.
- Whenever user do some action on the website, then in response content is dynamically updated without refreshing or loading a new page.



Q. What are the 5 Advantages of React? V. IMP.

1. Simple to build Single Page Application (by using Components)

2. React is cross platform and open source(Free to use)

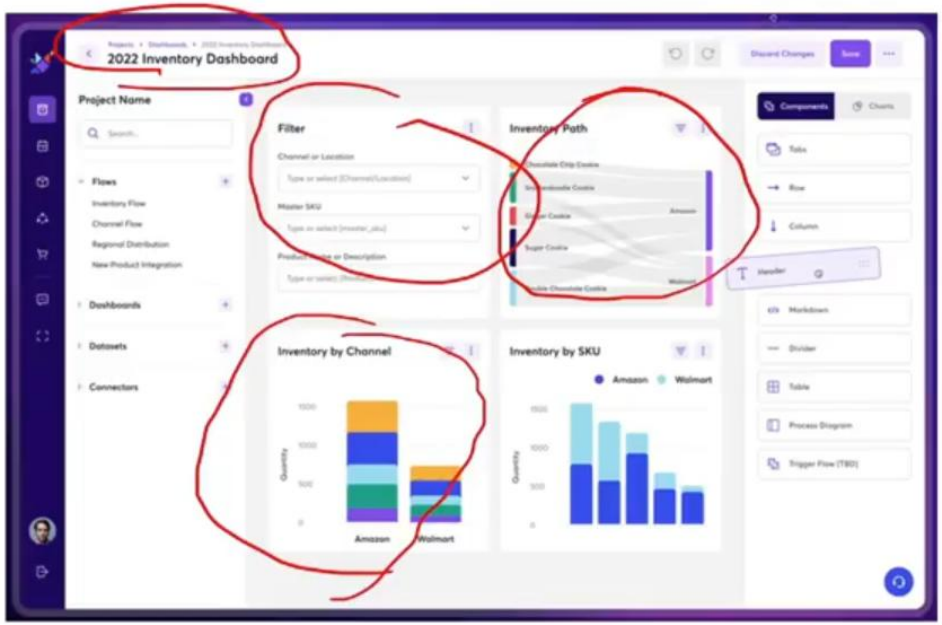
3. Lightweight and very fast (Virtual DOM)

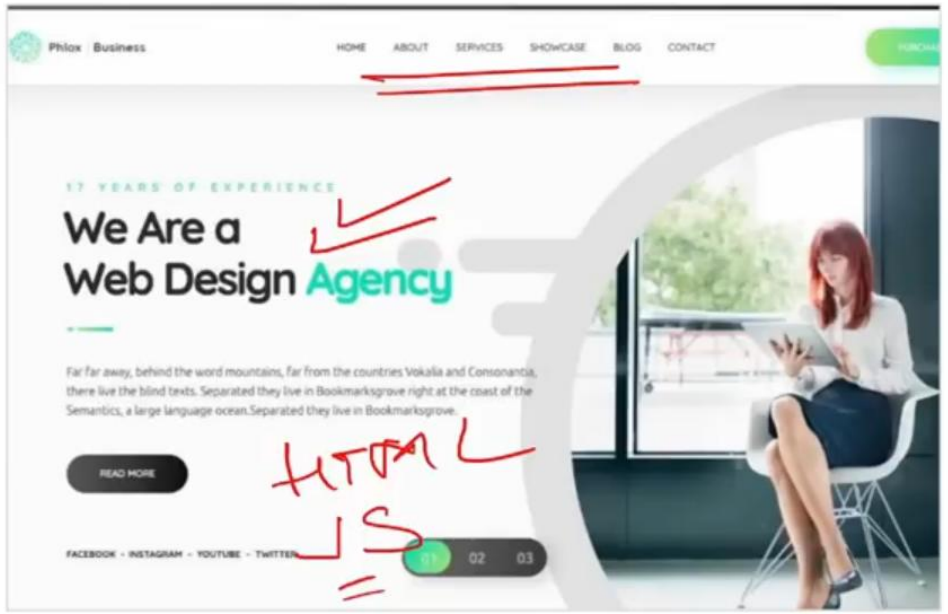
4. Large Community and Ecosystem

5. Testing is easy

Q. What are the Disadvantages of React?

- React is not a good choice for very small applications.





Q. What is the role of JSX in React? (3 points) V. IMP.

1. JSX stands for **JavaScript XML**.
2. JSX is used by React to write **HTML-like code**.
3. JSX is converted to JavaScript via tools like **Babel**.
   Because Browsers understand JavaScript not JSX.

```jsx
function App() {

  return (
    <div className="App">
      <h1>Hello!</h1>
      <p>Happy</p>
    </div>
  );

}
```

Babel

```jsx
function App() {

  return React.createElement(
    'div',
    { className: 'App' },
    React.createElement('h1', null, 'Hello!'),
    React.createElement('p', null, 'Happy')
  );

}
```

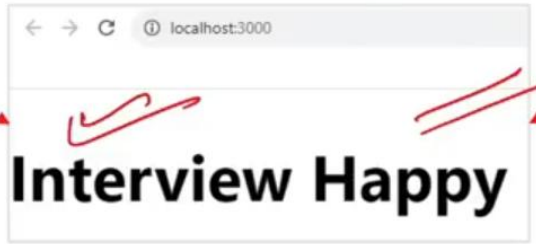
Q. What is the difference between Declarative & Imperative syntax?

1. Declarative syntax focuses on **describing the desired result** without specifying the step-by-step process.
2. **JSX** in React is used to write declarative syntax.

1. Imperative syntax involves **step by step process** to achieve a particular goal.
2. **JavaScript** has an imperative syntax.

```jsx
// Declarative syntax using JSX
function App() {

  return <h1>Interview Happy</h1>;

}
```

```jsx
// Imperative syntax(non-React) using JavaScript
function App() {
  const element = document.createElement("h1");
  element.textContent = "Interview Happy";
  document.body.appendChild(element);
}
```



2: React-Basics - II V. IMP.

Q1. What is Arrow Function Expression in JSX? V. IMP.

Q2. How to Setup React first project?

Q3. What are the Main Files in a React project?

Q4. How React App Load and display the components in browser? V. IMP.

Q5. What is the difference between React and Angular?

Q6. What are other 5 JS frameworks other than React?

Q7. Whether React is a Framework or a Library? What is the difference?

Q8. How React provides Reusability and Composition?

Q9. What are State, Stateless, Stateful and State Management terms?

Q10. What are Props n JSX? V. IMP.

Q. What is Arrow Function Expression in JSX?

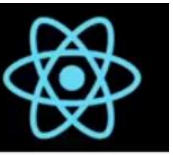
- The arrow function expression syntax is a concise way of defining functions.

```jsx
// Regular Function Declaration
function AppFunc(props) {
  return (
    <div>
      <h1>{props.name}</h1>
    </div>
  );
}
export default AppFunc;
```

=

```jsx
// Arrow Function Expression
const ArrowFunc = (props) => {
  return (
    <div>
      <h1>{props.name}</h1>
    </div>
  );
};
export default ArrowFunc;
```

Q. How to setup react project?



1. Install Node.js from this link:
https://nodejs.org

2. Install code editor for writing the code (VS Code link):
https://code.visualstudio.com/download

3. Open VS Code -> Terminal -> New Terminal and type command:
“npx create-react-app my-app”

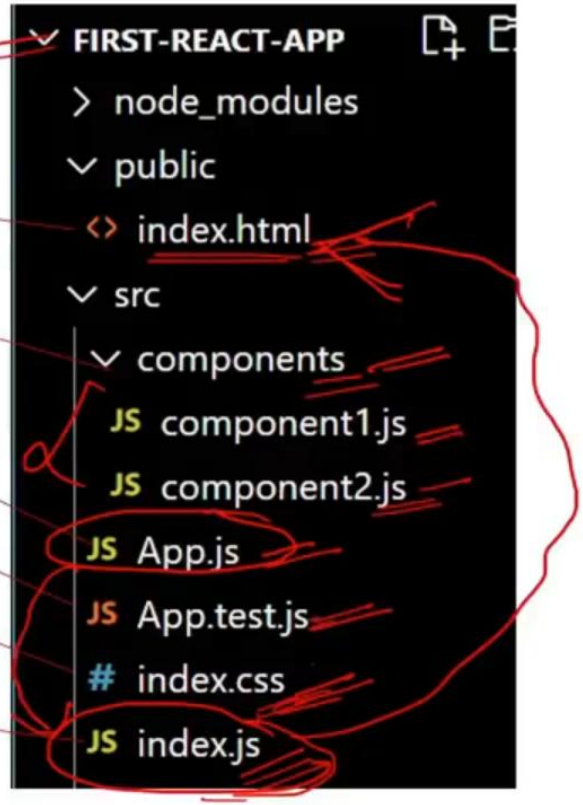
4. Click File -> Open Folder, go to the just created folder and open it.

5. Then open the terminal(ctrl + ‘) key and run this command.
“npm start”

Q. What are the Main Files in a React project?

❖ **Main Files in a React project:**

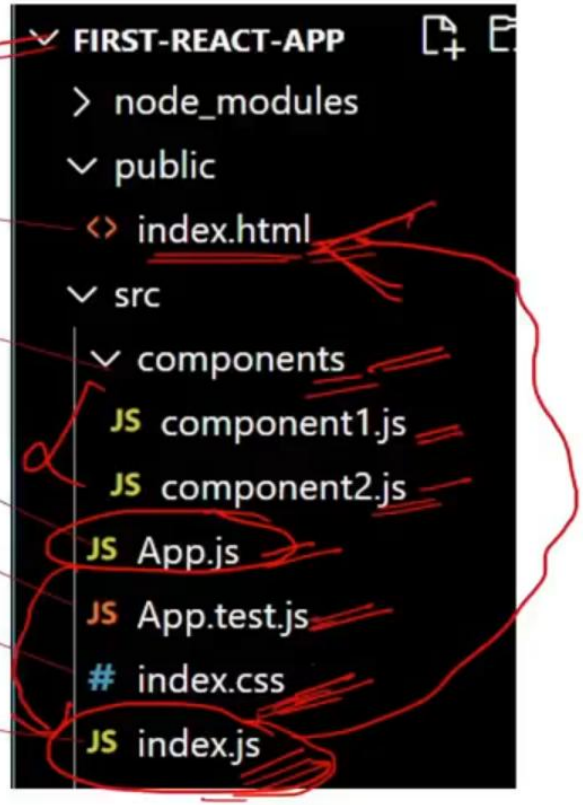
1. **index.html:** Single page for React application.
2. **Components/component1.js:** Your application components.
3. **App.js:** Main component or container or Root component.
4. **App.test.js(Optional):** Used for writing tests for the App.js file.
5. **Index.css(Optional):** This is a global CSS file that serves as the main stylesheet for the entire application.
6. **index.js:** Entry point for JavaScript. Renders the main React component (App) into the root DOM element.



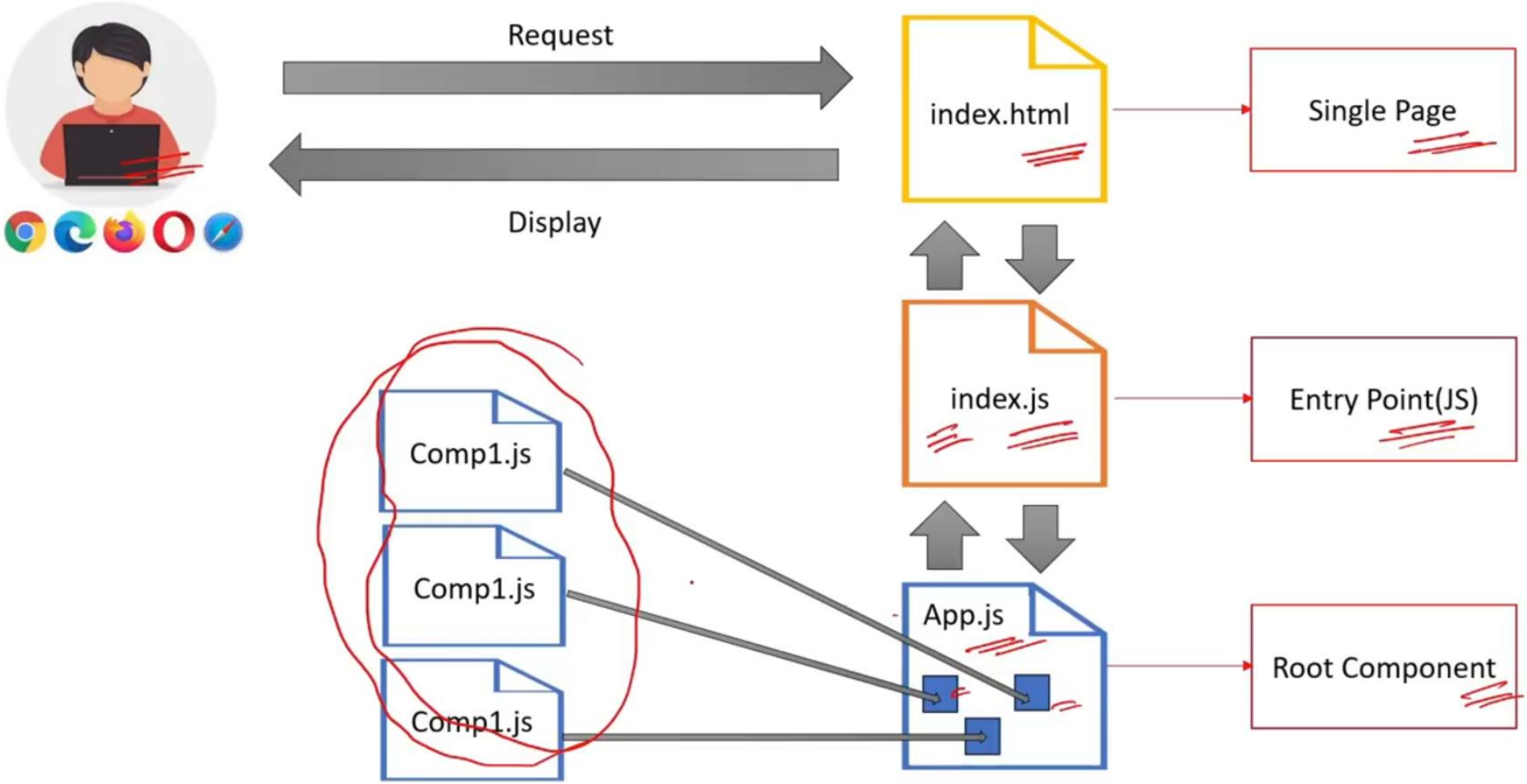
Q. What are the Main Files in a React project?

❖ **Main Files in a React project:**

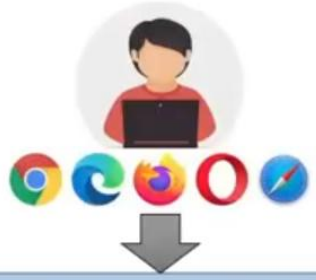
1. **index.html:** Single page for React application.
2. **Components/component1.js:** Your application components.
3. **App.js:** Main component or container or Root component.
4. **App.test.js(Optional):** Used for writing tests for the App.js file.
5. **Index.css(Optional):** This is a global CSS file that serves as the main stylesheet for the entire application.
6. **index.js:** Entry point for JavaScript. Renders the main React component (App) into the root DOM element.



Q. How React App Load and display the components in browser? V. IMP.



Q. How React App load and display the components in browser? V. IMP.



index.html

Single page which loads index.js by React libraries

index.js

Replace root element of index.html file by App component

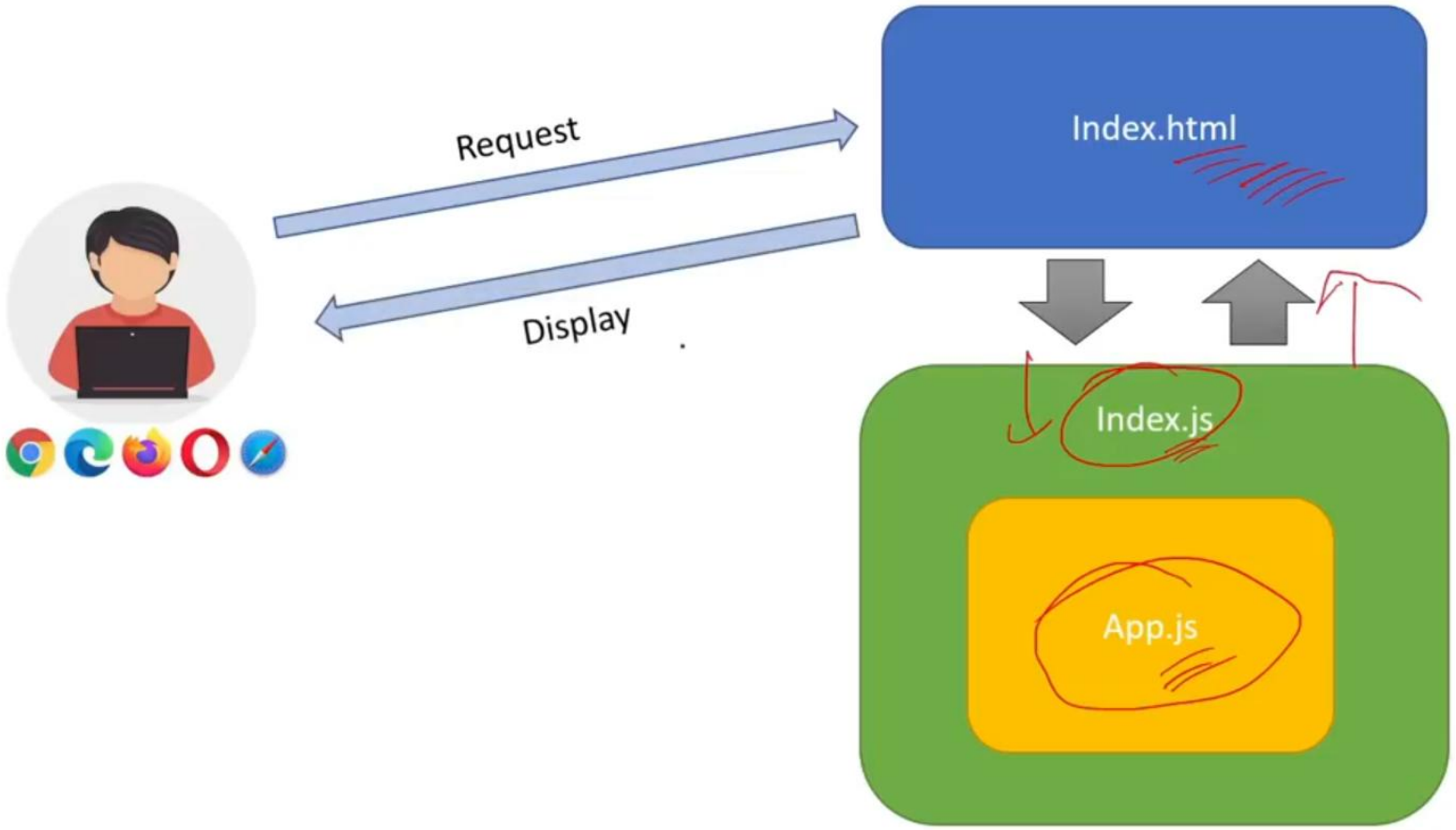
App.js

Root component which is the container of all the child components.

