



JavaScript Frameworks and Libraries

Language Technology and Web Applications

Johannes Graën, Nikolina Rajović, Igor Mustač

Department of Computational Linguistics &
Linguistic Research Infrastructure (LiRI)

November 15, 2023

Topics

1. jQuery & Modern Web Frameworks

2. D3.js

3. SVG vs Canvas

Topics

1. jQuery & Modern Web Frameworks

2. D3.js

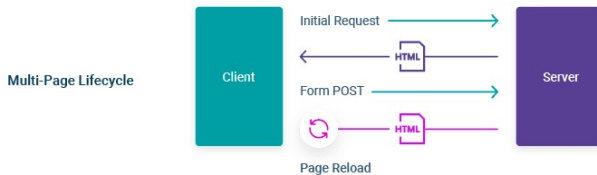
3. SVG vs Canvas

What is Frontend Web Development?

- Designing user interfaces and interactions
- Core technologies: HTML, CSS, JavaScript
- End Goal: Enhance user experience and interactivity

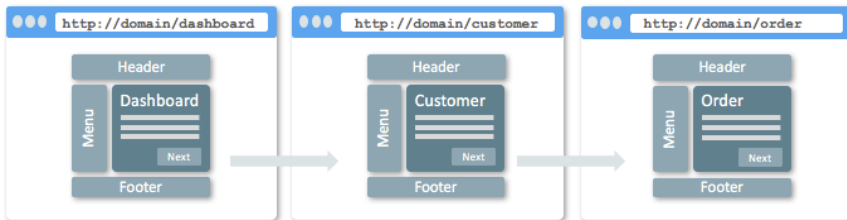
Evolution of Web Development

- Early 2000s: Static HTML Pages
- Mid-2000s: Introduction of AJAX for dynamic content
- Late 2000s: Rise of JavaScript and advanced CSS
- 2010s: Single Page Applications (SPAs)



Evolution of Web Development

- Early 2000s: Static HTML Pages
- Mid-2000s: Introduction of AJAX for dynamic content
- Late 2000s: Rise of JavaScript and advanced CSS
- 2010s: Single Page Applications (SPAs)



Evolution of Web Development

- Early 2000s: Static HTML Pages
- Mid-2000s: Introduction of AJAX for dynamic content
- Late 2000s: Rise of JavaScript and advanced CSS
- 2010s: Single Page Applications (SPAs)



The Role of JavaScript in Web Development

- Adds Interactivity: Forms, animations, page transitions
- More complex with AJAX, JSON, and frontend libraries
- Challenge: Managing complexity in larger web applications

The Role of JavaScript in Web Development

- Basic HTML **select** element:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/select>

- **Advanced** select element:

<https://select2.org/dropdown>

<https://jsfiddle.net/NathanFriend/rLmztr2d>



- Revolutionized JavaScript coding
- Easy DOM manipulation, event handling, and AJAX
- Cross-Browser Compatibility: Smoothed out inconsistencies across browsers

Impact of jQuery

- Easy selection and manipulation of HTML elements
- Separating JavaScript functionality from HTML structure
- Paved the way for modern frontend frameworks

jQuery examples

```
document.getElementById("demo").innerHTML = "Hello, JavaScript!"
```

jQuery examples

```
document.getElementById("demo").innerHTML = "Hello, JavaScript!"
```

```
$("#demo").text("Hello, jQuery!");
```

jQuery

jQuery examples

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'test.html', true);
xhr.onreadystatechange = function() {
    if (xhr.readyState == 4 && xhr.status == 200) {
        document.getElementById("div1").innerHTML = xhr.responseText;
    }
};
xhr.send();
```

jQuery examples

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'test.html', true);
xhr.onreadystatechange = function() {
    if (xhr.readyState == 4 && xhr.status == 200) {
        document.getElementById("div1").innerHTML = xhr.responseText;
    }
};
xhr.send();
```

```
fetch("test.html")
    .then(response => response.text())
    .then(data => {
        document.getElementById("div1").innerHTML = data;
    });
```

Modern JavaScript

jQuery examples

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'test.html', true);
xhr.onreadystatechange = function() {
    if (xhr.readyState == 4 && xhr.status == 200) {
        document.getElementById("div1").innerHTML = xhr.responseText;
    }
};
xhr.send();
```

```
$.ajax({
    url: "test.html",
    success: function(result) {
        $("#div1").html(result);
    }
});
```

jQuery

jQuery vs Modern Frameworks

- Library vs Framework
- DOM Manipulation: Direct vs. Virtual DOM (React)
- Ideal for smaller applications vs. complex SPAs
- Architectural Approach: Scripting enhancements vs. Component-based structure

Frontend Frameworks

- Collections of JavaScript tools and libraries for efficient web development
- Simplify and organize code, enhance performance, ensure scalability

Benefits of Frontend Frameworks

- Efficiency: Reusable components and predefined functions

- Consistency: Standardized coding practices



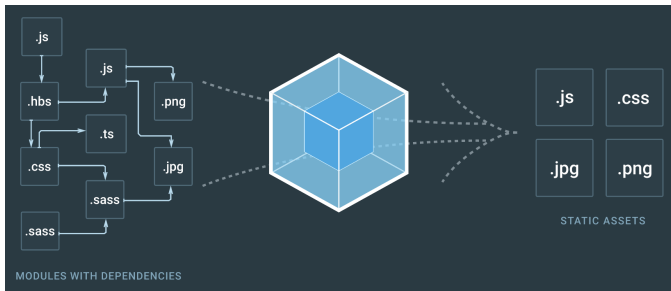
- Performance: Built-in optimization tools



- Active development, resources, and troubleshooting

Node.js

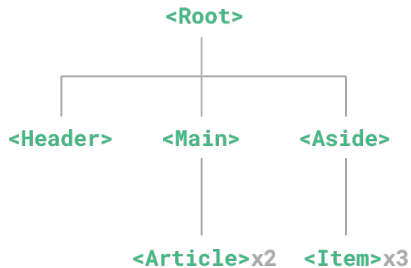
- A JavaScript runtime built on Chrome's V8 JavaScript engine
- A key technology in modern web development
- Extensive npm package ecosystem





- Core Features: Reactive data binding, virtual DOM, component-based architecture
- Ease of Integration: Can be gradually introduced into existing projects
- Flexibility: Easily integrates with other libraries and projects

Vue.js: Component based



Why Vue.js?

- Reactive Data Binding: Seamless synchronization between model and view
- Simplicity and Learning Curve: Easier to grasp for beginners
- Flexibility: Suited for both small-scale and large-scale applications

Vue.js in Action

- Let's play with Vue.js

Vue.js in Action

- Let's play with Vue.js
- Lifecycle: <https://vuejs.org/guide/essentials/lifecycle.html>

React.js: The UI Library

- Developed by Facebook
- Component-Based: Reusable components to manage UI
- Virtual DOM
- JSX: Writing HTML in JavaScript for easier component creation

React.js: JSX

