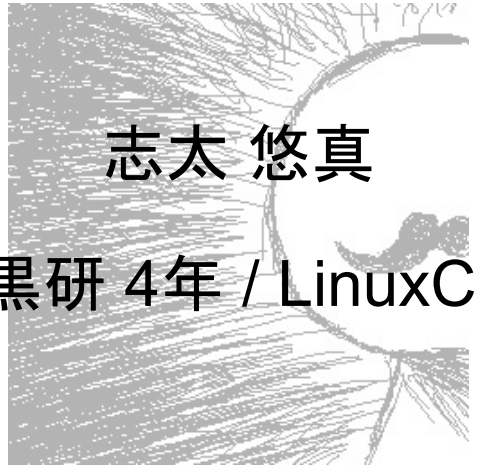
now loading

blanktar.jp

目黒研究室 基礎技術講座

blanktar.jp

講師について



志太 悠真

目黒研 4年 / LinuxClub

m@crat.jp

@mac2rat

この講座について

卒業研究で役に立ちそうな知識をお届けする講座。全3回。

ぶっちゃけ時間が無いのでざっくりやります。
詳しく知りたい方は直接聞いてください。

スライドは補足資料付きで公開してあります。

goo.gl/h8Kp7T

この講座について

卒業研究で役に立ちそうな知識をお届けする講座。全3回。

ぶっちゃけ時間が無いのでざっくりやります。
詳しく知りたい方は直接聞いてください。

スライドは補足資料付きで公開してあります。

goo.gl/h8Kp7T

講座の流れ

- 11月 2日 **プロトタイピングを支える技術**
  卒研で使うかもしれない技術の紹介。

- 11月16日 **デザインからプロトタイピングまで**
  要件を決めて、プロトタイプを作って動かすまでの流れ。

- 12月14日 **最低限の統計学**
  教授にボコボコにされないためのデータの集め方、使い方。

講座の流れ

- 11月 2日 **プロトタイピングを支える技術**
  卒研で使うかもしれない技術の紹介。

- 11月16日 **デザインからプロトタイピングまで**
  要件を決めて、プロトタイプを作って動かすまでの流れ。

- 12月14日 **最低限の統計学**
  教授にボコボコにされないためのデータの集め方、使い方。

目黒研究室 基礎技術講座 第一回

プロトタイピングを支える技術

blanktar.jp

補足

目黒研で卒業する方法

目黒研で卒業する方法

・調査研究をして論文を書く

既存のサービスの評価、市場ニーズや市場規模の調査、etc…

・ビジネスモデルで論文を書く

新しいサービスを試しに作ってみて、評価して論文にする

目黒研で卒業する方法

- 調査研究をして論文を書く

既存のサービスの評価、市場ニーズや市場規模の調査、etc…

- **ビジネスモデルで論文を書く**

新しいサービスを試しに作ってみて、評価して論文にする

目黒研究室 基礎技術講座 第一回

プロトタイピングを支える技術

blanktar.jp

目黒研究室 基礎技術講座 第一回

プロトタイピングを支える技術

blanktar.jp

プロトタイピング

＝

“試しに作ってみる”

プロトタイピング

作ってみるだけなので、なるべく簡単なものを、

なるべく素早く作ることが重要。

プロトタイプ

↓

アルファ版

↓

ベータ版

↓

製品版

プロトタイピングの鉄則

プロトタイピングの鉄則

手を抜く

プロトタイピングを支える技術

サービス提供の選択肢

1. Webサイト
2. アプリ
3. ハードウェア

サービス提供の選択肢

1. Webサイト

よくあるWebサイト。gmail的な高機能なやつも含む。

一番手っ取り早く作れるけれど、出来る事は限定される。

PCやスマホのようなデバイスや、AndroidやiOSといったOSをほとんど気にせず作れるのが最大の強み。

2. アプリ
3. ハードウェア

サービス提供の選択肢

1. Webサイト
2. アプリ

PC向けのソフトを含む。インストールして使うやつ全般。

Webサイトよりちょっと面倒臭いけれど、割と何でも出来る。

デバイスやOSに合わせて作らないといけないが、その分だけ環境に合わせた最適なものを作れるのが強み。

3. ハードウェア

サービス提供の選択肢

1. Webサイト
2. アプリ
3. ハードウェア

電子工作というやつ。別に日曜大工を含めても良いけど。

想像しうる限りありとあらゆるものを作ることが出来るが、
想像しうる限りありとあらゆる知識が必要になってつらい。

デバイス？ OS？ 最適なのを自分で作れば良いじゃない。

1. Webサイト

古典的なWebサイトの仕組み

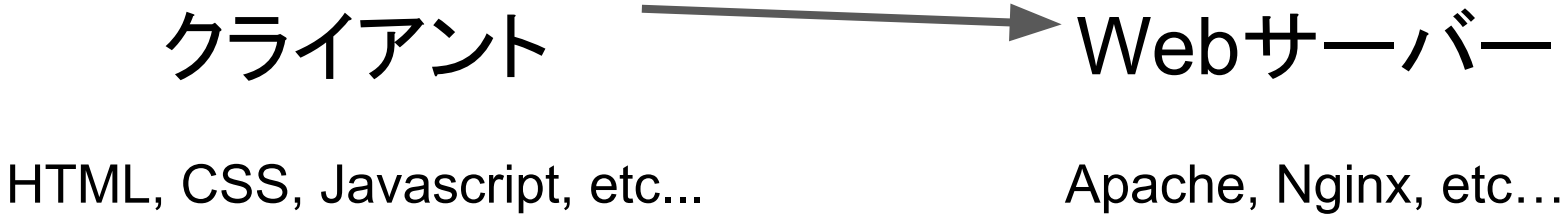
クライアント

HTML, CSS, Javascript, etc...

Webサーバー

Apache, Nginx, etc…

古典的なWebサイトの仕組み



古典的なWebサイトの仕組み

HTML, CSS, Javascript, etc...

Apache, Nginx, etc…

ちょっとモダンなWebサイトの仕組み

クライアント

HTML, CSS, Javascript, etc...

Webアプリサーバー

Python, Ruby, Perl, PHP, etc...

Webサーバー

Apache, Nginx, etc…

データベースサーバー

MySQL, PostgreSQL, MongoDB, etc...

ちょっとモダンなWebサイトの仕組み

Webアプリサーバー

Python, Ruby, Perl, PHP, etc...