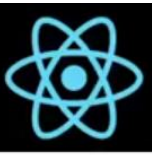
ChildComponent.js

Custom child components placed over app component.

Q. How React App load and display the components in browser?

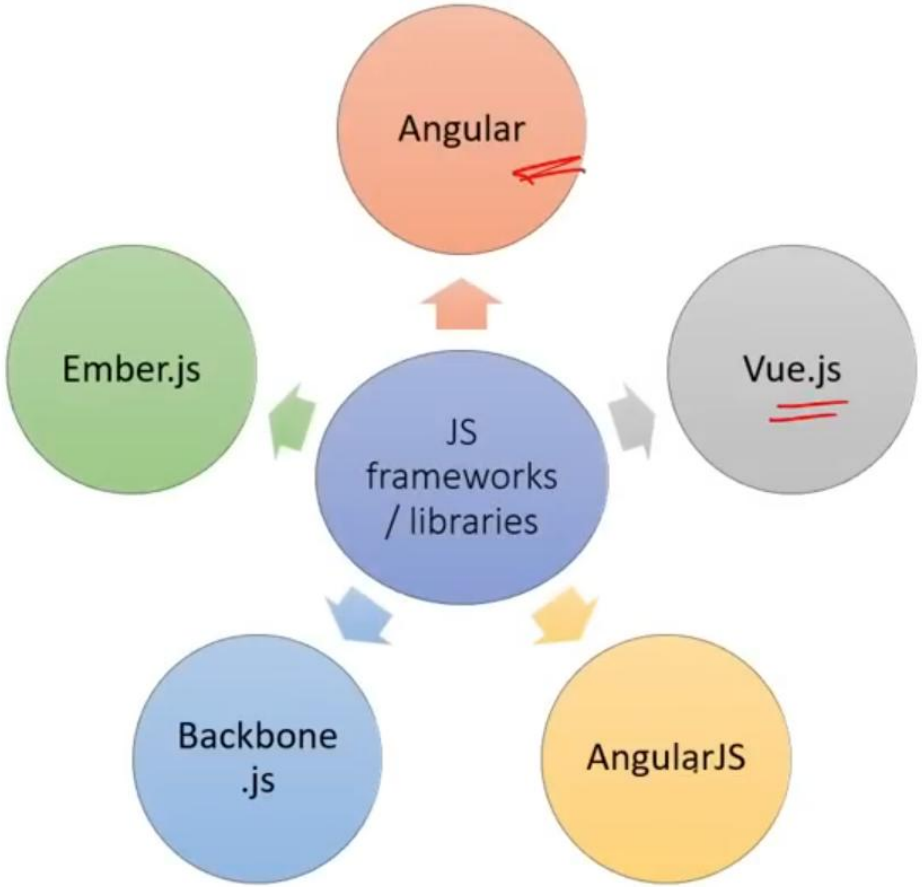


Q. What is the difference between React and Angular?



React	Angular
React and Angular both are used to create single page UI applications using components.	
1. React is a JavaScript library	Angular is a complete framework.
2. React uses a **virtual DOM** which makes it faster.	Angular uses a real DOM
3. React is smaller in size and **lightweight** and therefore faster sometime.	Angular is bigger because it is a complete framework.
4. React depends on external libraries for many complex features, so developer has to write many lines of code for complex functionalities.	Since Angular is a complete framework, therefore it provide **built-in support** for features like routing, forms, validation, and HTTP requests.
5. React is **simple to learn** and more popular than Angular.	5. Angular is slightly difficult to learn as it has Typescript, OOPS concept and many more thing.

Q. What are other 5 JS frameworks other than React?



Q. Whether React is a Framework or a Library? What is the difference?

- **Library:** Developers import the libraries at the top and then used its functions in components.
- React is commonly referred to as a JavaScript library.

```jsx
// 1. Import the React library
import React from "react";

function Component() {
  return (
    <div>
      <h1>Hello World</h1>
    </div>
  );
}

export default Component;
```

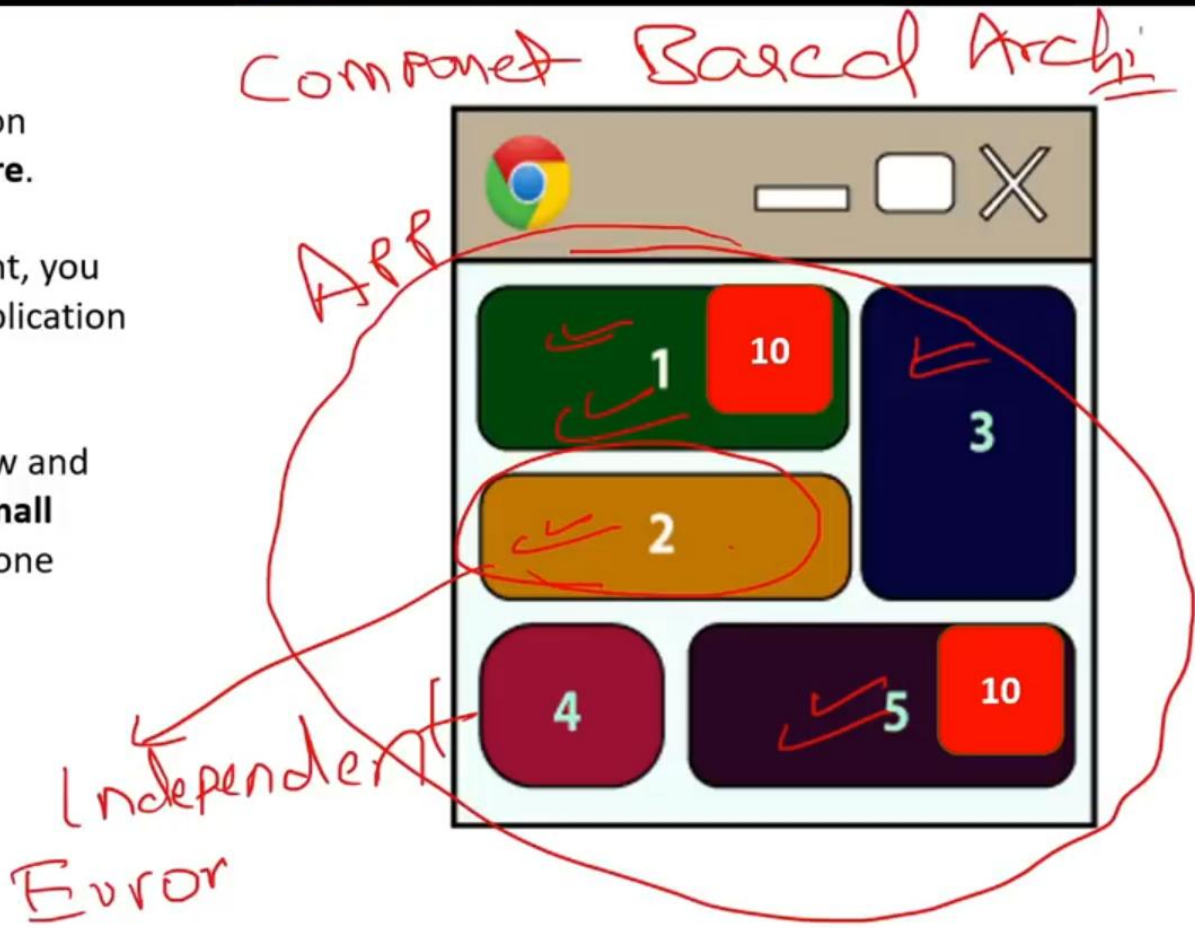
- **Framework:** Developers need to follow a specific structure or pattern defined by the framework.
- Angular is a framework.

app.component.ts | app.module.ts

```typescript
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Hello World';
}
```

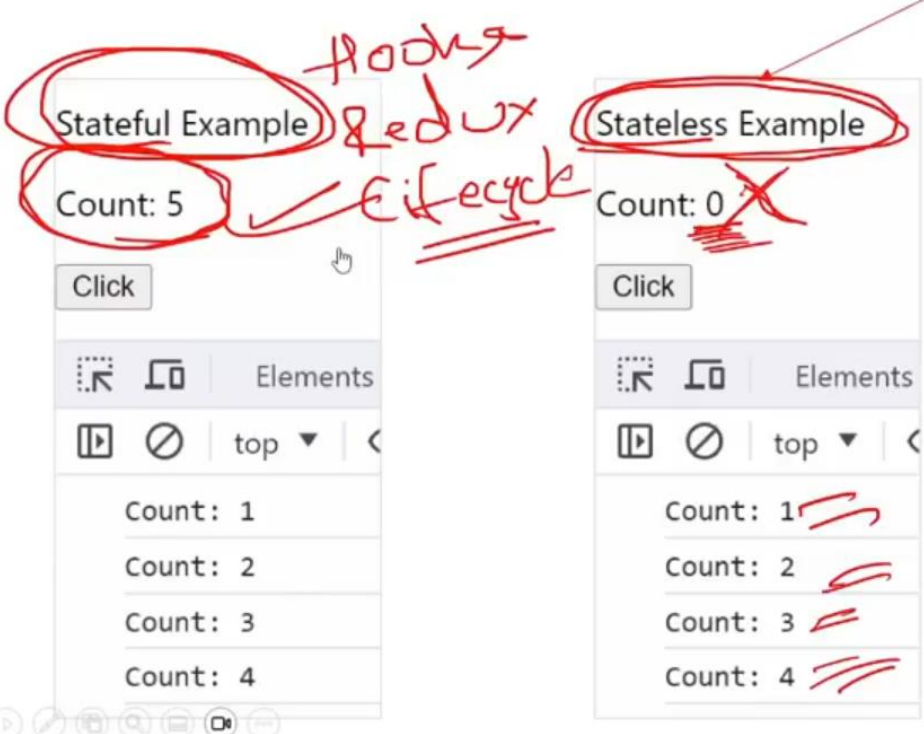
Q. How React provides Reusability and Composition?

- React provides reusability and composition through its **component-based architecture**.
- **Reusability:** Once you create a component, you can **re-use** it in different parts of your application or even in multiple projects.
- **Composition:** Composition is creating new and big components by **combining existing small components**. Its advantage is, change to one small component will not impact other components.



Q. What are state, stateless, stateful and state management terms?

- "state" refers to the current data of the component.
- Stateful or state management means, when a user performs some actions on the UI, then the React application should be able to **update and re-render that data or state** on the UI.




```
ooks > JS ComponentState.js > ...
import React from "react";

// Stateless Example
function ComponentState() {
  let count = 0; // Initial state

  const increment = () => {
    count += 1; // State updated
    console.log(`Count: ${count}`);
  };

  return (
    <div>
      <p>Stateless Example</p>
      <p>Count: {count}</p> {/* Not updating */}
      <button onClick={increment}>Click</button>
    </div>
  );
}
export default ComponentState;
```

Q. What are Props n JSX? V. IMP.

- props (properties) are a way to **pass data** from a parent component to a child component.

```jsx
function App() {
  return (
    <>
      <ChildComponent name="Happy" purpose="Interview" />
    </>
  );
}
```

```jsx
function ChildComponent(props) {

  return <div>{props.name}, {props.purpose}!</div>;

}
//Output: Happy, Interview!
```

3: React Project - Files & Folders

Q1. What is NPM? What is the role of node_modules folder? V. IMP.

Q2. What is the role of public folder in React?

Q3. What is the role of src folder in React?

Q4. What is the role of index.html page in React? V. IMP.

Q5. What is the role of index.js file and ReactDOM in React? V. IMP.

Q6. What is the role of App.js file in React? V. IMP.

Q7. What is the role of function and return inside App.js?

Q8. Can we have a function without a return inside App.js?

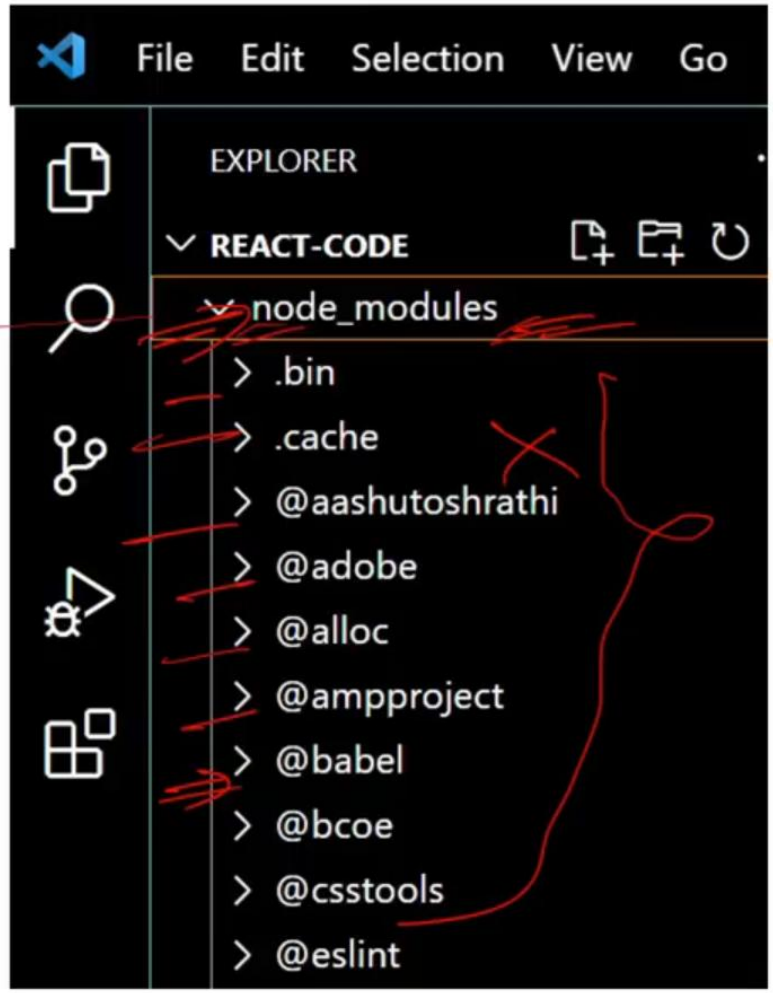
Q9. What is the role of export default inside App.js?

Q10. Does the file name and the component name must be same in React?

Q. What is NPM? What is the role of node_modules folder? V. IMP.

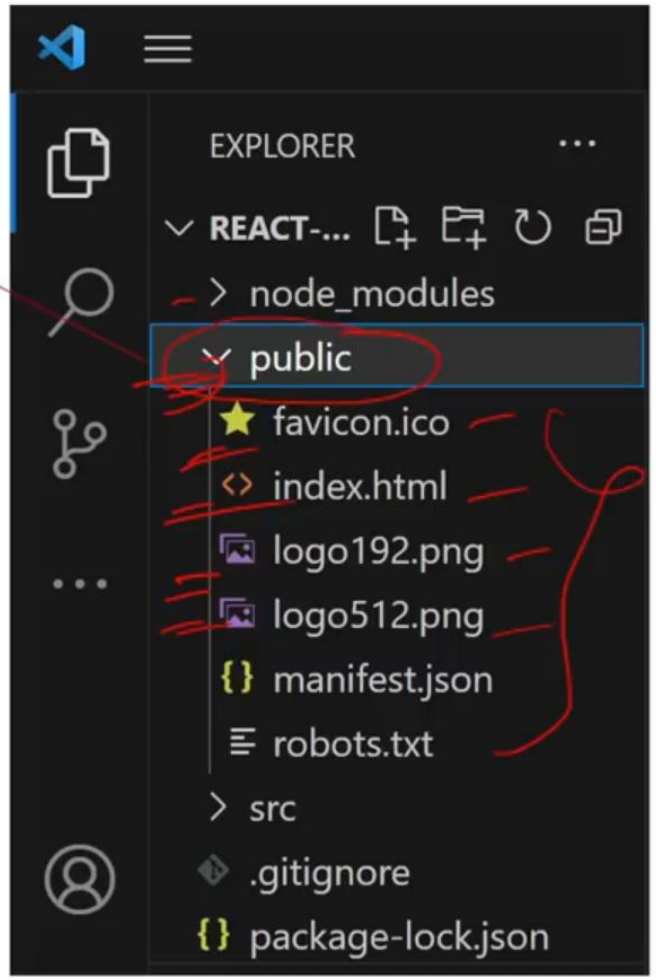
- NPM(Node Package Manager) is used to manage the **dependencies** for your React project, including the **React library** itself.

