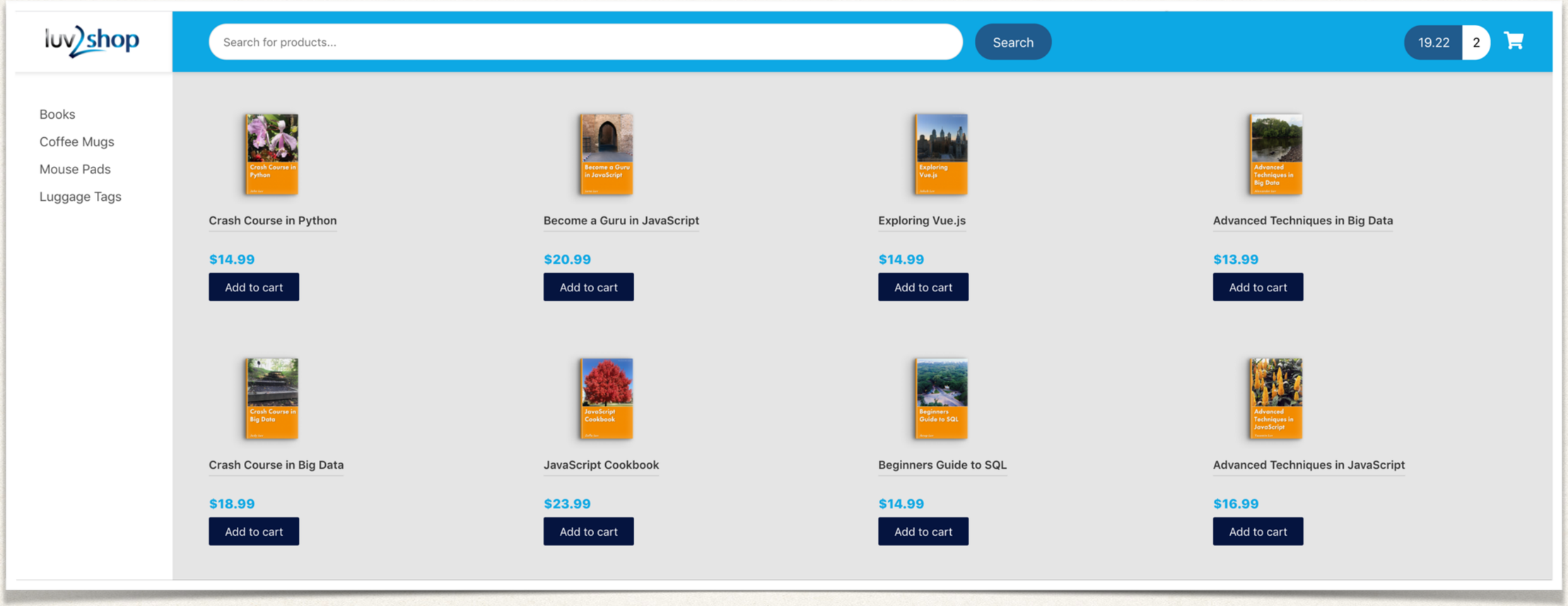
Search By Keyword

Search for Products by Keyword

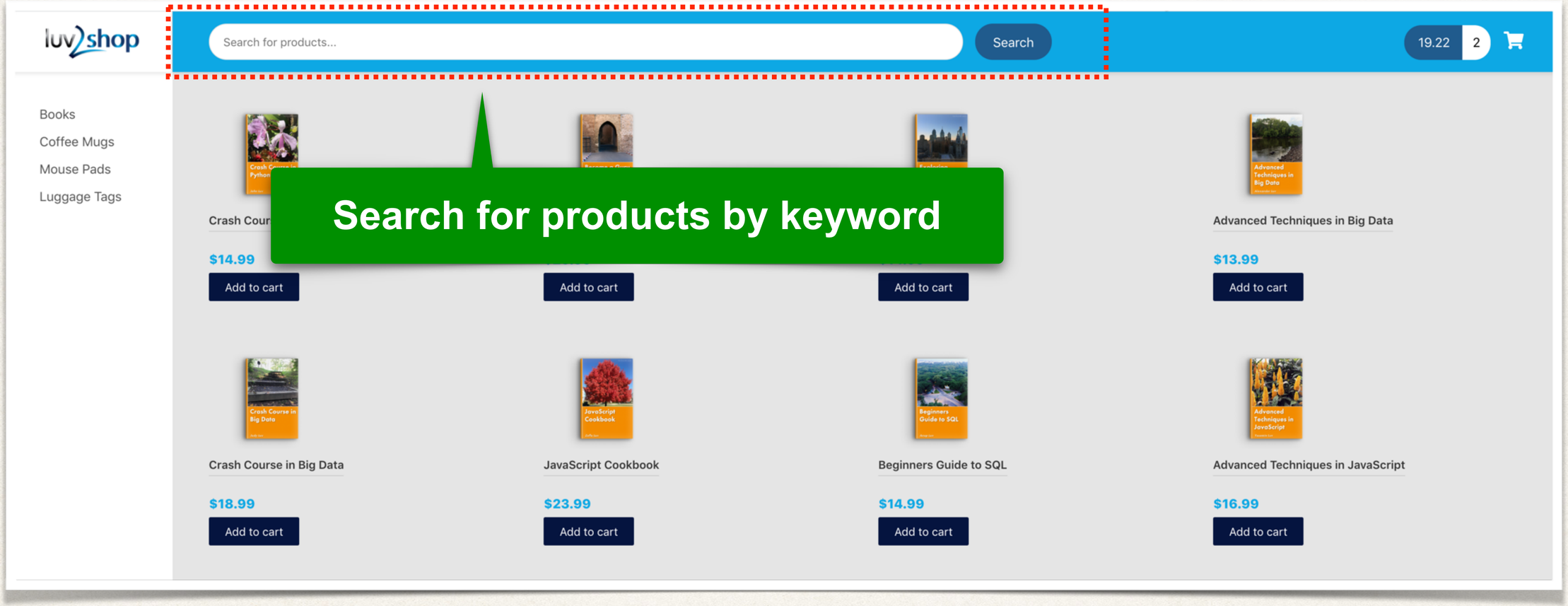
© luv2code LLC

Search for Products by Keyword



© luv2code LLC

Search for Products by Keyword



© luv2code LLC

Development Process

Step-By-Step

© luv2code LLC

Development Process

Step-By-Step

1. Modify Spring Boot app - Add a new search method

© luv2code LLC

Development Process

Step-By-Step

1. Modify Spring Boot app - Add a new search method
2. Create new component for search

© luv2code LLC

Development Process

Step-By-Step

1. Modify Spring Boot app - Add a new search method
2. Create new component for search
3. Add new Angular route for searching

© luv2code LLC

Development Process

Step-By-Step

1. Modify Spring Boot app - Add a new search method
2. Create new component for search
3. Add new Angular route for searching
4. Update SearchComponent to send data to search route

© luv2code LLC

Development Process

Step-By-Step

1. Modify Spring Boot app - Add a new search method
2. Create new component for search
3. Add new Angular route for searching
4. Update SearchComponent to send data to search route
5. Enhance ProductListComponent to search for products with ProductService

© luv2code LLC

Development Process

Step-By-Step

1. Modify Spring Boot app - Add a new search method
2. Create new component for search
3. Add new Angular route for searching
4. Update SearchComponent to send data to search route
5. Enhance ProductListComponent to search for products with ProductService
6. Update ProductService to call URL on Spring Boot app

© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

- Spring Data REST and Spring Data JPA supports "query methods"

© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

- Spring Data REST and Spring Data JPA supports "query methods"
- Spring will construct a query based on method naming conventions

© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

- Spring Data REST and Spring Data JPA supports "query methods"
- Spring will construct a query based on method naming conventions

Magic!

© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

- Spring Data REST and Spring Data JPA supports "query methods"
- Spring will construct a query based on method naming conventions

Magic!

File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

    ...

}
```


© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

- Find products based on name

© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

- Find products based on name

File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

    ...

}
```


© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

- Find products based on name

File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

    ...

}
```

"Containing" ... similar to SQL: "LIKE"

© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

    ...

}
```


© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);
```

Behind the scenes, Spring will execute a query similar to this

```
SELECT * FROM Product p
WHERE
p.name LIKE CONCAT('%', :name ,'%')
```


© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);
```

Behind the scenes, Spring will execute a query similar to this

```
SELECT * FROM Product p
WHERE
p.name LIKE CONCAT('%', :name ,'%')
```


© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

    ...