クライアント

Webサーバー

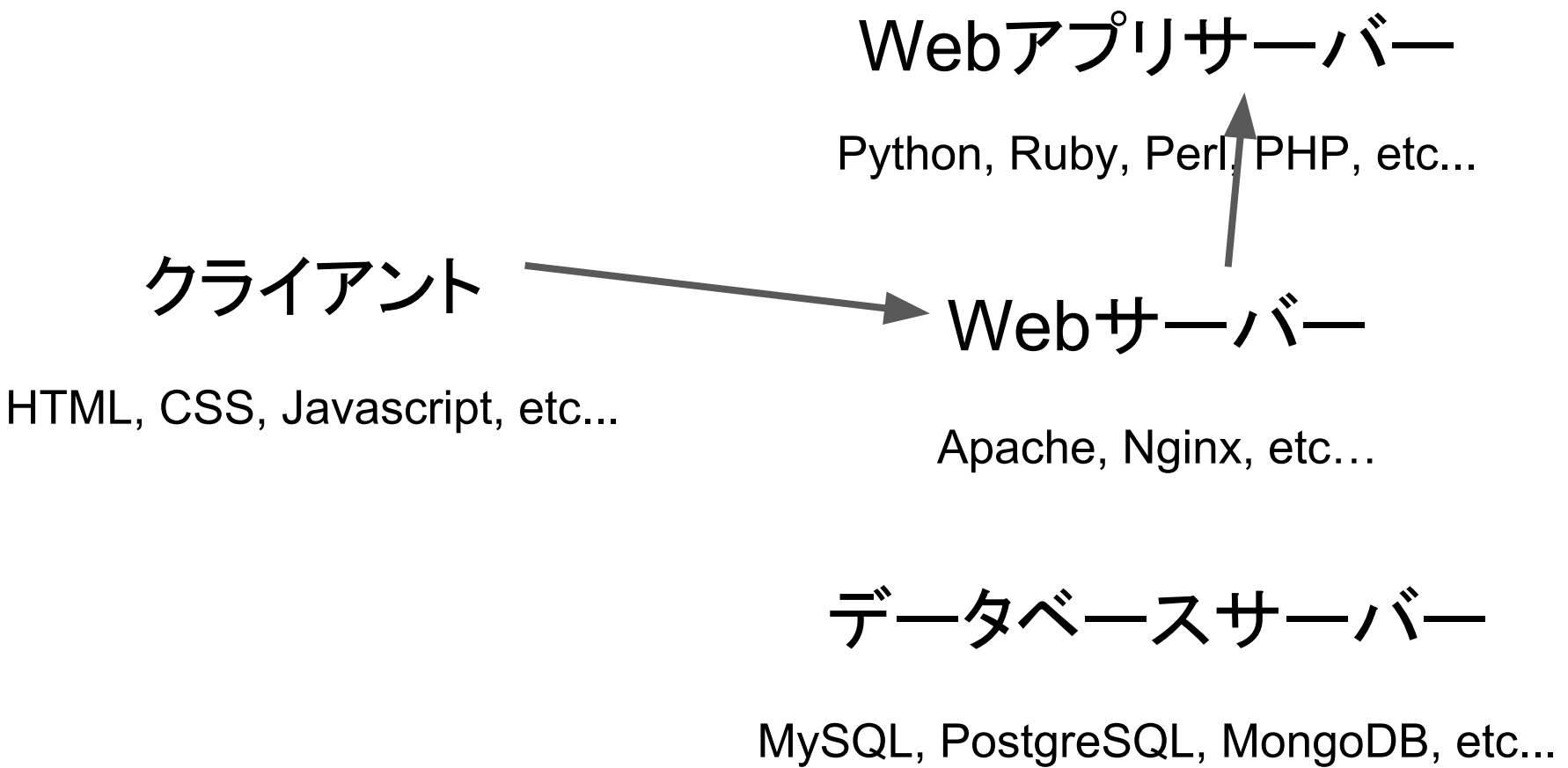
Apache, Nginx, etc…

HTML, CSS, Javascript, etc...

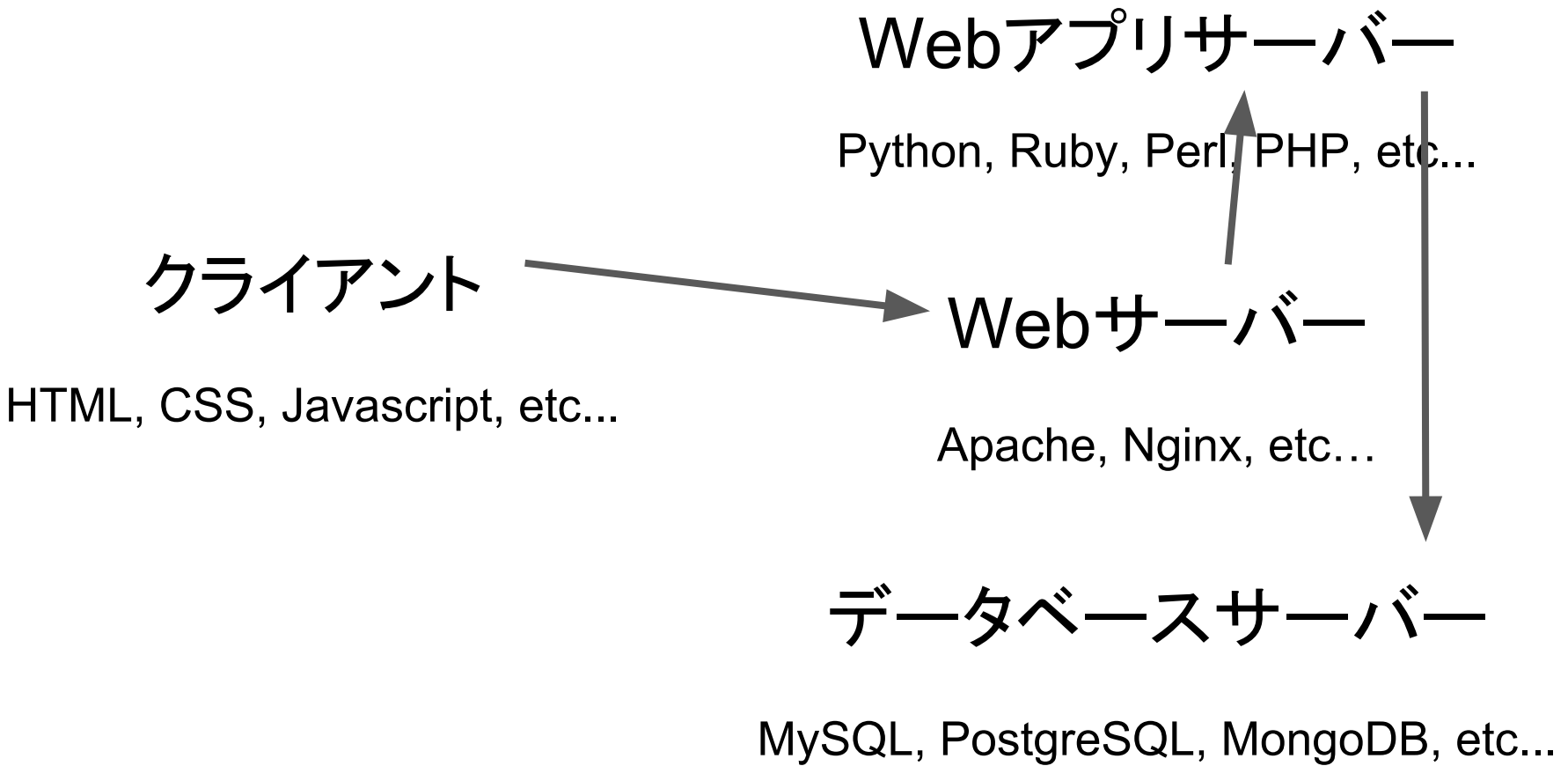
データベースサーバー

MySQL, PostgreSQL, MongoDB, etc...

