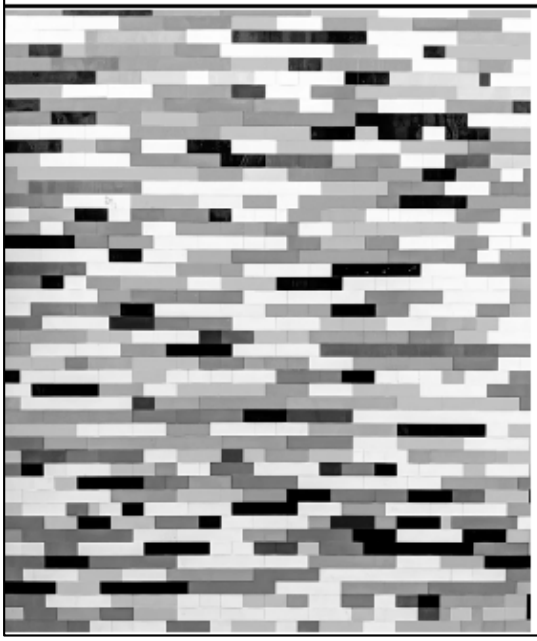
DEEPer

Bootstrap, jQuery and APIs
Week 2 Session 2

X

TODAY'S SESSION

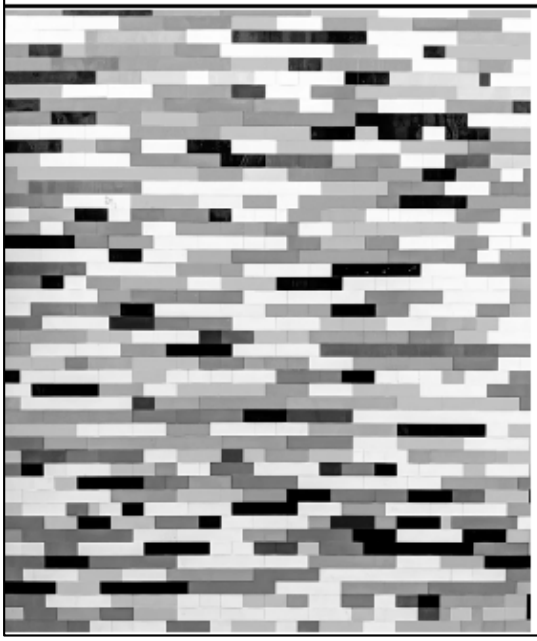


01 Bootstrap

02 jQuery

03 AJAX

TODAY'S SESSION



04 Data Formats

05 APIs

Bootstrap

01

What Is Bootstrap

- Open source HTML, CSS and JS library
- Current latest release: v4.5.1
- Founded initially by developers at Twitter, who left in 2012
- Drops support for IE9 and below

Why Use A UI Library?

- Facilitates easier mobile-first responsive front-end web development
- Provides a range of pre-styled UI components
- Also provides many JavaScript components
- Solves many cross-browser issues
- In the case of larger libraries, large pool of resources available

Getting Started

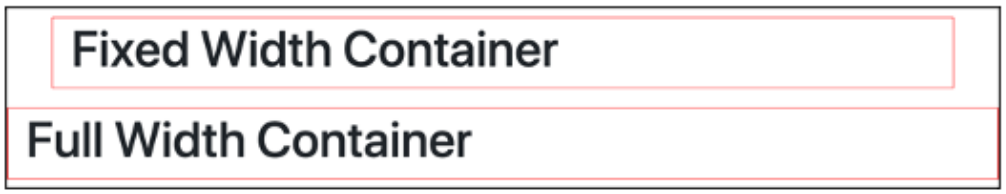
https://getbootstrap.com/docs/4.5/getting-started/introduction/