}
```


© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

    ...

}
```

http://localhost:8080/api/products/search/findByNameContaining?name=Python

© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

File: ProductRepository.java

```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

    ...

}
```

http://localhost:8080/api/products/search/findByNameContaining?name=Python

To pass data to REST API

© luv2code LLC

Step 1: Modify Spring Boot app - Search Method

File: ProductRepository.java

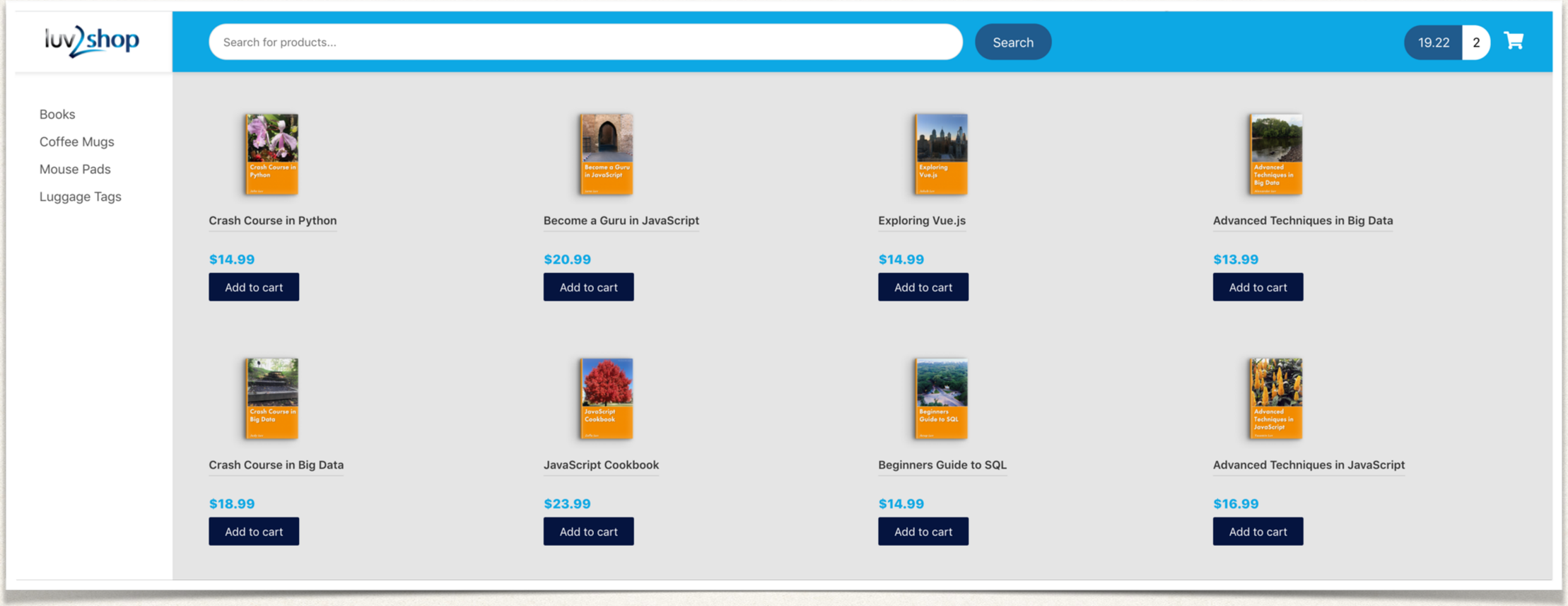
```
public interface ProductRepository extends JpaRepository<Product, Long> {

    Page<Product> findByNameContaining(@RequestParam("name") String name, Pageable pageable);

    ...

}
```

http://localhost:8080/api/products/search/findByNameContaining?name=Python

To pass data to REST API

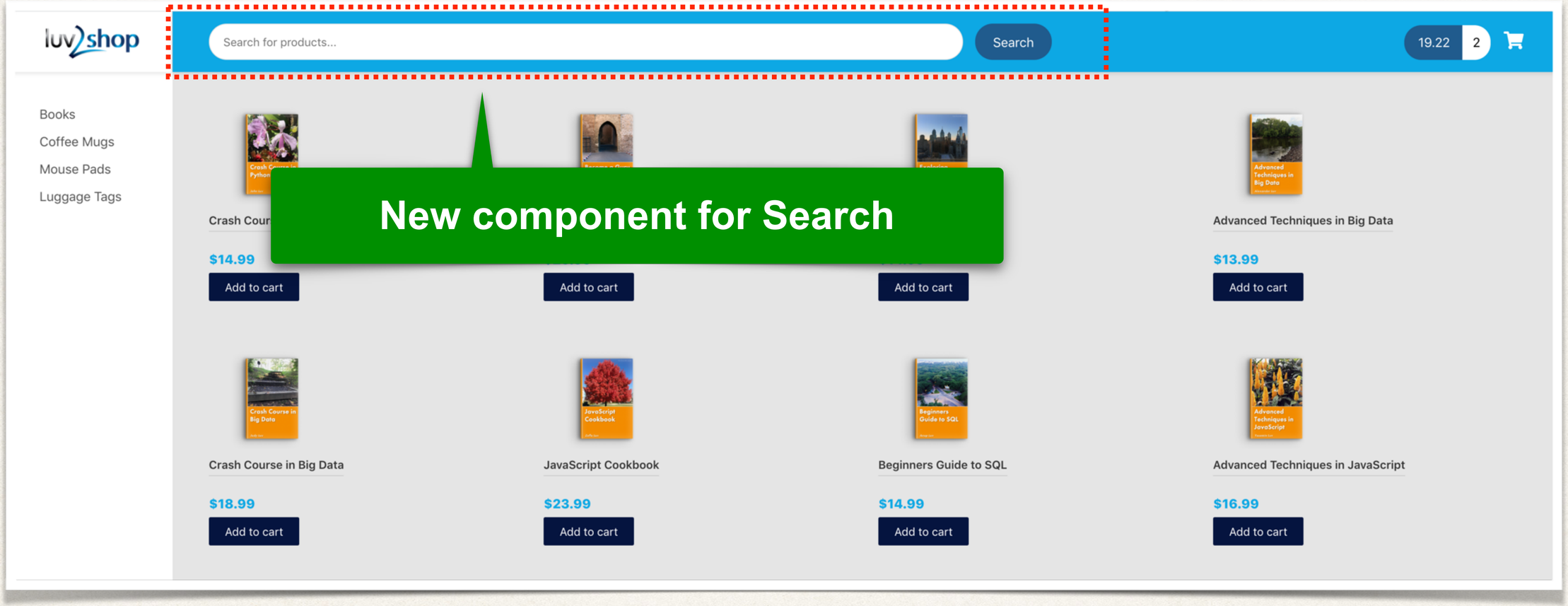
© luv2code LLC

Step 2: Create new component for Search



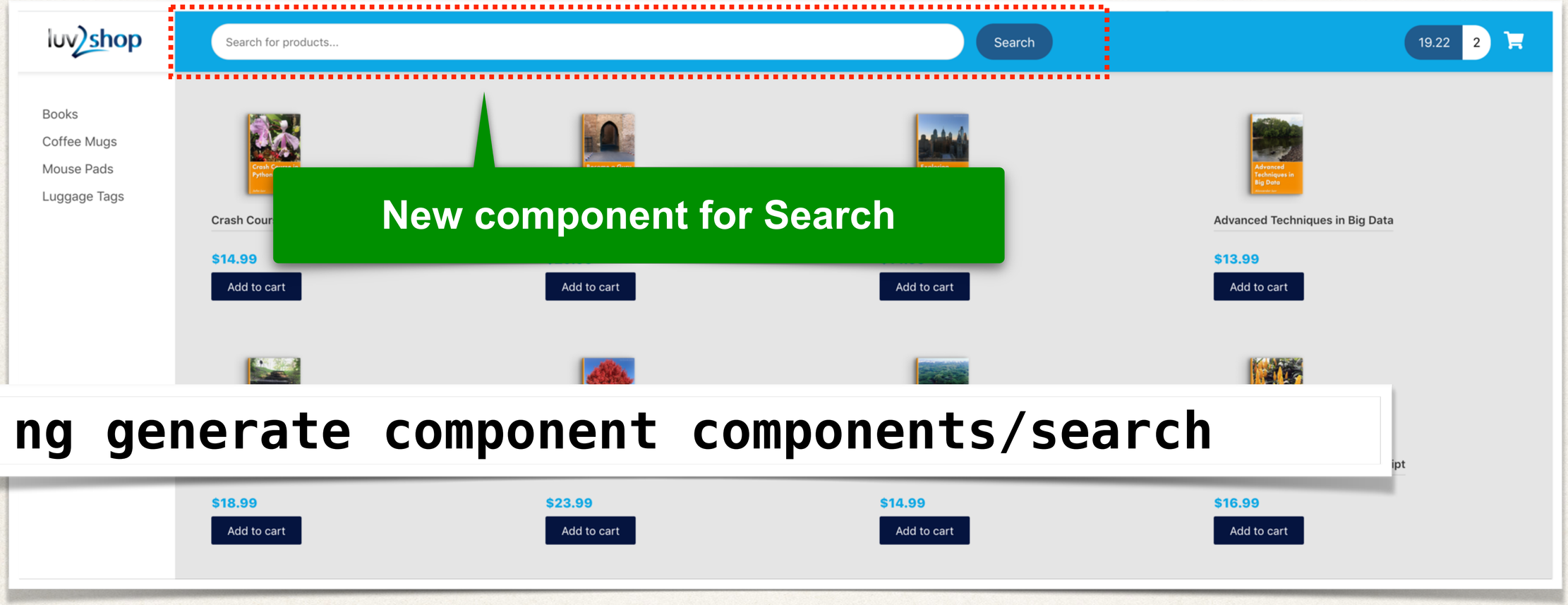
© luv2code LLC

Step 2: Create new component for Search



© luv2code LLC

Step 2: Create new component for Search




```
> ng generate component components/search
```


© luv2code LLC

Step 3: Add new Angular route for searching

File: app.module.ts

```
...
  {path: 'search/:keyword', component: ProductListComponent},
...
```


© luv2code LLC

Step 3: Add new Angular route for searching

File: app.module.ts

```
...
  {path: 'search/:keyword', component: ProductListComponent},
...
```

