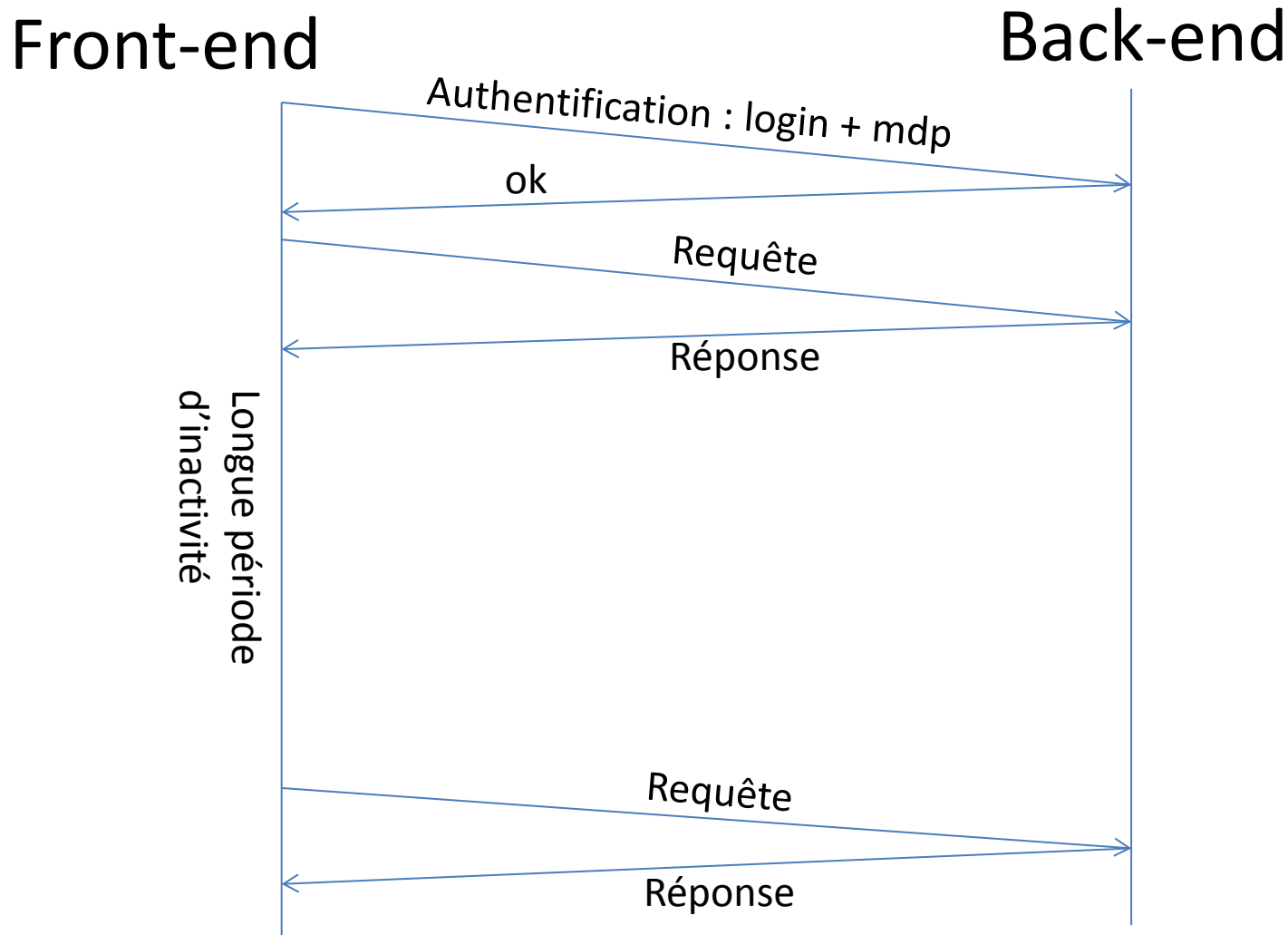


# **L'authentification**

# Authentication

- Consiste à identifier un utilisateur.
- Le cours de SQL vous montre comment retenir un login/mdp.
- Mais comment retenir cet utilisateur à travers les différentes requêtes du front-end ?