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.1/css/bootstrap.min.css"
      integrity="sha384-VCmXjywReHh4PwowAiWNagnWcLhlEJLA5buUprzK8rxFgeH0kww/aWY76TfkUoSX"
      crossorigin="anonymous"
    >
  </head>
  <body>
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script
      src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
      integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
      crossorigin="anonymous"
      ></script>
    <script
      src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
      integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
      crossorigin="anonymous"
      ></script>
    <script
    src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.1/js/bootstrap.min.js"
    integrity="sha384-XEerZL8cuoUbHE4nZReLT7nx9gQrQreJekYhJD9WNWhH8nEW+0c5qq7aIo2Wl30J"
    crossorigin="anonymous"
    ></script>
  </body>
</html>
```

https://getbootstrap.com/docs/4.5/getting-started/introduction/

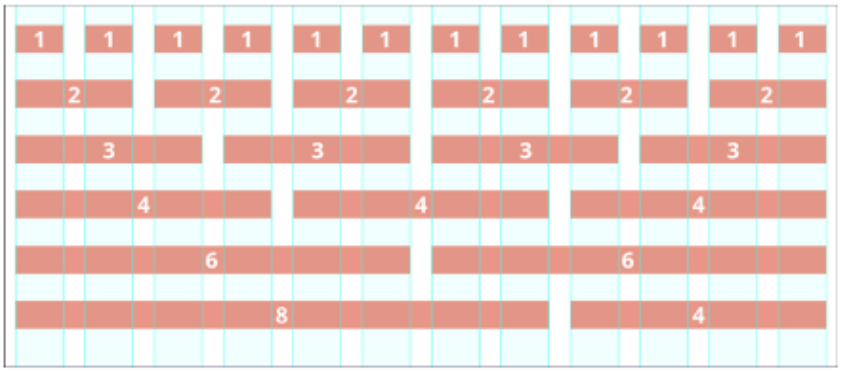
Containers


```
<body>
  <!-- Containers are required when using the grid system -->
  <div class="container container-example">
    <h1>Fixed Width Container</h1>
  </div>

  <div class="container-fluid container-example">
    <h1>Full Width Container</h1>
  </div>
</body>
```



https://getbootstrap.com/docs/4.4/layout/overview/#containers

Grid System



- 12 columns
- Fully responsive
- Allows complex layouts to be developed with ease

https://getbootstrap.com/docs/4.4/layout/overview/#containers

Grid System – Fixed Columns


```
<div class="container">
  <div class="row">
    <div class="col-8">col 8</div>
    <div class="col-4">col 4</div>
  </div>

  <div class="row">
    <div class="col-4">col 4</div>
    <div class="col-4">col 4</div>
    <div class="col-4">col 4</div>
  </div>
</div>
```

col-n classes retain their percentage width at all screen sizes

Responsive Breakpoints

Name	Code	Column Example	Screen Size
Extra Small		`.col-12`	< 576px
Small	sm	`.col-sm-8`	≥ 576px
Medium	md	`.col-md-6`	≥ 768px
Large	lg	`.col-lg-4`	≥ 992px
Extra Large	xl	`.col-xl-3`	≥1200px
Flex		`.col`	Any

These responsive codes apply to many elements across Bootstrap, but the screen size measurements are always the same. In this example if all of the listed size classes (all but flex) are applied to a single element, the column will resize depending on the size of the viewport. E.g. if the viewport is currently at 600px the column will be 6 columns wide, if the viewport is resized to 1100px it will shrink to 4 columns wide.

Grid System – Responsive Columns (800px)

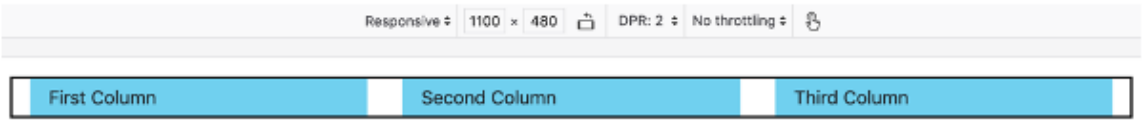

```
<div class="container-fluid">
  <div class="row">
    <div class="col-12 col-sm-8 col-md-6 col-lg-4 col-xl-3">
      <p>First Column</p>
    </div>

    <div class="col-12 col-sm-8 col-md-6 col-lg-4 col-xl-3">
      <p>Second Column</p>
    </div>

    <div class="col-12 col-sm-8 col-md-6 col-lg-4 col-xl-3">
      <p>Third Column</p>
    </div>
  </div>
</div>
```

Code	Screen Size
	< 576px
sm	≥ 576px
md	≥ 768px
lg	≥ 992px
xl	≥1200px

Grid System – Responsive Columns (1100px)