Parameter

© luv2code LLC

© luv2code LLC

Quick Discussion

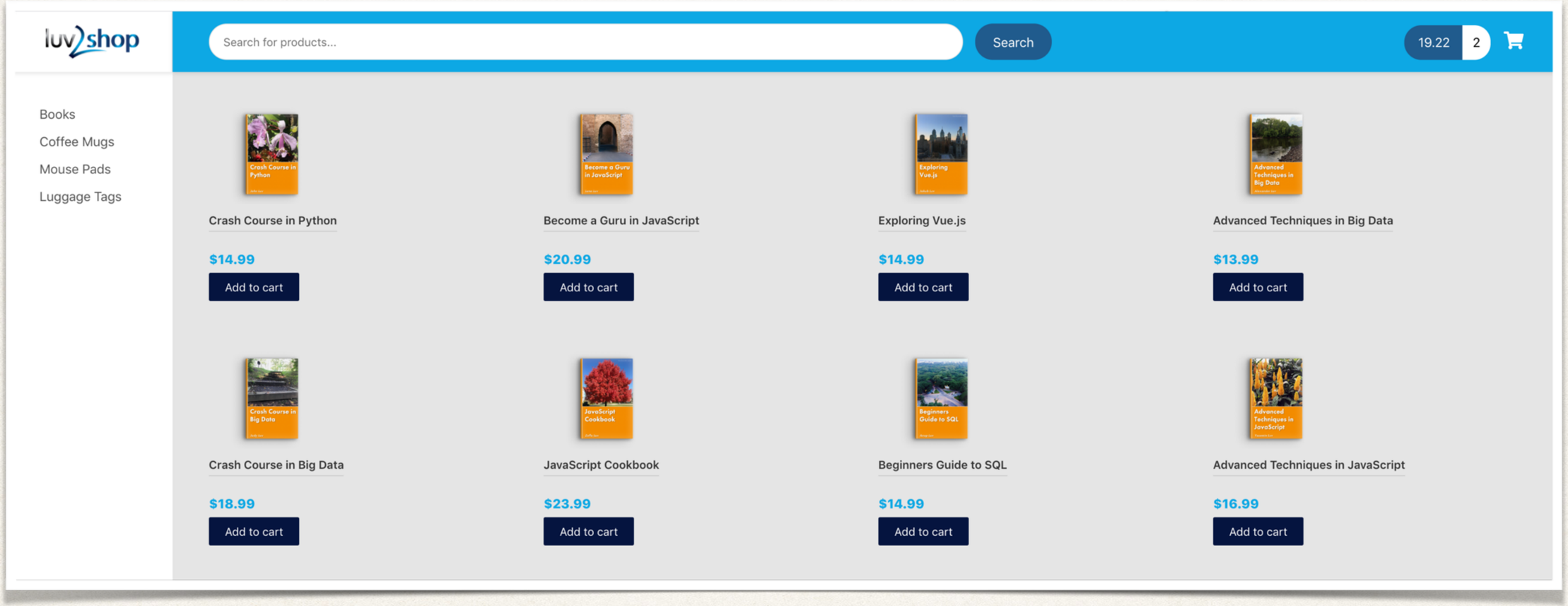
Event Binding

© luv2code LLC

Quick Discussion - Event Binding

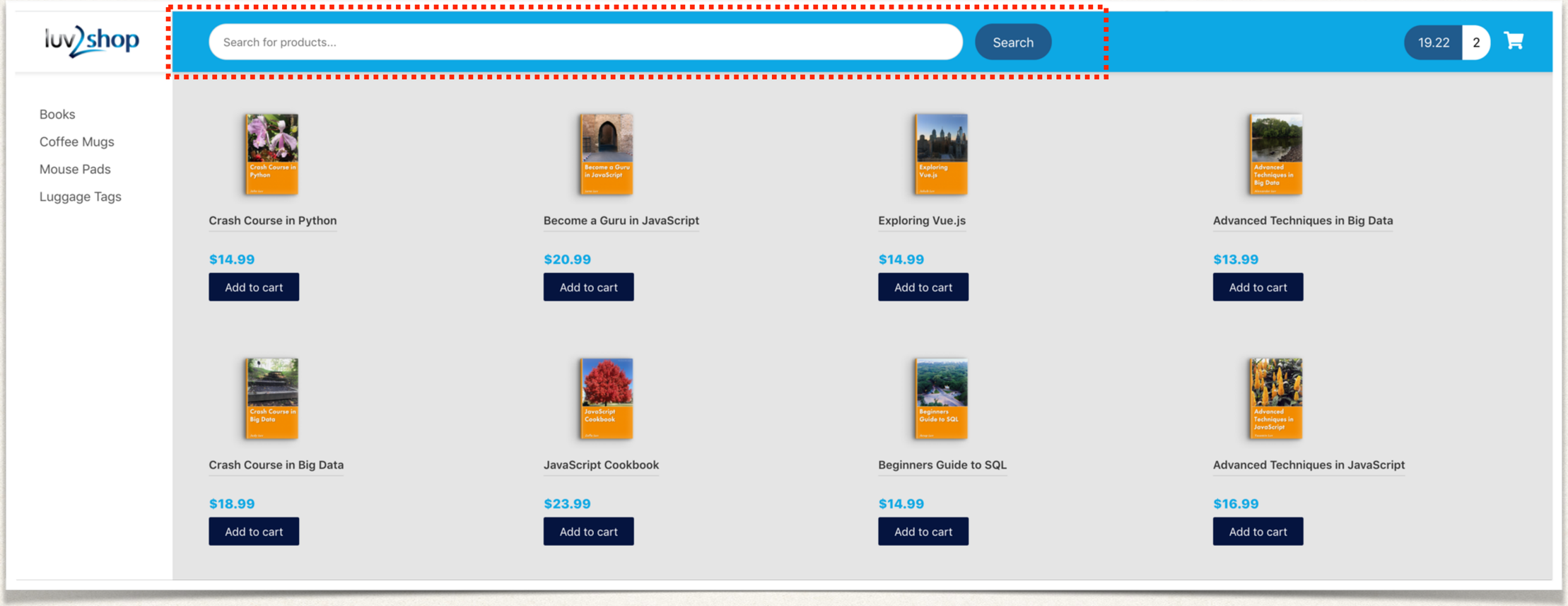
© luv2code LLC

Quick Discussion - Event Binding



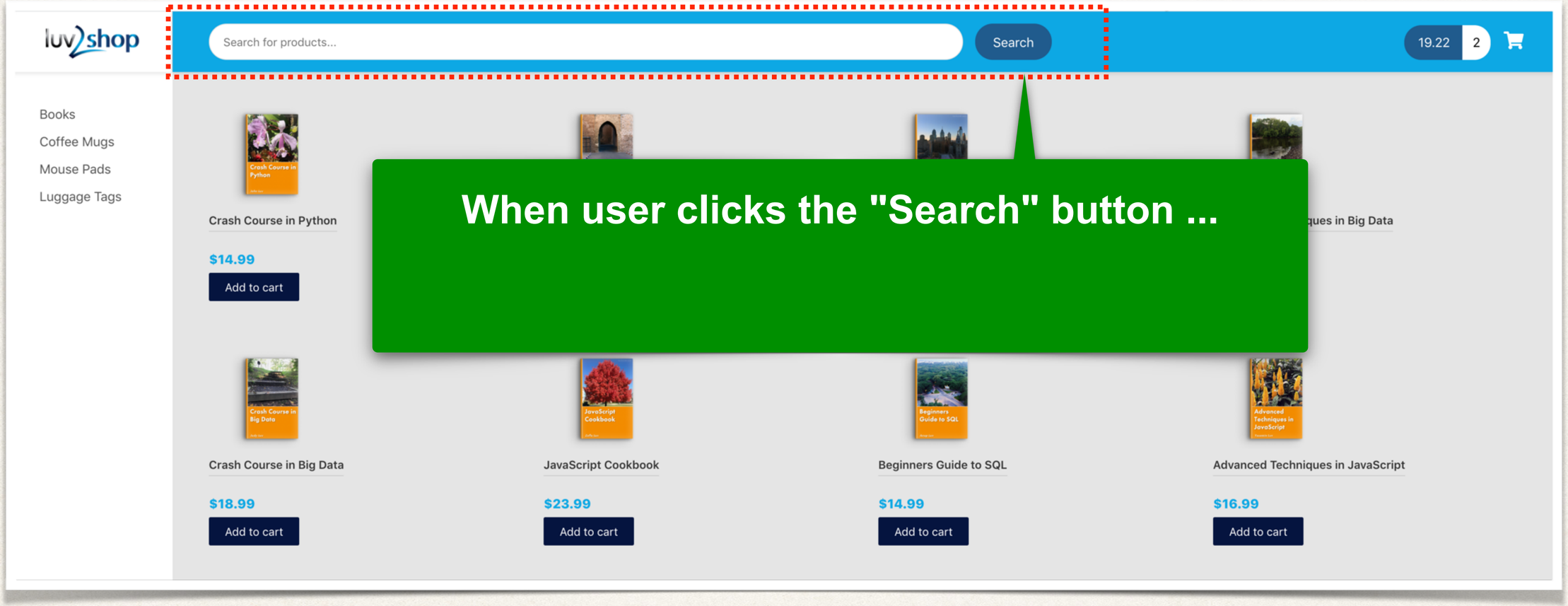
© luv2code LLC

Quick Discussion - Event Binding



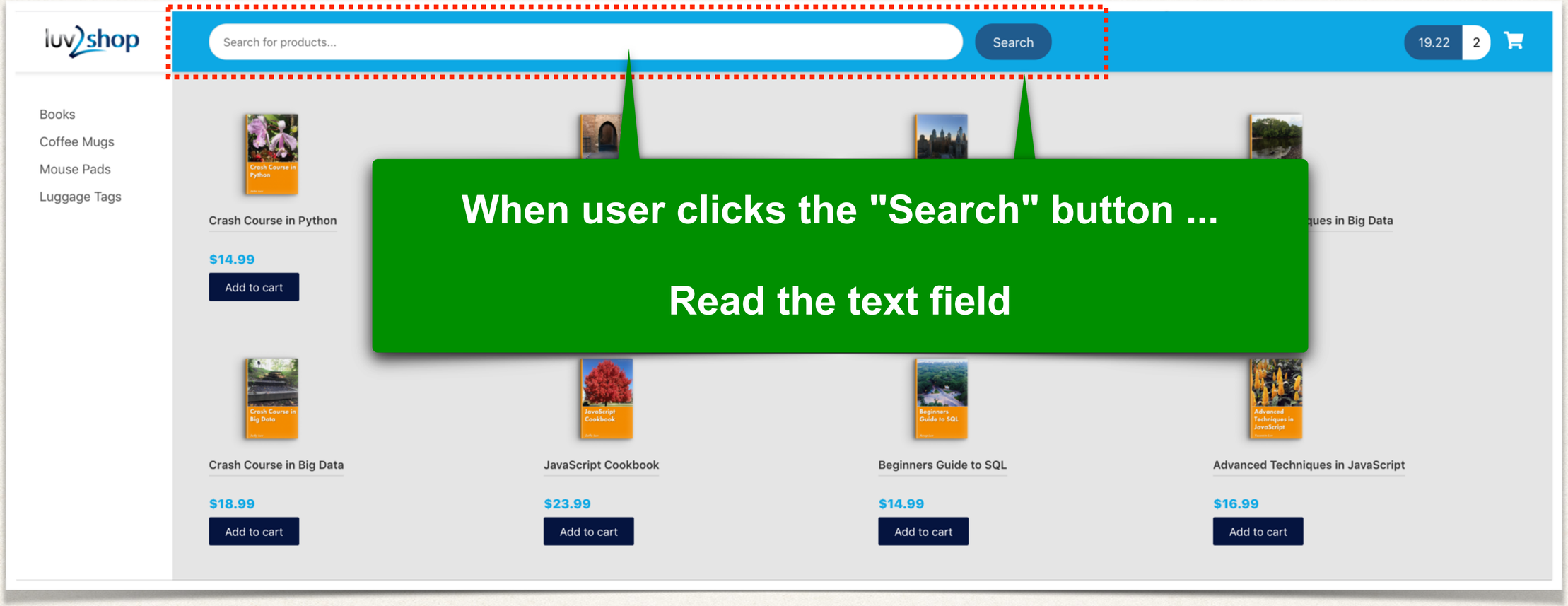
© luv2code LLC

Quick Discussion - Event Binding



© luv2code LLC

Quick Discussion - Event Binding



© luv2code LLC

Event Binding

© luv2code LLC

Event Binding

- In Angular, you can listen for events with "event binding"

© luv2code LLC

Event Binding

- In Angular, you can listen for events with "event binding"
- In other languages/frameworks, also known as "event handling"

© luv2code LLC

Event Binding

- In Angular, you can listen for events with "event binding"
- In other languages/frameworks, also known as "event handling"

```
<button (click)="doMyCustomWork()">Search</button>
```


© luv2code LLC

Event Binding

- In Angular, you can listen for events with "event binding"
- In other languages/frameworks, also known as "event handling"

```
<button (click)="doMyCustomWork()">Search</button>
```

Listen for "click" event

© luv2code LLC

Event Binding

- In Angular, you can listen for events with "event binding"
- In other languages / frameworks, also know "

Event handler

```
<button (click)="doMyCustomWork()">Search</button>
```

Listen for "click" event

© luv2code LLC

Event Binding

- In Angular, you can listen for events with "event binding"
- In other languages/frameworks, also know "

Event handler

```
<button (click)="doMyCustomWork()">Search</button>
```

Listen for "click" event

Call a method in our Angular component code

Can be any method name we define

© luv2code LLC

Event Binding - Example

© luv2code LLC

Event Binding - Example

```
<button (click)="doMyCustomWork()">Search</button>
```


© luv2code LLC

Event Binding - Example

```
<button (click)="doMyCustomWork()">Search</button>
```

Listen for "click" event

© luv2code LLC

Event Binding - Example

Call a method in our Angular component code

```
<button (click)="doMyCustomWork()">Search</button>
```

Listen for "click" event

© luv2code LLC

Event Binding - Example

Call a method in our Angular component code

```
<button (click)="doMyCustomWork()">Search</button>
```

Listen for "click" event

```
export class SearchComponent implements OnInit {

  doMyCustomWork() {
    console.log(`Hey! You pushed my button!`);
 }

  ...
}
```


© luv2code LLC

Reading User Input

© luv2code LLC

Reading User Input

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />
```


© luv2code LLC

Reading User Input

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />
```

Listen for the "enter" key

© luv2code LLC

Reading User Input

 # symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />
```

Listen for the "enter" key

© luv2code LLC

Reading User Input

 # symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />
```

Listen for the "enter" key

```
export class SearchComponent implements OnInit {

  doMyCustomWork(info: string) {
    console.log(`Hey! This is your data: ${info}`);
 }
 ...
}
```


© luv2code LLC

Reading User Input

 # symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />
```

Listen for the "enter" key

```
export class SearchComponent implements OnInit {

  doMyCustomWork(info: string) {
    console.log(`Hey! This is your data: ${info}`);
 }
 ...
}
```


© luv2code LLC

Reading User Input

symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />
```

The text the user typed in

Listen for the "enter" key

```
export class SearchComponent implements OnInit {

  doMyCustomWork(info: string) {
    console.log(`Hey! This is your data: ${info}`);
 }
 ...
}
```


© luv2code LLC

Reading User Input

symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />
```

The text the user typed in

Listen for the "enter" key

```
export class SearchComponent implements OnInit {

  doMyCustomWork(info: string) {
    console.log(`Hey! This is your data: ${info}`);
 }
 ...
}
```


© luv2code LLC

Reading User Input

 # symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />
```

```
export class SearchComponent implements OnInit {

  doMyCustomWork(info: string) {
    console.log(`Hey! This is your data: ${info}`);
 }
 ...
}
```


© luv2code LLC

Reading User Input

 # symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />

<button (click)="doMyCustomWork(myInput.value)">Search</button>
```

```
export class SearchComponent implements OnInit {

  doMyCustomWork(info: string) {
    console.log(`Hey! This is your data: ${info}`);
 }
 ...
}
```


© luv2code LLC

Reading User Input

 # symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />

<button (click)="doMyCustomWork(myInput.value)">Search</button>
```

Listen for "click" event

```
export class SearchComponent implements OnInit {

  doMyCustomWork(info: string) {
    console.log(`Hey! This is your data: ${info}`);
 }
 ...
}
```


© luv2code LLC

Reading User Input

 # symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />

<button (click)="doMyCustomWork(myInput.value)">Search</button>
```

Listen for "click" event

```
export class SearchComponent implements OnInit {

  doMyCustomWork(info: string) {
    console.log(`Hey! This is your data: ${info}`);
 }
 ...
}
```


© luv2code LLC

Reading User Input

symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />

<button (click)="doMyCustomWork(myInput.value)">Search</button>
```

The text the user typed in

Listen for "click" event

```
export class SearchComponent implements OnInit {

  doMyCustomWork(info: string) {
    console.log(`Hey! This is your data: ${info}`);
 }
 ...
}
```


© luv2code LLC

Reading User Input

symbol
Template reference variable

Provides access to the element

```
<input #myInput type="text"
       (keyup.enter)="doMyCustomWork(myInput.value)"  />

<button (click)="doMyCustomWork(myInput.value)">Search</button>
```

The text the user typed in

Listen for "click" event

```
export class SearchComponent implements OnInit {

  doMyCustomWork(info: string) {
    console.log(`Hey! This is your data: ${info}`);
 }
 ...
}
```


© luv2code LLC

Other Events

© luv2code LLC

Other Events

Name	Description
focus	An element has received focus

© luv2code LLC

Other Events

Name	Description
focus	An element has received focus
blur	An element has lost focus
keyup	Any key is released. For a specific key, the ***enter*** key, use: **keyup.*enter***

© luv2code LLC

Other Events

Name	Description
focus	An element has received focus
blur	An element has lost focus
keyup	Any key is released. For a specific key, the ***enter*** key, use: **keyup.*enter***
keydown	Any key is pressed

© luv2code LLC

Other Events

Name	Description
focus	An element has received focus
blur	An element has lost focus
keyup	Any key is released. For a specific key, the ***enter*** key, use: **keyup.*enter***
keydown	Any key is pressed
dblclick	The mouse is clicked twice on an element

