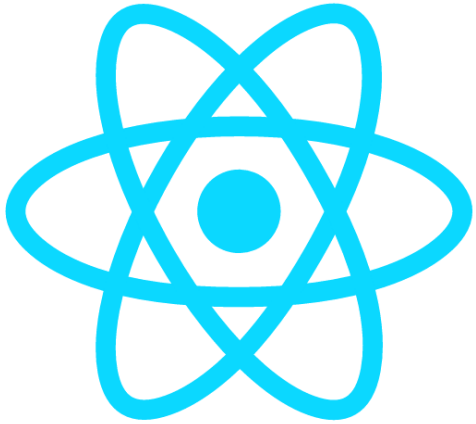
React 기초강좌

@IT KOREA 이정훈

목차

리액트에 대해서 가볍게 알아보도록 하겠습니다.

리액트란?



React

리액트는 2013년 페이스북에서 만든 UI 라이브러리 입니다.
MVC 프레임워크의 V를 담당하는 View를 담당하는 UI라이브러리입니다.

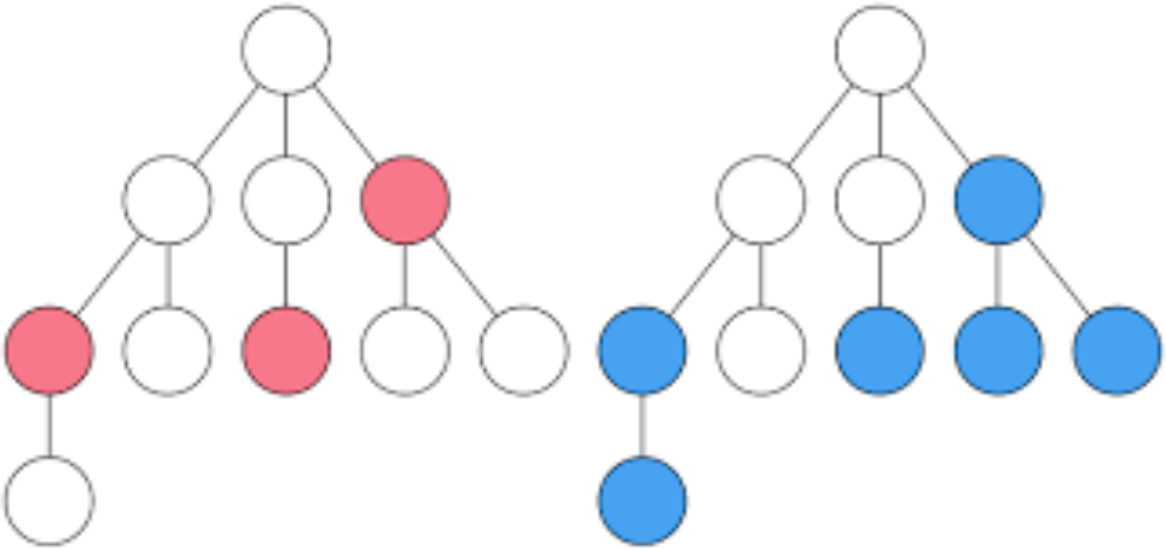
render

1. 초기 렌더링

```
render() {
    const { profiles } = this.state;
    return (
        <div className={profiles ? "App" : "App-loading"}>
            {profiles ? this._renderProfile() : "profiles are being loaded..."}
        </div>
    );
}
```