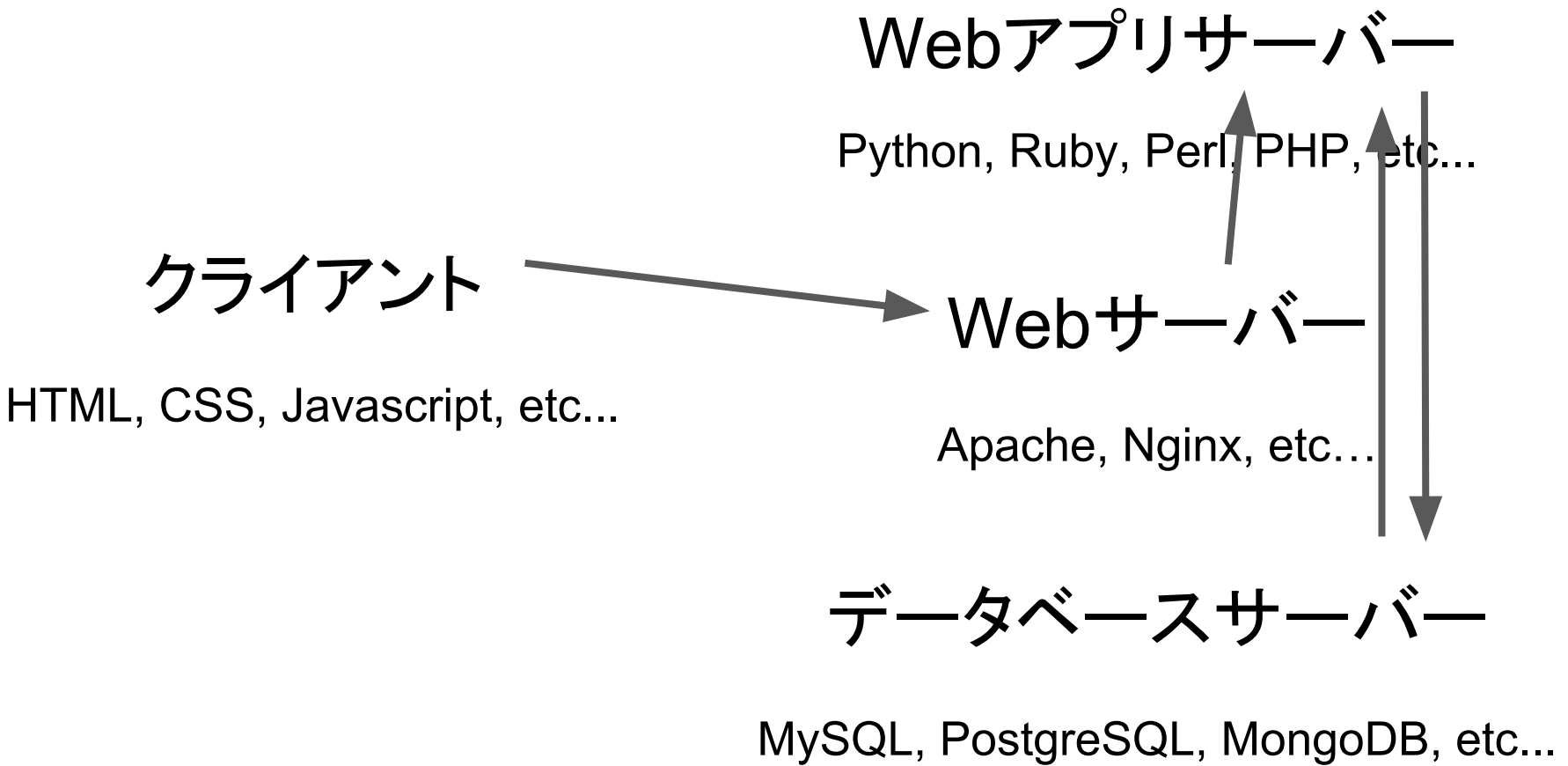
ちょっとモダンなWebサイトの仕組み



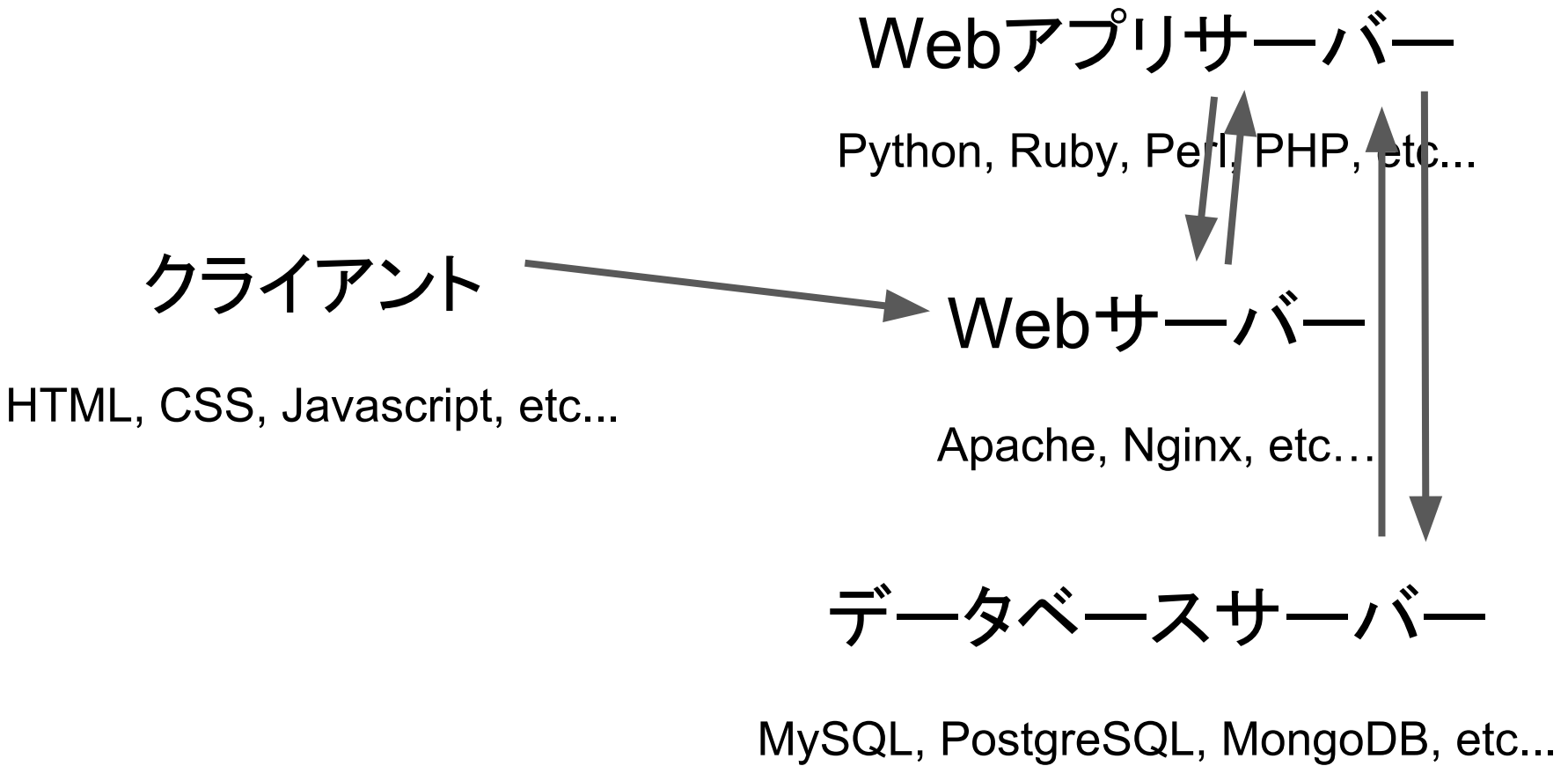
ちょっとモダンなWebサイトの仕組み



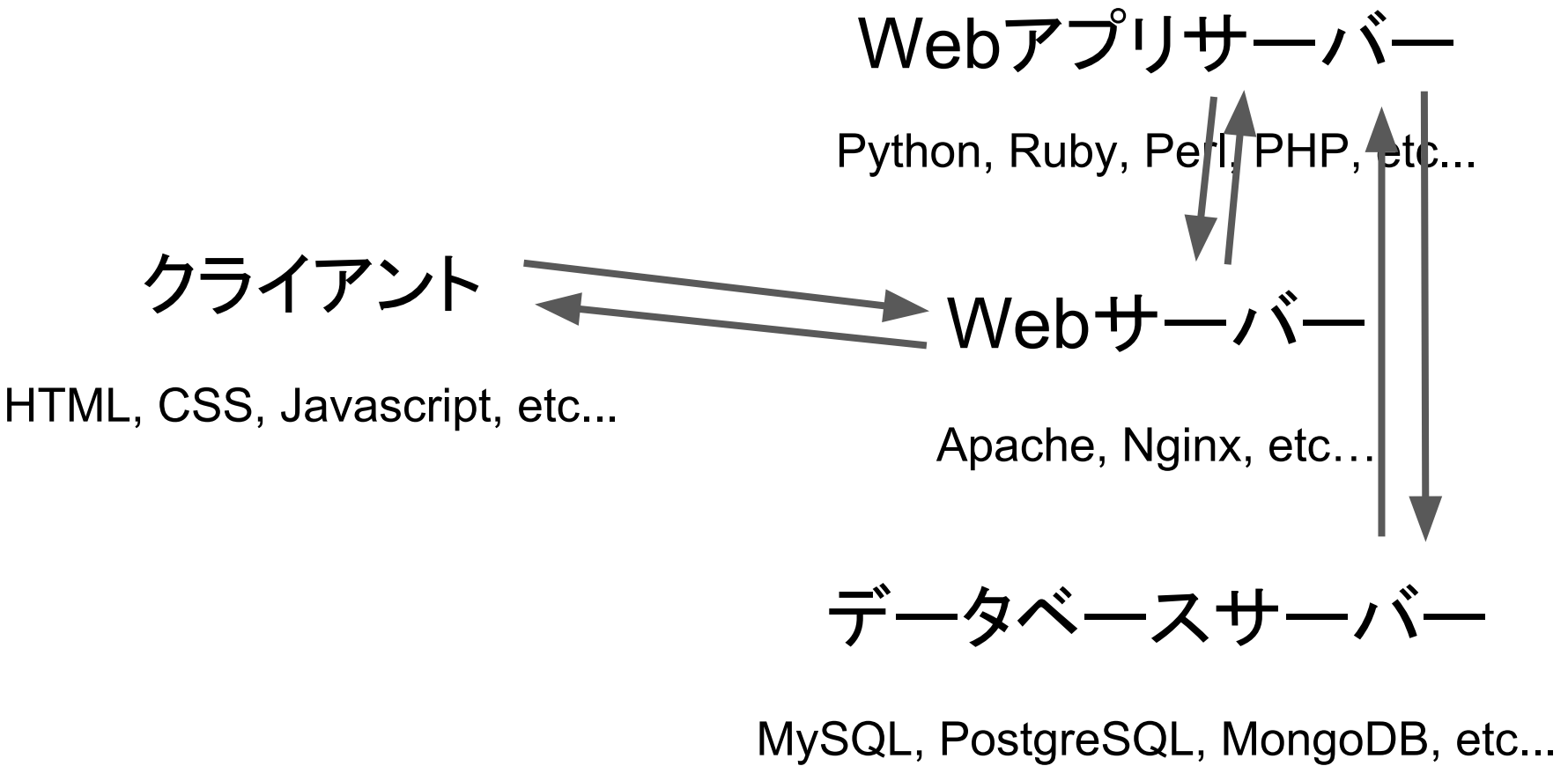
ちょっとモダンなWebサイトの仕組み



ちょっとモダンなWebサイトの仕組み



ちょっとモダンなWebサイトの仕組み



かなりモダンなWebサイトの仕組み

Webアプリケーション

Python, Ruby, Javascript, etc...

クライアント

HTML, CSS, Javascript, etc...

クラウドサービス

Google Cloud Platform,
Amazon Web Service,
Microsoft Azure

かなりモダンなWebサイトの仕組み

Webアプリケーション

Python, Ruby, Javascript, etc...