© luv2code LLC

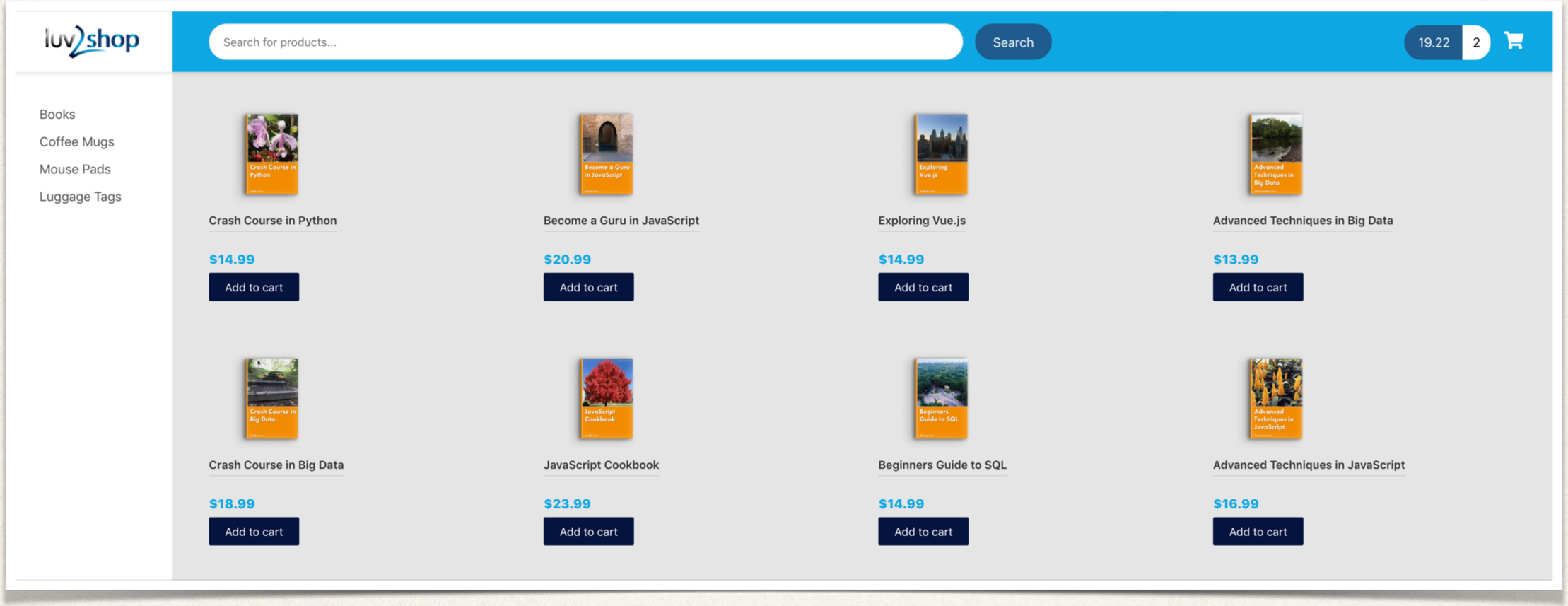
Other Events

Name	Description
focus	An element has received focus
blur	An element has lost focus
keyup	Any key is released. For a specific key, the ***enter*** key, use: **keyup.*enter***
keydown	Any key is pressed
dblclick	The mouse is clicked twice on an element
more ….	