- node_modules folder contains all the dependencies of the project, including the React libraries.



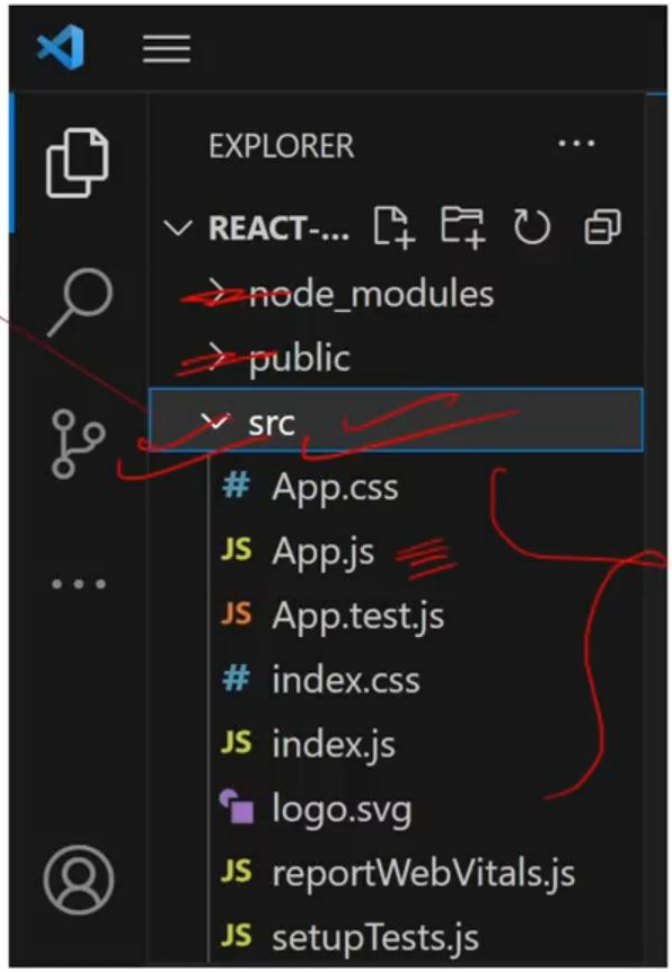
Q. What is the role of public folder in React?

- Public folder contains **static assets** that are served directly to the user's browser, such as images, fonts, and the index.html file.



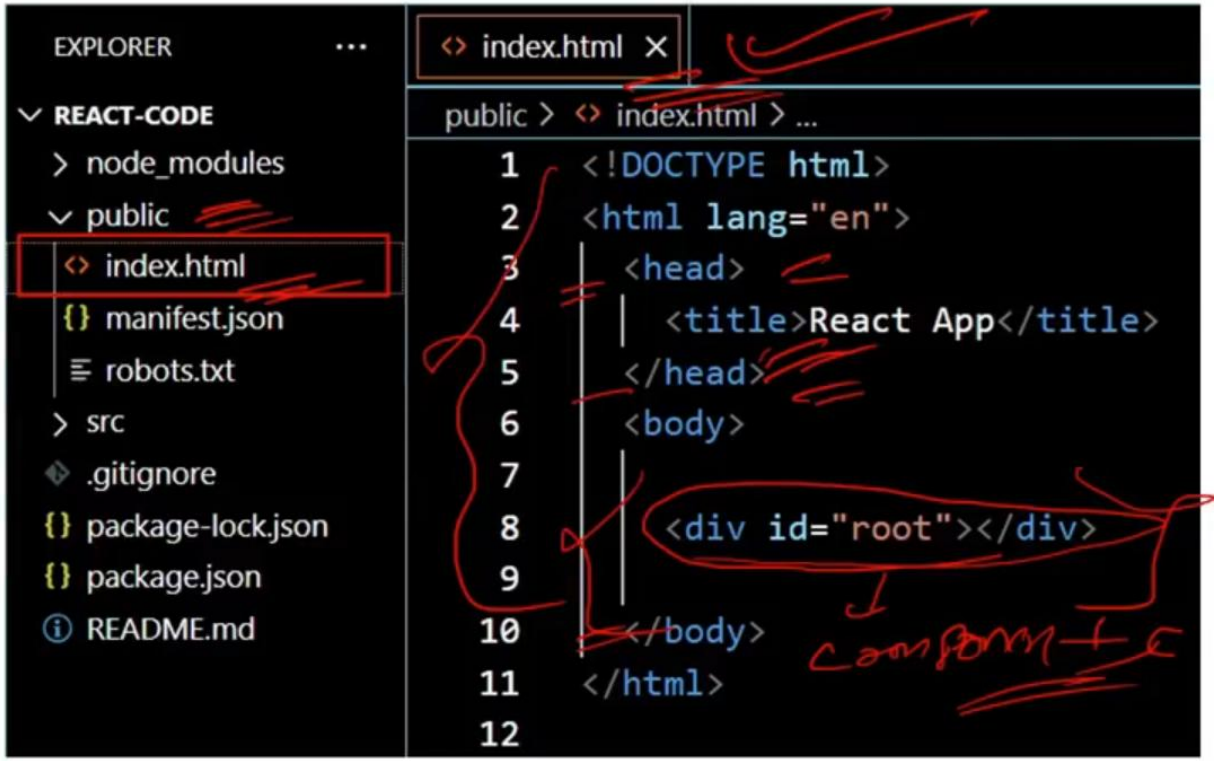
Q. What is the role of src folder in React?

- src folder is used to **store all the source code** of the application which is then responsible for the dynamic changes in your web application.



Q. What is the role of index.html page in React?

- index.html file is the main HTML file(SPA) in React applcication.
- Here the div with “id=root” will be replaced by the component inside index.js file.



Q. What is the role of index.js file and ReactDOM in React? V. IMP.

- ReactDOM is a JavaScript library that renders components to the DOM or browser.
- The index.js file is the JavaScript file that replaces the root element of the index.html file with the newly rendered components.

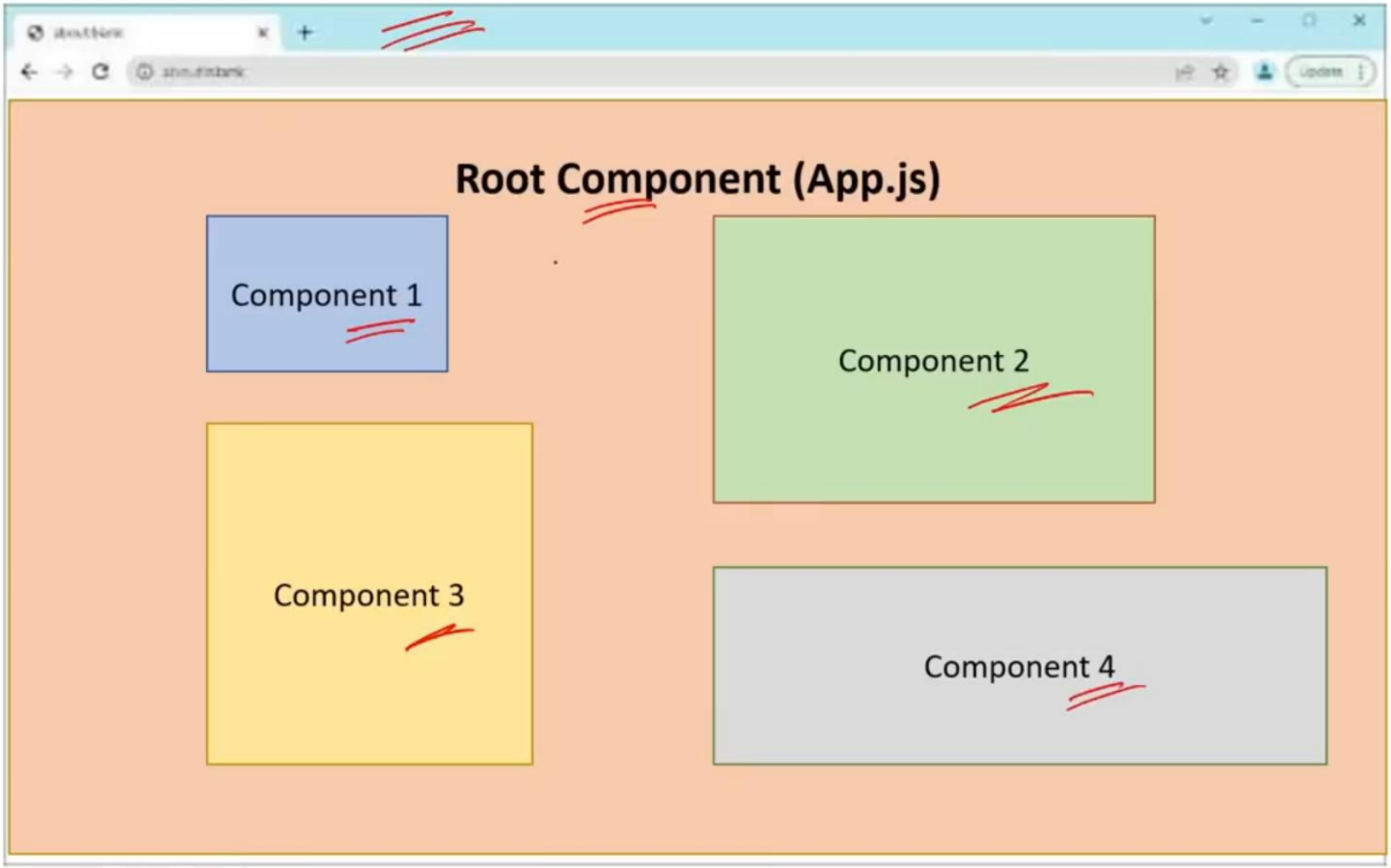
index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>React App</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

JS index.js > ...

```jsx
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";

const root = ReactDOM.createRoot(
    document.getElementById("root")
    );

root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Q. What is the role of App.js file in React? V. IMP.



Q. What is the role of App.js file in React? V. IMP.

- App.js file contain the **root component(App)** of React application.
- App component is like a **container** for other components.
- App.js defines the structure, layout, and routing in the application.

JS App.js > ...

```
import AppChild from "./Others/AppChild";

function App() {
  return (
    <div>
      <AppChild></AppChild>
    </div>
  );
}

export default App;
```

Q. What is the role of function and return inside App.js?

- The function keyword is used to **define a JavaScript function** that represents your React component.
- function is like a placeholder which contains all the code or logic of component.
- The function takes in props as its argument (if needed) and returns JSX

- **return** is used to return the element from the function.

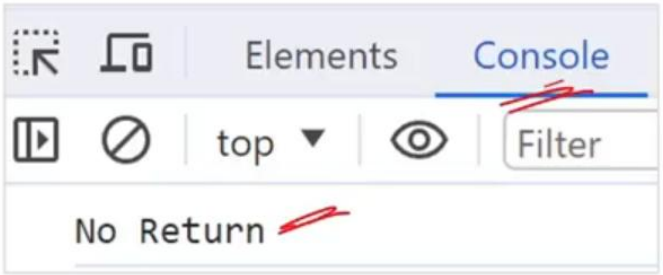
App.js > ...

```
import AppChild from "./Others/AppChild";

function App() {
  return (
    <div>
      <h1>Interview Happy</h1>
      <AppChild></AppChild>
    </div>
  );
}

export default App;
```

Q. Can we have a function without a return inside App.js?

- **Yes**, a function without a return statement is possible.
- In that case, your component will not render anything in UI.
- The common use case is for **logging** purpose.

```
const FuncWithoutReturn = () => {
  //   return (
  //     <div>
  //       <h1>Interview Happy</h1>
  //     </div>
  //   );

  console.log("No Return");
};

export default FuncWithoutReturn;
```



Q. What is the role of export default inside App.js?

- Export statement is used to make a component available for import using “import” statement in other files.

```jsx
import React from "react";

const AppChild = (props) => {
  return <h1>Hello, {props.name}!</h1>;
};

export default AppChild;
```

JS App.js > ...

```jsx
import AppChild from "./Others/AppChild";

function App() {
  return (
    <div>
      <AppChild></AppChild>
    </div>
  );
}

export default App;
```

Q. What is the role of export default inside App.js?

- Export statement is used to make a component available for import using “import” statement in other files.

```jsx
import React from "react";

const AppChild = (props) => {
  return <h1>Hello, {props.name}!</h1>;
};

export default AppChild;
```

JS App.js > ...

```jsx
import AppChild from "./Others/AppChild";

function App() {
  return (
    <div>
      <AppChild></AppChild>
    </div>
  );
}

export default App;
```

Q. Does the file name and the component name must be same in React?

- **No,** the file name and the component name don't have to be the same.
- However, it is recommended to keep them same for easier to organize and understand your code.

ComponentName.js

src > 1-Basics > ComponentName.js > ...

```
1   import React from "react";
2
3   const DifferentComponentName = () => {
4     return (
5       <div>
6         <h1>Interview Happy</h1>
7       </div>
8     );
9   };
10
11  export default DifferentComponentName;
12
```

4: JSX

Q1. What is the role of JSX in React? (3 points) V. IMP.

Q2. What are the 5 Advantages of JSX? V. IMP.

Q3. What is Babel?

Q4. What is the role of Fragment in JSX? V. IMP.

Q5. What is Spread Operator in JSX?

Q6. What are the types of Conditional Rendering in JSX? V. IMP.

Q7. How do you iterate over a list in JSX? What is map() method?

Q8. Can a browser read a JSX File?

Q9. What is Transpiler? What is the difference between Compiler & Transpiler?

Q10. Is it possible to use JSX without React?

Q. What is the role of JSX in React? (3 points) V. IMP.

1. JSX stands for **JavaScript XML**.
2. JSX is used by React to write **HTML-like code**.
3. JSX is converted to JavaScript via tools like **Babel**.
   Because Browsers understand JavaScript not JSX.

```jsx
function App() {

  return (
    <div className="App">
      <h1>Hello!</h1>
      <p>Happy</p>
    </div>
  );

}
```

Babel

```jsx
function App() {

  return React.createElement(
    'div',
    { className: 'App' },
    React.createElement('h1', null, 'Hello!'),
    React.createElement('p', null, 'Happy')
  );

}
```

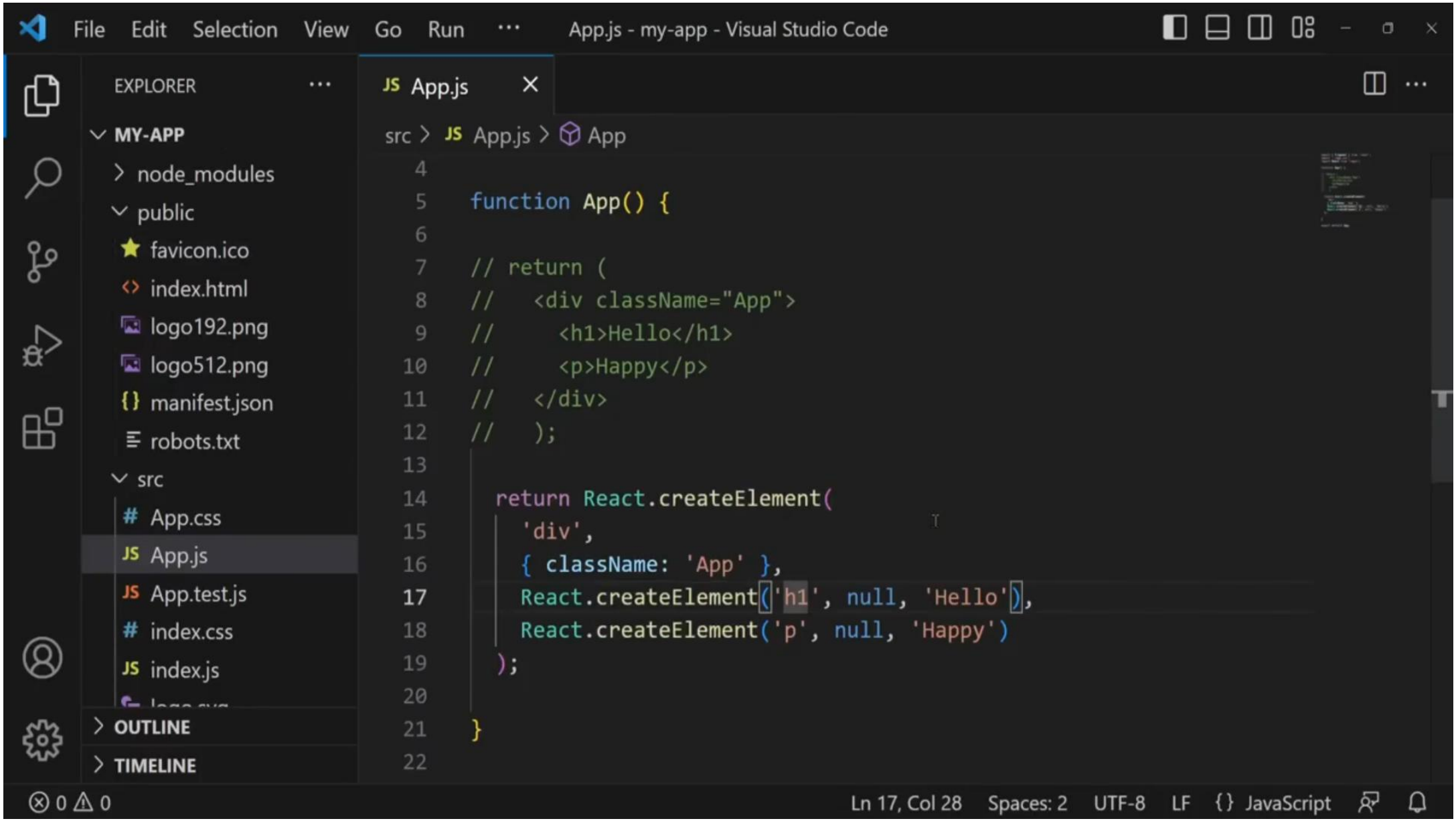
Q. What is the role of JSX in React? (3 points) V. IMP.

1. JSX stands for **JavaScript XML**.
2. JSX is used by React to write **HTML-like code**.
3. JSX is converted to JavaScript via tools like **Babel**.
   Because Browsers understand JavaScript not JSX.