```
<div class="container-fluid">
  <div class="row">
    <div class="col-12 col-sm-8 col-md-6 col-lg-4 col-xl-3">
      <p>First Column</p>
    </div>

    <div class="col-12 col-sm-8 col-md-6 col-lg-4 col-xl-3">
      <p>Second Column</p>
    </div>

    <div class="col-12 col-sm-8 col-md-6 col-lg-4 col-xl-3">
      <p>Third Column</p>
    </div>
  </div>
</div>
```

Code	Screen Size
	< 576px
sm	≥ 576px
md	≥ 768px
lg	≥ 992px
xl	≥1200px



Tables


```html
<!-- A class of "table" applies base styling to a table element -->
<table class="table">
  <thead>
      <tr>
        <th>ID</th>
        <th>Name</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>1</td>
        <td>John Smith</td>
      </tr>
      <tr>
        <td>2</td>
        <td>Jane Doe</td>
      </tr>
    </tbody>
</table>
```

ID	Name
1	John Smith
2	Jane Doe

Tables

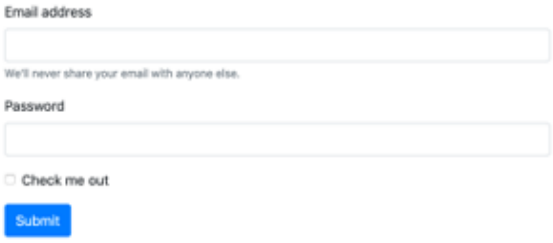

```
class="table table-striped"
```

ID	Name
1	John Smith
2	Jane Doe
3	Brian Robson
4	Sarah Brown

```
class="table table-striped table-dark"
```

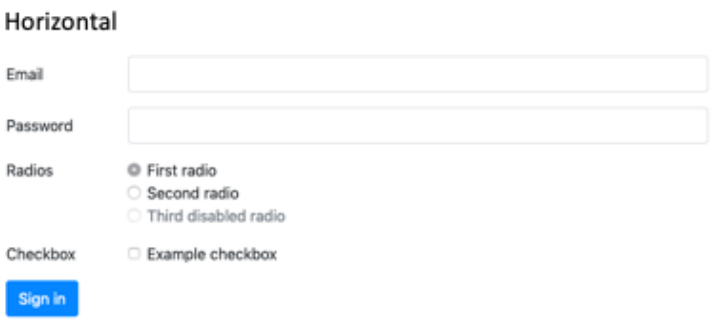
ID	Name
1	John Smith
2	Jane Doe
3	Brian Robson
4	Sarah Brown

Forms




```
<form>
  <div class="form-group">
    <label>Email address</label>
    <input type="email" class="form-control">
    <small class="form-text text-muted">We'll never share your email with anyone else.</small>
  </div>
  <div class="form-group">
    <label>Password</label>
    <input type="password" class="form-control">
  </div>
  <div class="form-group form-check">
    <input type="checkbox" class="form-check-input">
    <label class="form-check-label">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

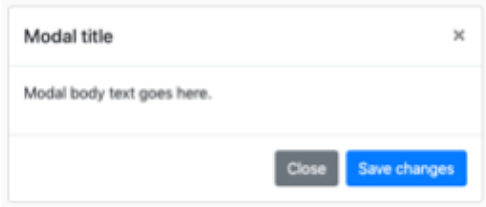
Forms – Layout



Buttons

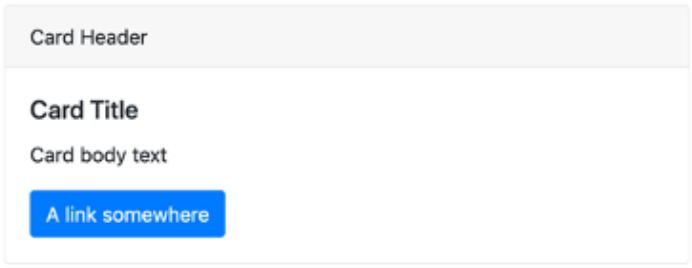

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>

<button type="button" class="btn btn-link">Link</button>
```

Modals



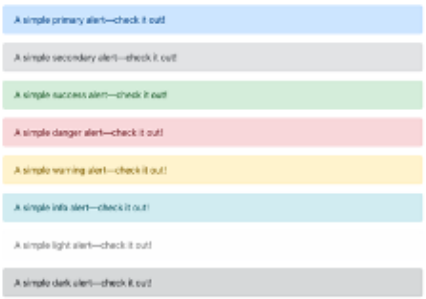