クライアント

HTML, CSS, Javascript, etc...

クラウドサービス

Google Cloud Platform,
Amazon Web Service,
Microsoft Azure

かなりモダンなWebサイトの仕組み

Webアプリケーション

Python, Ruby, Javascript, etc...

クライアント

HTML, CSS, Javascript, etc...

クラウドサービス

Google Cloud Platform,
Amazon Web Service,
Microsoft Azure

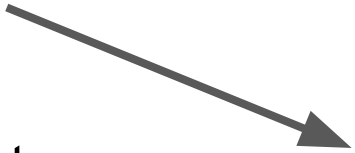
かなりモダンなWebサイトの仕組み

Webアプリケーション

Python, Ruby, Javascript, etc...

クライアント

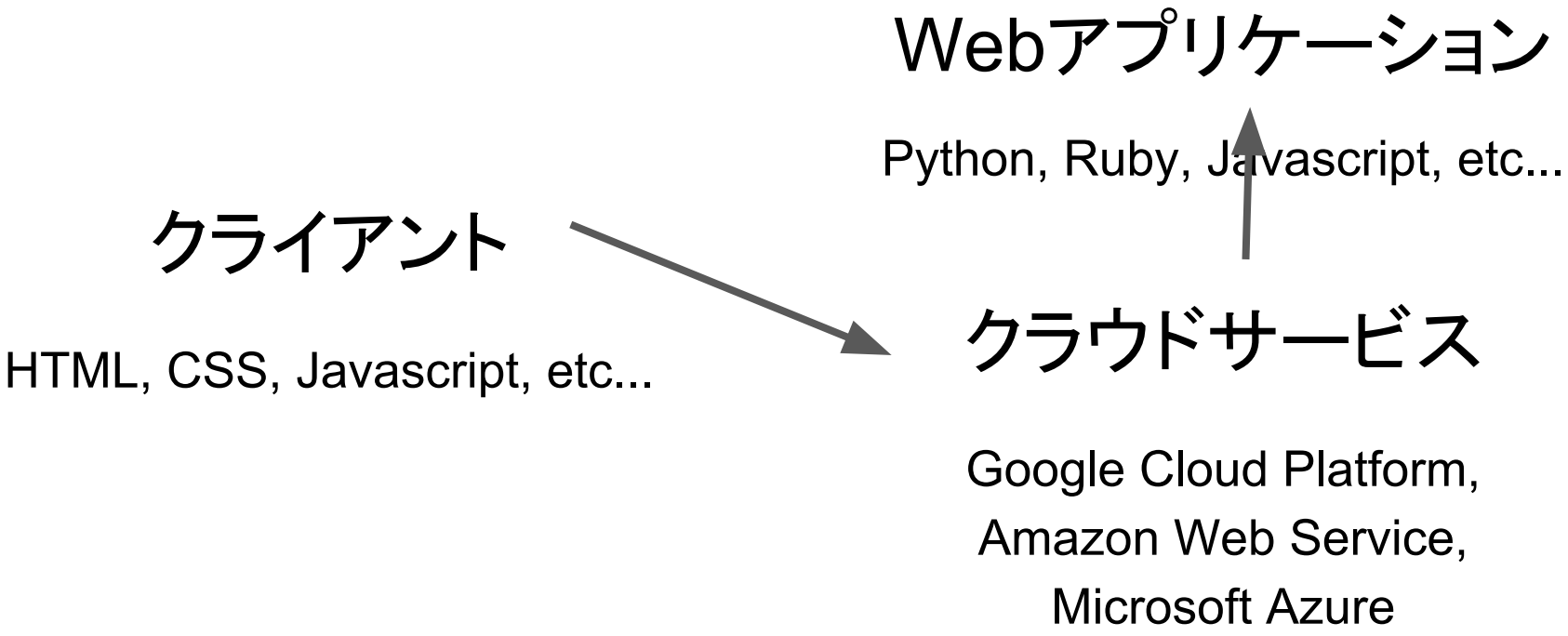
HTML, CSS, Javascript, etc...