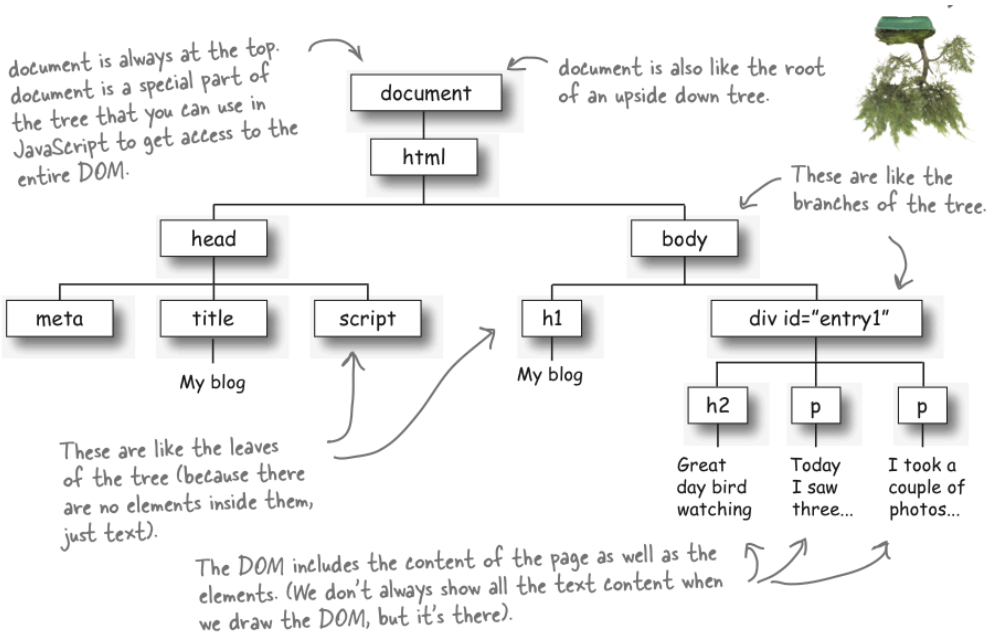
어떤 프레임워크, 라이브러리를 사용하던지간에 어떻게 보일지를 정하는 초기 렌더링이 필요합니다. 리액트에서는 render함수가 있습니다. render함수를 실행하면 렌더링이 끝나고, 가지고 있는 정보들로 HTML마크업을 만들고 이를 DOM에 주입하여 보여줍니다.

2. 조화과정



우리가 뷰를 업데이트할때, '업데이트과정을 거친다' 보다는 '조화과정을 거친다' 라고합니다. 뷰가 변형되는것처럼 보이지만, 새로운 요소를 갈아끼우기때문이죠. 이또한 render가 합니다. render함수는 데이터를 호출하면서 바로 반영하는것이 아닌, 이전에 render 가 만든 정보와 비교를 해서 차이를 최소한의 연산으로 업데이트 합니다.

DOM



Virtual DOM을 보기전에 DOM에 먼저 알아보겠습니다. Document Object Model의 약어로서, 객체로 문서구조를 표현하는 방법입니다. XML 또는 HTML로 작성합니다.
DOM은 느리다라고 하지만, DOM자체는 빠르지만, 웹브라우저에서 CSS를 읽고,레이아웃을 리페인트 하는 과정에서 시간이 걸리는것입니다.

Virtual DOM

1.데이터를 업데이트하면 전체 UI를 Virtual DOM에 리렌더링합니다.

2. 이전 Virtual DOM에 있던 내용과 현재 내용을 비교합니다.

3. 바뀐부분만 실제 DOM에 적용합니다.

위에 보이는 세가지 절차로 DOM에 업데이트됩니다.
Virtual DOM이 언제가 제공해주는것은 업데이트 처리 간결성입니다. UI 업데이트 과정에서 생기는 복잡함을 모두 해소하고, 더욱 쉽게 업데이트에 접근할수있습니다.

주요 ES6 문법

1. import

```
var React = require('react');
var MyComponent = require('./MyComponent')
```

```
import React {Component} from 'react';
import MyComponent from '.MyComponent';
```

2. class

```
function Dog(name){
    this.name = name;
}

Dog.Prototype.say = function(){
    console.log(this.name + ': 멍멍');
}
var dog = new Dog('검둥이');
dog.say(); //검둥이: 멍멍
```

```
class Dog{
    constructor(name){
        this.name = name;
    }

    say(){
        console.log(this.name + ': 멍멍');
    }
}

const dog = new Dog('흰둥이');
dog.say(); //흰둥이: 멍멍
```

JSX

```
var Hello = React.creatClass({
    render: function(){
        return (
            <div>Hello {this.props.name}</div>
        );
    }
});
```

JavaScript와 XML이 합쳐진 독특한 문법입니다.
React는 JSX을 지원함으로, 개발자가 javascript 코드 내부에
마크업 코드를 직접 작성할수있게 해줍니다.