# Cycles de vie front-end / back-end



# Le problème de l'authentification...

- Les requêtes du front-end dépendent de l'utilisateur authentifié
  - P.ex. affichage du mur Facebook
- Le serveur connaît cet utilisateur au moment de son authentification
- Les requêtes sont facilement manipulables :
  - Proxy qui en modifie le contenu
  - Appel Ajax bidouillé à la main dans la console
  - ....
- Le serveur doit donc trouver un moyen de retenir l'utilisateur une fois qu'il est authentifié pour répondre aux requêtes ultérieures correctement.

# Session

- L'Application Server (Jetty) va retenir une conversation entre un front-end et un back-end.
  - Une conversation = plusieurs requêtes successives du même front-end. Une telle conversation s'appelle une session.
  - Attention cependant, une conversation n'est pas une connexion TCP : l'application server fait de son mieux pour la tenir aussi exacte que possible, mais peut se tromper.
    - Exemple typique : lors du redémarrage de l'application server, toutes les sessions disparaissent.

# HTTPServlet HttpSession

- Méthode getSession() de l'HttpServletRequest permet d'obtenir la HttpSession.
  - L'instance de Session agit comme une Map<String,Object> via setAttribute et getAttribute.
  - <https://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpSession.html>

# Session

- Lors d'une authentification réussie, la session est remplie par l'id de l'utilisateur :

```
req.getSession().setAttribute("id", idUser);
```

- Lors des requêtes ultérieures, le serveur récupère l'id à partir de la session :

```
Object idUser=req.getSession().getAttribute("id");
```

```
if (idUser!=null) {...}
```

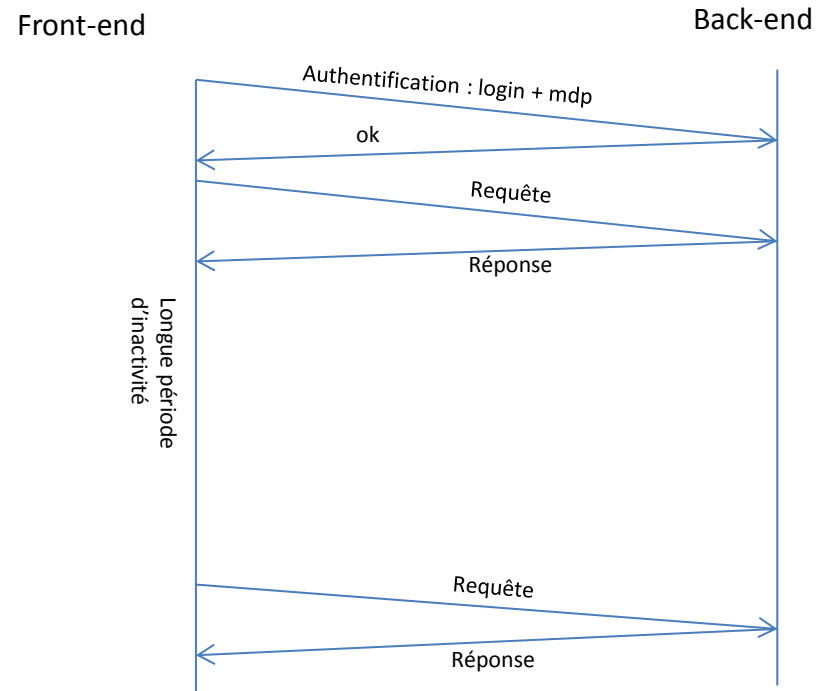
# Inconvénient 1 de la session

- La session utilise de la mémoire pour chaque utilisateur.
  - La mémoire nécessaire au fonctionnement du serveur = taille de la session X le nombre d'utilisateurs connectés.
  - Si le nombre d'utilisateurs connectés est très grand (>>millions), la mémoire nécessaire explose en proportion.



# Inconvénient 2 de la session

- Il peut y avoir de longues périodes d'inactivités du front-end.
- Soit les sessions n'expirent pas, mais alors la mémoire n'est jamais récupérée.
- Soit les sessions expirent, mais alors l'utilisateur devra régulièrement se réauthentifier.



