🎯 Node.js 및 모던 개발환경 구축 (2부)

Vite로 차세대 개발 경험하기

강사명: 조성호
소요시간: 30분

📋 7-2교시 학습 목표

- **Vite 빌드 도구** 설치 및 설정
- **Hot Module Replacement** 체험
- **프로덕션 빌드** 및 최적화
- **기존 프로젝트를 Vite로 마이그레이션**

🔗 7-1교시 복습

- Node.js 개념 이해 및 설치 완료
- npm 패키지 관리 시스템 기본 사용법
- 모듈 시스템 (CommonJS vs ES Modules)
- package.json 설정 및 스크립트 활용
- 첫 번째 Node.js 프로젝트 생성

⚡ Vite란 무엇인가?

🎯 정의 및 특징

Vite는 프랑스어로 "빠른"이라는 뜻의 차세대 프론트엔드 빌드 도구입니다.

Vite의 핵심 특징

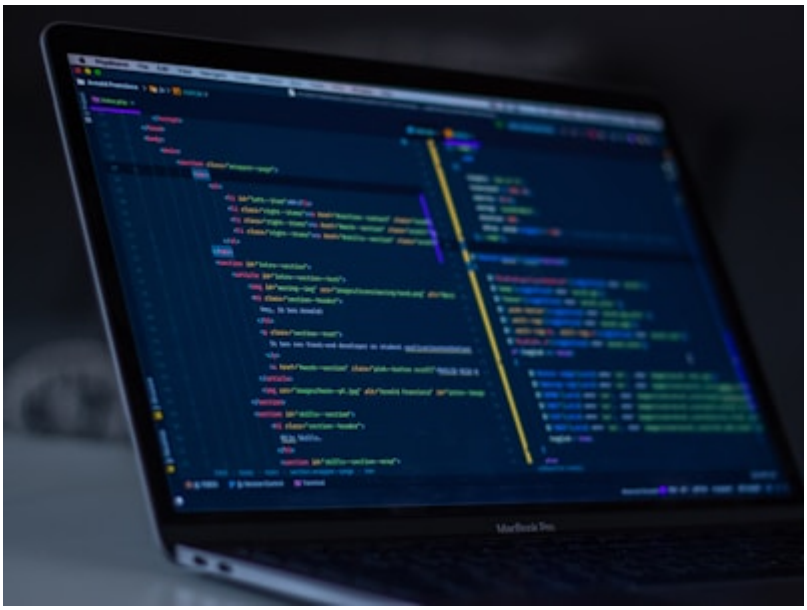
- **번개같은 개발 서버**: 즉시 시작되는 개발 환경
- **Hot Module Replacement**: 실시간 모듈 교체
- **ES Modules 네이티브**: 최신 브라우저 기능 활용
- **TypeScript 기본 지원**: 별도 설정 없이 사용 가능
- **프레임워크 친화적**: React, Vue, Svelte 등 지원

기존 도구와의 성능 비교

```
개발 서버 시작 시간:
Webpack: 15-30초 ⏰
Vite: 1-2초 ⚡

HMR 속도:
Webpack: 500ms-2s
Vite: 50-200ms

빌드 속도:
Webpack: 1-5분
Vite: 30초-2분
```



Vite의 동작 원리

개발 환경에서

```
브라우저 요청
    ↓
Vite 개발 서버
    ↓
ES Modules로 변환
    ↓
브라우저에 즉시 전달
```

프로덕션 환경에서

```
소스 코드
    ↓
Rollup 번들러
    ↓
최적화된 번들
    ↓
배포 준비 완료
```

Hot Module Replacement (HMR)

- **파일 변경 감지**: 저장 즉시 감지
- **모듈 단위 교체**: 전체 페이지 새로고침 없음
- **상태 유지**: 애플리케이션 상태 보존
- **즉시 반영**: 밀리초 단위 업데이트