JSX 문법

1. 감싸인 요소, Fragment

2. 자바스크립트 표현

3. if문 대신 조건부 연산자 a ? b : c

4. && 조건부 렌더링

5. 인라인 스타일링

6. class 대신 className

7. 꼭 닫아야하는 태그

컴포넌트

```jsx
import React {Component} from 'react';

class App extends Component{
    render(){
        return(
            <div>
                <h1>리액트 안녕!</h1>
                <h2>당신은 어썸한가요?</h2>
            </div>
        );
    }
}
export default App;
```

```jsx
import React {Component} from 'react';

class App extends Component{
    render(){
        const text = '당신은 어썸한가요?'
        return(
            <Fragment>
                <h1>리액트 안녕!</h1>
                <h2>{text}</h2>
            </Fragment>
        );
    }
}
export default App;
```

React.js에서는 기본적으로 component를 만들고 조합하여 애플리케이션을 구성합니다.
render라는 메소드를 이용하여 작성을 하게 됩니다.

React.creatClass

```
var Hello = React.createClass({
  render() {
    return <div>{this.props.name}</div>;
  }
});

React.render(React.createElement(Hello, {name: "foo"}), document.bo
// or
React.render(React.createFactory(Hello)({name: "foo"}), document.bo

// JSX는 이전과 같은 방식
React.render(<Hello name="foo" />, document.body);
```

createClass는 컴포넌트를 작성할때 사용하는 함수입니다.
마크업과 뷰의 로직을 안에 작성해줍니다.

props

```
//MyComponent.js
import React, {Component} from 'react';

import MyComponent from './MyComponent';

class MyComponent extends Component{
    render(){
        return (
            <div>
                안녕하세요 제 이름은 {this.props.name} 입니다.
            </div>
        )
    }
}
export default MyComponent;


//App.js
import React, {Component} from 'react';
import MyComponent from './MyComponent';

class App exte                t{
    render(){
        return (
            <MyComponent name="React"/>
        );
    }
}
export default App;
```

Definition:
untitled:7

props는 properties를 줄인 표현으로 컴포넌트 속성을 설정할때 사용합니다.
props값은 해당 컴포넌트를 불러와 사용하는 부모 컴포넌트에서만 설정 가능합니다.

defaultProps

```
//MyComponent.js
import React, {Component} from 'react';

import MyComponent from './MyComponent';

class MyComponent extends Component{
    static defaultProps={
        name: '기본 이름'
    }
    render(){
        return (
            <div>
                안녕하세요 제 이름은 {this.props.name} 입니다.
            </div>
        )
    }
}
// 2안
// MyComponent.defaultProps={
//   name: '기본 이름'
// }

export default MyComponent;
```

defaultProps는 props값을 지정하지 않았을때 기본 값으로 설정되는값입니다.

propTypes

```
import React, {Component} from 'react';
import PropTypes from 'prop-types';

class MyComponent extends Component{

    static defaultProps = {
        name: '기본 이름'
    }

    static propTypes = {
        name: PropTypes.string // name props 타입을 문자열로 설정
    }

    render(){
        return (
            <div>
                안녕하세요 제 이름은 {this.props.name} 입니다.
            </div>
        )
    }
}

//2안
// MyComponent.propTypes = {
//  name: PropTypes.string // name props 타입을 문자열로 설정
// }

export default MyComponent;
```

컴포넌트의 필수 props를 지정하거나 props타입을 지정할때는 propType을 사용합니다.
설정하는 방법은 defaultProps와 비슷하며, propTypes를 쓰려면 위에서 불러와야합니다.

필수 propTypes 설정

```
//MyComponent.js

import React, {Component} from 'react';
import PropTypes from 'prop-types';

class MyComponent extends Component{

    static defaultProps = {
        name: '기본 이름'
    }

    static propTypes = {
        name: PropTypes.string, // name props 타입을 문자열로 설정
        age: PropTypes.number.isRequired // 필수적으로 존재해야 하며, 숫자입니다.
    }

    render(){
        return (
            <div>
                <p>안녕하세요 제 이름은 {this.props.name} 입니다.</p>
                <p> 저는 {this.props.age}살 입니다. </p>
            </div>
        )
    }
}

export default MyComponent;
```

propTypes를 설정할 때 뒤에 isRequired를 붙여 주면 됩니다.