# Inconvénient 3 de la session

- En cas de redémarrage du serveur, toutes les sessions sont perdues.
  - Tous les utilisateurs devront se réauthentifier.

# Statefull

- Ces inconvénients sont dû à une seule chose :
  - Le serveur possède un état dépendant du front-end : l'occupation de la mémoire du serveur dépend du nombre d'utilisateurs connectés.
  - Le front-end peut disparaître sans notifier le serveur (p.ex. en fermant le navigateur) : fuite mémoire au niveau du serveur.
  - Le front-end peut revenir après une longue durée de temps : problème d'expiration de la session.

# Solution : serveur stateless

- stateless = sans état
- Un serveur sans état est un serveur :
  1. Fonctionnel.
  2. La réponse à une requête ne dépend que de la requête elle-même, pas d'un état retenu au serveur.

# Authentication dans un serveur stateless

- Une requête reste toujours trop facilement modifiable.
- Le serveur ne peut faire confiance qu'en lui-même.
  - Comme il ne retient plus l'utilisateur en mémoire : utilisation de la cryptographie.

# Gestion de l'authentification : **JWT**

- JSON Web Token
- C'est une chaîne de caractère qui :
  - Retient les informations d'authentification de l'utilisateur.
  - Est signée cryptographiquement par le serveur.
- Seul le serveur peut générer et valider un JWT valide.

# Référence

- <https://github.com/auth0/java-jwt>
- Création de la chaîne JWT lors d'une authentification réussie :

```
Map<String, Object> claims = new HashMap<String, Object>();  
claims.put("username", email);  
claims.put("id", id);  
claims.put("ip", req.getRemoteAddr());  
String ltoken = new JWTSigner(Config.JWTSecret).sign(claims);
```

- JWTSecret est une constante chaîne de caractère connue uniquement par le serveur.

# JWT et cookies

- La navigateur devra donc fournir la chaîne JWT à chacune de ses requêtes.
- Si on retient le JWT dans un cookie : cela sera automatique :

```
Cookie cookie = new Cookie("user", ltoken);  
cookie.setPath("/");  
cookie.setMaxAge(60 * 60 * 24 * 365);  
resp.addCookie(cookie);
```



# Et lors des requêtes suivantes...

- Lors des requêtes ultérieures, on pourra chercher la chaîne JWT dans les cookies :

```
String token = null;
Cookie[] cookies=req.getCookies();
if (cookies != null) {
    for (Cookie c : cookies) {
        if ("user".equals(c.getName()) && c.getSecure()) {
            token = c.getValue();
        } else if ("user".equals(c.getName()) && token == null){
            token = c.getValue();
        }
    }
}
```

# Validation d'une chaîne JWT

- Et finalement on pourra valider la chaîne JWT avant de récupérer l'utilisateur :

```
Object userID = null;
try {
    Map<String, Object> decodedPayload = new JWTVerifier(JWTSecret).verify(ltoken);
    userID = decodedPayload.get("id");
    if (!remoteHost.equals(decodedPayload.get("ip"))) userID=null;
} catch (Exception exception) {
    // ignore
}

if (userID!=null) { // ici on a pu récupérer un utilisateur valide
    ...
} else { // ici pas : envoi d'une erreur ou redirection vers page d'authentification ou autre...
    ...
}
```

# Ce qu'on met dans JWT

- On peut mettre un id, un nom, une date de péremption, l'IP du front-end, etc...
- On ne met surtout pas le mot de passe de l'utilisateur !
  - Une chaîne JWT ne peut être créée que par le serveur (signée sur base de son secret caché).
  - Sa signature ne peut être validée que par le serveur (toujours sur base de son secret caché).
  - Mais elle est facilement décryptable, le front-end peut en lire le contenu facilement, sans même connaître le secret.