🛠️ Vite 프로젝트 생성

create-vite로 시작하기

프로젝트 생성

```
# 최신 버전으로 프로젝트 생성
npm create vite@latest my-portfolio

# 또는 특정 템플릿 지정
npm create vite@latest my-portfolio -- --template vanilla

# 선택할 수 있는 템플릿들
# vanilla, vanilla-ts, vue, vue-ts, react, react-ts
```

초기 설정

```
# 프로젝트 폴더로 이동
cd my-portfolio

# 의존성 설치
npm install

# 개발 서버 시작
npm run dev
```

생성된 프로젝트 구조

```
my-portfolio/
├── public/
│   └── vite.svg          # 정적 파일
├── src/
│   ├── main.js           # 진입점
│   ├── style.css         # 메인 스타일
│   └── counter.js        # 예시 모듈
├── index.html            # HTML 템플릿
├── package.json          # 프로젝트 설정
└── vite.config.js        # Vite 설정 (선택적)
```

프로젝트 파일 분석

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite App</title>
</head>
<body>
    <div id="app"></div>
    <!-- 중요: type="module"로 ES 모듈 사용 -->
    <script type="module" src="/src/main.js"></script>
</body>
</html>
```

src/main.js

```
import './style.css'
import javascriptLogo from './javascript.svg'
import viteLogo from '/vite.svg'
import { setupCounter } from './counter.js'

document.querySelector('#app').innerHTML = `
  <div>
    <a href="https://vitejs.dev" target="_blank">
      <img src="${viteLogo}" class="logo" alt="Vite logo" />
    </a>
    <h1>Hello Vite!</h1>
    <div class="card">
      <button id="counter" type="button"></button>
    </div>
```

```
    </div>
`

setupCounter(document.querySelector('#counter'))
```

package.json 스크립트

```
{
  "scripts": {
    "dev": "vite",                    // 개발 서버 시작
    "build": "vite build",            // 프로덕션 빌드
    "preview": "vite preview"         // 빌드 결과 미리보기
  }
}
```

🔧 Vite 설정 커스터마이징

vite.config.js 설정

기본 설정 파일 생성

```
// vite.config.js
import { defineConfig } from 'vite'

export default defineConfig({
  // 개발 서버 설정
  server: {
    port: 3000,           // 포트 번호
    open: true,           // 자동으로 브라우저 열기
    host: true,           // 네트워크 접근 허용
  },

  // 빌드 설정
  build: {
    outDir: 'dist',       // 출력 디렉토리
    assetsDir: 'assets',  // 자산 디렉토리
    sourcemap: true,      // 소스맵 생성
  },

  // 기본 경로 설정
  base: './',             // 상대 경로 사용

  // CSS 설정
  css: {
    devSourcemap: true    // CSS 소스맵
```

```
  }
})
```

환경 변수 설정

.env 파일 생성

```
# .env (모든 환경에서 로드)
VITE_APP_TITLE=My Portfolio
VITE_API_URL=https://api.example.com

# .env.local (로컬 환경, git에서 제외)
VITE_SECRET_KEY=your-secret-key

# .env.development (개발 환경)
VITE_DEBUG=true
VITE_API_URL=http://localhost:8080

# .env.production (프로덕션 환경)
VITE_DEBUG=false
VITE_API_URL=https://api.production.com
```

환경 변수 사용

```
// main.js에서 환경 변수 사용
console.log('앱 제목:', import.meta.env.VITE_APP_TITLE)
console.log('API URL:', import.meta.env.VITE_API_URL)
console.log('개발 모드:', import.meta.env.DEV)
console.log('프로덕션 모드:', import.meta.env.PROD)

// 조건부 로직
if (import.meta.env.DEV) {
  console.log('개발 모드에서만 실행되는 코드')
}

// API 호출 예시
const apiUrl = import.meta.env.VITE_API_URL
fetch(`${apiUrl}/users`)
  .then(response => response.json())
  .then(data => console.log(data))
```

기존 프로젝트 마이그레이션

단계별 마이그레이션 가이드

기존 프로젝트 구조

