Slide Note

Learning Consolidation

Implement Interactions Between Angular Components

Slide Note

In this Sprint you learned to...

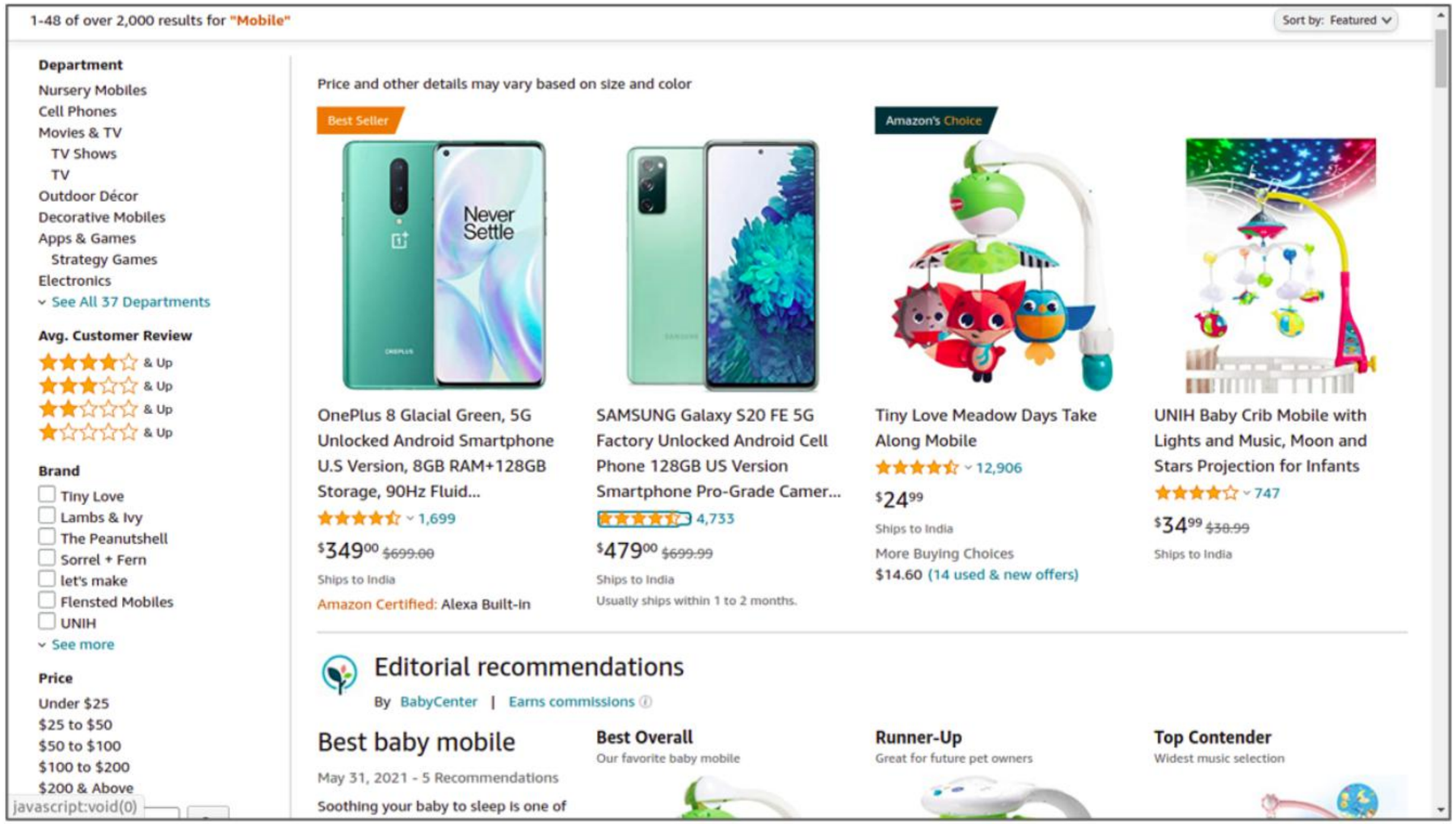
- Explain design principles for identifying components in an SPA
- Build component hierarchy
- Explain the relationship among components in a component hierarchy
- Explain different roles of the component in a component hierarchy
- Use the `@Input()` decorator to share data from the parent component to the child component
- Make use of the `@Output()` decorator to share data from child component to parent component

Slide Note

Here is another web page to identify components.