# Cookies

- Les cookies font parties du protocole HTTP.
- Elles sont placées au niveau d'un domaine (=URL), ont un nom et un contenu (String), et peuvent avoir une date de péremption.
- Le front-end et le back-end peuvent ajouter/supprimer/modifier les cookies, tant que le domaine est correct :
  - Une page de <http://localhost> peut placer un cookie sur <http://localhost> ou <http://localhost/chemin> mais pas sur <https://www.google.com>

# Cookies

- Toutes les requêtes HTTP émises par le front-end contiennent tous les cookies placés sur l'URL de la page.
  - Ceci est géré par le navigateur automatiquement.
- <http://www.journaldev.com/1956/servlet-cookie-example-tutorial>

# Authentification avec JWT

- Lorsque l'utilisateur rentre son login/mdp, une requête est envoyée sur une Servlet.
  - Validation du login/mdp
  - En cas d'échec : renvoi d'un message d'erreur
  - En cas de succès : création d'un JWT contenant l'id/login de l'utilisateur et placer cela dans un cookie + renvoi d'un message de succès
- Le front-end reçoit le message de retour et décide quoi faire sur base de ce dernier.

# Une fois authentifié

- Chaque requête contient le cookie avec la chaîne JWT.
  1. Les Servlets valident le JWT et récupèrent le login/id de l'utilisateur.
  2. Ensuite elles effectuent leur traitement habituel.
- Le point 1. prends du temps/demande du CPU pour chaque requête : optimisons cela.

# Session et JWT

- Valider la signature d'un JWT prends du temps : on va le faire une fois par session uniquement !
- Le processus d'une Servlet devient donc :
  - Est-ce que la session est authentifiée => ok on peut traiter la requête.
  - Sinon, on prends la chaîne JWT et on valide la signature => on retient cette authentification dans la session pour la fois prochaine et on peut traiter la demande.
  - Sinon, on redirige vers la page d'authentification.



# **L'orienté objet en Javascript**

# Orientation Objet en JS

- Nous avons esquivé l'OO jusqu'ici en JS car :
  - Les pseudo-objets et pseudo-classes fonctionnent déjà très bien
  - L'OO de JS est très différent de Java
- Nous allons survoler l'OO JS :
  - Pour votre culture personnelle, on ne vous demande pas de l'utiliser dans le cadre de ce cours et on ne vous évaluera pas dessus non plus.

# Rappel pseudo-classe

```
var createPerson=function(name,surname) {  
    var age,address;  
    function setAddress(a) { address=a;}  
    function setAge(a) { age=a;}  
    var self={  
        getName:function() {return name;},  
        getSurname:function() {return surname;},  
        getAge:function() {return age;},  
        getAddress:function() {return address;},  
        setAge:setAge,  
        setAddress:setAddress  
    }  
    return self;  
};  
var john=createPerson("John", "Snow");
```

# En réel OO

```
function Person(name,surname) {  
    var age,address;  
    function setAddress(a) { address=a;}  
    function setAge(a) { age=a;}  
    this.getName=function() {return name;};  
    this.getSurname=function() {return surname;};  
    this.getAge=function() {return age;};  
    this.getAddress=function() {return address;};  
    this.setAge=setAge;  
    this.setAddress=setAddress;  
};  
var john=new Person("John", "Snow");
```

```
    this.setAge=setAge;  
    this.setAddress=setAddress;  
};  
var john=new Person("John", "Snow");
```

- L'utilisation de new crée un this lors de l'exécution de la fonction.
  - this et new sont des mots-clefs du langage.
  - L'appel d'une fonction avec un new renvoie son this.
  - Le this est un objet-associatif : c'est l'instance.
  - La fonction sert de constructeur à cette instance.
  - Elle doit aussi définir les propriétés de ce this, notamment les fonctions (~ méthodes) utilisables.

# Prototype

On peut aussi définir Person comme ceci:

```
function Person(name,surname) {  
    this.name=name; this.surname=surname;  
}  
Person.prototype.setAddress=function(a) { this.address=a;}  
Person.prototype.setAge=function(a) { this.age=a;}  
Person.prototype.getName=function() {return this.name;};  
Person.prototype.getSurname=function() {return  
this.surname;};  
Person.prototype.getAge=function() {return this.age;};  
Person.prototype.getAddress=function() {return  
this.address;};  
};  
var john=new Person("John", "Snow");
```

# Prototype

```
Person.prototype.getAge=function() {return this.age;};  
var john=new Person("John", "Snow");
```

- john.getAge() continue à renvoyer son âge !
- Notez que this.age est accessible via john.age directement.
- Principe du prototype lors d'un accès à une propriété :
  - Si on ne trouve pas la propriété directement sur l'objet, alors JS cherche la propriété sur le prototype de la fonction pour l'appeler.