state

```
import React, {Component} from 'react';
import PropTypes from 'prop-types';

class MyComponent extends Component{

    static defaultProps = {
        name: '기본 이름'
    }

    static propTypes = {
        name: PropTypes.string,
        age: PropTypes.number.isRequired
    }

    constructor(props){
        super(props);
    }

    render(){
        return (
            <div>
                <p>안녕하세요 제 이름은 {this.props.name} 입니다.</p>
                <p> 저는 {this.props.age}살 입니다. </p>
            </div>
        )
    }
}

export default MyComponent;
```

props는 부모 컴포넌트가 설정하며, 컴포넌트 자신은 읽기전용으로 사용했습니다.
state는 컴포넌트 내부에서 읽고 업데이를 할 수있게 하기위해 사용합니다.
그리고 업데이트는 this.setState() 메소드로만 가능합니다.

초기값 설정 및 렌더링

```
    constructor(props){
        super(props);
        this.state ={
            number: 0
        }
    }

    render(){
        return (
            <div>
                <p>안녕하세요 제 이름은 {this.props.name} 입니다.</p>
                <p> 저는 {this.props.age}살 입니다. </p>
                <p> 저는 {this.state.number}살 입니다. </p>
            </div>
        )
    }
}
```

State 초기값을 설정해주고, 렌더링을 해줍니다.

setState()

```
render(){
    return (
        <div>
            <p>안녕하세요 제 이름은 {this.props.name} 입니다.</p>
            <p> 저는 {this.props.age}살 입니다. </p>
            <p> 숫자: {this.state.number}</p>
            <button onClick={()=>{
                this.setState({ number: this.state.number+1
                })
            }
            }>더하기</button>
        </div>
    )
}
```

이제 state 값을 업데이트 할건데, 이때 this.setState()메소드를 사용합니다.
위에서 ES6 문법인 arrow function을 사용했습니다.

* Arrow function

이벤트 핸들링

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title> JS Bin </title>
</head>
<body>
    <button onclick="alert('excuted')">
        Click Me
    </button>
</body>
</html>
```

유저가 브라우저에서 DOM요소들과 상호작용하는것을 이벤트라고 합니다.
마우스를 올리면 onmouseover 이벤트, 클릭하면 onclick 등 이벤트가 실행됩니다.
HTML에서 이미 해보셔서 익숙하실겁니다.

이벤트 사용시 주의사항

1. 이벤트 이름은 camelCase로 작성

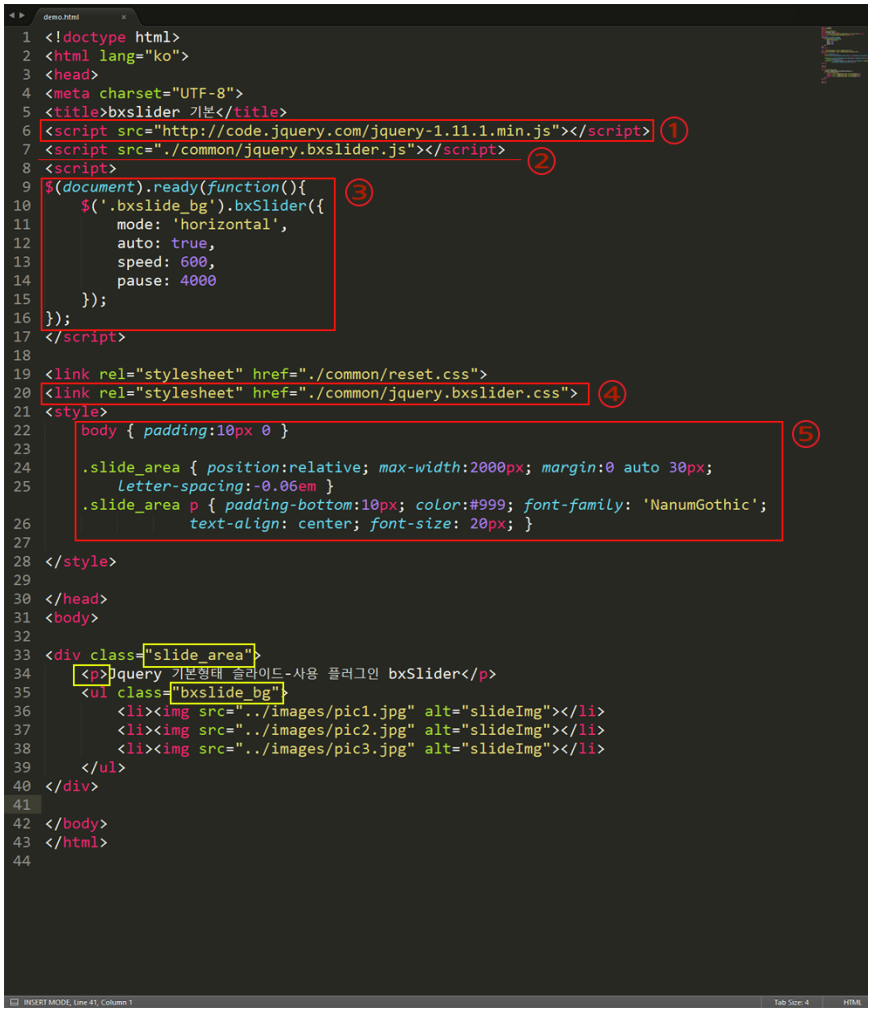
2. 이벤트에 실행할 자바스크립트 코드 전달하는것이 아니라, 함수 형태의 값을 전달합니다.

3. DOM 요소에만 이벤트를 설정 할 수있다.
즉 div, button, input, form, span등 DOM 요소에는 이벤트 설정이 되지만, 직접 만든 컴포넌트에는 설정이 불가능하다.