```jsx
function App() {

  return (
    <div className="App">
      <h1>Hello!</h1>
      <p>Happy</p>
    </div>
  );

}
```

Babel

```jsx
function App() {

  return React.createElement(
    'div',
    { className: 'App' },
    React.createElement('h1', null, 'Hello!'),
    React.createElement('p', null, 'Happy')
  );

}
```


File Edit Selection View Go Run ···
App.js - my-app - Visual Studio Code
EXPLORER
MY-APP
node_modules
public
favicon.ico
index.html
logo192.png
logo512.png
manifest.json
robots.txt
src
App.css
App.js
App.test.js
index.css
index.js
OUTLINE
TIMELINE
App.js
src > App.js > App
function App() {
// return (
//
// <h1>Hello</h1>
// Happy
//
//);
return React.createElement(
'div',
{ className: 'App' },
React.createElement('h1', null, 'Hello'),
React.createElement('p', null, 'Happy')
);
}
0 0
Ln 17, Col 28
Spaces: 2
UTF-8
LF
JavaScript

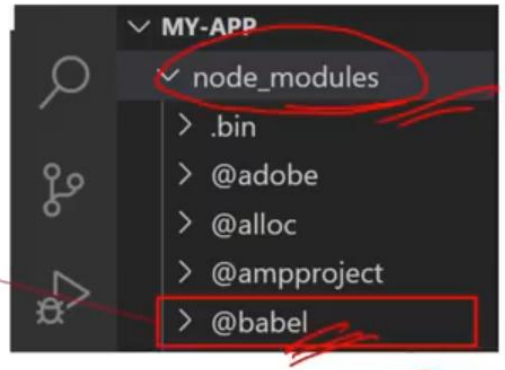
Q. What are the advantages of JSX? V. IMP.

- Advantage of JSX
  - 1. Improve code readability and writability
  - 2. Error checking in advance(Type safety)
  - 3. Support JavaScript expressions
  - 4. improved performance
  - 5. Code resusability

```jsx
function App() {

  const name = 'John';

  return (
    <div>
      {/* Javascript expressions*/}
      <h1>Hello, {name}!</h1>
      <p>{2 + 2} sum</p>
    </div>
  );
  //output: Hello John 4
}
```

Q. What is Babel?

- Babel in React is used to transpile JSX syntax into regular JavaScript which browser can understand.




```jsx
return (
  <div className="App">
    <h1>Hello!</h1>
    <p>Happy</p>
  </div>
);
```

Babel

```jsx
return React.createElement(
  'div',
  { className: 'App' },
  React.createElement('h1', null, 'Hello!'),
  React.createElement('p', null, 'Happy')
);
```

Q. What is the role of Fragment in JSX? V. IMP.

- In React, a fragment is a way to **group multiple children's** elements.
- Using a Fragment prevents the addition of unnecessary nodes to the DOM.

Separate elements(Error)

```
function App() {
  return (
    <div>Interview</div>
    <div>Happy</div>
  );
}
```

```
function App() {
  return (
    <div>
      <div>Interview</div>
      <div>Happy</div>
    </div>
  );
}
```

Fragment (<>, <Fragment>)

```
function App() {
  return (
    <>
      <div>Interview</div>
      <div>Happy</div>
    </>
  );
}
```

```
function App() {
  return (
    <Fragment>
      <div>Interview</div>
      <div>Happy</div>
    </Fragment>
  );
}
```

Q. What is Spread Operator in JSX?

- The spread operator (...) is used to expand or spread an array or object.

```
function App() {

  const props =  {name: 'Happy', purpose: 'Interview'};

  return (
    <>
      <ChildComponent {...props}/>
    </>
  );
}
```

```
function ChildComponent(props) {

  return <div>{props.name},
    {props.purpose}
  </div>;

}
//Output: Happy, Interview!
```

Q. What are the types of Conditional Rendering in JSX? V. IMP.

Conditional Rendering

1. If/else statements
2. Ternary operator
3. && operator
4. Switch statement

```jsx
function MyComponent() {
  if (2 > 1 )
  {
    return "abc";
  }
  else
  {
    return "xyz";
  }
}
```

```jsx
function MyComponent() {

return 2>1 ? "abc" : "xyz";

}
```

```jsx
function MyComponent() {

  {return 2 > 1 && "abc"}

}
```

```jsx
function MyComponent() {

  const value = 2;
  switch (value) {
    case 2:
      return 'abc';
    case 1:
      return 'xyz';
    default:
      return null;
  }
}
```

Q. How do you iterate over a list in JSX? What is map() method?

- map() method allows you to iterate over an array and modify its elements using a callback function.

```jsx
function App() {

  // Define an array of numbers
  const numbers = [1, 2, 3, 4, 5];

  return (
    <>
      {
        numbers.map((number) => (number * 2))
      }
    </>
  );
  //output: 2 4 6 8 10
}
```

x2

callback
Anonymous
Arrow

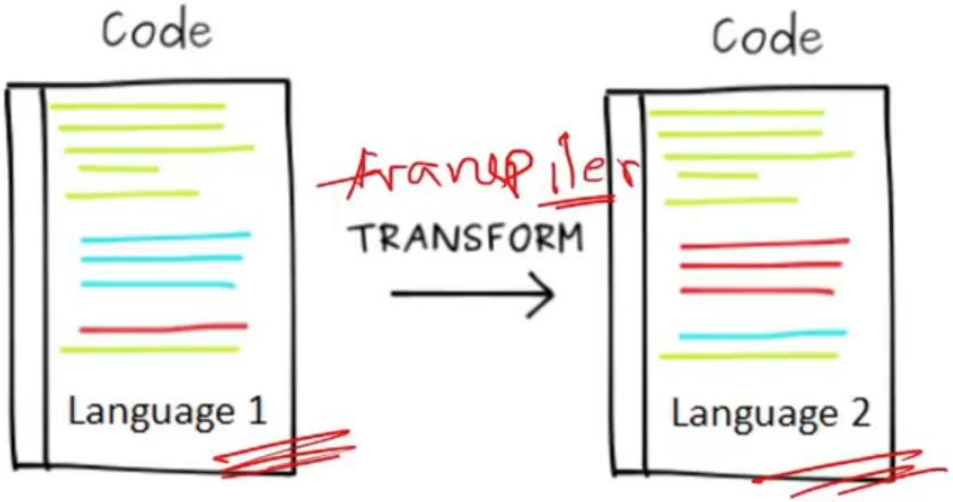
Q. Can a browser read a JSX File?

- **No**, browsers cannot directly interpret or understand JSX files.
- Babel takes JSX and converts it into equivalent JavaScript code that browsers can understand.

Q. What is Transpiler? What is the difference between Compiler & Transpiler?

- A Transpiler is a tool that converts source code from one high-level programming language(JSX) to another high-level programming language(JavaScript). Example: Babel

- A compiler is a tool that converts high-level programming language(Java) into a lower-level language(machine code or bytecode).



Q. Is it possible to use JSX without React?

- **Yes**, it's possible to use JSX without React by creating your own transpiler like Babel.
- However, this is not recommended since JSX is tightly integrated with React and relies on many React-specific features.

5: Components - Functional/ Class

Q1. What are React Components? What are the main elements of it? V. IMP.

Q2. What are the Types of React components? What are Functional Components? V. IMP.

Q3. How do you pass data between functional components in React?

Q4. What is Prop Drilling in React? V. IMP.

Q5. Why to Avoid Prop Drilling? In how many ways can avoid Prop Drilling? V. IMP.

Q6. What are Class Components In React? V. IMP.

Q7. How to pass data between class components in React?

Q8. What is the role of this keyword in class components?

Q9. What are the 5 differences btw Functional components & Class components? V. IMP.

Q. What are React Components? What are the main elements of it? V. IMP.

- In React, a component is a **reusable building block** for creating user interfaces.


```jsx
// 1. Import the React library
import React from "react";

// 2. Define a functional component
function Component() {
  // 3. Return JSX to describe the component's UI
  return (
    <div>
      <h1>I am a React Reusable Component</h1>
    </div>
  );
}

// 4. Export the component to make it available
// for use in other files
export default Component;
```

I am a React Reusable Component

Q. What are React Components? What are the main elements of it? V. IMP.

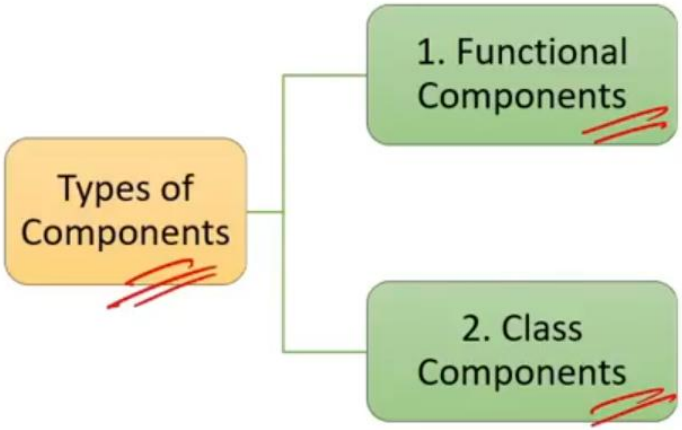
- In React, a component is a **reusable building block** for creating user interfaces.


```jsx
// 1. Import the React library
import React from "react";

// 2. Define a functional component
function Component() {
  // 3. Return JSX to describe the component's UI
  return (
    <div>
      <h1>I am a React Reusable Component</h1>
    </div>
  );
}

// 4. Export the component to make it available
// for use in other files
export default Component;
```

I am a React Reusable Component

Q. What are the Types of React components? What are Functional Components? V. IMP.

- Functional components are declared as a **JavaScript function**.
- They are **stateless component**, but with the help of hooks, they can now manage state also.




```jsx
// 1. Import the React library
import React from "react";

// 2. Define a functional component
function Component() {
  // 3. Return JSX to describe the component's UI
  return (
    <div>
      <h1>I am a React Reusable Component</h1>
    </div>
  );
}

// 4. Export the component to make it available
// for use in other files
export default Component;
```

I am a React Reusable Component

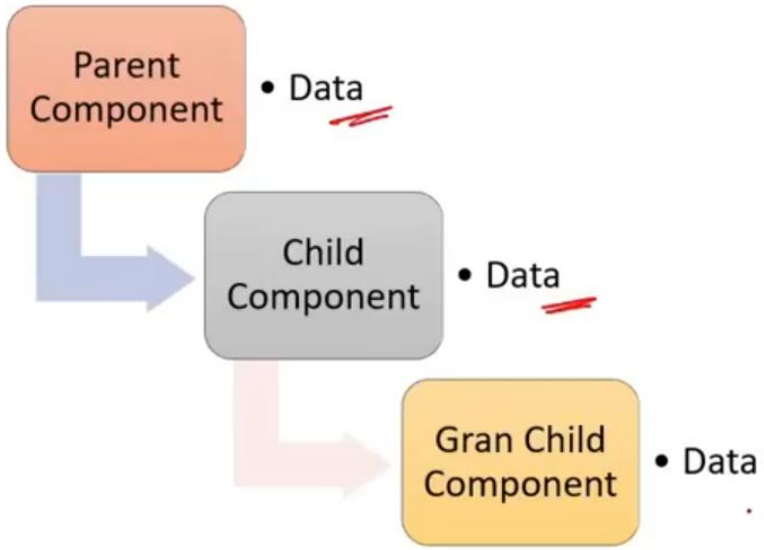
Q. How do you pass data between functional components in React?

- **props (properties)** are a way to pass data from a parent component to a child component.

```jsx
function App() {
  return (
    <>
      <ChildComponent name="Happy" purpose="Interview" />
    </>
  );
}
```

```jsx
function ChildComponent(props) {

  return <div>{props.name}, {props.purpose}!</div>;

}
//Output: Happy, Interview!
```

Q. What is Prop Drilling in React? V. IMP.



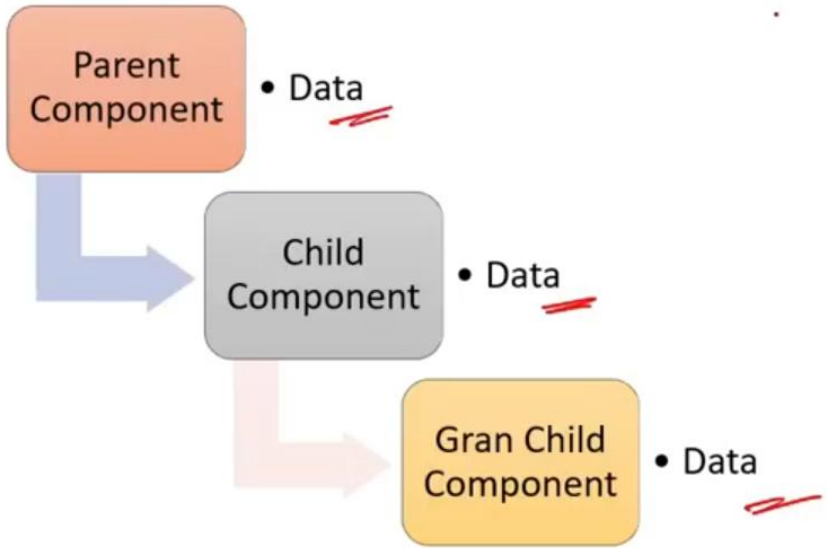

```jsx
function PropParent() {
  return (
    <div>
      <PropChild message={'data'} />
    </div>
  );
}
```

```jsx
function PropChild({ message }) {
  return (
    <div>
      <PropGrandChild message={message} />
    </div>
  );
}
```

```jsx
function PropGrandChild({ message }) {
  return (
    <div>
      <h3>{message}</h3>
    </div>
  );
}
```

Q. What is Prop Drilling in React? **V. IMP.**

- Prop drilling is the process of **passing down props** through multiple layers of components.

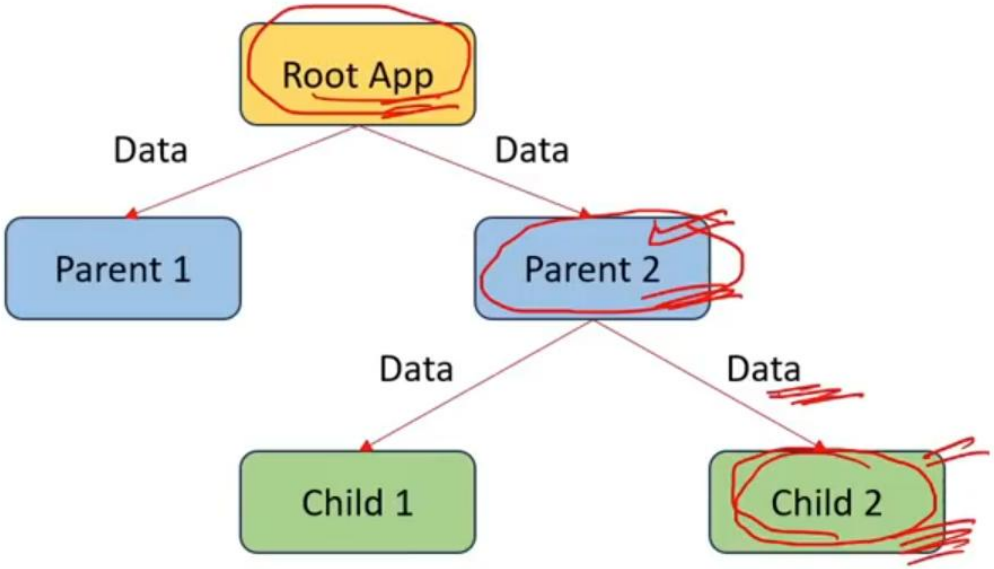



```jsx
function PropParent() {
  return (
    <div>
      <PropChild message={'data'} />
    </div>
  );
}
```

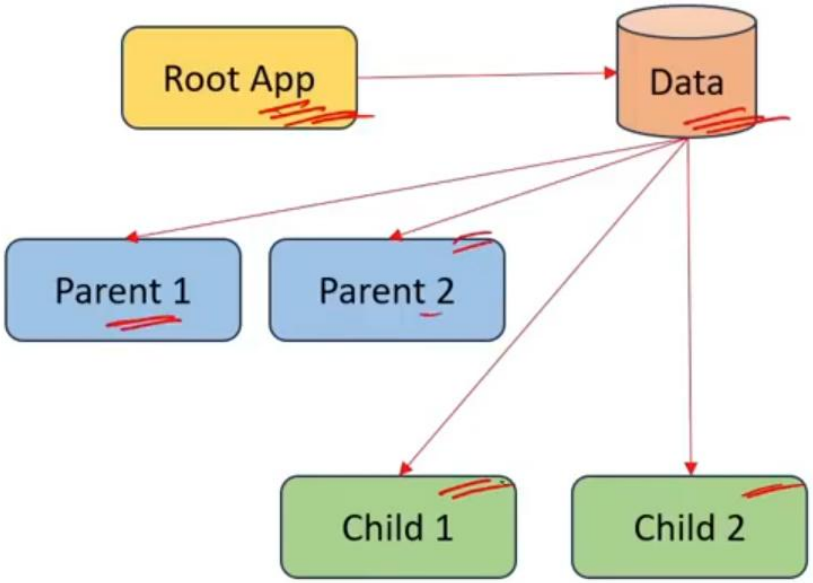
```jsx
function PropChild({ message }) {
  return (
    <div>
      <PropGrandChild message={message} />
    </div>
  );
}
```

```jsx
function PropGrandChild({ message }) {
  return (
    <div>
      <h3>{message}</h3>
    </div>
  );
}
```

Q. Why to Avoid Prop Drilling? In how many ways can avoid Prop Drilling? V. IMP.



Using Prop Drilling



Avoiding Prop Drilling

Q. Why to Avoid Prop Drilling? In how many ways can avoid Prop Drilling? V. IMP.

❖ **Why to avoid Prop Drilling:**

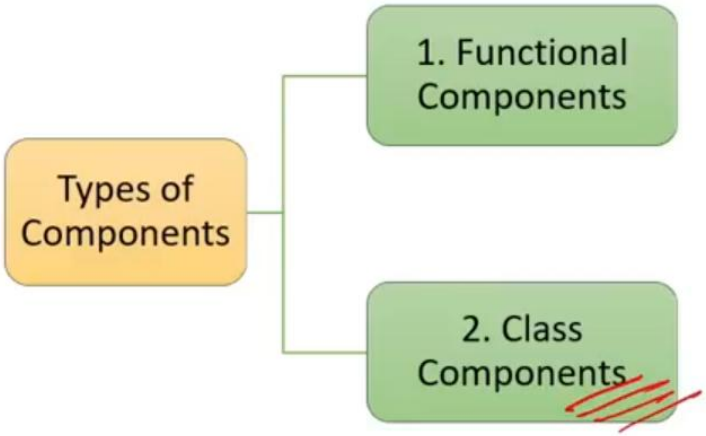
1. **Maintenance:** Prop drilling can make code harder to maintain as changes in data flow require updates across multiple components.
2. **Complexity:** It increases code complexity and reduces code readability.
3. **Debugging:** Debugging becomes challenging when props need to be traced through numerous components.

5 Ways to avoid Prop Drilling

1. Using Context API
2. Using Redux
3. Using Component Composition
4. Using Callback Functions
5. Using Custom Hooks

Q. What are Class Components In React? V. IMP.

- Class components are defined using **JavaScript classes**.
- They are stateful components by using the lifecycle methods.
- The render method in a class component is responsible for returning JSX.