# Héritage par prototype

```
// définition de Lanister qui hérite de Person :  
Lanister.prototype=new Person("Lanister","");  
// redirection du constructeur par défaut (sinon celui de  
Person est utilisé) :  
Lanister.prototype.constructor=function(surname) {  
    this.surname=surname;  
}
```

<http://phrogz.net/js/classes/OOPinJS2.html>



# **L'écosystème Javascript**

# Ecosystème JS, c'est quoi ?

- C'est l'ensemble des librairies, techniques et technologies autour de Javascript
- Très riche, probablement le plus actif actuellement, principalement grâce à Internet.
  - Et pas seulement au front-end : Node.js est un back-end en Javascript !
  - Grosse croissance dans le monde du mobile.

# Suite du cours...

- Nous allons brièvement aborder certaines parties de cet écosystème.
  - Pour votre culture personnelle, ne fait pas partie de la matière du cours qui sera évaluée pour cette activité d'apprentissage.
- Profitez du projet pour les découvrir.
  - Ce dernier n'est pas évalué : profitez-en pour être audacieux et découvrir un maximum possible.
  - L'examen sera un mini-projet simplifié.

# Bootstrap

- Le web moderne = ordinateur de bureau, tablette, mobile, etc...
- Chaque appareil a une taille d'écran spécifique et une interaction spécifique aussi.
- Une seule adresse pour fournir différents contenus ?

# User-agent

- Dans les détails d'une requête, l'identification du navigateur est fournie : le user-agent.
    - Le back-end peut discriminer et servir un contenu différent et adapté.
- => il faut donc développer plusieurs sites Web en fonction des différents types d'appareils fournis.

# Responsive design

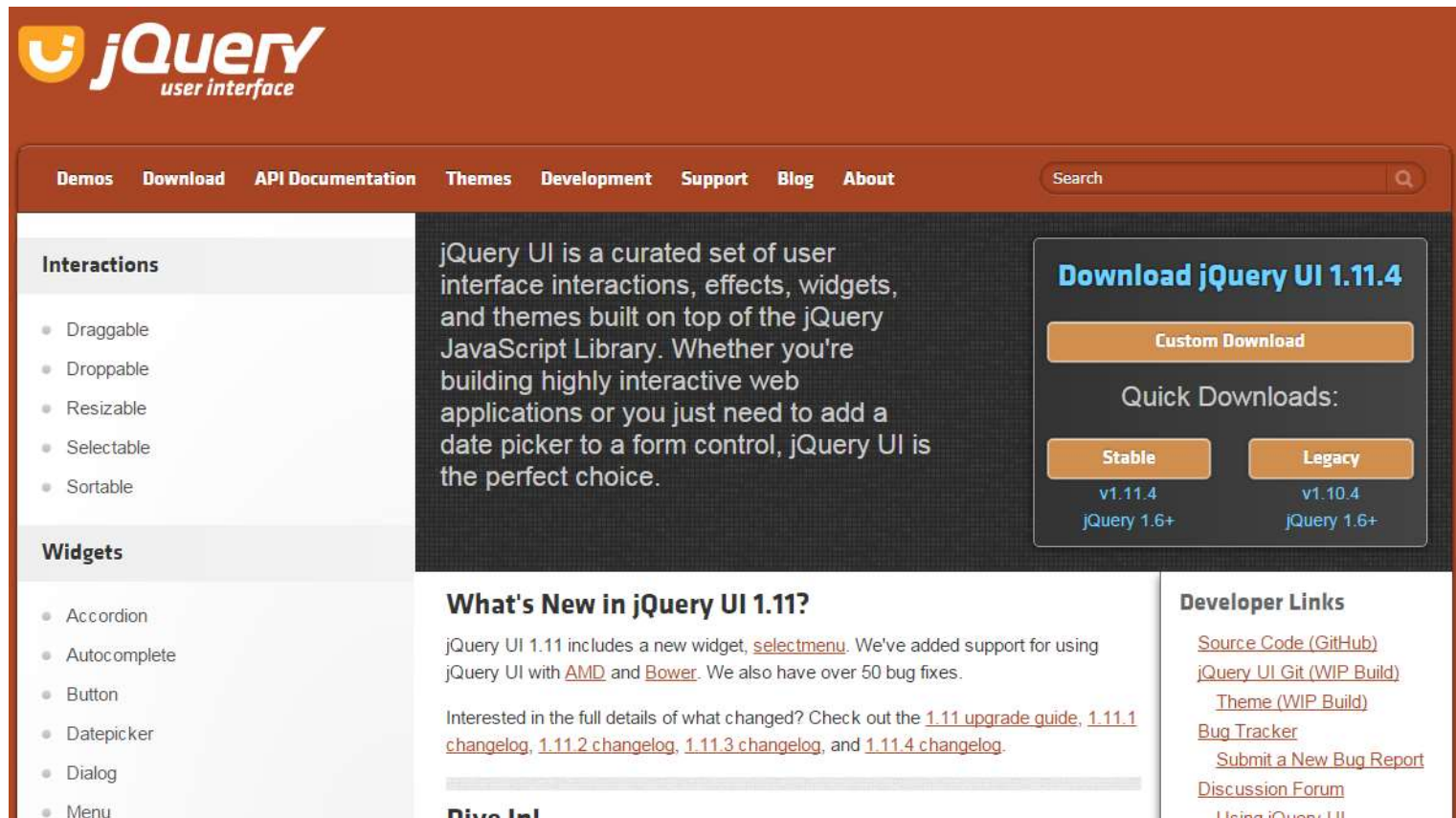
- Approche différente : un seul contenu qui s'adapte au mieux automatiquement.
- Bootstrap est le plus connu, créé par Twitter.
- <http://www.w3schools.com/bootstrap/>

