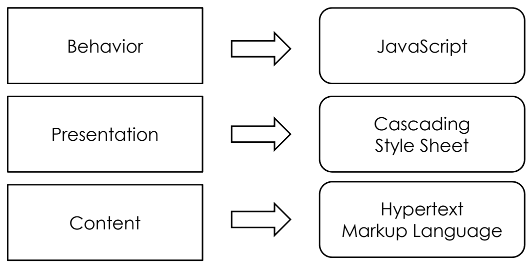
#04 Web Client

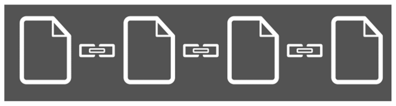
(HTM5, React.js)

CLIENT/SERVER COMPUTING AND WEB TECHNOLOGIES

Web Page Layers



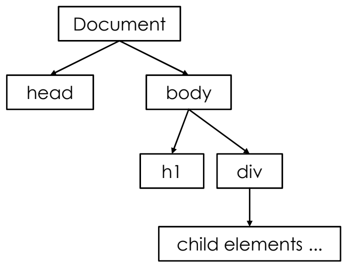
HTML



- Hypertext: A software system that links topics on the screen to related information and graphics, which are typically accessed by a point-and-click method.
- Markup Language: A set of markup tags for grouping and describing page content.

Document Object Model

```
<html>
    <head> </head>
    <body>
        <h1></h1>
        <div> ... </div>
    </body>
</html>
```



Document Hierarchy: Parents, children and siblings

HTML Elements

<tag>Content</tag>

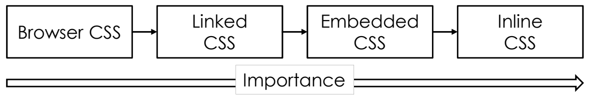
- An HTML element includes both the HTML tag and everything between the tag (the content).
- Tags normally come in pairs. The first tag is the start tag, and the second tag is the end tag.
- HTML has a defined set of tag names (also called keywords) that the browser understands.
- Most elements can have attributes, which provides additional information about the element.
  - <div class="left-nav"></div>

Essential Element Tags

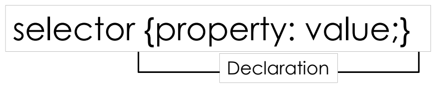
Primary Structure	Structural Elements (block)	Formatting Elements (inline)
`html`	`p`	`em`
`head`	`br`	`i`
`body`	`h1 - h6`	`strong`
	`ul`	`b`
Head Elements	`ol`	`q`
`title`	`a`	`blockquote`
`meta`	`img`	`(span)`
`link`	`(div)`	

CSS

- Stylesheet
  - Rules defining how an html element will be "presented" in the browser.
  - Targeted to specific elements in the html document.
- Cascading
  - Rules for resolving conflicts with multiple CSS rules applied to the same elements.
  - For example, if there are two rules defining the color or your `h1` elements, the rule that comes last in the cascade order will "trump" the other.



CSS Syntax



- Every style is defined by a selector and a declaration. The declaration contains at least one property/value pair.
  - Together they are called a CSS Rule.

```
body {font-family: Arial, Helvetica}
p {color: #666666}
h1 {font-size: 24px}
a {color: blue}
```

CSS Selector

- Type Selector
  - targets an html element by name
- Id Selector
  - An ID is an html attribute added to a html markup.
  - Reference that ID with a hash (#)
    - `#logo { declaration }`
    - `<img id="logo" src="" alt="">`
- Class Selector
  - A class is an html attribute added to a html markup.
  - Reference that ID with a period (.)
    - `.ingredients {declaration}`
    - `<ul class="ingredients">`

p

#

.

JavaScript

- JavaScript as HTML element

```
<script type="text/javascript">
    …
</script>
```

Refer to Chapter #03 for syntaxes.