```
mponents > JS AppClass.js > ...
import React, { Component } from 'react';

class AppClass extends Component {
  render() {
    return <h1>Interview Happy</h1>;
  }
}

export default AppClass;
```

Q. How to pass data between class components in React?

- **this.props** can be used in child component to access properties/ data passed from parent component.

Props

```
class ParentComponent extends Component {
  render() {
    const dataToSend = "Hello from Parent!";

    return (
      <div>
        <ChildComponent message={dataToSend} />
      </div>
    );
  }
}
export default ParentComponent;
```

```
class ChildComponent extends Component {
  render() {
    return (
      <div>
        <p>Message: {this.props.message}</p>
      </div>
    );
  }
}
export default ChildComponent;
```

Message: Hello from Parent!

Q. What is the role of this keyword in class components?

- this keyword is used to refer to the **instance of the class**.

```
class ParentComponent extends Component {
  render() {
    const dataToSend = "Hello from Parent!";

    return (
      <div>
        <ChildComponent message={dataToSend} />
      </div>
    );
  }
}
export default ParentComponent;
```

```
class ChildComponent extends Component {
  render() {
    return (
      <div>
        <p>Message: {this.props.message}</p>
      </div>
    );
  }
}
export default ChildComponent;
```

Message: Hello from Parent!

Q. What are the 5 differences btw Functional components & Class components? V. IMP.

Functional Component	Class Component
1. Syntax: Defined as a JS function.	Defined as a JS(ES6) class.
2. State: Originally stateless but can now maintain state using hooks.	Can manage local state with this.state.
3. Lifecycle methods: No	Yes
4. Readability: more readable & concise.	Verbose(complex).
5. this keyword: No	Yes (Access props using this.props)
6. Do not have render method.	Have **render** method.

6: Routing

Q1. What is Routing and Router in React? V. IMP.

Q2. How to Implement Routing in React? V. IMP.

Q3. What are the roles of <Routes> & <Route> component in React Routing? V. IMP.

Q4. What are Route Parameters in React Routing?

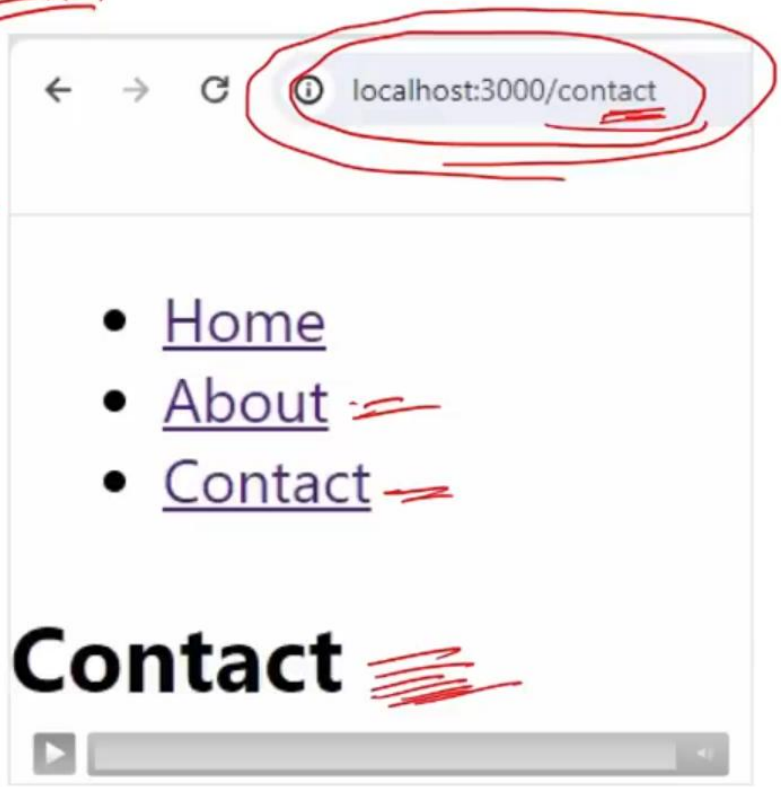
Q5. What is the role of Switch Component in React Routing?

Q6. What is the role of exact prop in React Routing?

Q. What is Routing and Router in React? V. IMP.

Navigation

Routing	Routing allows you to create a single-page web application with **navigation**, without the need for a full-page refresh.
React Router	React Router is a library for handling routing and enables navigation and rendering of different components **based on the URL**.



Q. How to Implement Routing in React?

- Command to install router:

```
npm install react-router-dom
```

```jsx
import React from "react";
import { Routes, Route, Link } from "react-router-dom";

// Elements or imported components
const Home = () => <h2>Home</h2>;
const About = () => <h2>About</h2>;
const Contact = () => <h2>Contact</h2>;

const AppRoute = () => (
  <div>
    {/* Navigation links */}
    <nav>
      <ul>
        <li><Link to="/">Home</Link></li>
        <li><Link to="/about">About</Link></li>
        <li><Link to="/contact">Contact</Link></li>
      </ul>
    </nav>
```

```jsx
// index.js
import AppRoute from "./Others/AppRoute";
import { BrowserRouter as Router }
        from "react-router-dom";

const root = ReactDOM.createRoot(
  document.getElementById("root"));
root.render(
  <Router>
    <AppRoute />
  </Router>
);
```

```jsx
    {/* Routes */}
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/about" element={<About />} />
      <Route path="/contact" element={<Contact />} />
    </Routes>
  </div>
);
export default AppRoute;
```

Q. How to Implement Routing in React?

Step 1
- Install React Router

Step 2
- Create Navigation

Step 3
- Create Routes

Q. What are the roles of <Routes> & <Route> component in React Routing? V. IMP.

- The <Routes> component is used as the root container for declaring your **collection of routes**.
- The <Route> component is used to define a route and specify the component that should render when the **route matches**.
  - For example, in this code if user enter "websitename.com/about" in url, then matching "About" component will be rendered.

```jsx
import React from "react";
import { Routes, Route, Link } from "react-router-dom";

{/* Routes */}
<Routes>
  <Route path="/" element={<Home />} />
  <Route path="/about" element={<About />} />
  <Route path="/contact" element={<Contact />} />
</Routes>
```

Q. What are Route Parameters in React Routing?

- Route parameters in React Router are a way to pass dynamic values(data) to the component **as part of the URL path**.

```
{/* userId is the route parameter */}
<Route path="/users/:userId" component={UserProfile} />
```

Q. What is the role of Switch Component in React Routing?

- Switch component ensures that only the **first matching <Route> is rendered** and rest are ignored.
  - For example, Switch is commonly used to handle 404 or "not found" routes.

```jsx
import { Switch, Route } from 'react-router-dom';

<Switch>
  <Route path="/users" element={<UsersList />} />
  <Route path="/users/:id" element={<UserProfile />} />
</Switch>
```

Q. What is the role of exact prop in React Routing?

- exact prop is used with the <Route> component to **match exactly** to the provided path.

```
{/* without exact  (default behavior)*/}
{/* match "about", "/about/team", "/about/contact" etc. */}
<Route path="/about" component={About} />
```

```
{/* with exact */}
{/* only match "about"*/}
<Route path="/about" exact component={About} />
```

7: Hooks -useState/ useEffect

Q1. What are React Hooks? What are the Top React Hooks? V. IMP.

Q2. What are State, Stateless, Stateful and State Management terms? V. IMP.

Q3. What is the role of useState() hook and how it works? V. IMP.

Q4. What is the role of useEffect(). How it works and what is its use? V. IMP.

Q5. What is Dependency Array in useEffect() hook?

Q6. What is the meaning of the empty array [] in the useEffect()?

Q. What are React Hooks? What are the Top React Hooks? V. IMP.

1. React Hooks are inbuilt functions provided by React that allow functional components to **use state** and **lifecycle features**.
2. Before Hooks, class components lifecycle methods were used to maintain state in React applications.
3. To use React Hook first we first have to import it from React library:

```
// Import Hook from the React library
import React, { useState } from "react";
```

1. useState:	State
2. useEffect:	Side effects
3. useContext:	Context
4. useReducer:	Complex state
5. useCallback:	Memoization
6. useMemo:	Performance
7. useRef:	Refs
8. useLayoutEffect:	Synchronous Side effects.

Q. What are React Hooks? What are the Top React Hooks? V. IMP.

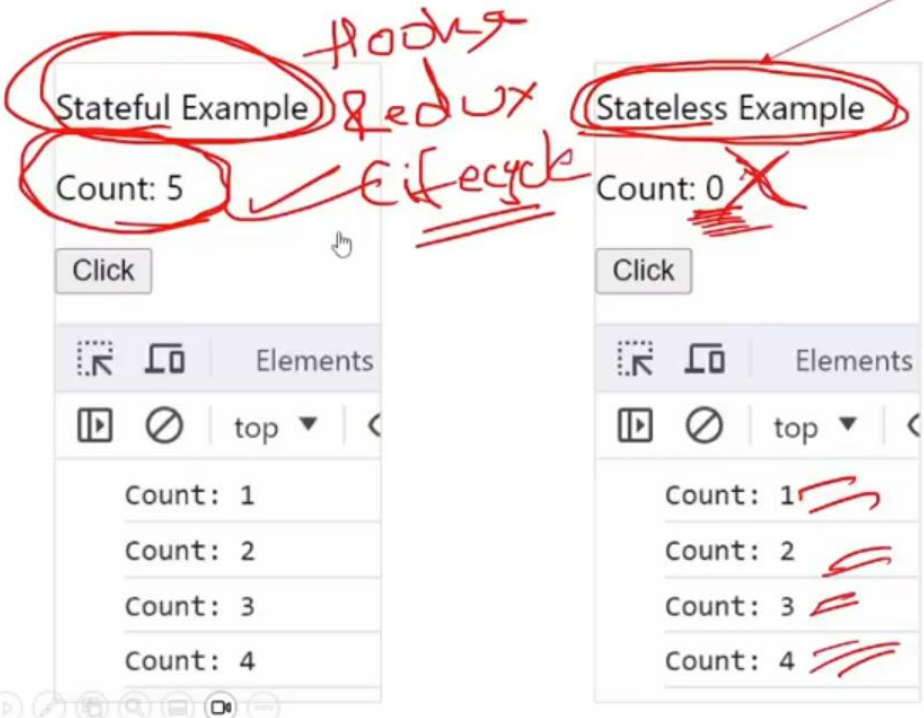
1. React Hooks are inbuilt functions provided by React that allow functional components to **use state** and **lifecycle features**.
2. Before Hooks, class components lifecycle methods were used to maintain state in React applications.
3. To use React Hook first we first have to import it from React library:

```
// Import Hook from the React library
import React, { useState } from "react";
```

1. useState:	State
2. useEffect:	Side effects
3. useContext:	Context
4. useReducer:	Complex state
5. useCallback:	Memoization
6. useMemo:	Performance
7. useRef:	Refs
8. useLayoutEffect:	Synchronous Side effects.

Q. What are state, stateless, stateful and state management terms?

- "state" refers to the current data of the component.
- Stateful or state management means, when a user performs some actions on the UI, then the React application should be able to **update and re-render that data or state** on the UI.




```
ooks > JS ComponentState.js > ...
import React from "react";

// Stateless Example
function ComponentState() {
  let count = 0; // Initial state

  const increment = () => {
    count += 1; // State updated
    console.log(`Count: ${count}`);
  };

  return (
    <div>
      <p>Stateless Example</p>
      <p>Count: {count}</p> {/* Not updating */}
      <button onClick={increment}>Click</button>
    </div>
  );
}
export default ComponentState;
```

Q. What is the role of useState() hook and how it works? V. IMP.

Stateful Example

Count: 5

Click

Elements

top

Count: 1

Count: 2

Count: 3

Count: 4

Count: 5

```jsx
import React, { useState } from "react";

function UseState() {
  // array destructuring
  const [count, setCount] = useState(0);

  const increment = () => {
    setCount(count + 1);
    console.log(`Count: ${count + 1}`);
  };

  return (
    <div>
      <p>Stateful Example</p>
      <p>Count: {count}</p> {/* Updating */}
      <button onClick={increment}>Click</button>
    </div>
  );
}
export default UseState;
```

Q. What is the role of useState() hook and how it works? V. IMP.

- The useState hook enables functional components to **manage state**.

- useState() working: usestate() function accept the initial state value as the parameter and returns an array with two elements:

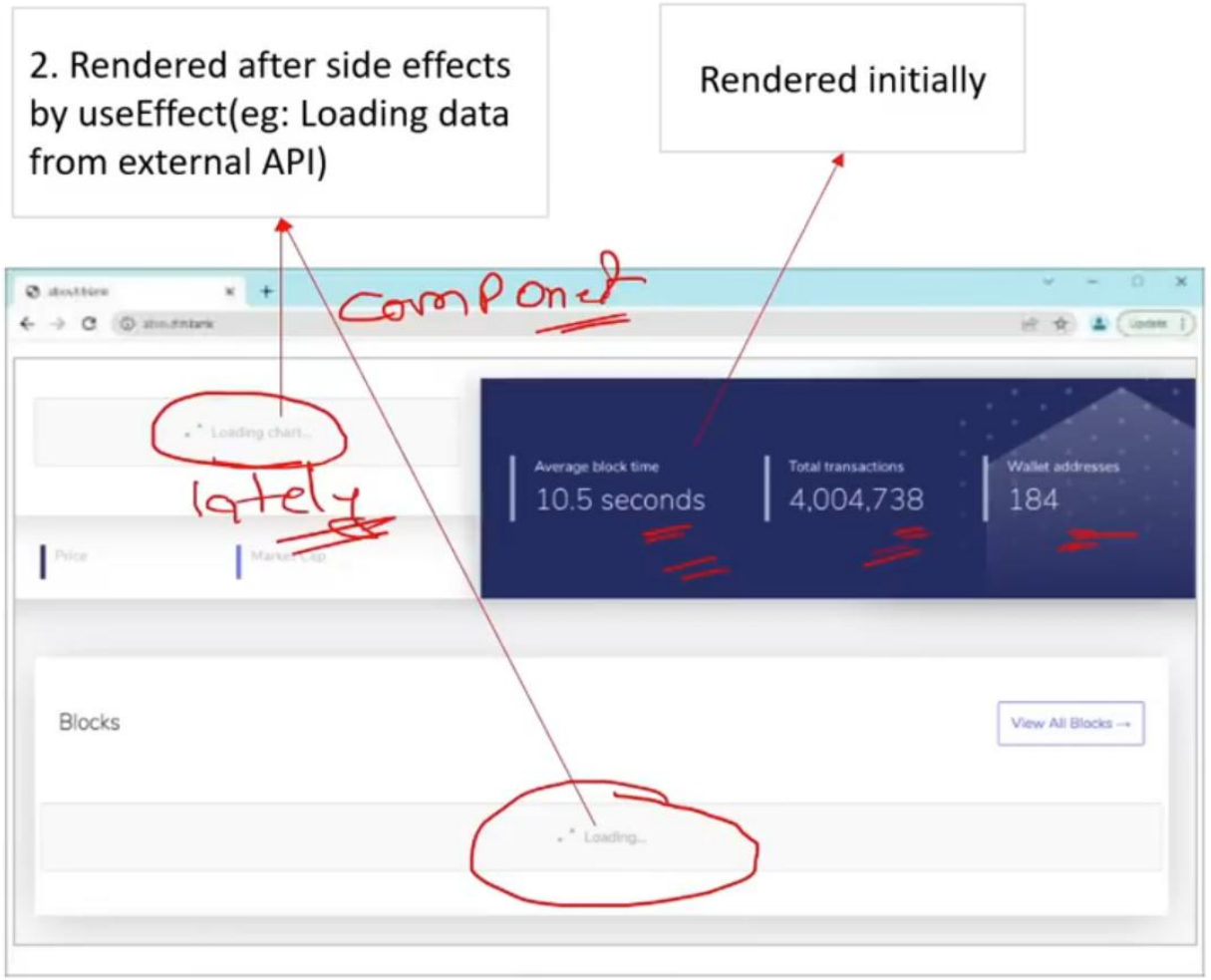
1. The first element is the current state value (count in this code).
2. Second element is the function that is used to update the state (setCount in this code).

- The concept of assign array elements to individual variables is called **array destructuring**.