# Bootstrap

- Composé d'un .js et d'un .css
- Requiert d'utiliser des classes spécifiques pour les composants HTML.
- L'adaptation est alors automatique.

# jQuery UI

- jQuery = boîte à outils de base pour manipuler le DOM.
- jQuery UI = composants graphiques riches.



The screenshot shows the jQuery UI website with a dark orange header. The logo 'jQuery user interface' is in the top left. A navigation bar contains links: Demos, Download, API Documentation, Themes, Development, Support, Blog, and About. A search bar is on the right. The main content area has a left sidebar with 'Interactions' (Draggable, Droppable, Resizable, Selectable, Sortable) and 'Widgets' (Accordion, Autocomplete, Button, Datepicker, Dialog, Menu). The main text describes jQuery UI as a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. A 'Download jQuery UI 1.11.4' section offers a 'Custom Download' button and 'Quick Downloads' for 'Stable' (v1.11.4, jQuery 1.6+) and 'Legacy' (v1.10.4, jQuery 1.6+). Below this, 'What's New in jQuery UI 1.11?' mentions a new 'selectmenu' widget and support for AMD and Bower. A 'Developer Links' section provides links to Source Code (GitHub), jQuery UI Git (WIP Build), Theme (WIP Build), Bug Tracker, Submit a New Bug Report, Discussion Forum, and Using jQuery UI.

**jQuery**  
user interface

[Demos](#) [Download](#) [API Documentation](#) [Themes](#) [Development](#) [Support](#) [Blog](#) [About](#)

**Interactions**

- Draggable
- Droppable
- Resizable
- Selectable
- Sortable

**Widgets**

- Accordion
- Autocomplete
- Button
- Datepicker
- Dialog
- Menu

jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice.

**Download jQuery UI 1.11.4**

[Custom Download](#)

Quick Downloads:

[Stable](#) [Legacy](#)

v1.11.4 v1.10.4  
jQuery 1.6+ jQuery 1.6+

**What's New in jQuery UI 1.11?**

jQuery UI 1.11 includes a new widget, [selectmenu](#). We've added support for using jQuery UI with [AMD](#) and [Bower](#). We also have over 50 bug fixes.