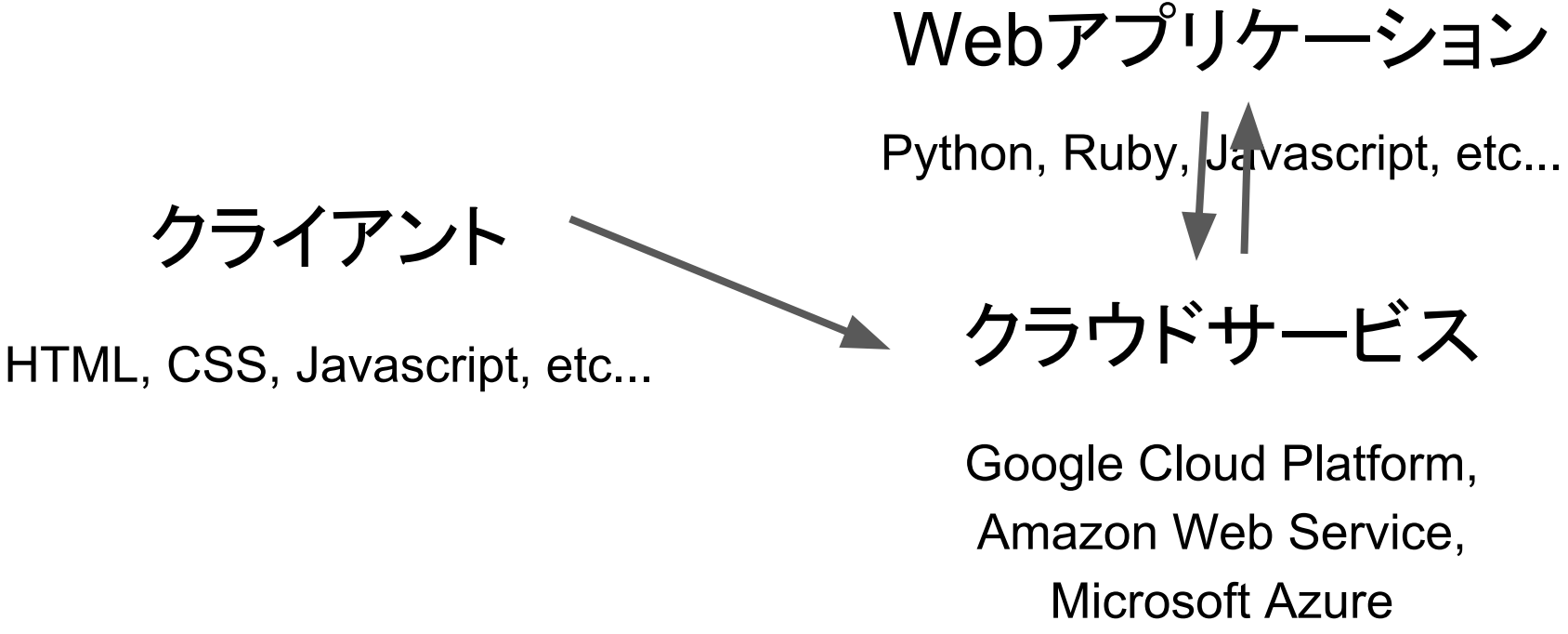
クラウドサービス

Google Cloud Platform,
Amazon Web Service,
Microsoft Azure

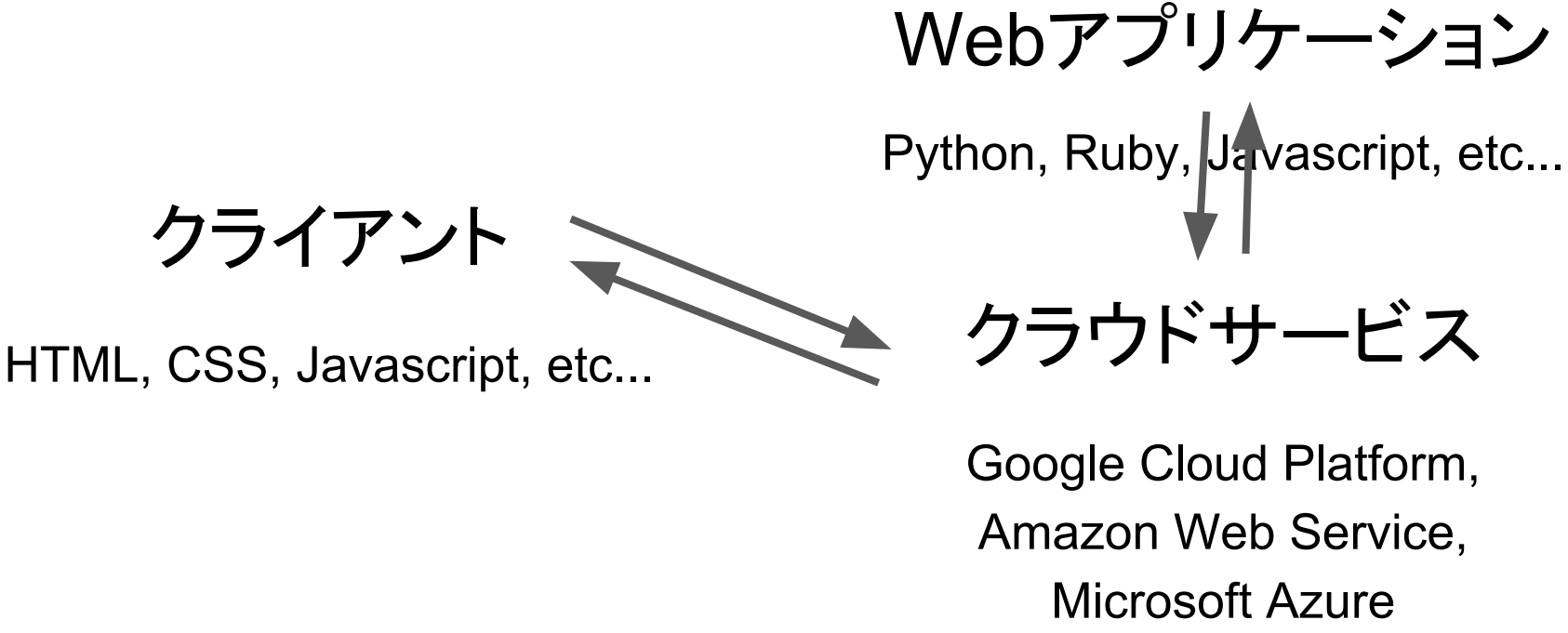
かなりモダンなWebサイトの仕組み



かなりモダンなWebサイトの仕組み



かなりモダンなWebサイトの仕組み



すごくモダンなWebサイトの仕組み

Webアプリケーション
(クライアント)

HTML, CSS, Javascript, etc...

クラウドサービス

Google Cloud Platform,
Amazon Web Service

すごくモダンなWebサイトの仕組み

Webアプリケーション
(クライアント)

HTML, CSS, Javascript, etc...

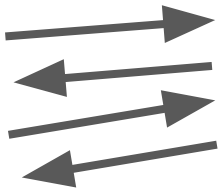
クラウドサービス

Google Cloud Platform,
Amazon Web Service

すごくモダンなWebサイトの仕組み

Webアプリケーション
(クライアント)

HTML, CSS, Javascript, etc...