- JavaScript as external resources

```
<script type="text/javascript" src="e.js"></script>
```

- Purposes
  - Manipulate HTML DOM via document object
    `document.getElementById("logo")...`
  - Handle Event from HTML element
    `<p onclick="do_smth()"> ... </p>`
  - Implement application logics, e.g., form validations

Libraries

http://www.monolinea.com/css-frameworks-comparison/

- CSS Framework
  - Heavyweights: Bootstrap, Foundation
  - Middleweights: Gummy, Groundwork
  - Lightweights: Pure, Base, Kube CSS
- JavaScript Library
  - DOM manipulation, animation, events, HTTP requests
    - jQuery, minified.js
  - Supports: underscore.js, moment.js
- JavaScript Framework
  - jQuery, Dojo, Ember.js, AngularJS, ReactJS, VueJS

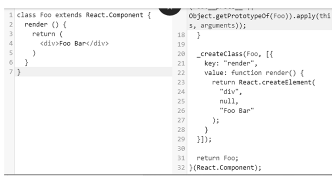
http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks

ReactJS

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

React features

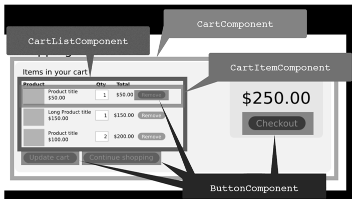
- JSX
  - JavaScript extension
  - Try it: http://babeljs.io/repl



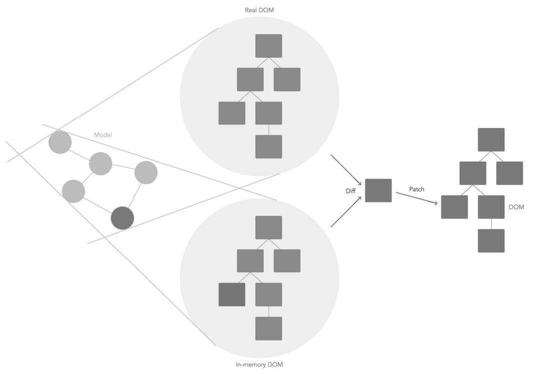
```
class Foo extends React.Component {
  render () {
    return (
      <div>Foo Bar</div>
    )
  }
}
```

```
Object.getPrototypeOf(Foo)).apply(thi
s, arguments));
  }

  _createClass(Foo, [{
    key: "render",
    value: function render() {
      return React.createElement(
        "div",
        null,
        "Foo Bar"
      );
    }
  }]);

  return Foo;
}(React.Component);
```

- Components
  - Reusable, Maintainable, Testable
- The virtual DOM



The virtual DOM



Reference: https://stackoverflow.com/questions/21109361/why-is-reacts-concept-of-virtual-dom-said-to-be-more-performant-than-dirty-mode

Setup

- Softwares
  - node & npm
  - IDE: Web storm, VS Code, Atom, Sublime, vi
- Quick start
  - npm install -g create-react-app
  - create-react-app my-app
  - cd my-app
  - npm start

Reference: https://reactjs.org/tutorial/tutorial.html

React: Start from scratch

- Prepare and create package.json:
  - npm init -y
- Install global package:
  - npm install -g babel babel-cli
  - npm install -g webpack-dev-server
- Add dependencies and plugins:
  - npm install webpack webpack-dev-server --save
  - npm install react react-dom --save
  - npm install babel-core babel-loader --save
  - npm install babel-preset-react babel-preset-es2015 --save

Reference: https://www.tutorialspoint.com/reactjs/reactjs_environment_setup.htm

Compiler, Server and Loaders

- create webpack.config.js

```
var config = {
   entry: './src/index.js',
   output: {
      path:'/',
      filename: 'bundle.js',
   },
   devServer: {
      inline: true,
      port: 8080
   },
   module: {
      loaders: [
         {
            exclude: /node_modules/,
            loader: 'babel-loader',
            query: {
               presets: ['es2015', 'react']
            }
         }
      ]
   }
}
module.exports = config;
```

Compiler, Server and Loaders

- edit package.json

```
"scripts": {
  "start": "webpack-dev-server --hot"
  "test": "echo \"Error: no test specified\" && exit 1"
},
```

index.html

```
<!DOCTYPE html>
<html lang = "en">
<head>
      <meta charset = "UTF-8">
      <title>React App</title>
</head>
       <body>
         <div id = "app"></div>
          <script src = "index.js"></script>
             </body>
</html>
```

app.jsx

```
import React from 'react';

class App extends React.Component
{
   render() {
      return ( <div> Hello World!!! </div> );
   }
}

export default App;
```

main.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './app.jsx';
ReactDOM.render(
        <App />, document.getElementById('app')
);
```

- npm start

Try to modify in app.jsx and check result at browser

Component based