```
<button type="button" class="btn btn-primary" data-target="#exampleModal">
  Launch demo modal
</button>

<div class="modal fade" id="exampleModal" role="dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h3 class="modal-title" id="exampleModalLabel">Modal title</h5>
      </div>
      <div class="modal-body">
        Modal text body goes here.
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```

Cards




```
<div class="card">
  <div class="card-header">Card Header</div>
  <div class="card-body">
    <h5 class="card-title">Card Title</h5>
    <p class="card-text">Card body text</p>
    <a href="#" class="btn btn-primary">A link somewhere</a>
  </div>
</div>
```

Alerts




```
<div class="alert alert-primary">...</div>
<div class="alert alert-secondary">...</div>
<div class="alert alert-success">...</div>
<div class="alert alert-danger">...</div>
<div class="alert alert-warning">...</div>
<div class="alert alert-info">...
<div class="alert alert-light">...</div>
<div class="alert alert-dark">...</div>
```

Customising Boostrap

- Bootstrap makes it **very** easy to build well structured, attractive interfaces
- This however leads to many very similar looking websites
- When getting started, using default Bootstrap styling is perfectly fine
- When more comfortable, you should look to customise its appearance
- Bootstrap is a framework; it should serve as the **foundation** for your UI **not the full solution**

The primary point here is that Bootstrap should be the foundation to your UI build, not the solution

Customising Boostrap

- Every element of Bootstrap can be changed, from colours to sizing, borders to animations
- CSS rules provided by Bootstrap can still be manually overridden by your own custom CSS so long as
  - Your CSS selector is more specific than the one provided by Bootstrap, or
  - Your selector is equally specific but is loaded **after** Bootstrap's CSS

The primary point here is that Bootstrap should be the foundation to your UI build, not the solution

Other CSS Frameworks

- Foundation - https://get.foundation/
- Tailwind - https://tailwindcss.com/
- Semantic UI - https://semantic-ui.com/
- Bulma - https://bulma.io/

The primary point here is that Bootstrap should be the foundation to your UI build, not the solution

02

jQuery

What is jQuery

- A JavaScript library which offers simplified ways of manipulating the DOM over raw JS
- Also provides a simple interface for AJAX (to come) and event management
- Offers many animation techniques, which have fallen out of favour thanks to CSS animations
- Provides developers with a simple way of creating reusable interactive elements

Syntax

- When jQuery is loaded, a global variable is defined containing its functionality
- jQuery use generally consists of selecting element(s), then manipulating them

```
jQuery(selector).action();
$(selector).action();
```

Selecting Elements

- jQuery allows for the selection of DOM elements using **any** CSS selector
- Any jQuery function run on the returned value will affect all elements

```
// el will contain 0, 1 or more elements as found by the selector

var el = $('#page-title');     // id of page-title
```

Selecting Elements


```
// Traverse up the DOM hierarchy
var el = $('#my-element').parent().parent().find('.my-class');
```

Creating Elements

- jQuery also provides a simpler method of dynamically creating HTML elements to insert into the page
- Passing an opening HTML tag to the jQuery function will create that element
- Methods exist to manipulate the element's attributes and content

```
var dynamicElement = document.createElement('p');
dynamicElement.innerHTML = 'My dynamic content';
dynamicElement.id = 'my-id';
dynamicElement.className = 'my-class';

$('<p>').id('my-id').addClass('my-class').text('My dynamic content');
```

Appending Elements


```
<ul id="shopping-list">
  <li>Milk</li>
</ul>

<script type="text/javascript">
  var listItem = document.createElement('li');
  listItem.innerHTML = 'Bread';

  var list = document.getElementById('shopping-list');
  list.appendChild(listItem);
</script>


<script type="text/javascript">
  $('<li>').text('Bread').appendTo('#shopping-list');
</script>
```

Element Visibility

- Useful methods for making an element visible or hidden include:
  - `$('.my-elements').show();`
  - `$('.my-elements').hide();`
  - `$('.my-elements').toggle();`