https://developer.mozilla.org/en-US/docs/Web/Events

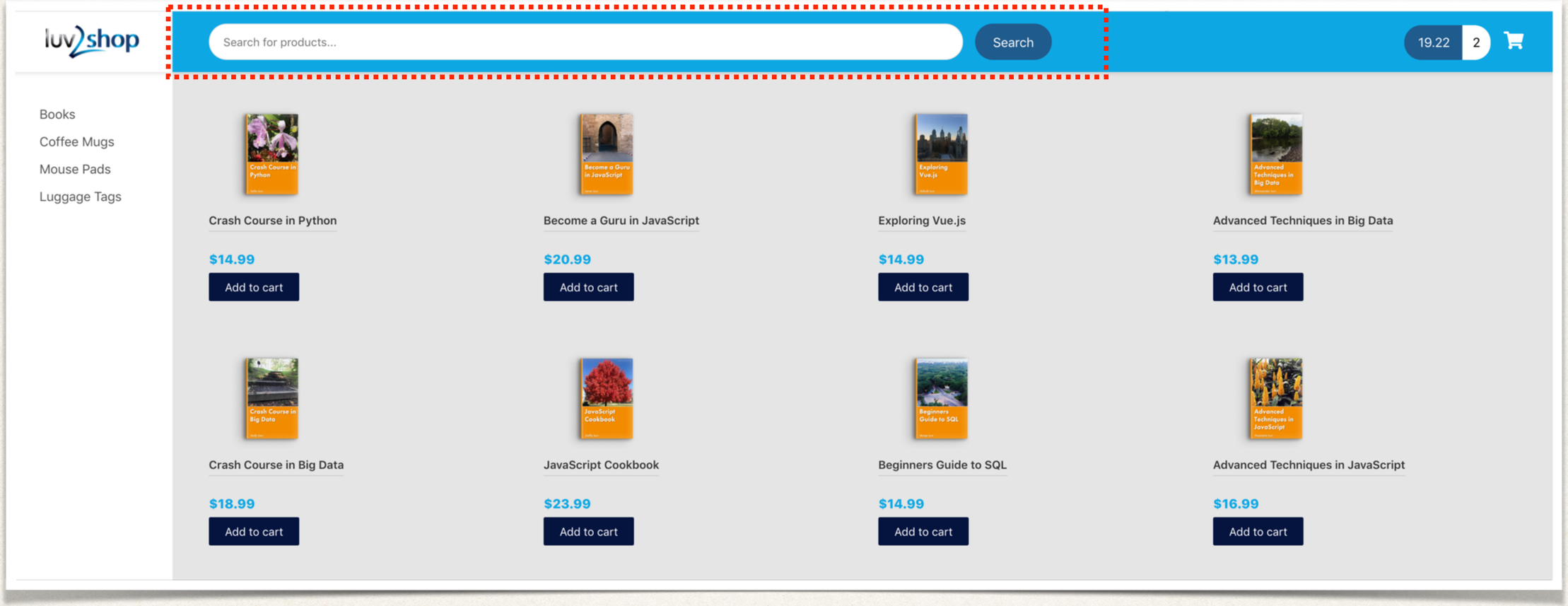
© luv2code LLC

Step 4: Update SearchComponent to send data to search route



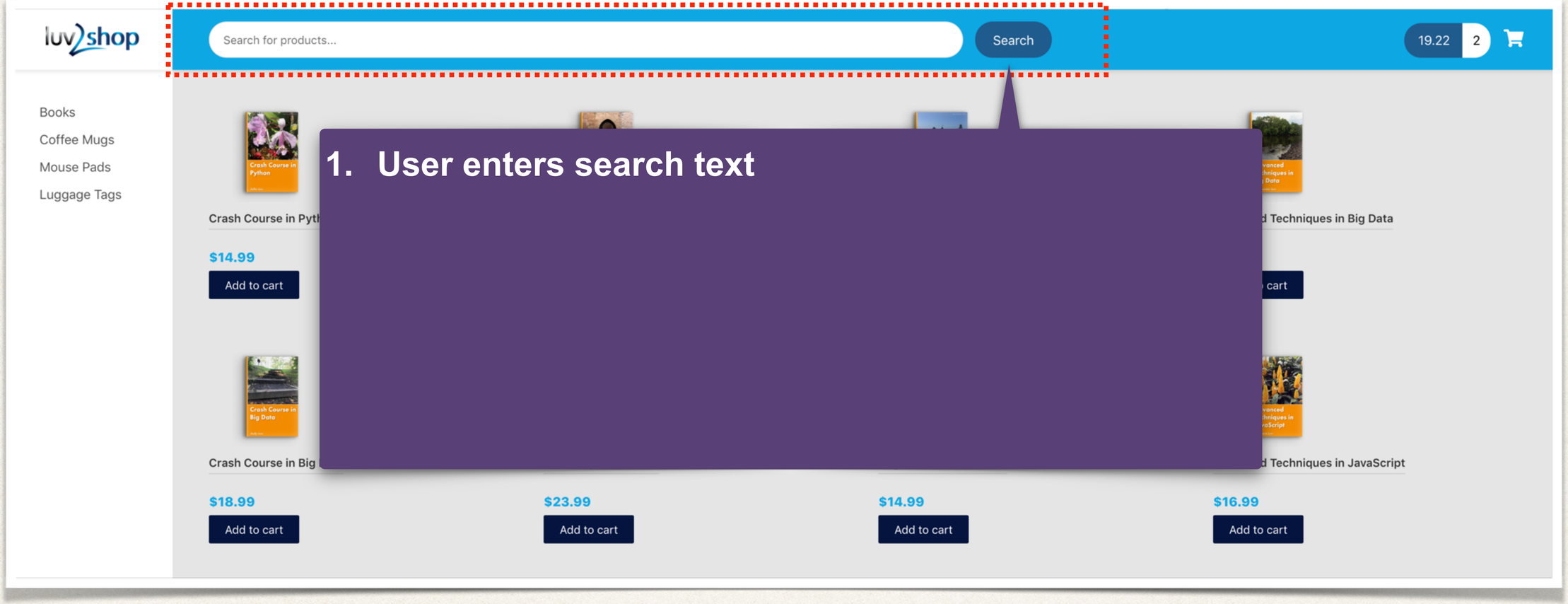
© luv2code LLC

Step 4: Update SearchComponent to send data to search route



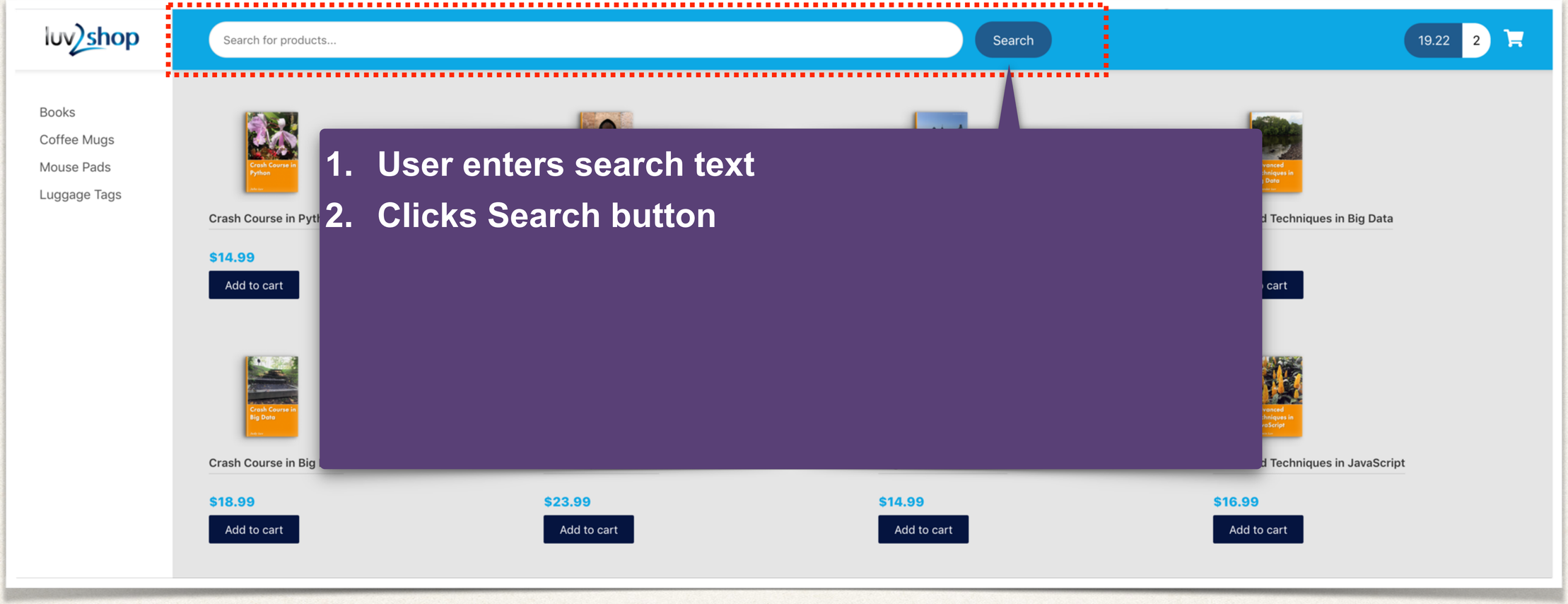
© luv2code LLC

Step 4: Update SearchComponent to send data to search route



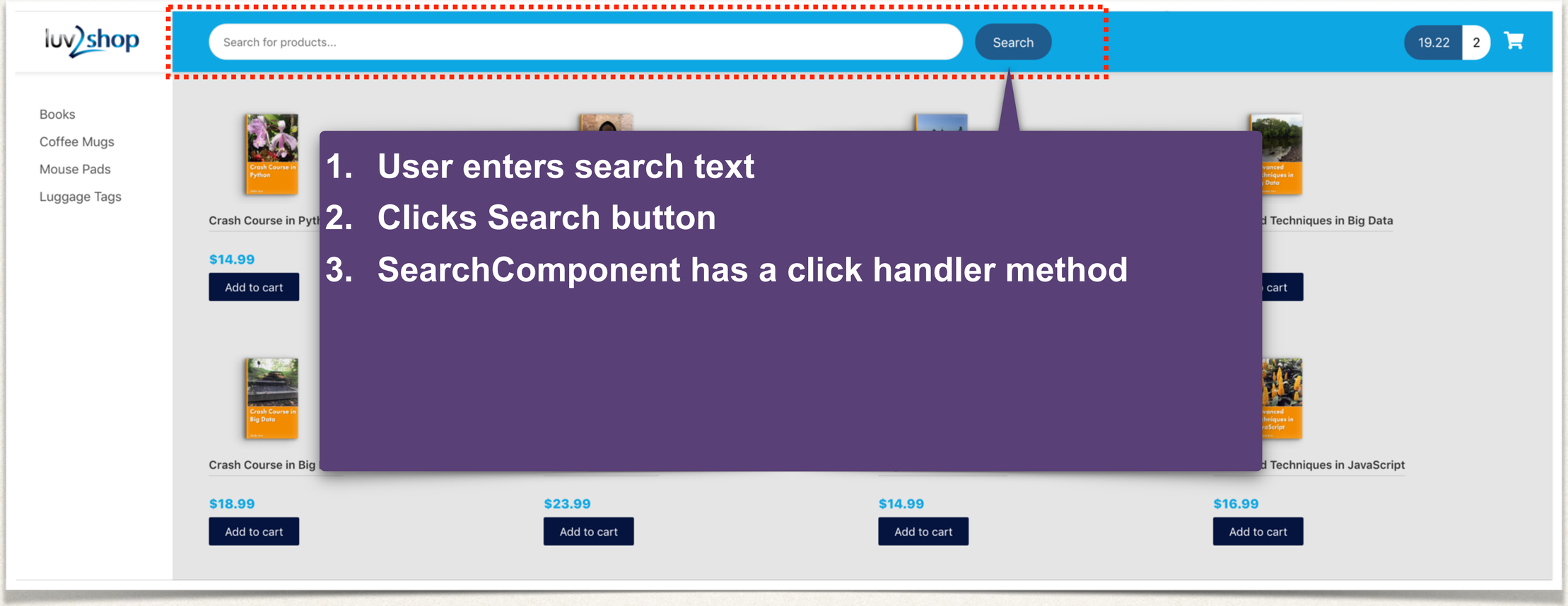
© luv2code LLC

Step 4: Update SearchComponent to send data to search route



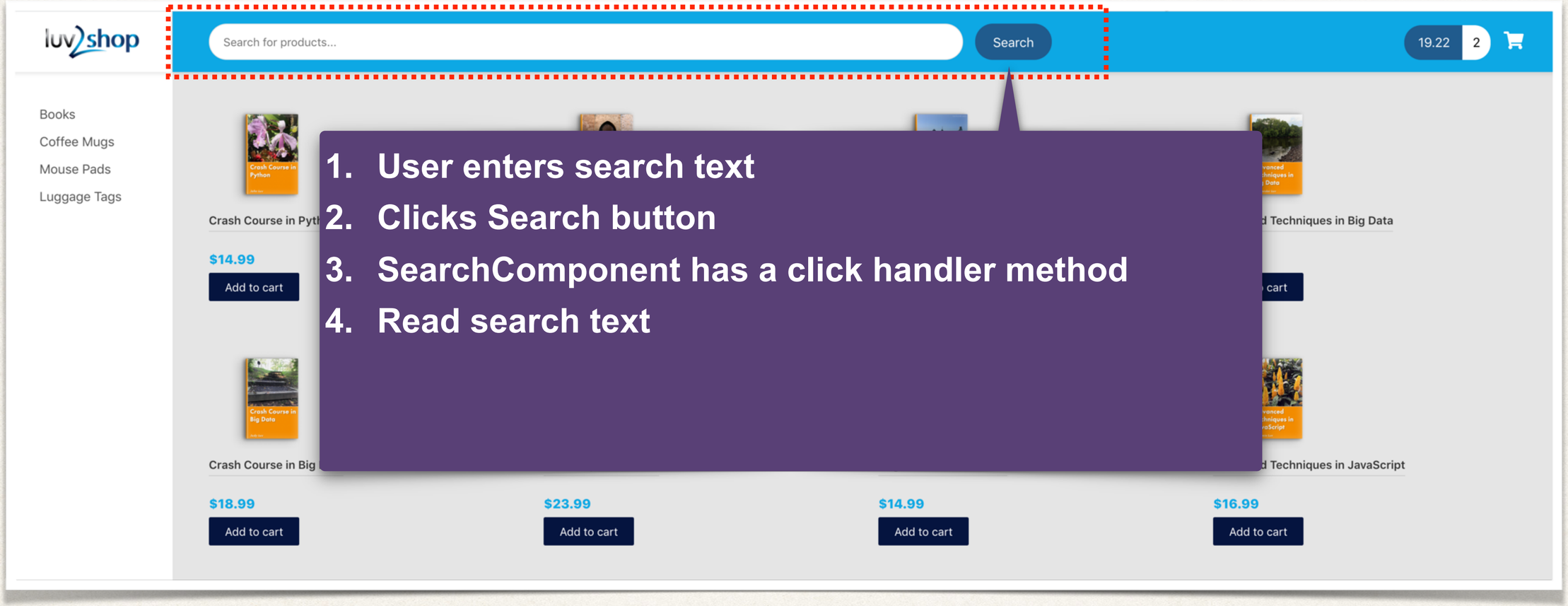
© luv2code LLC

Step 4: Update SearchComponent to send data to search route



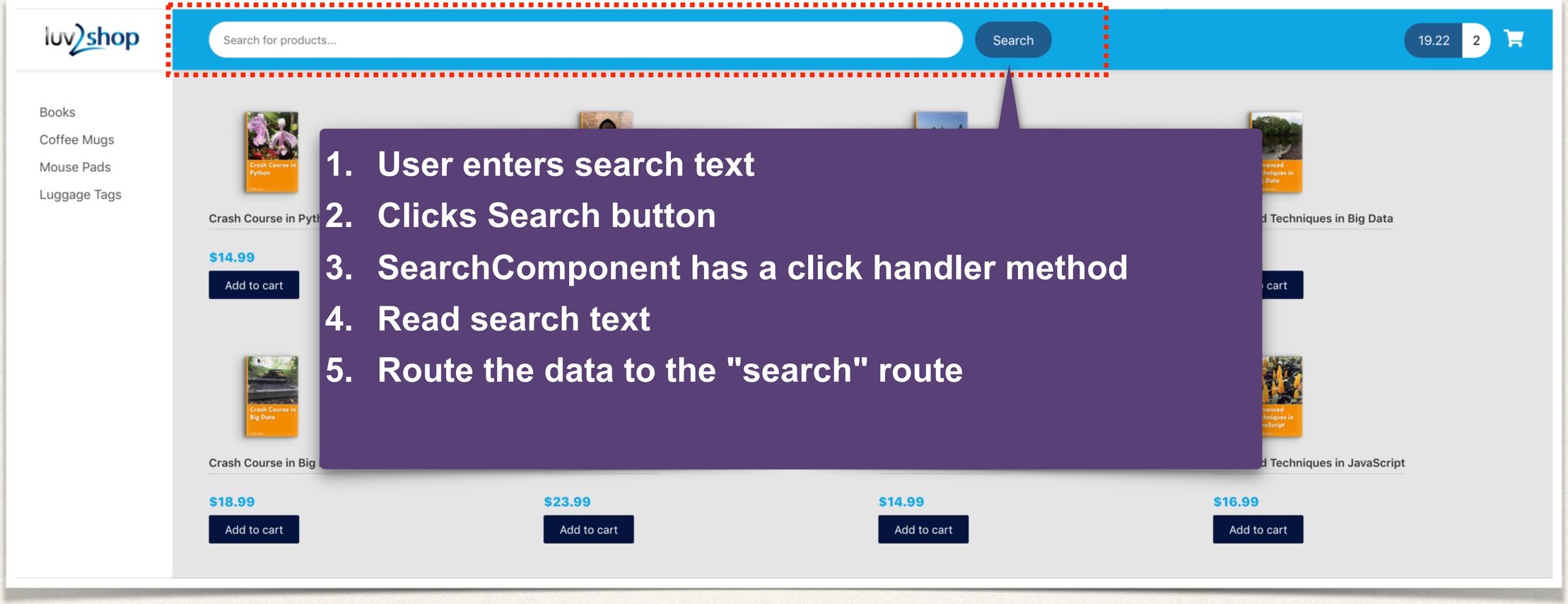
© luv2code LLC

Step 4: Update SearchComponent to send data to search route



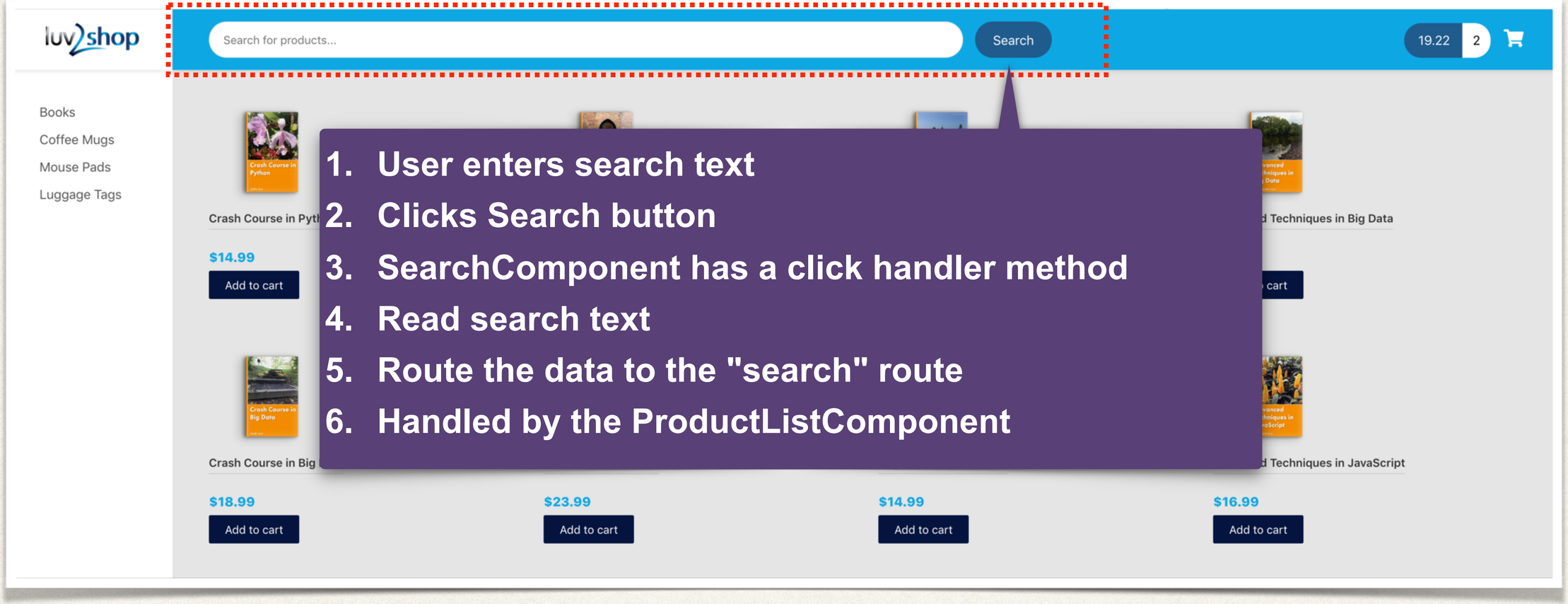
© luv2code LLC

Step 4: Update SearchComponent to send data to search route



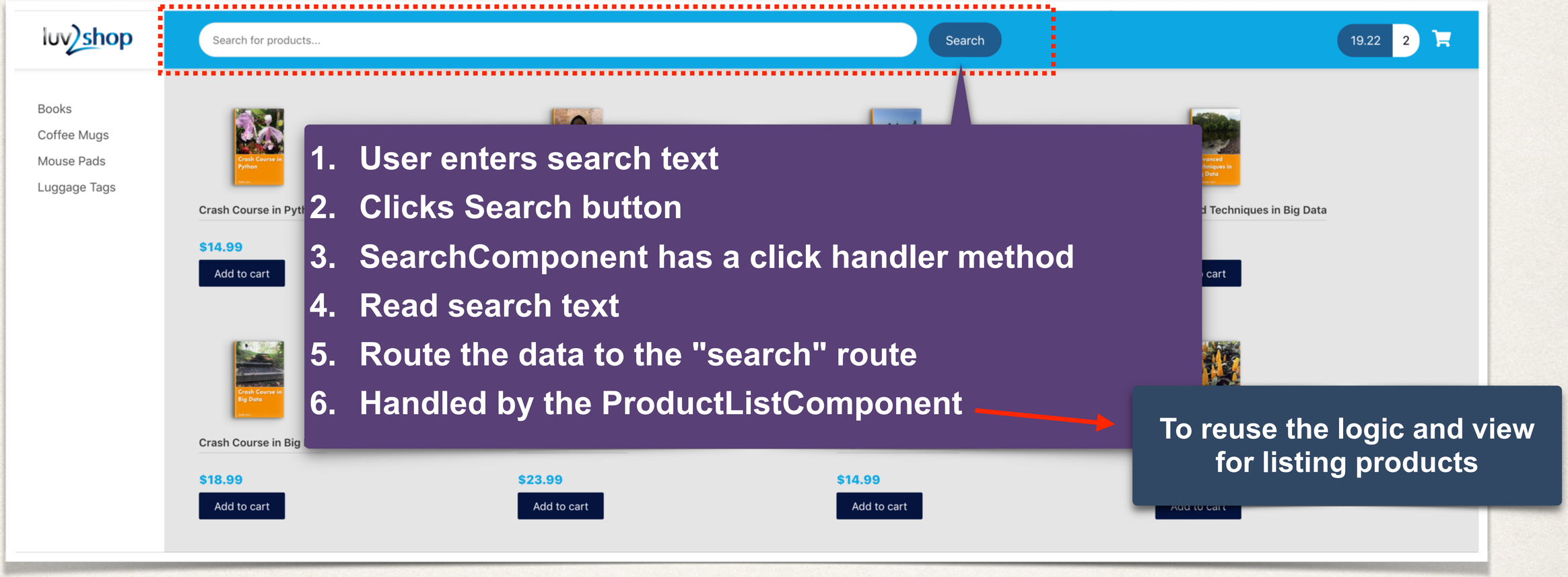
© luv2code LLC

Step 4: Update SearchComponent to send data to search route



© luv2code LLC

Step 4: Update SearchComponent to send data to search route



© luv2code LLC

Step 4: Update SearchComponent to send data to search route

© luv2code LLC

Step 4: Update SearchComponent to send data to search route

File: search.component.html

```
<div class="form-header">

    <input #myInput type="text"
           placeholder="Search for products ..."
           class="au-input au-input-xl"
           (keyup.enter)="doSearch(myInput.value)"  />

    <button (click)="doSearch(myInput.value)" class="au-btn-submit">
        Search
    </button>

</div>
```


© luv2code LLC

Step 4: Update SearchComponent to send data to search route

Template reference variable

Provides access to the element

File: search.component.html

```
<div class="form-header">

    <input #myInput type="text"
           placeholder="Search for products ..."
           class="au-input au-input-xl"
           (keyup.enter)="doSearch(myInput.value)"  />

    <button (click)="doSearch(myInput.value)" class="au-btn-submit">
        Search
    </button>

</div>
```


© luv2code LLC

Step 4: Update SearchComponent to send data to search route

Template reference variable

Provides access to the element

File: search.component.html

```
<div class="form-header">

    <input #myInput type="text"
           placeholder="Search for products ..."
           class="au-input au-input-xl"
           (keyup.enter)="doSearch(myInput.value)"  />

    <button (click)="doSearch(myInput.value)" class="au-btn-submit">
        Search
    </button>

</div>
```