Learners may give various answers. Discuss with them after moving to the next slide.

Product Catalog View



Slide Note

Identified components:

1-App

2-Side Navbar

3-Product Grid

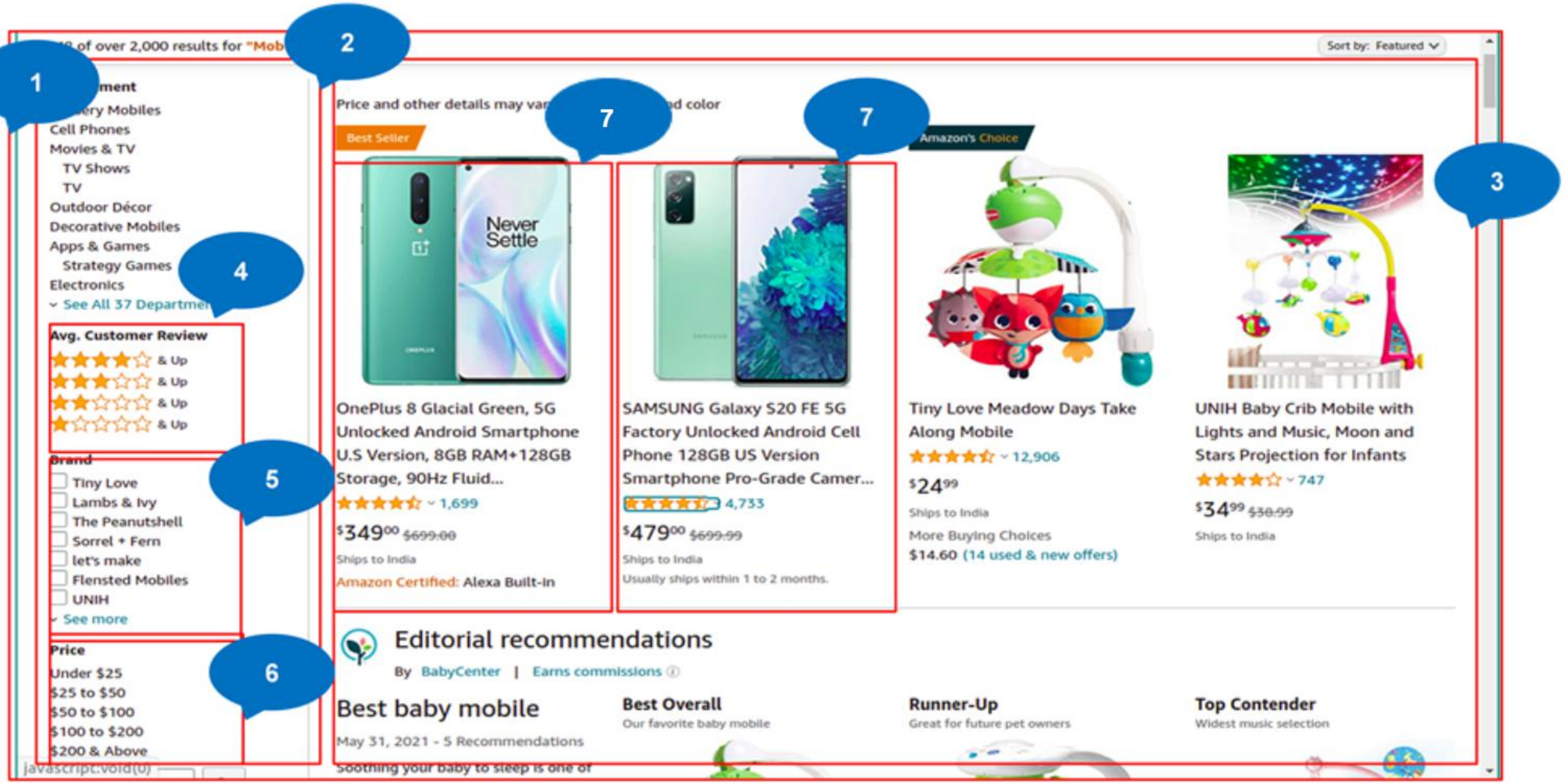
4-Rating Filter

5-Brand Filter

6-Price Filter

7-Product Card

Product Catalog View – Outlining Components



Menu

00:04:00:00:17 04/ 17

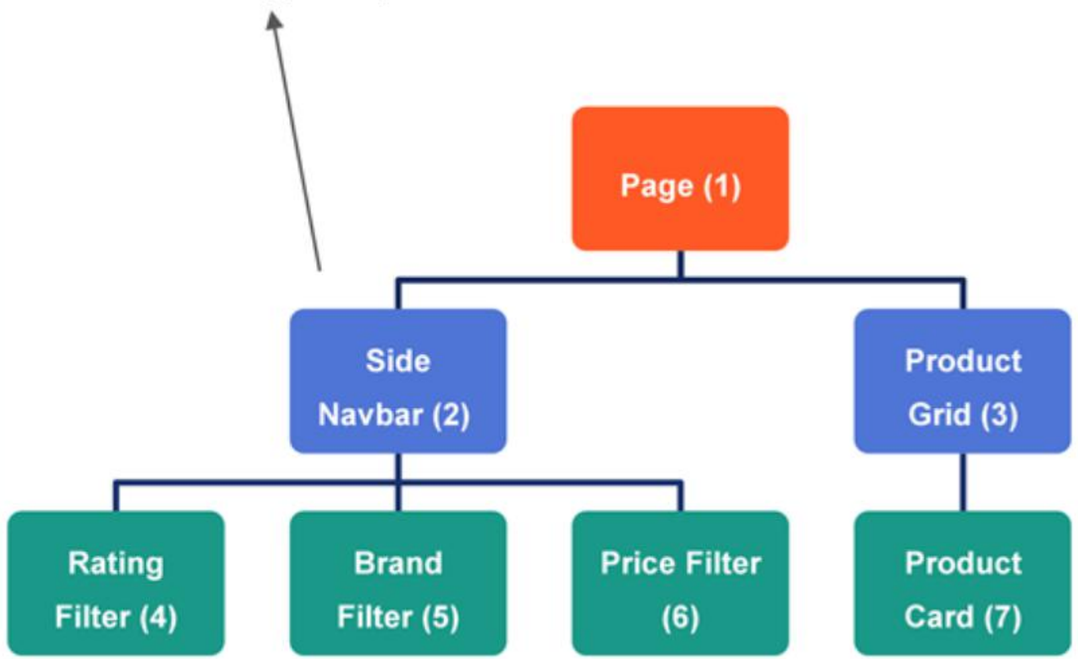
Slide Note

The image depicts the tree of components identified with hierarchy.

In a component hierarchy of an SPA, a component can be a child of another component, and a component can be a parent of another component.

Suppose you have a component called P. In the template of the P component, you write the selector of another component, say C. In this case, component C is the child of the P component, or P is the parent of C.

- Parent to Rating, Brand Filter, and Price Filter components
- Child of App component



Product Catalog View – Component Hierarchy

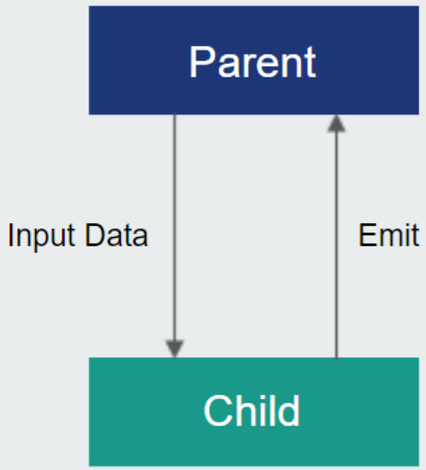
- After you know your app's components, define some relationships between them.
- Start from the root component (App). It will be the root node of the tree.
- Find its direct child components (Side Navbar View and Product Grid).
- Make a tree and represent the app as the root node and Side Navbar View and Product Grid as children.
- Now, repeat the same process with other component nodes.