Interested in the full details of what changed? Check out the [1.11 upgrade guide](#), [1.11.1 changelog](#), [1.11.2 changelog](#), [1.11.3 changelog](#), and [1.11.4 changelog](#).

**Developer Links**

- [Source Code \(GitHub\)](#)
- [jQuery UI Git \(WIP Build\)](#)
- [Theme \(WIP Build\)](#)
- [Bug Tracker](#)
- [Submit a New Bug Report](#)
- [Discussion Forum](#)
- [Using jQuery UI](#)



# jQuery Datatables

- <https://www.datatables.net/>

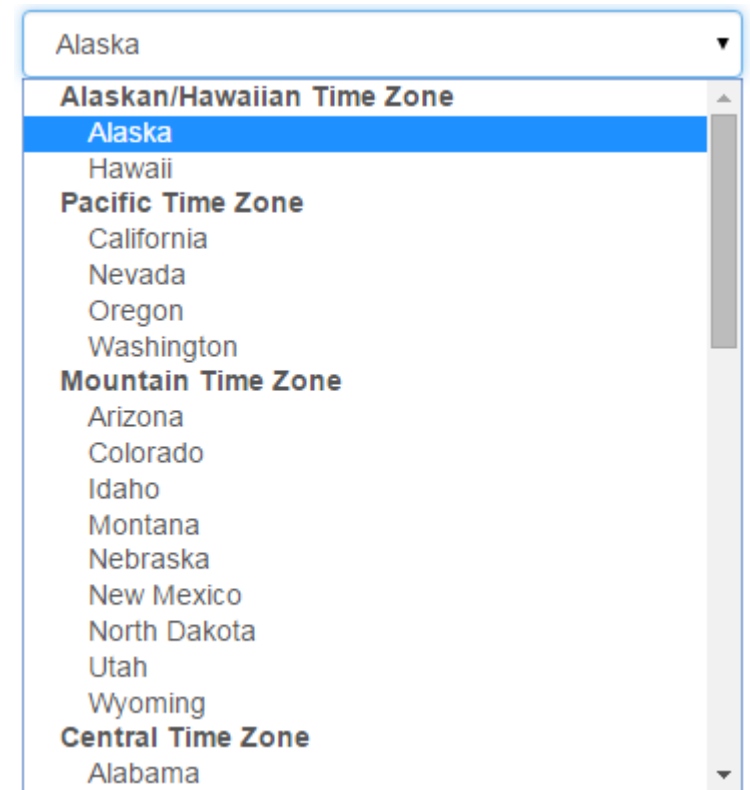
Show  entries Search:

Name	Position	Office	Age	Start date	Salary
Airi Satou	Accountant	Tokyo	33	2008/11/28	\$162,700
Angelica Ramos	Chief Executive Officer (CEO)	London	47	2009/10/09	\$1,200,000
Ashton Cox	Junior Technical Author	San Francisco	66	2009/01/12	\$86,000
Bradley Greer	Software Engineer	London	41	2012/10/13	\$132,000
Brenden Wagner	Software Engineer	San Francisco	28	2011/06/07	\$206,850
Brielle Williamson	Integration Specialist	New York	61	2012/12/02	\$372,000
Bruno Nash	Software Engineer	London	38	2011/05/03	\$163,500
Caesar Vance	Pre-Sales Support	New York	21	2011/12/12	\$106,450
Cara Stevens	Sales Assistant	New York	46	2011/12/06	\$145,600
Cedric Kelly	Senior Javascript Developer	Edinburgh	22	2012/03/29	\$433,060

Showing 1 to 10 of 57 entries Previous 1 2 3 4 5 6 Next

# Select2

- <https://select2.github.io/examples.html>



# Canvas

- Composant introduit par HTML5, permettant de dessiner ce qu'on veut dans une page Web.
- <http://www.williammalone.com/articles/html5-canvas-example/>
- [http://www.w3schools.com/html/html5\\_canvas.asp](http://www.w3schools.com/html/html5_canvas.asp)

# Exemple canvas

- <https://jsfiddle.net/gg19b2do/4/>