```
function GreetingComponent() {  
  const name = 'Alice';  
  const greeting = 'Hello';  
  
  return (  
    <div>  
      <h1>{greeting}, {name}!</h1>  
      <p>Welcome to the world of React and JSX.</p>  
      <button onClick={() => alert('Button clicked!')}>  
        Click Me  
      </button>  
    </div>  
  );  
}  
  
export default GreetingComponent;
```

Angular: Enterprise-Level Framework

- Developed by Google
- TypeScript-based
- Suited for large-scale applications

Compare Frameworks

Feature / Aspect	Vue.js	Angular	React.js
Type	JavaScript Framework	JavaScript Framework	JavaScript Library
Initiator	Evan You	Google	Facebook
Learning Curve	Easy, good for beginners	Steep, comprehensive	Moderate, requires understanding of JSX
Popularity	High, growing community	Very popular, well-established	Extremely popular, large community
Performance	Fast, lightweight	Fast, optimized for larger projects	Fast, efficient updates
Architecture	Component-based, MVVM pattern	Component-based, MVC pattern	Component-based
Data Binding	Two-way binding	Two-way binding	One-way binding

Compare Frameworks

Feature / Aspect	Vue.js	Angular	React.js
TypeScript Support	Supported, not required	Strongly encouraged	Supported, not required
State Management	Vuex (official), Pinia	RxJS, NgRx (commonly used)	Redux, Context API (commonly used)
Template Syntax	HTML-based templates	HTML with Angular directives	JSX (JavaScript XML)
Use Case	Small to medium scale applications	Large-scale, enterprise applications	Flexible, suitable for various scales
Community & Ecosystem	Large, growing	Very large, mature	Largest, extensive ecosystem
Documentation	Excellent, easy to understand	Comprehensive, steep learning curve	Good, focused on practical examples

Topics

1. jQuery & Modern Web Frameworks

2. D3.js

3. SVG vs Canvas

d3.js: Data Visualization Powerhouse

- JavaScript library for data visualization
- Manipulate DOM based on data
- Providing basic math & layout algorithms



d3.js: Examples

- <https://ialab.it.monash.edu/webcola/>
- <https://d3js.org/>

Topics

1. jQuery & Modern Web Frameworks

2. D3.js

3. SVG vs Canvas

- XML-based format for 2D graphics
- Supports interactivity and animation
- Images are scalable without loss of quality

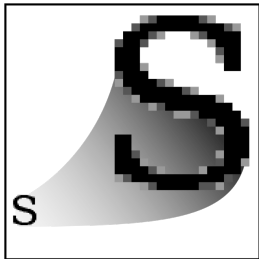
```
<svg width="100" height="100"  
  xmlns="http://www.w3.org/2000/svg">  
  <circle cx="50" cy="50" r="40"  
    stroke="black" stroke-width="3" fill="red" />  
</svg>
```

- HTML element used for drawing graphics via scripting (usually JavaScript)
- Suitable for raster graphics (pixel-based)
- Ideal for complex, real-time graphics (like games)

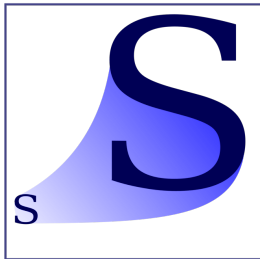
Canvas

```
<canvas id="myCanvas" width="200" height="100"  
        style="border: 1px solid #000000;">  
</canvas>  
  
<script>  
    var canvas = document.getElementById("myCanvas");  
    var ctx = canvas.getContext("2d");  
    ctx.fillStyle = "#0000FF";  
    ctx.fillRect(20, 20, 150, 50);  
</script>
```

SVG vs Canvas



Raster
GIF, JPEG, PNG



Vector
SVG

- The bitmap image is composed of a fixed set of pixels
- The vector image is composed of a fixed set of shapes

- **Scalability:** Perfect for high-quality visuals at any zoom level
- **Accessibility:** Text in SVGs is selectable and searchable
- **Editability:** Easily edited with any text editor
- **SEO Friendly:** Text in SVG can be indexed by search engines

- **Flexibility:** Great for complex, interactive animations and games
- **Performance:** Better for intensive, real-time graphic manipulations
- **Control:** Offers pixel-level control over images
- **Compatibility:** Widely supported across modern browsers