Slide Note

Role of Angular Components in SPA

- Components can be essentially classified into two types:
  - Smart or Container components
    - This component handles storing data (the data could be defined within the component or fetched from an external source).
    - The component is usually a parent component that passes this data to the child component for presentation.
    - Since it's bound to data, it would be very hard to reuse this component in another application.
  - Dumb or Presentation components
    - The component's responsibility is to present the data received from the parent component to the end user and not to fetch it from a particular location.
    - This component is highly reusable since it's not tightly bound with data but provides presentation logic to serve the data that it receives.

Slide Note



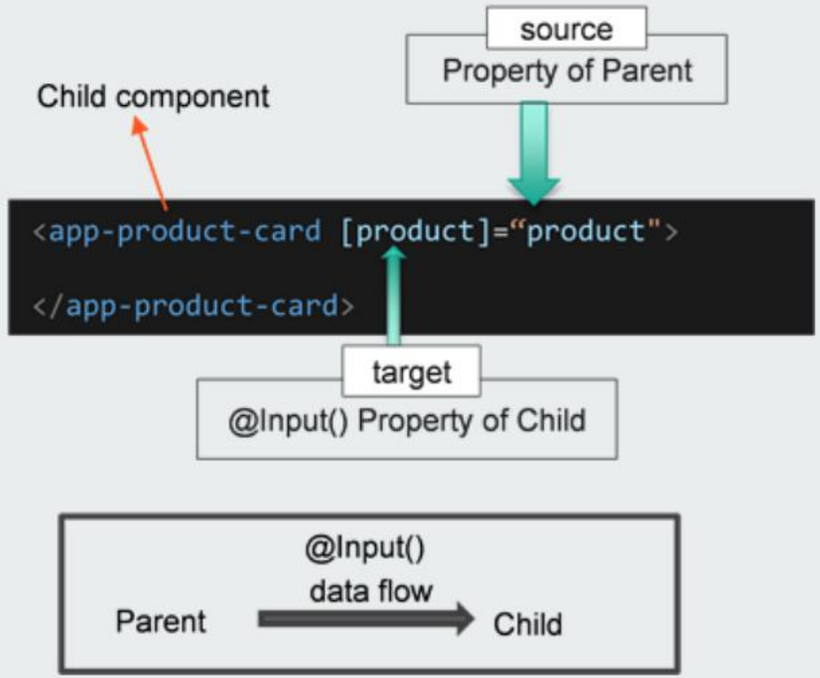
Component Interaction

- In a component hierarchy, parent and child components interact to share data.
- The data flow is bidirectional between Angular components.
- The parent component renders the child component and, while rendering, may pass data to the child component.
  - This data is input for the child component.
- When an action occurs in the child component, the child component may want to notify the parent about this action.
  - The child component notifies the parent component by emitting an event.

Slide Note

```
@Component({
  selector: 'app-product-card',
  templateUrl: './product-card.component.html',
  styleUrls: ['./product-card.component.css'],
})
export class ProductCardComponent {
  @Input() product?: Product;
}
```

Input property



@Input() Decorator

- The child component accepts the value for its property from the parent component by annotating the property with the `@Input()` decorator.
- In the snippet shown on the slide, the `app-product-card` is the child component selector used by the parent component to render the `Product-Card` component.
- While rendering, the parent component passes the product's value to the child component, which is received by the product property of the child component.
- The `Product-Card` component declares the `product` property with the `@Input()` decorator to inform Angular that this property will be inputted with the value provided by the parent component.

Slide Note

@Output() Decorator

- The child component emits an event to notify the parent component of the changes and shares the data with the event emitted.
- The event is an `EventEmitter` object declared by the child component by annotating the object with the `@Output()` decorator.
- In the snippet shown on the slide, the `app-search` is the child component selector used by the parent component to render the `Search` component.
- The `Search` component class emits `searchTextChanged` event with the value of the `searchText` variable.
- While rendering the `Search` component, the parent component associates the event with event handler `onSearchTextChanged()` and receives the data passed via the `$event` argument.