```
import React from 'react';

class  App  extends React.Component {
      render() {
     return(
        <div>
          <Header/>
          <Content/>
        </div>
     );
  }
}

class  Header    extends React.Component {
  render() {
    return(    <div> <h1>Header</h1> </div> );
  }
}

class  Content   extends React.Component {
  render() {
     return(
       <div>
           <h2>Content</h2> <p>The content text!!!</p>
       </div>
     );
  }
}

export default App;
```

In practical, Header and Content should be separately created and exported.

Data passing (props vs. state)

- React has 2 objects of data passing in order to control data into a component
  - Props
    - Pass from parent to child components
    - Immutable
      - **Props CANNOT** be **CHANGED** inside a component
        - Single source of the truth
      - Fixed throughout the component
  - State
    - Reside within component
    - Mutable
      - State **CAN** be **CHANGED**

Props: pass to a component

```
import React, { Component } from 'react';
class Foo extends Component {
  render() {
    return (
        <div> <h1> Foo: {this.props.name} </h1></div>
      )
  }
}

class App extends Component {
  render() {
    return (
      <div>
        <Foo name="FooName" />
      </div>
    );
  }
}

export default App;
```

Define a new property 'name'

State: initial and update

```
class App extends Component {
  constructor(props){
    super(props)
    this.state = { fooState: "Foo State"  }
  }

  render() {
    return (
      <div>
          Message: {this.state.fooState} <br/>
      </div>
    );
  }
}
```

Initial state object

Read state object

State: bind method to context

```
class App extends Component {
  constructor(props){
    super(props)
    this.state = { fooState: "Foo State"  }
    this.updateMessage = this.updateMessage.bind(this)
  }

  updateMessage(e) {
      this.setState( {fooState: "New Foo State: "
                + e.target.value })
  }

  render() {
    return (
      <div>
         <div>
            Message:
             <input type='text' onChange={this.updateMessage}/> <br/>
             {this.state.fooState} <br/>
         </div>
      </div>
    );
  }
}
```

Have to bind method to 'App' context, otherwise a new method will not be known

Define the method to update state

Trig the method

State: automatically bind

```
class App extends Component {
  constructor(props){
    super(props)
    this.state = { fooState: "Foo State"  }
  }

  updateMessage = (e) => {
      this.setState( {fooState: "New Foo State: "  + e.target.value })
  }

  render() {
    return (
      <div>
         <div>
            Message:
             <input type='text' onChange={this.updateMessage}/> <br/>
             {this.state.fooState} <br/>
         </div>
      </div>
    );
  }
}
```

Arrow function binds a method automatically

State: Parent and child component

```
class Foo extends Component {
    render() {
        return (
            <div>
                <h3> Foo: {this.props.name} </h3>
                {this.props.fooState}
            </div>
        )
    }
}

class App extends Component {
        ...

    render() {
        return (
            <div>
                <div>
                    Message:
                    <input type='text' onChange={this.updateMessage}/> <br/>
                    {this.state.fooState} <br/>
                </div>
                <Foo
                    name="FooName" fooState={this.state.fooState}
                    updateMessage={this.updateMessage.bind(this)}
                />
            </div>
        );
    }
}
```

Read ‘state’ as ‘props’

Pass ‘state’ as ‘props’

Update ‘state’ from parent but it affects to child component

React – AJAX Request

PROMISES: AXIOS LIBRARY

HTTP Library: Axios

- Target API: https://api.github.com/users/wwarodom
- Example: axios
  - npm install axios --save

```
import React, { Component } from 'react';
import axios from 'axios';

const USER = 'wwarodom';

class Profile extends Component {

    constructor(props){
        super(props)
        this.state = {  data: {} }
    }

    componentDidMount() {
        axios.get(`https://api.github.com/users/${USER}`)
            .then(response => {
                    this.setState({data: response.data})
                    console.log(response.data)
                }
            )
    }
```

Send Http request

Read object

```
    render() {
        const dataOption = Object.keys(this.state.data)
            .map( (key,index) =>
                <option value={index}>
                    {index+1 +'. ' +key+ ': '  + this.state.data[key]}
                </option>
            )

        return (
            <div>
                    <h2> Github Profile</h2>
                    <ul>
                        <li>{this.state.data.url}</li>
                        <li>{this.state.data.login}</li>
                        <li>{this.state.data['blog']}</li>
                    </ul>

                    <dd><select>{dataOption}</select></dd>
            </div>
        );
    }
}

export default Profile;
```

Pick a value