```
// state is the current state value.
// setState is a function that used to update the state.
const [state, setState] = useState(initialValue);
```

Q. What is the role of useEffect(). How it works and what is its use? V. IMP.

- The useEffect Hook in React is used to perform **side effects** in functional components.
- For example, data fetching from API, subscriptions or any other operation that needs to be performed after the component has been rendered.

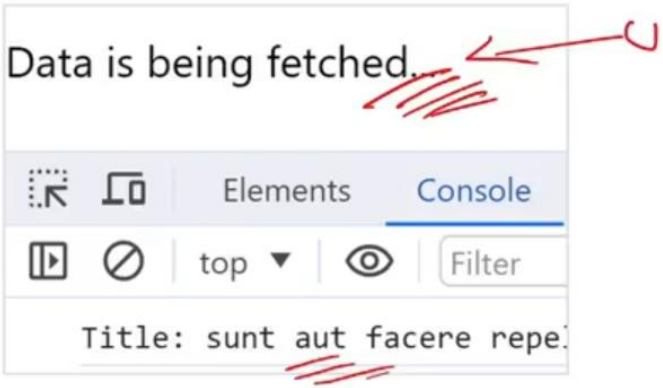


Q. What is useEffect() hook and when to use it? V. IMP.

❖ **2 points to remember about useEffect():**

1. useEffect() is called after the component renders. Example, side effects.
2. useEffect() function will accept two parameter: (Effect function, dependency array).

Browser Output




```jsx
import React, { useEffect } from "react";
function UseEffect() {
  useEffect(() => {
    const fetchData = async () => {
      const response = await fetch(
        "https://jsonplaceholder.typicode.com/posts/1"
      );
      const result = await response.json();
      console.log("Title:", result.title);
    };
    fetchData();
  }, []);
  return (
    <div>
      <p>Data is being fetched...</p>
    </div>
  );
}
export default UseEffect;
```

Q. What is the meaning of the empty array [] in the useEffect()?

- An empty array [] indicates that the effect function should only **run once**.

ooks > JS UseEffect.js > ...

```
function UseEffect() {
  const [data, setData] = useState({});

  useEffect(() => {
    const fetchData = async () => {
      const response = await fetch(
        "https://jsonplaceholder.typicode.com/posts/1"
      );
      const result = await response.json();
      setData(result);
    };
    fetchData();
  }, []);

  return (
    <div>
      <p>Title: {data.title}</p>
    </div>
  );
}
```

8: Hooks - useContext/ useReducer

Q1. What is the role of useContext() hook? V. IMP.

Q2. What is createContext() method? What are Provider & Consumer properties?

Q3. When to use useContext() hook instead of props in real applications?

Q4. What are the similarities between useState() and useReducer() hook?

Q5. What is useReducer() hook? When to use useState() and when useReducer()? V. IMP.

Q6. What are the differences between useState() and useReducer() Hook?

Q7. What are dispatch & reducer function in useReducer Hook?

Q8. What is the purpose of passing initial state as an object in UseReducer?

Q. What is the role of useContext() hook? V. IMP.

- useContext in React provides a way to pass data from parent to child component without using props.

ooks > JS MyContext.js > ...

```
import { createContext } from "react";

const MyContext = createContext();

export default MyContext;
```

Hello from Context!

ooks > JS Parent.js > ...

```
const Parent = () => {
  const contextValue = "Hello from Context!";

  return (
    <MyContext.Provider value={contextValue}>
      {/* Your component tree */}
      <Child></Child>
    </MyContext.Provider>
  );
};
export default Parent;
```

ooks > JS Child.js > ...

```
const Child = () => {
  const contextValue = useContext(MyContext);

  return <p>{contextValue}</p>;
  // return (
  //   <MyContext.Consumer>
  //     {(contextValue) => <div>{contextValue}</div>}
  //   </MyContext.Consumer>
  // );
};
export default Child;
```

Q. What is createContext() method? What are Provider & Consumer properties?

- createContext() function returns an object with Provider and Consumer properties.
- The Provider property is responsible for providing the context value to all its child components.
- useContext() method or Consumer property can be used to consume the context value in child components.

ooks > JS MyContext.js > ...

```
import { createContext } from "react";

const MyContext = createContext();

export default MyContext;
```

ooks > JS Parent.js > ...

```
const Parent = () => {
  const contextValue = "Hello from Context!";

  return (
    <MyContext.Provider value={contextValue}>
      {/* Your component tree */}
      <Child></Child>
    </MyContext.Provider>
  );
};
export default Parent;
```

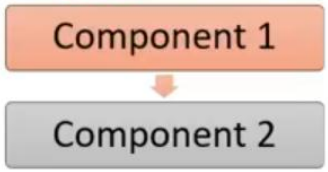
ooks > JS Child.js > ...

```
const Child = () => {
  const contextValue = useContext(MyContext);

  return <p>{contextValue}</p>;
  // return (
  //   <MyContext.Consumer>
  //     {(contextValue) => <div>{contextValue}</div>}
  //   </MyContext.Consumer>
  // );
};
export default Child;
```

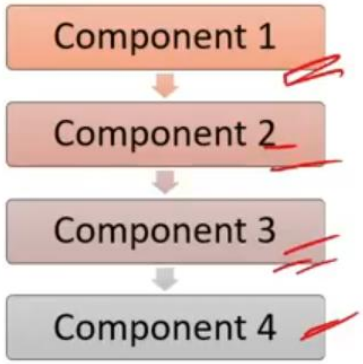
Q. When to use useContext() hook instead of props in real applications?

- Use useContext instead of props when you want to **avoid prop drilling** and access context values directly within deeply nested components.

- Props are good



- useContext is good



1. Theme Switching (Dark/ Light)

- You can centralize and pass the theme selection of the application from the parent to all the deep child components.

2. Localization (language selection)

- You can centralize and pass the language selection of the application from the parent to all the child components.

3. Centralize Configuration Settings

- Common configuration settings like API endpoints can be centralized and change in the parent component will pass the setting to all its child components

4. User Preferences

- Any other user preferences apart from theme and localization can also be centralized.

5. Notification System

- Components that trigger or display notifications can access the notification state from the context.

11: Components LifeCycle Methods - I

Q1. What are Component life cycle phases? V. IMP.

Q2. What are Component life cycle methods? V. IMP.

Q3. What are Constructors in class components? When to use them?

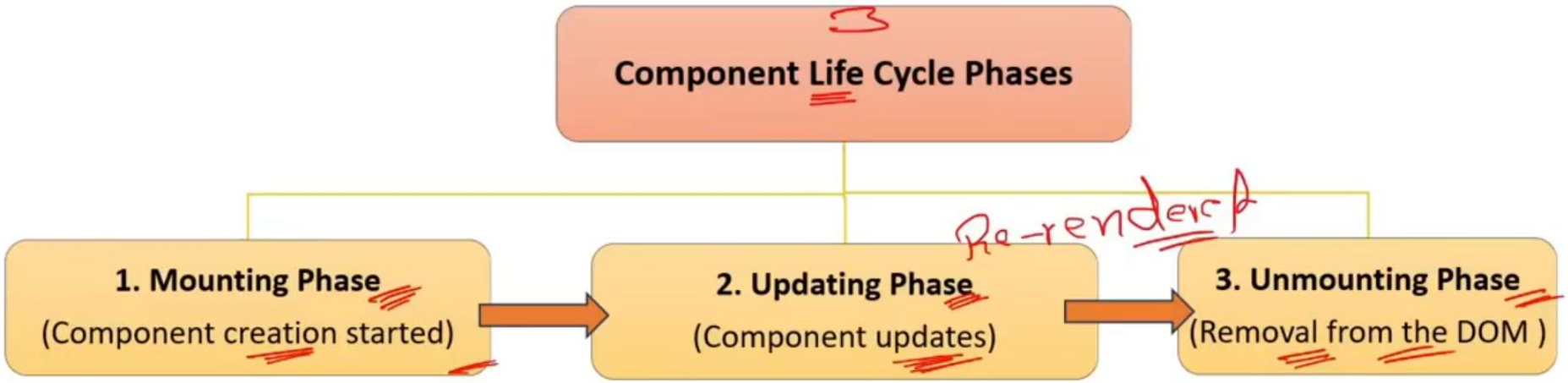
Q4. What is the role of super keyword in constructor?

Q5. What is the role of render() method in component life cycle?

Q6. How the State can be maintained in a class component?

Q7. What is the role of componentDidMount() method in component life cycle?

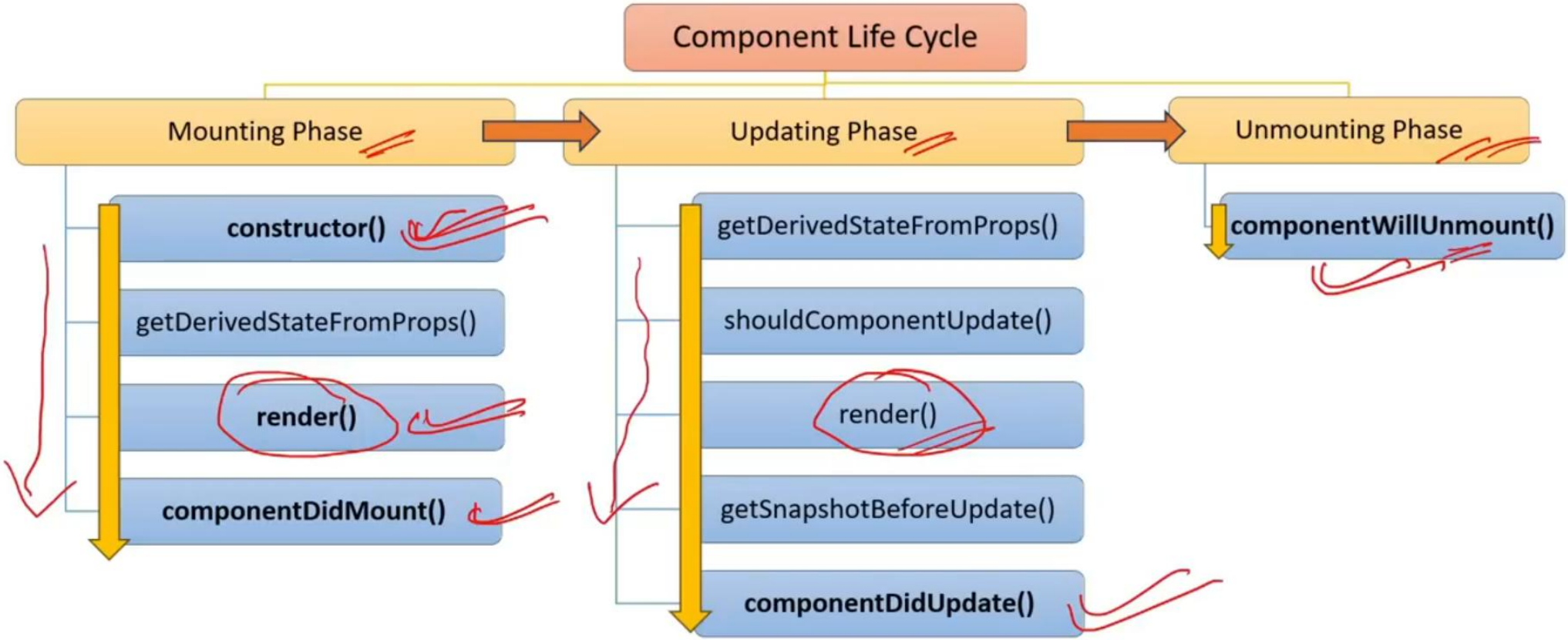
Q. What are component Life Cycle Phases?



- This phase occurs when an instance of a component is being created and inserted into the DOM.
- This phase occurs when a component is being re-rendered as a result of changes to either its props or state.
- This phase occurs when a component is being removed from the DOM.

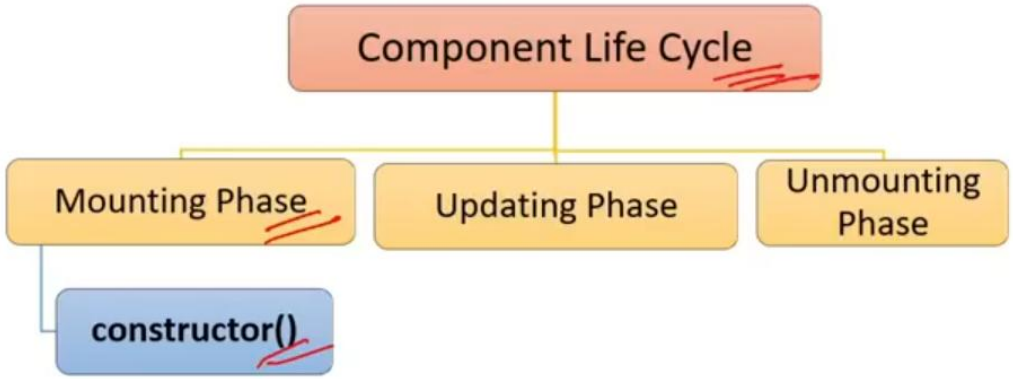
Q. What are component Life Cycle Methods? V. IMP.

- Component lifecycle methods are special methods that get called at various stages of a component's life.



Q. What are Constructors in class components? When to use them?

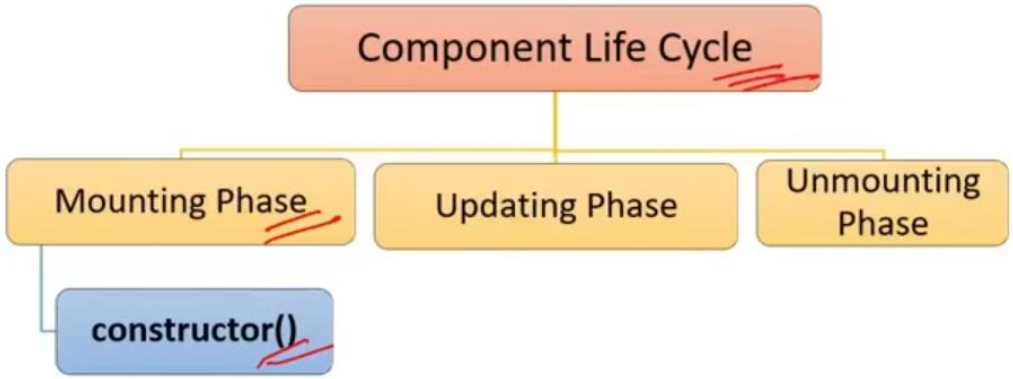
- constructor is a **special method** that is called when an instance of the class is created.
- Constructor is **used for initializing the component's state** or performing any setup that is needed before the component is rendered.