Event fired when user clicks the mouse button

© luv2code LLC

Step 4: Update SearchComponent to send data to search route

File: search.component.html

```
<div class="form-header">

    <input #myInput type="text"
           placeholder="Search for products ..."
           class="au-input au-input-xl"
           (keyup.enter)="doSearch(myInput.value)"  />

    <button (click)="doSearch(myInput.value)" class="au-btn-submit">
        Search
    </button>

</div>
```

Template reference variable

Provides access to the element

Event fired when user hits the Enter key

Event fired when user clicks the mouse button

© luv2code LLC

Step 4: Update SearchComponent to send data to search route

File: search.component.html

```
<div class="form-header">

    <input #myInput type="text"
           placeholder="Search for products ..."
           class="au-input au-input-xl"
           (keyup.enter)="doSearch(myInput.value)"  />

    <button (click)="doSearch(myInput.value)" class="au-btn-submit">
        Search
    </button>

</div>
```

Template reference variable

Provides access to the element

Retrieve the value entered by the user

Event fired when user clicks the mouse button

© luv2code LLC

Step 4: Update SearchComponent to send data to search route

File: search.component.html

```
<div class="form-header">

    <input #myInput type="text"
           placeholder="Search for products ..."
           class="au-input au-input-xl"
           (keyup.enter)="doSearch(myInput.value)"  />

    <button (click)="doSearch(myInput.value)" class="au-btn-submit">
        Search
    </button>

</div>
```

Template reference variable

Provides access to the element

Retrieve the value entered by the user

Event fired when user clicks the mouse button

Method defined in our component

We can give it any name

© luv2code LLC

Step 4: Update SearchComponent to send data to search route

File: search.component.ts

```
export class SearchComponent implements OnInit {

  constructor(private router: Router) { }

  doSearch(value: string) {
    console.log(`value=${value}`);
    this.router.navigateByUrl(`/search/${value}`);
  }

  ...
}
```


© luv2code LLC

Step 4: Update SearchComponent to send data to search route

Method defined in our component

We can give it any name

File: search.component.ts

```
export class SearchComponent implements OnInit {

  constructor(private router: Router) { }

  doSearch(value: string) {
    console.log(`value=${value}`);
    this.router.navigateByUrl(`/search/${value}`);
  }

  ...
}
```


© luv2code LLC

Step 4: Update SearchComponent to send data to search route

Method defined in our component

We can give it any name

File: search.component.ts

```
export class SearchComponent implements OnInit {

  constructor(private router: Router) { }

  doSearch(value: string) {
    console.log(`value=${value}`);
    this.router.navigateByUrl(`/search/${value}`);
  }

  ...
}
```

File: search.component.html

```
<div class="form-header">

    <input #myInput type="text"
           placeholder="Search for products ..."
           class="au-input au-input-xl"
           (keyup.enter)="doSearch(myInput.value)"  />

    <button (click)="doSearch(myInput.value)" class="au-btn-submit">
        Search
    </button>

</div>
```


© luv2code LLC

Step 4: Update SearchComponent to send data to search route

Method defined in our component

We can give it any name

File: search.component.ts

```
export class SearchComponent implements OnInit {

  constructor(private router: Router) { }

  doSearch(value: string) {
    console.log(`value=${value}`);
    this.router.navigateByUrl(`/search/${value}`);
  }

  ...
}
```

Route the data to our "search" route

It will be handled by the ProductListComponent

File: search.component.html

```
<div class="form-header">

    <input #myInput type="text"
           placeholder="Search for products ..."
           class="au-input au-input-xl"
           (keyup.enter)="doSearch(myInput.value)"  />

    <button (click)="doSearch(myInput.value)" class="au-btn-submit">
        Search
    </button>

</div>
```


© luv2code LLC

Step 4: Update SearchComponent to send data to search route

Method defined in our component

We can give it any name

File: search.component.ts

```
export class SearchComponent implements OnInit {

  constructor(private router: Router) { }

  doSearch(value: string) {
    console.log(`value=${value}`);
    this.router.navigateByUrl(`/search/${value}`);
  }

  ...
}
```

Route the data to our "search" route

It will be handled by the ProductListComponent

File: search.component.html

```
<div class="form-header">

    <input #myInput type="text"
           placeholder="Search for products ..."
           class="au-input au-input-xl"
           (keyup.enter)="doSearch(myInput.value)"  />

    <button (click)="doSearch(myInput.value)" class="au-btn-submit">
        Search
    </button>

</div>
```

To reuse the logic and view for listing products

© luv2code LLC

Step 5: Enhance component to search for products with product service

© luv2code LLC

Step 5: Enhance component to search for products with product service

File: product-list.component.ts

```
...
    const theKeyword: string = this.route.snapshot.paramMap.get('keyword');

    // now search for the products using keyword
    this.productService.searchProducts(theKeyword).subscribe(
      data => {
        this.products = data;
      }
    )
...
```


© luv2code LLC

Step 5: Enhance component to search for products with product service

File: app.module.ts

```
...
  {path: 'search/:keyword', component: ProductListComponent},
...
```

File: product-list.component.ts

```
...
    const theKeyword: string = this.route.snapshot.paramMap.get('keyword');

    // now search for the products using keyword
    this.productService.searchProducts(theKeyword).subscribe(
      data => {
        this.products = data;
      }
    )
...
```


© luv2code LLC

Step 5: Enhance component to search for products with product service

File: app.module.ts

```
...
  {path: 'search/:keyword', component: ProductListComponent},
...
```

File: product-list.component.ts

```
...
    const theKeyword: string = this.route.snapshot.paramMap.get('keyword');

    // now search for the products using keyword
    this.productService.searchProducts(theKeyword).subscribe(
      data => {
        this.products = data;
      }
    )
...
```

Passed in from SearchComponent

© luv2code LLC

Step 5: Enhance component to search for products with product service

File: app.module.ts

```
...
  {path: 'search/:keyword', component: ProductListComponent},
...
```

File: product-list.component.ts

```
...
    const theKeyword: string = this.route.snapshot.paramMap.get('keyword');

    // now search for the products using keyword
    this.productService.searchProducts(theKeyword).subscribe(
      data => {
        this.products = data;
      }
    )
...
```

File: search.component.ts

```
export class SearchComponent implements OnInit {

 constructor(private router: Router) { }

  doSearch(value: string) {
    console.log(`value=${value}`);
    this.router.navigateByUrl(`/search/${value}`);
  }

  ...
}
```

Passed in from SearchComponent

© luv2code LLC

Step 6: Update product service to call URL on Spring Boot app

© luv2code LLC

Step 6: Update product service to call URL on Spring Boot app

File: product.service.ts

```
...

  searchProducts(theKeyword: string): Observable<Product[]> {

    // need to build URL based on the keyword
    const searchUrl = `${this.baseUrl}/search/findByNameContaining?name=${theKeyword}`;

    return this.httpClient.get<GetResponseProducts>(searchUrl)
                                .pipe(
                                   map(response => response._embedded.products));

  }

...

interface GetResponseProducts {
  _embedded: {
    products: Product[];
  }
}
```


© luv2code LLC

Step 6: Update product service to call URL on Spring Boot app

File: product.service.ts

URL for searching products

```
...

  searchProducts(theKeyword: string): Observable<Product[]> {

    // need to build URL based on the keyword
    const searchUrl = `${this.baseUrl}/search/findByNameContaining?name=${theKeyword}`;

    return this.httpClient.get<GetResponseProducts>(searchUrl)
                                .pipe(
                                   map(response => response._embedded.products));

  }

...

interface GetResponseProducts {
  _embedded: {
    products: Product[];
  }
}
```


© luv2code LLC

Step 6: Update product service to call URL on Spring Boot app

File: product.service.ts

URL for searching products

```
...

  searchProducts(theKeyword: string): Observable<Product[]> {

    // need to build URL based on the keyword
    const searchUrl = `${this.baseUrl}/search/findByNameContaining?name=${theKeyword}`;

    return this.httpClient.get<GetResponseProducts>(searchUrl)
                                .pipe(
                                   map(response => response._embedded.products));

  }

...

interface GetResponseProducts {
  _embedded: {
    products: Product[];
  }
}
```

Call REST API

© luv2code LLC

Step 6: Update product service to call URL on Spring Boot app

File: product.service.ts

```
...

  searchProducts(theKeyword: string): Observable<Product[]> {

    // need to build URL based on the keyword
    const searchUrl = `${this.baseUrl}/search/findByNameContaining?name=${theKeyword}`;

    return this.httpClient.get<GetResponseProducts>(searchUrl)
                                .pipe(
                                   map(response => response._embedded.products));

  }

...

interface GetResponseProducts {
  _embedded: {
    products: Product[];
  }
}
```

URL for searching products

Call REST API

Returns an observable

Map the JSON data from Spring Data REST to Product array

© luv2code LLC

Step 6: Update product service to call URL on Spring Boot app

File: product.service.ts

```
...

  searchProducts(theKeyword: string): Observable<Product[]> {

    // need to build URL based on the keyword
    const searchUrl = `${this.baseUrl}/search/findByNameContaining?name=${theKeyword}`;

    return this.httpClient.get<GetResponseProducts>(searchUrl)
                                .pipe(
                                   map(response => response._embedded.products));

  }

...

interface GetResponseProducts {
  _embedded: {
    products: Product[];
  }
}
```

URL for searching products

Call REST API

Returns an observable

Map the JSON data from Spring Data REST to Product array

Unwraps the JSON from Spring Data REST _embedded entry

© luv2code LLC