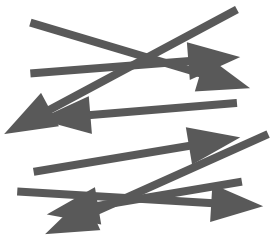
クラウドサービス

Google Cloud Platform,
Amazon Web Service

すごくモダンなWebサイトの仕組み

Webアプリケーション
(クライアント)

HTML, CSS, Javascript, etc...



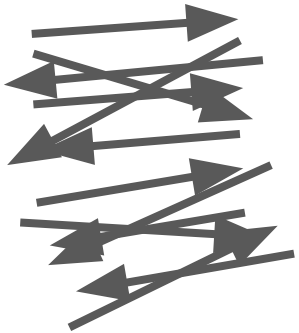
クラウドサービス

Google Cloud Platform,
Amazon Web Service

すごくモダンなWebサイトの仕組み

Webアプリケーション
(クライアント)

HTML, CSS, Javascript, etc...



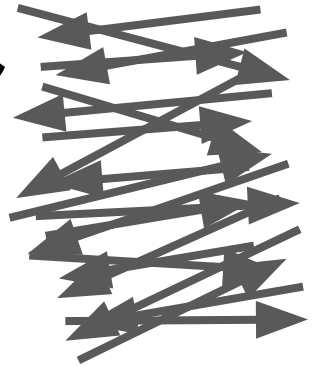
クラウドサービス

Google Cloud Platform,
Amazon Web Service

すごくモダンなWebサイトの仕組み

Webアプリケーション
(クライアント)

HTML, CSS, Javascript, etc...



クラウドサービス

Google Cloud Platform,
Amazon Web Service

かなりモダンなWebサイトの仕組み

Webアプリケーション

Python, Ruby, Javascript, etc...

クライアント

HTML, CSS, Javascript, etc...

クラウドサービス

Google Cloud Platform,
Amazon Web Service,
Microsoft Azure

クライアント

Webアプリケーション

クラウドサービス

クライアント

Webブラウザとかそんなやつの上で動く
コンテンツとかデザインとか。

簡単な動作はクライアントだけでやらせるのが良い。

Webアプリケーション

クラウドサービス

クライアント

Webアプリケーション

サーバー上で動くやつ。

コンテンツを準備したり、複雑な処理をやったりする。

細かいことはやらせない方が後々楽かもしれない。

クラウドサービス

クライアント

Webアプリケーション

クラウドサービス

SaaSとかPaaSとかFaaSとか呼ぶべき。本当は。

Webサーバーや、データベースの管理、Webアプリの実行をユーザーの代わりにやってくれる。

便利な機能が色々あるので、可能な限り使うべき。

クライアント

Webアプリケーション

クラウドサービス

クライアント

Webアプリケーション

クラウドサービス

クライアントに関わるキーワード

Javascript

JQuery

SPA (Single Page Application)

クライアントに関わるキーワード

Javascript

クライアントで何か動かすならほぼ必須。

無理に避けようとすると十中八九地獄を見る。
でも書かなくて済むなら書かない方が幸せ。**手を抜く**べき。

JQuery

SPA (Single Page Application)

クライアントに関わるキーワード

Javascript

```
window.body.onLoad = function() {
  var button = document.querySelector('#button');
  button.addEventListener('click', function() {
    alert('hello world!');
  });
```

JQuery

SPA (Single Page Application)

クライアントに関わるキーワード

Javascript

JQuery

JavascriptでWebサイトを動かすのを楽にしてくれるやつ。

避けて通っても良いけど避けない方が幸せになれる。

手を抜くためには使うべきかもしれない。

SPA (Single Page Application)

クライアントに関わるキーワード

Javascript

JQuery

```
$(function(){
   var button = $('#button');
   button.click(function() {
       alert('hello world!');
   });
});
```

SPA (Single Page Application)

クライアントに関わるキーワード

Javascript

JQuery

SPA (Single Page Application)