```jsx
class ConstructorExample extends Component {
  constructor(props) {
    super(props);

    // Initialize the state
    this.state = {
      count: 0,
    };
  }

  render() {
    return (
      <h2>Count: {this.state.count}</h2>
    );
  }
}

export default ConstructorExample;
```

Q. What are Constructors in class components? When to use them?

- constructor is a **special method** that is called when an instance of the class is created.
- Constructor is **used for initializing the component's state** or performing any setup that is needed before the component is rendered.



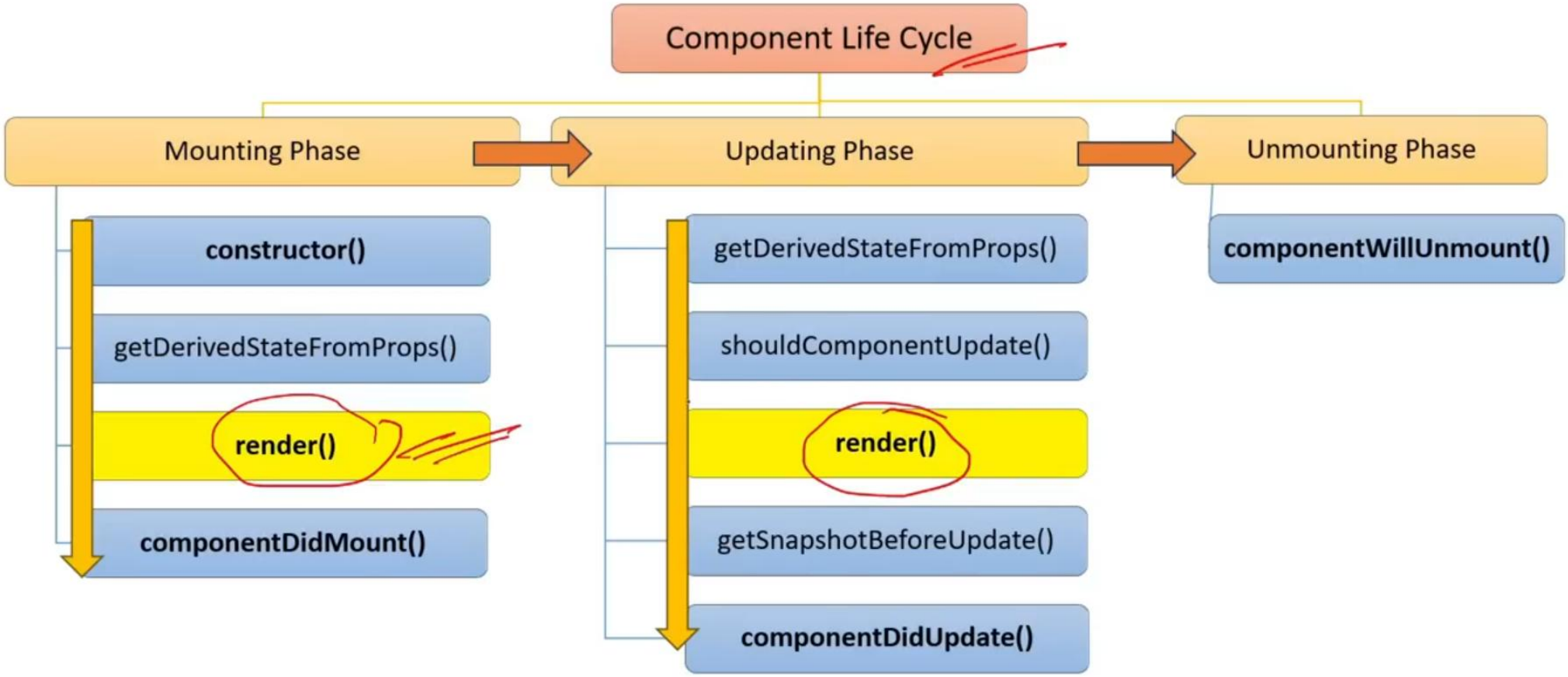

```
class ConstructorExample extends Component {
  constructor(props) {
    super(props);

    // Initialize the state
    this.state = {
      count: 0,
    };
  }

  render() {
    return (
      <h2>Count: {this.state.count}</h2>
    );
  }
}

export default ConstructorExample;
```

Q. What is the role of super keyword in constructor?

- super keyword is used in the constructor of a class component to **call the constructor of the parent class**.
- This is necessary to ensure that the initialization logic of the parent class is executed.

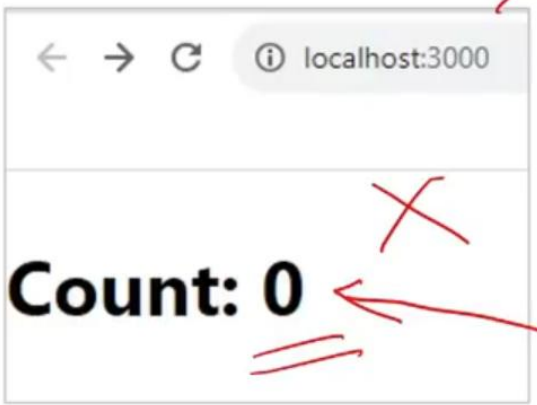
```
class ConstructorExample extends Component {
  constructor(props) {
    super(props);

    // Initialize the state
    this.state = {
      count: 0,
    };
  }

  render() {
    return (
      <h2>Count: {this.state.count}</h2>
    );
  }
}

export default ConstructorExample;
```

Q. What is the role of render() method in component life cycle?



Q. What is the role of render() method in component life cycle?

- Render() method **returns the React elements** that will be rendered to the DOM.




```jsx
class ConstructorExample extends Component {
  constructor(props) {
    super(props);

    // Initialize the state
    this.state = {
      count: 0,
    };
  }

  render() {
    return (
      <h2>Count: {this.state.count}</h2>
    );
  }
}

export default ConstructorExample;
```

Q. How the State can be maintained in a class component? V. IMP.

❖ **Two step process to maintain state:**

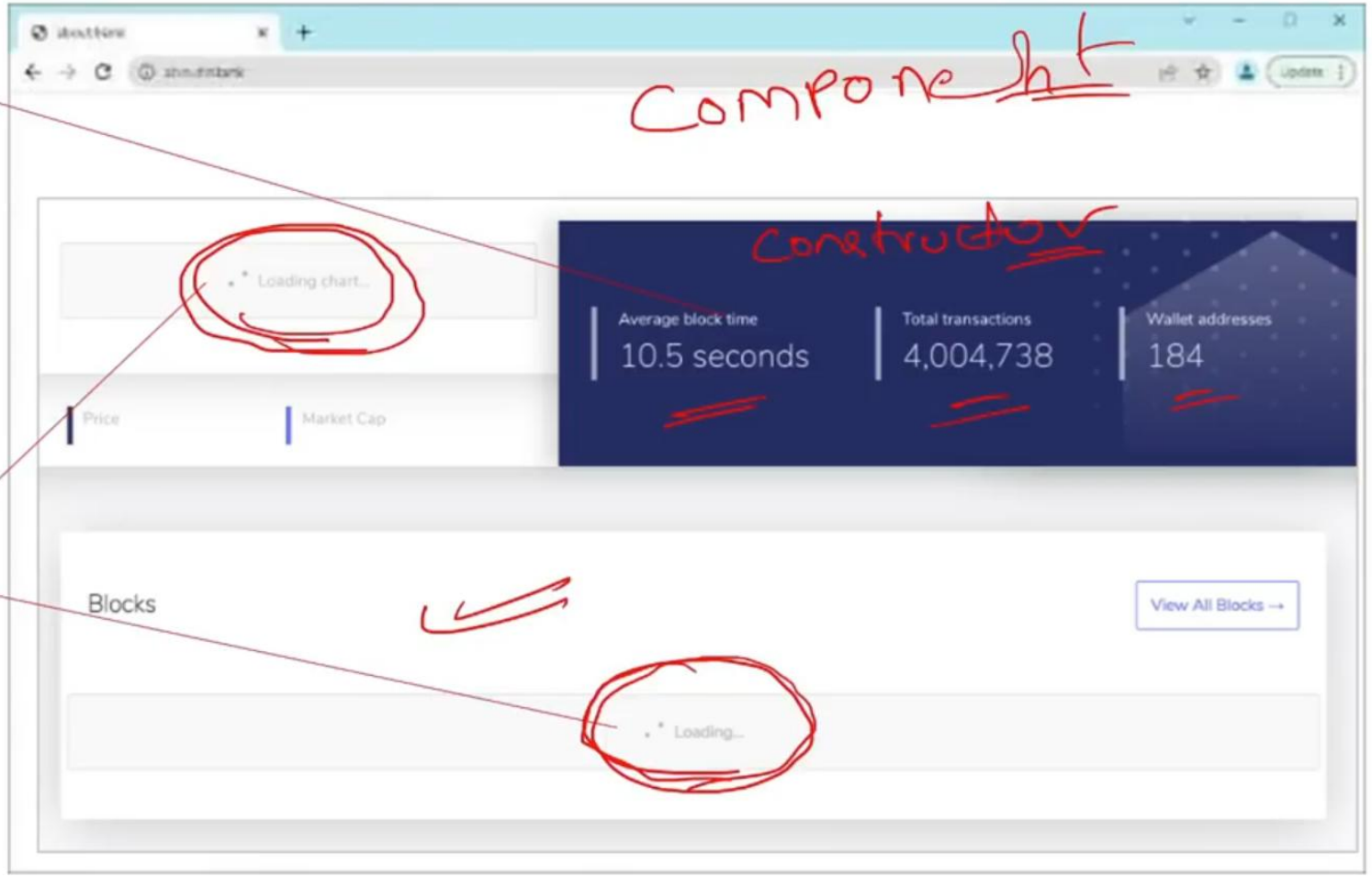
1. **this.setState()** method is used to update the state.
2. **this.state** property is used to render the update state in DOM.


```
class StateComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0, // Initializing the state
    };
  }
  render() {
    return (
      <div>
        <h2>Counter: {this.state.count}</h2>
        <button onClick={this.handleInc}>Inc</button>
      </div>
    );
  }

  handleInc = () => {
    this.setState((prevState) => ({
      count: prevState.count + 1,
    }));
  };
}
export default StateComponent;
```

Q. What is the role of componentDidMount() method in component life cycle?

1. Rendered after constructor initialization.

2. Rendered after componentDidMount() to run **Side effects**(ex: loading data from external API) and then call render() method of updating phase again.



Q. What is the role of componentDidMount() method in component life cycle?

- componentDidMount() lifecycle method in React is the part of mounting phase and is **called after a component has been rendered** to the DOM.
- Mostly used for **side effects**. For example, external data fetching or setting up subscriptions.

```
// Fetch data from API example using
// componentDidMount life cycle method
class CompDidMount extends Component {
  constructor(props) {
    // 1. Component intialization
    this.state = {
      data: null
    };
  }

  // 2. componentDidMount is called after
  // the component is added to the DOM
  componentDidMount() {
    // 3. Fetch data from an API and
    // update state with fetch data
    fetchData().then((data) => {
      this.setState({
        data: data
      });
    });
  }

  // 4. Render the component's UI using
  // the fetched data
  render() {
    return (
      <div>
        <p>Data: {this.state.data}</p>
      </div>
    );
  }
}
export default CompDidMount;
```

13: Controlled & Uncontrolled Components

Q1. What are Controlled Components in React? V. IMP.

Q2. What are the Differences btw Controlled & Uncontrolled Components? V. IMP.

Q3. What are characteristics of controlled components?

Q4. What are the advantages of using controlled components in React forms?

Q5. How to handle forms in React?

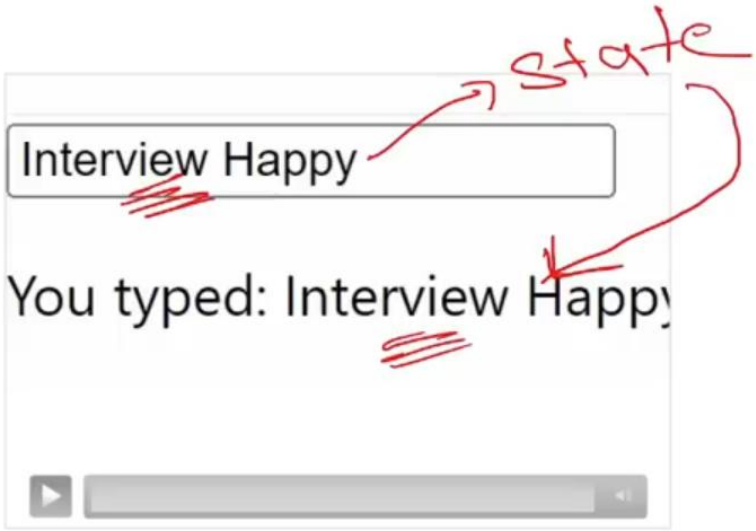
Q6. How can you handle multiple input fields in a controlled form?

Q7. How do you handle form validation in a controlled component?

Q8. In what scenarios might using uncontrolled components be advantageous?

Q. What are Controlled Components in React? V. IMP.

- A controlled component is a component whose form elements (like input fields or checkboxes) are **controlled by the state** of the application.




```jsx
const Controlled = () => {
  // State to store the input value
  const [inputValue, setInputValue] = useState("");

  // Event handler for input changes
  const handleInputChange = (e) => {
    // Update the state with the new input value
    setInputValue(e.target.value);
  };

  return (
    <div>
      {/* Input controlled by the state*/}
      <input type="text" value={inputValue}
        onChange={handleInputChange} placeholder="Type..."/>

      {/* Display the current state value */}
      <p>You typed: {inputValue}</p>
    </div>
  );
};
export default Controlled;
```

Q. What are the Differences btw Controlled & Uncontrolled Components? V. IMP.

Controlled Components	Uncontrolled Components
1. Values are controlled by **React state**.	Values are not controlled by React state.
2. Event handlers **update React state**.	No explicit state update; values can be accessed **directly from the DOM**.
3. Don't depend on useRef().	Commonly uses **useRef()** to access form element values.
4. **Re-renders** on state changes.	Less re-rendering since values are not directly tied to React state.
5. A **recommended** and standard practice for form handling in React.	Useful in certain scenarios but less commonly considered a best practice.

Q. What are characteristics of controlled components?

❖ **Characteristics of controlled components:**

1. **State Control:** The value of the form element is stored in the component's state.
2. **Event Handling:** Changes to the form element trigger an event (e.g., onChange for input fields).
3. **State Update:** The event handler updates the component's state with the new value of the form element.
4. **Re-rendering:** The component re-renders with the updated state, and the form element reflects the new value.

```jsx
const Controlled = () => {
  // State to store the input value
  const [inputValue, setInputValue] = useState("");

  // Event handler for input changes
  const handleInputChange = (e) => {
    // Update the state with the new input value
    setInputValue(e.target.value);
  };

  return (
    <div>
      {/* Input controlled by the state*/}
      <input type="text" value={inputValue}
        onChange={handleInputChange} placeholder="Type..."/>

      {/* Display the current state value */}
      <p>You typed: {inputValue}</p>
    </div>
  );
};
export default Controlled;
```

Q. What are the advantages of using controlled components in React forms?

❖ Top 3 benefits of using controlled components in React forms:

1. In controlled components, form elements have their values managed by React state, ensuring a **single source of truth**.

2. This approach facilitates predictable and synchronized updates, making it easier to implement features such as form validation, and dynamic rendering, and seamless integration with React's lifecycle methods.

3. Controlled components offer better control and maintainability compared to uncontrolled components, making them the best practice for handling forms in React applications.

state → Single source of truth

Q. How to handle forms in React?

- The preferred and recommended approach for handling forms in React is **by using controlled components**.

Q. How can you handle multiple input fields in a controlled form?

- Maintain **separate state variables** for each input field and update them individually using the **onChange event**.

Q. How do you handle form validation in a controlled component?

- By using conditional rendering based on the state and validate input values before updating the state.

Q. In what scenarios might using uncontrolled components be advantageous?

- Uncontrolled components can be beneficial when integrating with non-React libraries, or when dealing with forms where controlled components are not possible.

14: Code Splitting

Q1. What is Code Splitting in React? V. IMP.

Q2. How to Implement Code Splitting in React? V. IMP.

Q3. What is the role of Lazy and Suspense methods in React? V. IMP.

Q4. What are the Pros and Cons of Code Splitting?

Q5. What is the role of the import() function in code splitting?

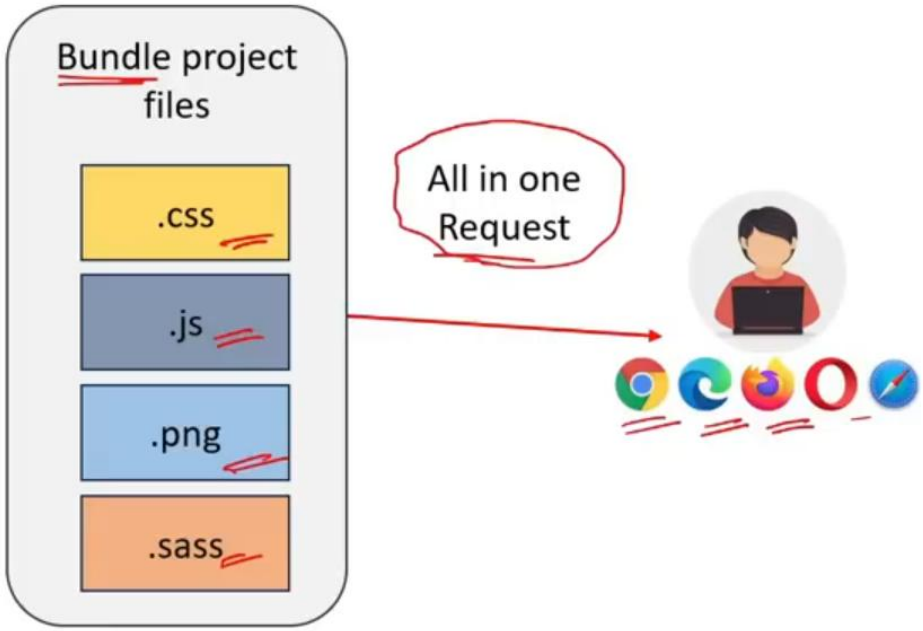
Q6. What is the purpose of the fallback prop in Suspense?

Q7. Can you dynamically load CSS files using code splitting in React?

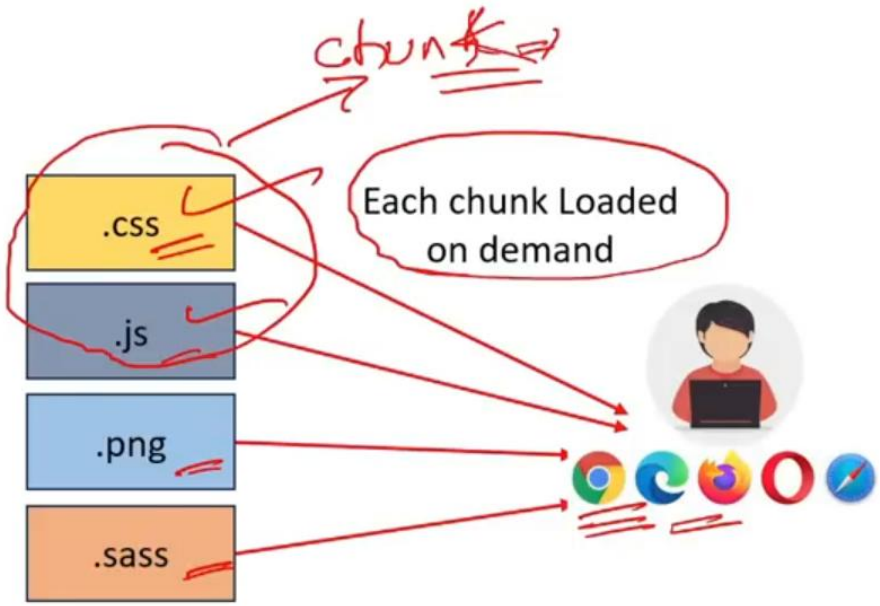
Q8. How do you inspect and analyze the generated chunks in a React application?

Q. What is Code Splitting in React? V. IMP.

- Code splitting is a technique to split the JavaScript bundle into **smaller chunks**, which are **loaded on-demand**.



Without Code Splitting



With Code Splitting Implemented

Q. How to Implement Code Splitting in React? V. IMP.

❖ **3 steps for Code splitting in React:**

1. Use **React.lazy()** to lazily import components.
2. Wrap components with **Suspense** to handle loading.
3. Configure your build tool (e.g., **Webpack**) for dynamic imports.

Q. How to Implement Code Splitting in React? V. IMP.