Element Animations

- Useful methods for animating elements include:
  - `$('.my-elements').fadeIn();`
  - `$('.my-elements').fadeOut();`
  - `$('.my-elements').fadeToggle();`

Document Ready

- Sometimes, an element may not have rendered before we try to access it with jQuery
- We can wait until the document is rendered (ready):

```
$(document).ready(function() {
  // ...
});
```

Extending Elements

- When accessed with $('…'), elements are wrapped with extra jQuery-provided methods and properties
- There is a wide array of jQuery plugins which offer further extensibility – e.g. UX elements such as sortable tables, tabbed content, and accordions

Extending Elements


```
<p class="reverse">Reverse me!</p>

<script>
  $.fn.reverseText = function() {
    var text = $(this).text();
    var characterArray = text.split('');
    var reversedArray = characterArray.reverse();
    var reversedText = reversedArray.join('');

    $(this).text(reversedText);

    return this;
  }

  $('.reverse').reverseText();
</script>
```

Choosing a Plugin

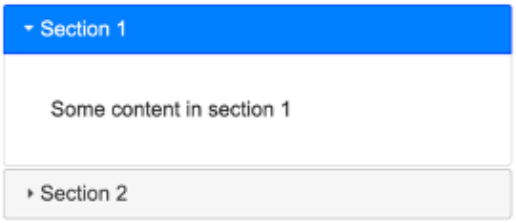
- Often, there are many plugins which achieve the same thing
- A set of criteria which may help us select the right plugin:
  - How many active users?
  - When was the last commit on GitHub?
  - How many open issues are there?
  - Is it compatible with the version I'm using?

jQuery UI

- Provides skinnable interactive elements
- In some ways like Bootstrap, although elements need to be manually declared (whereas Bootstrap auto-applies based on classes)

jQuery UI


```
<div id="accordion">
  <h3>Section 1</h3>
  <div>
    <p>Some content in section 1</p>
  </div>
  <h3>Section 2</h3>
  <div>
    <p>Some content in section 2</p>
  </div>
</div>

<script>
  $('#accordion').accordion();
</script>
```



AJAX

03

What is AJAX

41

- Stands for Asynchronous XML and JavaScript
- Allows us to make a request to an external resource without refreshing the page
- Commonly used in form submissions

- TODO: Find or make a diagram

Native AJAX Requests


```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  /**
   * readyState can be:
   * 0: UNSENT
   * 1: OPENED
   * 2: HEADERS_RECEIVED
   * 3: LOADING
   * 4: DONE
   */
  if (this.readyState === 4 && this.status === 200) {
     console.log(xhttp.responseText);
  }
};

xhttp.open("GET", "https://api.jokes.one/jod", true);
xhttp.send();
```

- TODO: Find or make a diagram

AJAX Libraries

- Many libraries exist for making AJAX requests
- jQuery offers a method which allows us to provide callbacks or return a promise
- Axios is a widely used library, offering request interception and other useful features

- TODO: Find or make a diagram

jQuery AJAX


```
$.ajax({
  url: 'https://api.postcodes.io/postcodes',
  type: 'POST',
  data: { postcodes: ['PR7 7DW', 'SW1A 1AA'] },
  success: function(response) {
    console.log(response);
  },
  error: function(error) {
    console.log(error);
  }
});
```

Axios AJAX


```
axios.post('https://api.postcodes.io/postcodes', {
  postcodes: ['PR7 7DW', 'SW1A 1AA']
})
  .then(res => console.log(res))
  .catch(err => console.log(err));
```

04

Data Formats

JSON

- Stands for “JavaScript Object Notation”
- Most commonly used data format in modern APIs
- Built on two structures:
- Name / value pairs
- An ordered list of values (array)

JSON


```json
{
  "stringProperty": "lorem ipsum",
  "intProperty": 7,
  "arrayProperty": [1, 2, 3],
  "objectProperty": {
    "foo": "bar"
  }
}
```

JSON


```
[
  {
    id: 1,
    name: "Joe Bloggs",
    review: "Love this stuff!",
    rating: 5,
    dateTime: "2020-08-04T21:42:02+00:00"
  },
  {
    id: 2,
    name: "Jane Doe",
    review: "I've had better.",
    rating: 2,
    dateTime: "2020-08-02T17:29:47+00:00"
  }
]
```