Javascriptだけでサイトを全て完結させるイケイケな技術。

あまりにもイケイケなので、余程腕に自信があるか、
ものすごいやる気があるのでなければ触れない方が良い。

クライアント

Webアプリケーション

クラウドサービス

クライアント

Webアプリケーションサーバー

クラウドサービス

Webアプリケーションサーバーに関するキーワード

Python

Ruby

Javascript

Webアプリケーションサーバーに関するキーワード

Python

最近人気のある言語。（感想には個人差があります）

とりあえず作りたいならとりあえず使っとけばという感がある。

Ruby

Javascript

Pythonとゆかいな仲間たち

Flask

手軽にWebアプリを作れるやつ。

複雑なサイトになってくるとややこしくなってくる。
卒研で作る程度なら多分どうということはない。

超おすすめ。

Pythonとゆかいな仲間たち

Django

かなり複雑なWebアプリでも簡単に作れるやつ。

数ページしか作らないならオーバーキルな感じがする。

がっつり作りたい人におすすめ。

Webアプリケーションサーバーに関するキーワード

Python

最近人気のある言語。（感想には個人差があります）

とりあえず作りたいならとりあえず使っとけばという感がある。

Ruby

Javascript

Webアプリケーションサーバーに関するキーワード

Python

Ruby

最近人気がない言語。（感想には個人差があります）

慣れると滅茶苦茶気持ち良く書けるけれど、中々慣れない。

Javascript

Ruby on Rails

さくさくっとWebアプリを作れるやつ（らしい）。

一時期爆発的に流行ったものの、今はどうなったんだろう。

SPAに挑戦するのなら是非これを使うべき（という話を聞いた）

Webアプリケーションサーバーに関するキーワード

Python

Ruby

最近人気がない言語。（感想には個人差があります）

慣れると滅茶苦茶気持ち良く書けるけれど、中々慣れない。

Javascript

Webアプリケーションサーバーに関するキーワード

Python

Ruby

Javascript

最近アツい言語。猫も杓子もJavascript。

元がWeb専用言語だけあって、かなり最適解な言語仕様。
しかし、人類にはまだ早すぎるように思える…。

Javascriptとゆかいな仲間たち

Node.js

サーバーでもJavascript使おうぜ的な
ヤバい発想の元に誕生したモノ。ヤバい。

かなり良い感じに何でも出来るのだけれど、
かなり独特なので学習コストが高い。

がっつりやりたい人におすすめ。

Webアプリケーションサーバーに関するキーワード

Python

Ruby

Javascript

最近アツい言語。猫も杓子もJavascript。

元がWeb専用言語だけあって、かなり最適解な言語仕様。
しかし、人類にはまだ早すぎるように思える…。

クライアント

Webアプリケーション

クラウドサービス

クラウドサービスに関するキーワード

Google Cloud Platform

Amazon Web Service

Microsoft Azure

クラウドサービスに関するキーワード

Google Cloud Platform

Amazon Web Service

Microsoft Azure

クラウドサービスに関するキーワード

Google Cloud Platform

Amazon Web Service

クラウドサービスに関するキーワード

Google Cloud Platform / Amazon Web Service

どっちも大体一緒。

Webアプリケーションサーバー

Google App Engine

Webサービスを簡単に動かすやつ。

サーバーの事は気にせず動かせる。

ちょっとだけ手順が特殊。

Amazon EC2

Webサービス他、何にでも使える
レンタルサーバー。

サーバーソフトは自分で管理する。

ファイルサーバー

Cloud Storage

静的ファイルを保存したりするやつ。

HTTPでそのまま配信も出来る。

かなり安いので動きの少ないデータはなるべくここに。

Amazon S3

これも静的ファイルを保存するやつ。

やっぱりHTTPでそのまま配信出来る。

値段もやっぱりかなり安い。

使い方もほぼ一緒。

データベースサーバー

Google Datastore

Google App Engineのおまけ。

データを手軽に保存出来る。

使い方が特殊なので、設計の段階から
使うかどうか決めておくべき。

Amazon DynamicDB

データを手軽に保存出来る。

これも特殊なので、設計の段階から
使うかどうか決めておくべき。

SQLサーバー

Google Cloud SQL

どうしても一般的なデータベースを
使いたい人向け。

割高で性能が低い。
正直おすすめできない。

Amazon RDS

どうしても一般的なデータベースを
使いたい人向け。

やっぱりあまり幸せではなさそう。

クラウドサービスに関するキーワード

Google Cloud Platform / Amazon Web Service

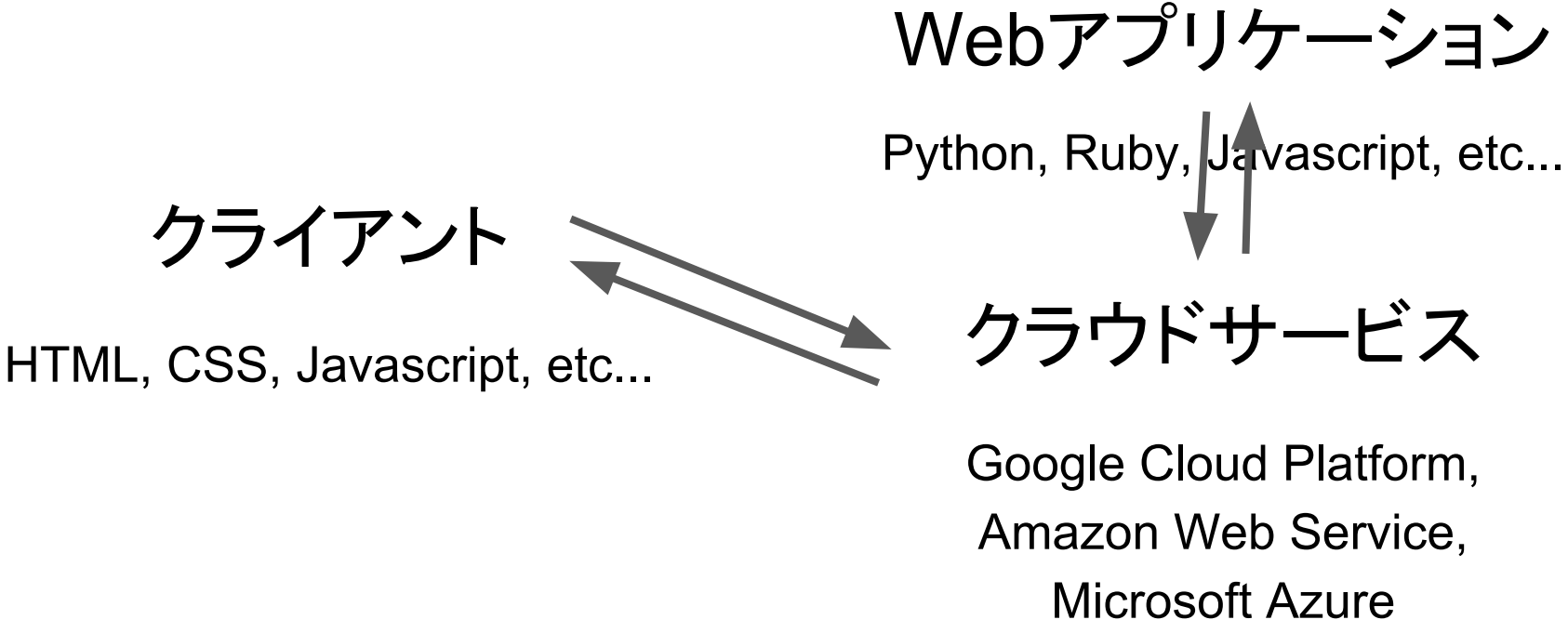
どっちも大体一緒。

クライアント

Webアプリケーション

クラウドサービス

かなりモダンなWebサイトの仕組み



Recommend

Recommend

Jquery
Python / Flask
Google App Engine

この組み合わせが多分一番幸せになれる。

補足

WebサイトとWebアプリの違い

WebサイトとWebアプリの違い

Webサイト / サイト

よくある普通のサイト。

大学のホームページとか個人のブログとかはただのWebサイト。

Webアプリ / Webアプリケーション

仕組みはただのサイトなんだけれど、ちょっと高機能なもの。

GoogleドライブとかTweetDeckとか。

※分類には個人差があります。そんなに重要な話じゃないです。

プロトタイピングを支える技術

サービス提供の選択肢

1. Webサイト
2. アプリ
3. ハードウェア

2. アプリ

アプリを動かす環境

- Android
- iOS
- Windows / mac / Linux / etc...

アプリを動かす環境

- Android

Googleのあれ。

世界では普及率が高い。日本ではイマイチ。

開発も公開もとても楽なのでおすすめ。でも普及率がイマイチ。

- iOS
- Windows / mac / Linux / etc...

アプリを動かす環境

- Android
- iOS

Apple様。

日本での普及率がすごい。世界には打って出れない。

開発から公開までがしんどい。凄くしんどい。でもおしゃれ感。

- Windows / mac / Linux / etc...

アプリを動かす環境

- Android
- iOS
- Windows / mac / Linux / etc...

正直今更作る意味があるのか謎。

スマホと比べると自由度がすごい。

でも作る意味があるのかとても謎。今回はお話しません。

アプリを動かす環境

- Android
- iOS
- Windows / mac / Linux / etc...

Androidで開発する場合

Java

あのJava。みんな大好きJava。

…本当に好き？

Kotlin

最近出てきたモダンなやつ。すごくイケイケで幸せに書ける。

でも学習コストが少し高くて、Javaの知識も結局必要。

みんな大好きJava。

```
class Application extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        findViewById(R.id.button).setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                ((TextView)findViewById(R.id.message)).setText("hello world!");
            }
        });
    }
}
```

Androidで開発する場合

Java

あのJava。みんな大好きJava。

…本当に好き？

Kotlin

最近出てきたモダンなやつ。すごくイケイケで幸せに書ける。

でも学習コストが少し高くて、Javaの知識も結局必要。

しあわせなKotlin

```
class Application: Activity {
    override fun onCreate(savedInstanceState: Bundle) {
        button.setOnClickListener {
            message.text = "hello world!";
        }
    }
}
```

アプリを動かす環境

- Android
- iOS
- Windows / mac / Linux / etc...

iOSで開発する場合

Objective-C

やばいやばいと噂のやばい言語。

マジでヤバい。

Swift

最近出てきたモダンなやつ。恐しくイケイケで簡単に書ける。

でも学習コストが高くて、Objective-Cの知識も結局必要。

やばいぞObjective-C

```
@interface ViewController ()
@end

@implementation ViewController
- (void)viewDidLoad {
    [super viewDidLoad];
    UILabel *label = [[UILabel alloc] init];
    label.text = @"Hello World!";
    [label sizeToFit];
    label.center = self.view.center;
    [self.view addSubview:label];
}

- (void)didReceiveMemoryWarning {
 [super didReceiveMemoryWarning];
}
@end
```

iOSで開発する場合

Objective-C

やばいやばいと噂のやばい言語。

マジでヤバい。

Swift

最近出てきたモダンなやつ。恐しくイケイケで簡単に書ける。

でも学習コストが高くて、Objective-Cの知識も結局必要。

簡単らしいぞSwift

```
class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()

        let label = UILabel(frame: CGRect(x: 100, y: 100, width: 200, height: 50))
        label.text = "Hello World!"
        self.view.addSubview(label)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}
```

アプリを動かす環境

- Android
- iOS
- Windows / mac / Linux / etc...

