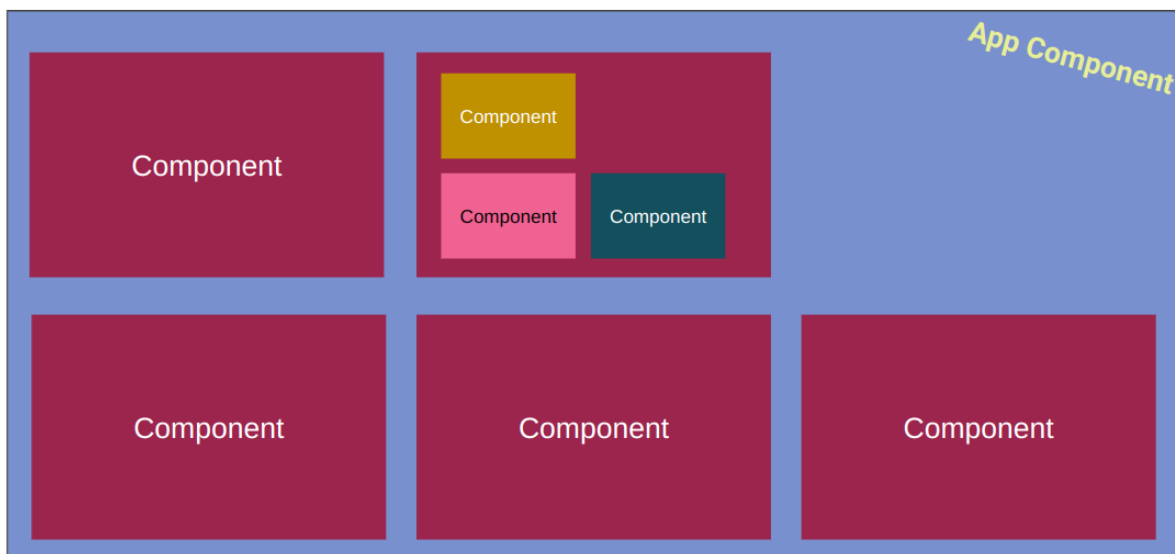


Angular Components Handouts

Angular Components

- Components are the visual building blocks of an Angular application.
- You can compose different components together to create a page.
- APP Component is the root component of our application. It's called the bootstrap component. In an angular application, the other components usually load into the app component.



- A component should be included in the `declarations` array of a `@Module`.

```
@NgModule({  
  declarations: [  
    AppComponent,  
  ],  
})
```

- To create a new component using the CLI run: `ng g c <component name>`.
- Like Directives, Components can have one or more selectors that specify how to load the component in the HTML Dom.

Use the following table as a cheat sheet for how to specify different types of selectors.

Selector	Definition	Usage
Tag selector	'tag'	<tag>
Class selector	'class-name'	<whatever class='class-name'> <whatever>
Attribute Selector	[attribute-name]	<whatever attribute-name> <whatever>
Attribute Value Selector	[attribute-value=false]	<whatever attribute-name="false"> <whatever>
Not Selector	[omit-me]:not([omit-me=true])	<whatever omit-me="false"> <whatever>

```
@Component({
  selector: 'task-table, todo-table, .task-table-class, [task-table-attr], [is-task-table=true]:not([omit-me=true])',
  templateUrl: './task-table.component.html',
  styleUrls: ['./task-table.component.css']
})
export class TaskTableComponent {
  ...
}
```

- Bind a field value to a component's template like this:

```
@Component({...})
export class CalendarComponent {
  selectedDate = new Date();
}

<mat-calendar
  [(selected)]="selectedDate">
</mat-calendar>

<a [routerLink]="['/tasks',
  {date:selectedDate }]">
  Tasks
</a>
```

Input / Output Binding

To pass a value from a parent component to a child component decorate the field with the `@Input` decorator.

```
<task-table  
  [tasks]="..."
```

```
  (onRemove)="f()"  
></task-table>
```

```
@Component({  
  selector: 'task-table',  
  ...})  
export class TaskTableComponent {
```

```
  @Input() tasks: TaskItem[]
```

```
  @Output() onRemove: EventEmitter
```

```
    onRemove.next()
```

To pass a value from a child component to the parent component create a field of type `EventEmitter` and decorate it with the `@Output` decorator.