```
<MyComponent onClick={doSomething}/>

<div onClick={this.props.onClick}>{/* ... */}</div>
```

이벤트 실습


demo.html
1 <!doctype html>
2 <html lang="ko">
3 <head>
4 <meta charset="UTF-8">
5 <title>bxslider 기본</title>
6 <script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
7 <script src="./common/jquery.bxslider.js"></script>
8 <script>
9 $(document).ready(function(){
10 $('.bxslide_bg').bxSlider({
11 mode: 'horizontal',
12 auto: true,
13 speed: 600,
14 pause: 4000
15 });
16 });
17 </script>
18
19 <link rel="stylesheet" href="./common/reset.css">
20 <link rel="stylesheet" href="./common/jquery.bxslider.css">
21 <style>
22 body { padding:10px 0 }
23
24 .slide_area { position:relative; max-width:2000px; margin:0 auto 30px;
25 letter-spacing:-0.06em }
.slide_area p { padding-bottom:10px; color:#999; font-family: 'NanumGothic';
text-align: center; font-size: 20px; }
26
27
28 </style>
29
30 </head>
31 <body>
32
33
34 Jquery 기본형태 슬라이드-사용 플러그인 bxSlider
35 <ul class="bxslide_bg">
36 <img src="../images/pic1.jpg" alt="slideImg">
37 <img src="../images/pic2.jpg" alt="slideImg">