XML

- Stands for “Extensible Markup Language”
- Not as widely used as it used to be (but still very common!)
- Similar structure to HTML where elements have attributes
- Always has a single root node (like the <html> tag)

XML


```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<root>
  <child>
    <grandchild attribute1="foo" attribute2="bar">
      <greatgrandchild />
    </grandchild>
  </child>
</root>
```

XML


```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<checkins>
  <checkin>
    <id>1</id>
    <name>Joe Bloggs</name>
    <review>Love this stuff!</review>
    <rating>5</rating>
    <dateTime>2020-08-04T21:42:02+00:00</dateTime>
  </checkin>
    <checkin>
      <id>2</id>
      <name>Jane Doe</name>
      <review>I've had better.</review>
      <rating>2</rating>
      <dateTime>2020-08-02T17:29:47+00:00</dateTime>
    </checkin>
</checkins>
```

YAML

- Stands for “YAML Ain’t Markup Language”
- Commonly used for application configurations
- Uses whitespace to identify parent/child relations

YAML


```yaml
---
 xmas-fifth-day:
   golden-rings: 5
   calling-birds: four
   french-hens: 3
   turtle-doves: two
   partridges:
     count: 1
     location: "a pear tree"
```

APIs

05

APIs

- Stands for “Application Programming Interface”
- Provide a method of interacting with data from an external service
- These services can be internal or external systems
- They are commonly called using AJAX to retrieve data from the server without needing to reload the page

RESTful APIs

- Stands for “Representational State Transfer”
- Defines one or more resources that a user can then interact with
- Utilises standardized URL structures along with HTTP verbs to determine actions for each resource
- Most common verbs are GET, POST, PUT, PATCH, DELETE and OPTIONS
- JSON is the most used data format
- Requests are stateless - no contextual information is stored on the server between requests (e.g. cookies)

RESTful Verbs

- All web requests use an HTTP method, depending on the intended action
- GET: Retrieve a resource
- POST: Create a resource
- PUT: Replace a resource
- PATCH: Partially update a resource
- DELETE: Remove a resource
- OPTIONS: Which actions are available for this resource?

Example Resource Actions


```
GET /products           // Return a collection of products
GET /products/123       // Return a single product identified by 123
POST /products          // Create a new product
PUT /products/123       // Replace product 123
PATCH /products/123     // Partially update product 123
DELETE /products/123    // Delete product 123
```

- URLs should **not** contain verbs
- Singular vs Plural resource names is preference (but plural makes more sense!)

Example Resource Actions


```
GET /products/123/reviews   // Return a collection of reviews for product 123
POST /products/123/reviews  // Create a new review for product 123
```

- Resources can have children, referred to as Sub-Resources

Query

- Additional parameters can be provided to API requests using the query string
- This is generally used for GET requests
- APIs often use these parameters for tasks such as filtering and pagination

```
GET /products?where[category]=5&where[title]=macbo
```

RESTful Status Codes

- 2xx: Success
- 3xx: Redirection
- 4xx: Client Error
- 5xx: Server Error

- 200: Success
- 201: Created
- 400: Bad Request
- 401: Forbidden
- 404: Not Found
- 500: Internal Server Error

RESTful Examples


```
GET /products
[
  {
    "id": 1,
    "title": "Some Product",
    "rating": 3.5
  },
  {
    "id": 2,
    "title": "Another Product",
    "rating": 4.5
  }
]
```

RESTful Examples


```
GET /products/1
{
  "id": 1,
  "title": "Some Product",
  "rating": 3.5
}
```

RESTful Examples


```
POST /products
Request Body:
{
  "title": "Yet Another Product",
  "rating": 4
}
Response: [201 Created]
{
  "id": 3,
  "title": "Yet Another Product",
  "rating": 4
}
```

RESTful Examples


```
PATCH /products/1
Request Body:
{
  "title": "Altered Product"
}
Response: [200 OK]
{
  "id": 1,
  "title": "Altered Product",
  "rating": 3.5
}
```

RESTful Examples


```
DELETE /products/1 [204 No Content]
```

RPC APIs

- Stands for “Remote Procedural Calls”
- Often uses XML or JSON as a data format
- While REST is used for sharing resource representations, RPC is about calling remote actions
- Think of this as URLs like /create-product