```
old-portfolio/
├── index.html
├── css/
│   ├── style.css
│   └── responsive.css
├── js/
│   ├── main.js
│   └── utils.js
└── images/
    ├── profile.jpg
    └── project1.png
```

1단계: Vite 초기화

```
# 기존 프로젝트 폴더에서
npm init -y
npm install -D vite
```

2단계: 폴더 구조 재구성

```
# 새 폴더 구조 생성
mkdir src public

# 파일 이동
mv css/* src/
mv js/* src/
mv images/* public/

# 메인 JavaScript 파일명 변경
mv src/main.js src/main.js.bak
touch src/main.js
```

3단계: index.html 수정

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>내 포트폴리오</title>
</head>
<body>
    <div id="app">
```

```
        <!-- 기존 HTML 내용을 여기에 복사 -->
        <header>
            <h1>조성호</h1>
            <nav>
                <a href="#about">About</a>
                <a href="#projects">Projects</a>
                <a href="#contact">Contact</a>
            </nav>
        </header>

        <main>
            <section id="about">
                <h2>About Me</h2>
                <p>프론트엔드 개발자입니다.</p>
            </section>
        </main>
    </div>

    <!-- Vite 진입점 -->
    <script type="module" src="/src/main.js"></script>
</body>
</html>
```

4단계: main.js 재작성

```
// src/main.js
// CSS 파일들 import
import './style.css'
import './responsive.css'

// 유틸리티 함수들 import
import {
  initNavigation,
  initThemeToggle,
  initContactForm
} from './utils.js'

// 기존 JavaScript 코드를 모듈화하여 작성
document.addEventListener('DOMContentLoaded', () => {
  // 네비게이션 초기화
  initNavigation()

  // 테마 토글 초기화
  initThemeToggle()

  // 연락처 폼 초기화
  initContactForm()

  // 기타 초기화 코드들...
})
```

```
// HMR을 위한 코드 (개발 환경에서만)
if (import.meta.hot) {
  import.meta.hot.accept()
}
```

5단계: utils.js 모듈화

```
// src/utils.js
// 네비게이션 관련 함수
export function initNavigation() {
  const navLinks = document.querySelectorAll('nav a')

  navLinks.forEach(link => {
    link.addEventListener('click', (e) => {
      e.preventDefault()
      const targetId = link.getAttribute('href').slice(1)
      const targetElement = document.getElementById(targetId)

      if (targetElement) {
        targetElement.scrollIntoView({ behavior: 'smooth' })
      }
    })
  })
}

// 테마 토글 관련 함수
export function initThemeToggle() {
  const themeToggle = document.querySelector('.theme-toggle')

  if (themeToggle) {
    themeToggle.addEventListener('click', () => {
      document.body.classList.toggle('dark-theme')

      const isDark = document.body.classList.contains('dark-theme')
      localStorage.setItem('theme', isDark ? 'dark' : 'light')
    })
  }

  // 저장된 테마 적용
  const savedTheme = localStorage.getItem('theme')
  if (savedTheme === 'dark') {
    document.body.classList.add('dark-theme')
  }
}

// 연락처 폼 관련 함수
export function initContactForm() {
  const form = document.querySelector('#contact-form')

  if (form) {
```

```
    form.addEventListener('submit', (e) => {
      e.preventDefault()

      const formData = new FormData(form)
      const data = Object.fromEntries(formData)

      console.log('폼 데이터:', data)
      alert('메시지가 전송되었습니다!')
    })
  }
}
```

6단계: package.json 스크립트 추가

```
{
  "name": "my-portfolio",
  "version": "1.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview",
    "clean": "rm -rf dist"
  },
  "devDependencies": {
    "vite": "^4.4.0"
  }
}
```

