**웹 앱 개발**

**\* 배열의 함수형 메소드**

기술의 발전으로 Ajax를 통한 싱글 페이지 작업이 필요해지면서 Forntend에서도 데이터의 복잡한 조작을 필요로 하게 되었다. 그러면서 배열에서도 다양한 메소드를 지원하게 된다. 배열의 forEach, map, filter 등등의 함수에 대하여 검색하여 자세히 알아보자.

**\* 객체 리터럴과 this**

자바스크립트에서 this 키워드는 현재 실행 컨텍스트 정보를 나타낸다. 현재 실행되는 함수가 참조하고 있는 객체의 참조 지점을 알려준다. 아래 예를 보면 this가 healthObj가 된다. 즉 this의 실제 값을 실행 타임에 결정이 된다.

|  |
| --- |
| var healthObj= {  name: “달리기”  showHealth: function() {  console.log(this.name);  }  }  // 현재 showHealt를 호출하는 객체는 healthObj이기 때문에 this는 healObj가 된다.  healthObj.showHealth(); |

좀 더 자세히 말하면, 자바스크립트에는 전역 스크립트나 함수가 실행될 때 실행 영역(context)이 생성되고 실제 실행은 stack 공간에 올라가서 실행된다. 모든 context에는 참조하고 있는 객체가 존재하는데 현재 context가 참조하고 있는 객체를 의미하는 것이 this 키워드이다.

|  |
| --- |
| var others= {  todos: “난 절대로 아무것도 안해”  }  var todo= {  todos: “자바스크립트 공부”  showTodo: function() {  return this.todos;  }  }  todo.showTodo();  // 실행 context의 참조 객체를 todo에서 others로 바꾼다. 때문에 난 절대로.. 가 출력된다.  todo.showTodo.**call(others)**; |

**\* bind 메소드로 this 제어하기**

상황에 따라 this가 변경될 수 있으므로 혼돈하지 않도록 조심해야 한다. 아래 예제를 보면 달라지는 this를 확인할 수 있다.

|  |
| --- |
| var todo= {  todos: “자바스크립트 공부”  showTodo: function() {  setTimeout(function() {  console.log(this.todos);  }, 500);  }  }  // 이 경우, setTimeout이 호출되면서 실행 context가 참조하는 객체가 todo에서 window 객체  // 로 변경되어, todos가 undefined 되게 된다.  todo.showTodo(); |

위와 같은 문제를 해결하기 위해 실행 context가 window 객체가 아닌 todo 객체를 가리킬 수 있도록 bind 함수를 통해 지정해주는 방법이 있다.

|  |
| --- |
| var todo= {  todos: “자바스크립트 공부”  showTodo: function() {  setTimeout(function() {  console.log(this.todos);  }**.bind(this)**, 500);  }  } |

**\* 자바스크립트 라이브러리**

자바스크립트를 개발하는데 지난 10년간 jQuery라는 라이브러리가 큰 인기를 끌었다. 그 이유는 웹브라우저간 API의 차이를 줄여주고, DOM 조작을 더 쉽게 할 수 있게 해주었기 때문이다. 하지만 브라우저간 호환성 이슈가 줄어들고 DOM 접근을 좀 더 추상화된 방식으로 지원하는 자바스크립트 프레임워크들이 등장하면서 그 인기가 많이 줄어들었다.

자바스크립트 프레임워크는 싱글페이지 개념이 등장하면서 필요성이 생기게 되었다. 복잡한 데이터 처리, Ajax와 같은 기능을 사용하면서, 컴포넌트화가 될 필요성이 생겼고 실제 Data조작을 뷰에서 분리하도록 도와주는 프레임워크가 생기게 된다.

**\* Handlebar를 활용한 templating 작업**

복잡한 templating 작업을 쉽게 할 수 있도록 도와주는 라이브러리가 handlebars이다. 사용 방법은 구글링 검색을 통해 알아보자. Java의 velocity 템플릿 엔진과 같이 for, if와 같은 문법까지 지원해준다. 추가적으로 문법을 아래와 같이 직접 커스터마이징 할 수도 있다.

|  |
| --- |
| Handlebars.registerHelper(“likes”, function(likes) {  if(like > 4) {  return “<span>축하합니다. 좋아요가 4개 이상입니다.</span>”;  } else {  return “...”;  }  }); |
| <script type=”myTemplate” id=”listTemplate”>  {{#likes like}}  {{like}}  {{/likes}}  </script>  var template= document.querySelector(“#listtemplate”).innerText;  var bindTemplate= Handlebars.compile(template);  var data= {like: 5};  bindTemplate(data); |