AndroidとiOSの両方で開発する場合

Xamarin / C#

色んなデバイス向けのアプリをまとめて作れるやつ。

スマホだけでなくPCでも動く。すごい。

簡単なアプリを作るときにはとても幸せになれる。

ややこしい事をしようとすると輪を掛けてややこしくなる。

簡単なことは簡単に出来るXamarin

```
namespace test {
    public class TestPage: ContentPage {
        public TestPage() {
            Button.Clicked += (sender, e) => {
                Message.Text = "hello world!";
            };
        }
    }
}
```

アプリを動かす環境

- Android

Java or **Kotlin** or **Xamarin**

- iOS

Objective-C or **Swift** or **Xamarin**

- Windows / mac / Linux / etc...

Recommend

Recommend

簡単なアプリであればXamarinが幸せ。

（でもそれはWebで良いだろうという話もある）

センサーとか無線とかのハードを扱うなら
Androidが楽で良い。

iOSはやめよう。つらいから。

プロトタイピングを支える技術

サービス提供の選択肢

1. Webサイト
2. アプリ
3. ハードウェア

3. ハードウェア

3. ハードウェア

サービス提供の選択肢

1. Webサイト

← ここにもう一つの選択肢

2. アプリ
3. ハードウェア

PWA

Progressive Web Application

PWA

Webアプリとスマホアプリの中間

PWAとは

Webの手軽さで、アプリのような機能を使える。

PWAという名前のツールとかではなく、色んなものの集合体。
なので、新しく勉強するのはちょっと面倒かもしれない。

Webサイトを作って、それをPWAに変換するようなイメージで
実装すると良いかも。

Recommend

Recommend

上手くやればかなり良いものが出来る。

でも最初からやろうとすると多分すごくつらい。

Recommend

とりあえずWebサイトを作ってみる。

で、それからPWAにするか考える。

と、幸せかもしれない。

プロトタイピングを支える技術

サービス提供の選択肢

1. Webサイト
2. アプリ
3. ハードウェア

3. ハードウェア

ハードなんかやりたいやつ居る？

~~ハードなんかやりたいやつ居る？~~

ハードウェア

計測機器

電子工作

ハードウェア

計測機器

アイトラッキングとかは研究室でも度々話題に出たりする。
Kinectで姿勢取るとかいう話も出る。

どちらも自分でプログラムを掛こうとするとそこそこ大変。

電子工作

ハードウェア

計測機器

電子工作

Arduinoとか使えばそこそこ幸せに作れる。C言語。

性能が足りなければ**mbed**とかが良いかも。C++になる。

それでも足りなければ**RX**とか？ というか工学部に行くべき。

Recommend

Recommend

電子工作はおすすめしない。

計測機器はやりたい事によって無限の選択肢がある。

興味があれば個別に聞いてください。

プロトタイピングを支える技術

サービス提供の選択肢

1. Webサイト
2. アプリ
3. ハードウェア

サービス提供の選択肢

1. Webサイト
2. アプリ
3. ハードウェア

< PWA

では、何で作るか

どうやって決めるべきか

Q1. 普通のスマホ or PC以外で動く？

Q2. 常に起動してなきゃいけない？

Q3. 沢山リソースを使う？

Q4. 頻繁に使うもの？

Q5. 全部NOだった？

Q1. 普通のスマホ or PC以外で動く？

eg. 特殊なセンサーが必要、専用の筐体が必要、etc…

→ YESならハードウェア。さようなら。

Q2. 常に起動してなきゃいけない？

Q3. 沢山リソースを使う？

Q4. 頻繁に使うもの？

Q5. 全部NOだった？

Q1. 普通のスマホ or PC以外で動く？

Q2. 常に起動してなきゃいけない？

eg. スマホのGPSを追跡する、フィットネスデータの記録、etc…

→ YESならアプリ。多分Androidだけにするべき。

Q3. 沢山リソースを使う？

Q4. 頻繁に使うもの？

Q5. 全部NOだった？

Q1. 普通のスマホ or PC以外で動く？

Q2. 常に起動してなきゃいけない？

Q3. 沢山リソースを使う？

eg. 大量の画像を使うゲーム、重い計算をするツール、etc…

→ YESならアプリ。Xamarinでも良いかも。

Q4. 頻繁に使うもの？

Q5. 全部NOだった？

Q1. 普通のスマホ or PC以外で動く？

Q2. 常に起動してなきゃいけない？

Q3. 沢山リソースを使う？

Q4. 頻繁に使うもの？

eg. よく使うツール、毎日やるゲーム、隙あらば見るSNS、etc…

→ YESならWebアプリ。PWAに出来ると幸せ。

Q5. 全部NOだった？

Q1. 普通のスマホ or PC以外で動く？

Q2. 常に起動してなきゃいけない？

Q3. 沢山リソースを使う？

Q4. 頻繁に使うもの？

Q5. 全部NOだった？

eg. 観光や不動産ほかの情報提供、ショッピング、etc…

→ YESならWebサイト。

Q1. 普通のスマホ or PC以外で動く？

Q2. 常に起動してなきゃいけない？

Q3. 沢山リソースを使う？

Q4. 頻繁に使うもの？

Q5. 全部NOだった？

Q1. 普通のスマホ or PC以外で動く？ → ハード

Q2. 常に起動してなきゃいけない？ → Android

Q3. 沢山リソースを使う？ → アプリ

Q4. 頻繁に使うもの？ → Webアプリ

Q5. 全部NOだった？ → Webサイト

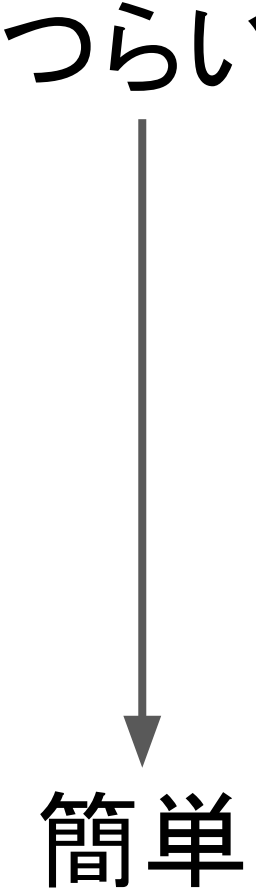
Q1. 普通のスマホ or PC以外で動く？

Q2. 常に起動してなきゃいけない？

Q3. 沢山リソースを使う？

Q4. 頻繁に使うもの？

Q5. 全部NOだった？



Recommend

Recommend

なるべく高いレイヤーでやろう。

なるべく手を抜こう。

目黒研究室 基礎技術講座 第一回

プロトタイピングを支える技術

blanktar.jp

Thank you for listening!

goo.gl/3YRc8d

研究室内での連絡用

4年次では先生からの
連絡もこれを使います。

goo.gl/3YRc8d

IDは何でも良い。

プロフィールの氏名は
必ず登録してください。

goo.gl/3YRc8d

私のIDは @macrat です。

聞きたいことあったら
気軽に聞いてね。

goo.gl/3YRc8d

本日の資料はこっち

goo.gl/h8Kp7T

Slack(連絡網)はこっち

goo.gl/3YRc8d

Thank you for listening!