⚡ HMR (Hot Module Replacement) 체험

HMR 동작 확인

CSS 변경 테스트

```
/* src/style.css */
body {
  background-color: #f0f0f0; /* 이 값을 변경해보세요 */
  font-family: Arial, sans-serif;
}

.header {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  /* 그라데이션 색상을 변경해보세요 */
  color: white;
```

```
  padding: 2rem;
}
```

JavaScript 변경 테스트

```
// src/utils.js
export function updateTime() {
  const timeElement = document.querySelector('#current-time')

  if (timeElement) {
    const now = new Date()
    timeElement.textContent = now.toLocaleTimeString('ko-KR')
    // 이 포맷을 변경해보세요
  }
}

// 1초마다 시간 업데이트
setInterval(updateTime, 1000)
```

HMR API 활용

```
// src/main.js
import { updateTime } from './utils.js'

// HMR 핫 리로드 설정
if (import.meta.hot) {
  // 모듈이 업데이트될 때 실행
  import.meta.hot.accept('./utils.js', (newModule) => {
    console.log('utils.js가 업데이트되었습니다!')
    // 필요한 경우 상태 복원 로직
  })

  // 전체 페이지 리로드 방지
  import.meta.hot.accept()
}
```

🚀 프로덕션 빌드

빌드 프로세스

빌드 실행

```
# 프로덕션 빌드
npm run build
```

```
# 빌드 결과 미리보기
npm run preview

# 빌드 분석 (번들 크기 확인)
npm run build -- --report
```

빌드 결과 구조

```
dist/
├── index.html              # 최적화된 HTML
├── assets/
│   ├── index-abc123.js     # 번들된 JavaScript (해시 포함)
│   ├── index-def456.css    # 번들된 CSS (해시 포함)
│   └── logo-ghi789.svg     # 최적화된 이미지
└── vite.svg               # 정적 파일
```

빌드 최적화 확인

```
// 빌드된 파일들의 특징
// 1. 코드 압축 (Minification)
// 2. 트리 쉐이킹 (사용하지 않는 코드 제거)
// 3. 번들링 (여러 파일을 하나로 합침)
// 4. 해시 추가 (캐싱 최적화)
// 5. 이미지 최적화
```

빌드 최적화 설정

성능 최적화

```
// vite.config.js
export default defineConfig({
  build: {
    // 압축 설정
    minify: 'terser',
    terserOptions: {
      compress: {
        drop_console: true,    // console.log 제거
        drop_debugger: true,   // debugger 제거
      }
    },

    // 청크 분할
    rollupOptions: {
      output: {
```

```
          manualChunks: {
            // 벤더 라이브러리 분리
            vendor: ['lodash', 'axios'],
            // 유틸리티 함수 분리
            utils: ['./src/utils.js']
          }
        }
      },

      // 번들 크기 경고 임계값
      chunkSizeWarningLimit: 1000
    }
  })
```

🎪 7-2교시 마무리

오늘 배운 내용 정리

- **Vite 빌드 도구** 설치 및 설정 완료
- **Hot Module Replacement** 실시간 개발 경험
- **기존 프로젝트 마이그레이션** 완료
- **프로덕션 빌드** 및 최적화 기법
- **모던 개발 워크플로우** 완전 정착

모던 개발환경의 장점 체험

개발 생산성

- ⚡ **즉시 피드백**: 코드 변경 시 실시간 반영
- 🔧 **모듈화**: 체계적인 코드 구조 관리
- 📦 **패키지 관리**: npm으로 라이브러리 손쉽게 활용
- 🛠️ **자동화**: 빌드, 최적화 자동 처리

다음 단계 제안

- TypeScript 도입으로 타입 안정성 확보
- ESLint, Prettier로 코드 품질 관리
- Vitest로 테스트 자동화 구축
- CI/CD 파이프라인으로 배포 자동화


Hey Balázs!
17°
87