38 <img src="../images/pic3.jpg" alt="slideImg">
39
40
41
42 </body>
43 </html>
44
① ② ③ ④ ⑤
INSERT MODE, Line 41, Column 1
Tab Size: 4
HTML

컴포넌트의 라이프사이클

1. 마운트

2. 업데이트

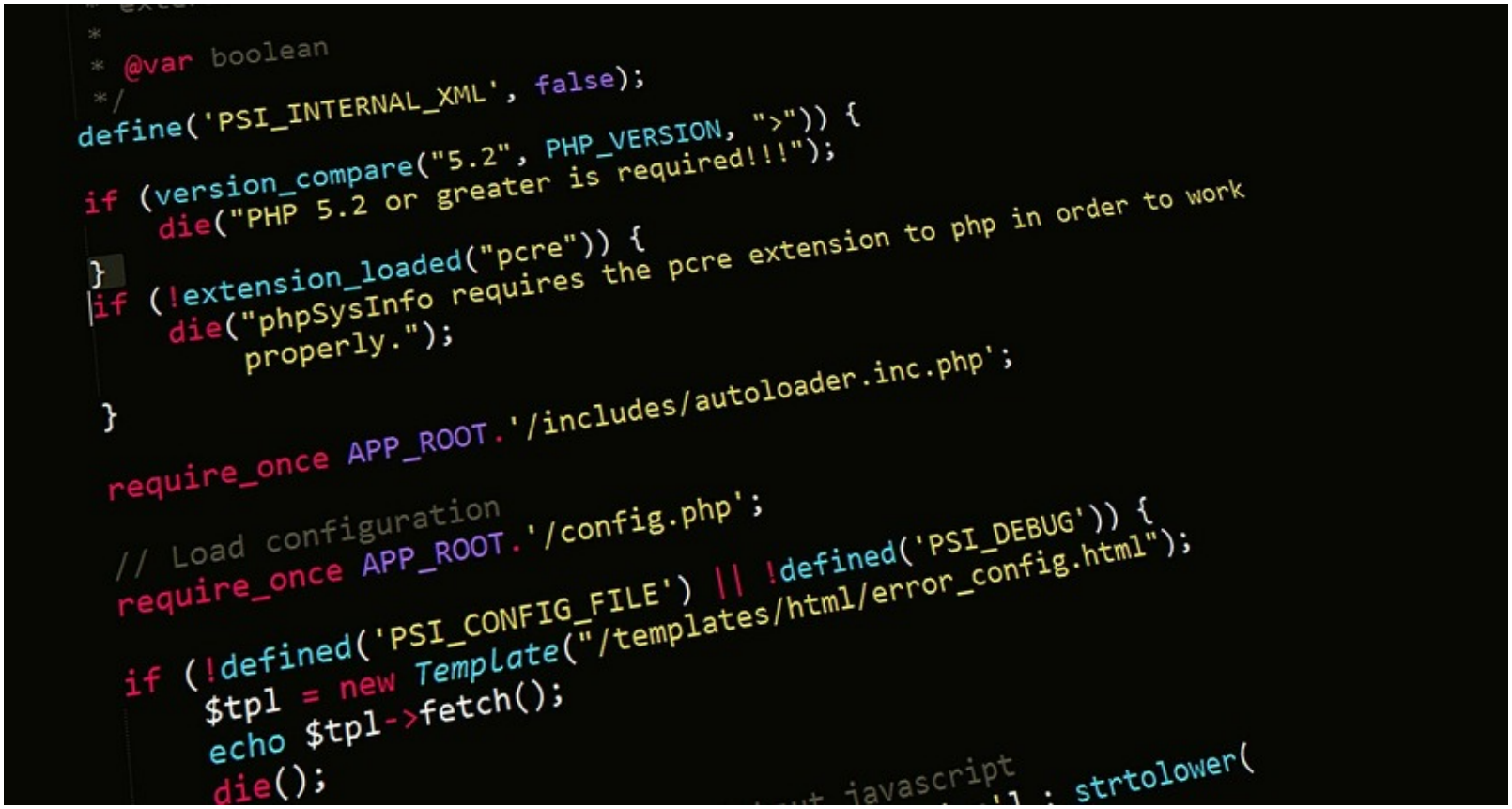
3. 언마운트

라이프사이클은 컴포넌트의 수명주기를 말합니다. 라이프사이클메소드는 총 10가지입니다.
Will 접두사가 붙으면 작동하기전에 실행, Did가 붙으면 작동한 후에 실행되는 메소드입니다.
그리고 라이프 사이클은 마운트,업데이트, 언마운트 세가지 카테고리로 나뉩니다.

라이프 사이클 함수 및 메소드

1. render()
2. constructor
3. getDerivedStateFromProps
4. componentDidMount
5. shouldComponentUpdate
6. getSnapshotBeforeUpdate
7. componentDidUpdate
8. componentWillUnmount

라이프사이클 사용 실습


* @var boolean
*/
define('PSI_INTERNAL_XML', false);
if (version_compare("5.2", PHP_VERSION, ">")) {
die("PHP 5.2 or greater is required!!!");
}
if (!extension_loaded("pcre")) {
die("phpSysInfo requires the pcre extension to php in order to work
properly.");
}
require_once APP_ROOT.'/includes/autoloader.inc.php';
// Load configuration
require_once APP_ROOT.'/config.php';
if (!defined('PSI_CONFIG_FILE') || !defined('PSI_DEBUG')) {
$tpl = new Template("/templates/html/error_config.html");
echo $tpl->fetch();
die();