Child component

Event emitted by Child

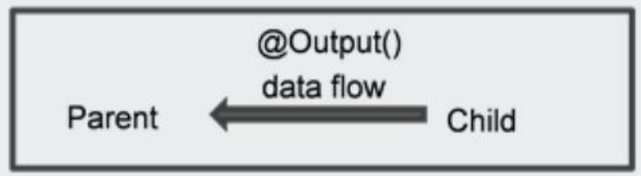
Event handler in Parent

Data received by Parent

```
<app-search
(searchTextChanged)="onSearchTextChanged($event)">
</app-search>
```

```
@Component({
  selector: 'app-search',
  templateUrl: './search.component.html',
  styleUrls: ['./search.component.css'],
})
export class SearchComponent implements OnInit {
  @Output()
  searchTextChanged: EventEmitter<string> =
            new EventEmitter<string>();
  searchText: string = '';
   search() {
    this.searchTextChanged.emit(this.searchText);
  }
}
```

Data passed to Parent



Slide Note

Self-Check

What is the functionality of the `@Output()` decorator?

1. Used to annotate the child component property to share data with the parent component.
2. Used to annotate the parent component property to share data with the child component.
3. Used to annotate the child component to share data with the parent component.
4. Used to annotate the parent component to share data with the child component.

Slide Note

Self-Check: Solution

What is the functionality of the `@Output()` decorator?

1. **Used to annotate the child component property to share data with the parent component.**
2. Used to annotate the parent component property to share data with the child component.
3. Used to annotate the child component to share data with the parent component.
4. Used to annotate the parent component to share data with the child component.

Slide Note

Self-Check

What is the functionality of the `@Input()` decorator?

1. Used to annotate the child component property to receive input from the parent component.
2. Used to annotate the parent component property to receive input from the child component.
3. Used to annotate the child component to receive input from the parent component.
4. Used to annotate the parent component to receive input from the child component.

Slide Note

Self-Check: Solution

What is the functionality of the `@Input()` decorator?

1. **Used to annotate the child component property to receive input from the parent component.**
2. Used to annotate the parent component property to receive input from the child component.
3. Used to annotate the child component to receive input from the parent component.
4. Used to annotate the parent component to receive input from the child component.

Slide Note

Self-Check

The below code is written in the template code of the **`ProductGrid`** component. Answer the questions given below to comprehend the code.

```
<app-product (selectionChanged)="onSelectionChanged($event)"
         *ngFor="let _product of products" [product]="_product">

</app-product>
```

1. Which component is rendered by the ProductGrid component?
2. In the component hierarchy, between ProductGrid and Product component, who is the parent component and who is the child component?
3. Which data does the parent component share with the child component?
4. How is the child component receiving the input from its parent component?
5. In the code snippet, what is `selectionChanged`?
6. In the code snippet, what is `onSelectionChanged`?

Slide Note

Self-Check: Solution

The below code is written in the template code of the **ProductGrid** component. Answer the questions given below to comprehend the code.

```
<app-product (selectionChanged)="onSelectionChanged($event)"
          *ngFor="let _product of products" [product]="_product">
</app-product>
```

1. Which component is rendered by the ProductGrid component?

Answer: Product

2. In the component hierarchy, between ProductGrid and Product component, who is the parent component and the child component?

Answer: Parent Component: ProductGrid, Child Component: Product

3. Which data does the parent component share with the child component?

Answer: The value of _product is shared with the child component's product property using property binding.

Slide Note

Self-Check: Solution (Cont'd.)

The below code is written in the template code of **ProductGrid** component. Answer the questions given below to comprehend the code.

```
<app-product (selectionChanged)="onSelectionChanged($event)"
          *ngFor="let _product of products" [product]="_product">
</app-product>
```

4. How is the child component receiving the input from its parent component?

Answer: The child component (Product) should have product property declared with @Input() decorator

5. In the code snippet, what is `selectionChanged`?

Answer: It should be the child (`Product`) component's property declared with `@Output()` emitter and of type `EventEmitter`.

6. In the code snippet, what is `onSelectionChanged`?

Answer: It is the method of `ProductGrid` component that gets invoked when `selectionChanged` event is emitted by `Product` component and receives data passed by child to parent component via $event argument.