```jsx
// CodeSplit.js: Component
const CodeSplit = () => {
  return <div>My component!</div>;
};

export default CodeSplit;
```


```jsx
// App.js
import React, { lazy, Suspense } from "react";

const CodeSplit = lazy(() => import("./Others/CodeSplit"));

function App() {
  return (
    <div>
      <Suspense fallback={<div>Loading...</div>}>
        <CodeSplit />
      </Suspense>
    </div>
  );
}
export default App;
```

```jsx
// webpack installation command
// npm install webpack webpack-cli --save-dev

// webpack.config.js: under root
const path = require('path');

module.exports = {
  // other webpack configuration...
  output: {
    filename: '[name].bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
};
```

Q. What is the role of Lazy and Suspense methods in React? V. IMP.

- React.lazy is a function that allows you to **load a component lazily**.
- It enables code splitting by allowing you to import a component asynchronously/ dynamically, meaning component is loaded only when needed only.

- The Suspense component is used to display a fallback UI while the lazily loaded component is being fetched.

```jsx
// App.js
import React, { lazy, Suspense } from "react";

const CodeSplit = lazy(() => import("./Others/CodeSplit"));

function App() {
  return (
    <div>
      <Suspense fallback={<div>Loading...</div>}>
        <CodeSplit /> => On demand
      </Suspense>
    </div>
  );
}
export default App;
```

Q. What are the Pros and Cons of Code Splitting?

5 Pros of Code Splitting:

chunks

1. Faster Initial Load Time:

data

2. Optimized Bandwidth Usage:

Focused

3. Improved Caching:

4. Parallel Loading:

Independent

5. Easier Maintenance:

Q. What are the Pros and Cons of Code Splitting?

5 Cons of Code Splitting:

1. Complexity:	Implementing code splitting introduces additional complexity to your application. This complexity can make the development process slow.
2. Tooling Dependencies:	Proper code splitting often requires specific build tools and configurations, such as Webpack and Babel. Managing these tools is challenging.
3. Potential for Runtime Errors:	Dynamically loading code at runtime can introduce the possibility of runtime errors. Careful testing is necessary to catch such issues.
4. Increased Number of Requests:	Code splitting may increase the number of HTTP requests needed to fetch all the necessary chunks. This can impact performance.
5. Learning Curve:	Developers who are new to code splitting may need time to understand the concepts and best practices. This can be a challenging.

Webpack

Each chunk separate request

Q. What is the role of the import() function in code splitting?

- The import() function returns a promise that allow **dynamic loading of modules**.

```jsx
// App.js
import React, { lazy, Suspense } from "react";

const CodeSplit = lazy(() => import("./Others/CodeSplit"));

function App() {
  return (
    <div>
      <Suspense fallback={<div>Loading...</div>}>
        <CodeSplit />
      </Suspense>
    </div>
  );
}
export default App;
```

Q. What is the purpose of the fallback prop in Suspense?

- The fallback prop provides a **loading indicator** or UI while the dynamically imported component is being loaded.

```jsx
// App.js
import React, { lazy, Suspense } from "react";

const CodeSplit = lazy(() => import("./Others/CodeSplit"));

function App() {
  return (
    <div>
      <Suspense fallback={<div>Loading...</div>}>
        <CodeSplit />
      </Suspense>
    </div>
  );
}
export default App;
```

Q. Can you dynamically load CSS files using code splitting in React?

- **Yes**, using dynamic import() for CSS files allows you to load styles on-demand along with the corresponding components.

15: React - Others

Q1. What is a Higher-Order Component in React? V. IMP.

Q2. What are the 5 ways to Style React components? Explain Inline Styles?

Q3. What are the differences between React & React Native?

Q4. What is GraphQL?

Q5. What are the Top 3 ways to achieve state management? When to use what in React?

Q6. How can you Implement Authentication in a React application? V. IMP.

Q7. What is the use of React Profiler?

Q8. What is the difference between fetch & axios for api calls in React?

Q9. What are the popular Testing Libraries for React?

Q10. How can you Optimize Performance in a React application? V. IMP.

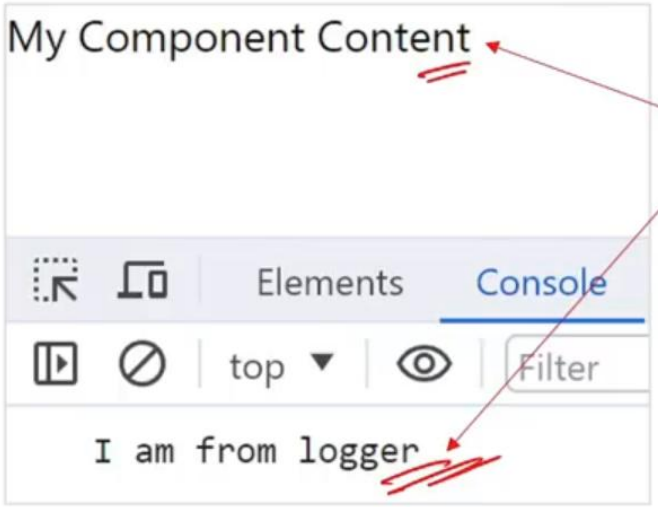
Q11. Explain Reactive Programming with example?

Q12. In how many ways can we implement Reactive Programming in React?

Q13. How to pass data from child component to parent Component in Re[illegible]

Q. What is a Higher-Order Component in React? V. IMP.

- A Higher-Order Component is a component which takes **another component as an argument** and adds extra features to another component.
- HOC can be used for providing **logging functionality** to all the components in a reusable way.




```
// HocLogger.js: High Order Component
const HocLogger = (HocUse) => {

  return function WithLogger(props) {
    console.log("I am from logger");
    return <HocUse />;
  };

};
export default HocLogger;
```

```
// HocUse.js: Normal Component will HOC
const HocUse = () => {
  return <div>My Component Content</div>;
};

export default Hoc(HocUse);
```

HocUse

```
// index.js
root.render(<HocUse></HocUse>);
```

Q. What are the 5 ways to Style React components? Explain Inline Styles?

5 ways to Style React components

1. Inline Styles
2. CSS Stylesheets
3. CSS-Modules
4. Global Stylesheets
5. CSS Frameworks

```jsx
import React from "react";

const AppInlineStyle = () => {
  return (
    <div>
      <h1 style={inlineStyles.title}>Inline</h1>
    </div>
  );
};

const inlineStyles = {
  title: {
    color: "blue",
    fontSize: "24px",
  },
};

export default AppInlineStyle;
```

Inline

Q. What are the differences between React & React Native?

React	React Native
1. React is a library	React Native is a framework.
2. React is used for building web interfaces.	React Native is used for building mobile applications.
3. Run on Web browsers.	Run on iOS and Android platforms.
4. HTML and CSS are used for UI.	Native UI components (e.g., View, Text) are used for UI.
5. Deployed as web applications.	Deployed through app stores(e.g., App Store, Google Play).

Q. What is GraphQL?

- GraphQL is a **query language for APIs** (Application Programming Interfaces) and a runtime for executing those queries with your existing data.
- GraphQL and React are often used together. React components can use GraphQL queries to fetch the data required for rendering.

GraphQL

```
<!--GraphQL query example -->
query {
  user(id: 1) {
    id
    name
    email
    posts {
      title
      content
    }
  }
}
```

Q. What are the Top 3 ways to achieve state management? When to use what in React?

1. useState Hook:

- **When to use:** Simple component-level state.
- **Reason:** Ideal for applications having small components and isolated state because it is Lightweight and built into React only.

2. Context API:

- **When to use:** Prop drilling avoidance for sharing global data.
- **Reason:** Simplifies data passing through the component tree, reducing the need for manual prop drilling.

3. Redux:

- **When to use:** Large-scale applications with complex state.
- **Reason:** Centralized store and actions provide a predictable state management pattern, aiding in debugging and scalability.

Q. How can you Implement Authentication in a React application? V. IMP.

Front-end/ Client-side

Middleware/ Server-side/ Backend

1. POST: {username, password}

2. Authenticate & create JWT **Token**

3. Return Response {JWT token}

4. Store JWT token at local storage

5. Request Data {JWT token: Header}

6. Validate token signature

7. Send Data

8. Display data on browser

Q. What is the use of React Profiler?

- React Profiler is a set of tools in React that allows developers to profile(analyze) the **performance** of a React application.

```
// Wrap the section of code you want to profile
// with the React.Profiler component.
<React.Profiler id="example" onRender={callback}>
  {/* Your code to profile */}
</React.Profiler>;

// Define a callback function (onRender) that
// will be called whenever the component tree
// within the Profiler is committed.
function callback(id,phase,actualDuration,baseDuration,
  startTime,commitTime
) {
  // Process profiling data
  // Start time, End Time, Execution Time
}
```

Q. What is the difference between fetch & axios for api calls in React?

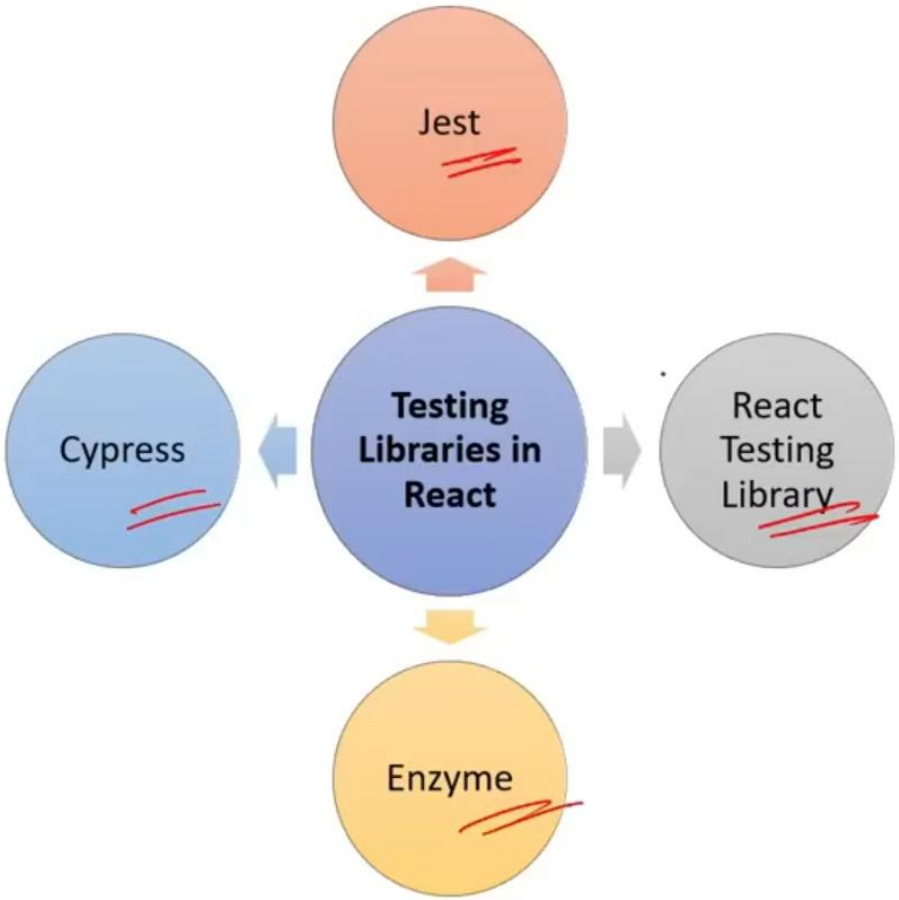
```
fetch("https://api.example.com/data")
  .then((response) => response.json())
  .then((data) => console.log(data))
  .catch((error) => console.error("Error:", error));
```

1. fetch is a **built-in JavaScript function**, so it doesn't require any additional libraries.
2. fetch **returns Promises**, making it easy to work with asynchronous code using async/await syntax.
3. If you want to keep **http requests simple,** fetch is a good choice.

```
// installation command: npm install axios
import axios from "axios";
axios
  .get("https://api.example.com/data")
  .then((response) => console.log(response.data))
  .catch((error) => console.error("Error:", error));
```

1. Axios is a **third-party library,** that simplifies the process of making HTTP requests.
2. Axios allows you to **use interceptors**, which can be good for tasks like request/response logging, authentication, and error handling.
3. If you want to intercept http request/response, or improve error handling then Azios has more features to do it.

Q. What are the popular Testing Libraries for React?



Q. How can you Optimize Performance in a React application?

1. Memoization with useMemo and useCallback:	Use this hooks to memoize values and, reducing unnecessary recalculations.
2. Optimizing Renders with React.Fragment:	Use it to avoid unnecessary wrapper elements that could cause additional DOM nodes.
3. Lazy Loading with React.lazy:	Use it to load components lazily, reducing the initial bundle size and improving initial loading performance.
4. Code Splitting:	Employ code splitting to divide your application into smaller chunks that are loaded on demand, improving initial load times.
5. Optimizing Images and Assets:	Compress and optimize images, use responsive images, and leverage lazy loading for images to reduce network and rendering overhead.

Q. In how many ways can we implement Reactive Programming in React?

1. State and Props	Reacting to changes in local component state and passing data reactively through props.
2. React Hooks:	Leveraging **useState** and **useEffect** hooks for managing state and side effects in functional components.
3. Event Handling:	Reacting to user interactions through event handling and updating state accordingly.
4. Context API:	Sharing and managing global state reactively across components using the Context API.
5. Redux:	Using state management libraries like Redux for managing complex application state reactively.
6. Component Lifecycle Methods:	Using class components and lifecycle methods for handling side effects and updates.
7. Async/Await:	Utilizing **async/await** syntax for handling asynchronous operations reactively.
8. RxJS and Observables:	Leveraging RxJS for handling asynchronous operations and data streams in a reactive manner.

Q. How to pass data from child component to parent Component in React?

- Parent provides a **callback function** to child and then child component can then invoke this callback to pass data back to the parent.

```jsx
const ParentComponent = () => {
  // Callback to receive data from the child
  const callback = (data) => {
    console.log("Data from Child:", data);
  };

  return (
    <div>
      {/* Pass the callback to the child */}
      <ChildComponent fromChild={callback} />
    </div>
  );
};
export default ParentComponent;
```

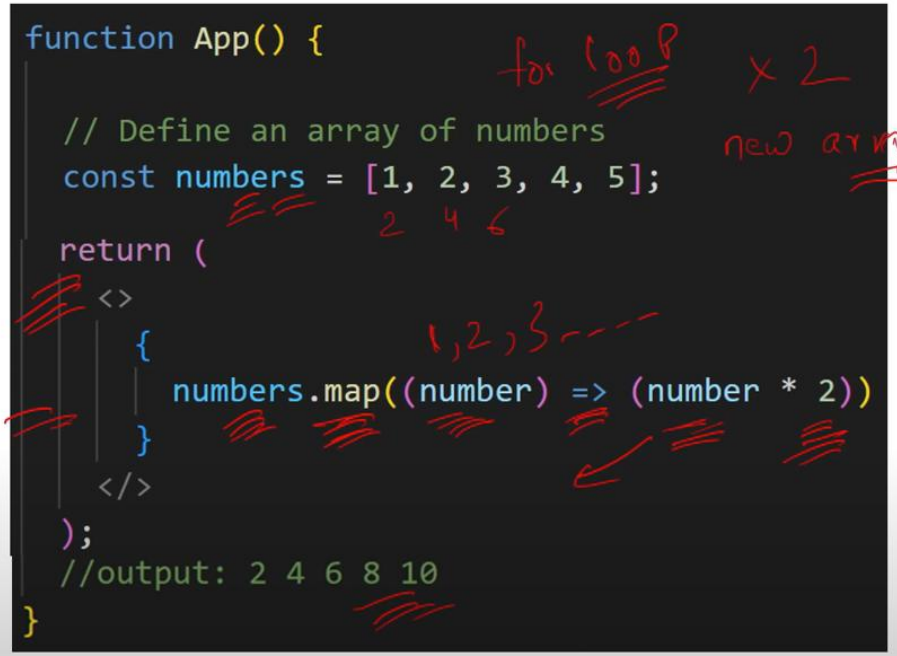
```jsx
const ChildComponent = ({ fromChild }) => {
  // No local state in the child component
  const dataToParent = () => {
    // Directly pass the input value to the parent
    fromChild(document.getElementById
      ("inputField").value);
  };

  return (
    <div>
      <input type="text" id="inputField" />
      <button onClick={dataToParent}>Send</button>
    </div>
  );
};
export default ChildComponent;
```

Screenshot (551).png

React js - Top 30 Interview Questions and Answers

Q. How do you iterate over a list in JSX? What is map() method?

V. IMP.



```jsx
function App() {

  // Define an array of numbers
  const numbers = [1, 2, 3, 4, 5];

  return (
    <>
      {
        numbers.map((number) => (number * 2))
      }
    </>
  );
  //output: 2 4 6 8 10
}
```


45:26 / 50:42 • Q27. How do you iterate over a list in JSX? What I is the use of map method? >

84%