JS

1. Rappels
2. **Environnement de debug**
3. ES6+
4. POO
5. jQuery
6. Expressions régulières
7. Échange de données
8. APIs HTML5

2. ENV. DE DEBUG

- **Les devtools**
  - Gérer les erreurs
  - L'objet console
  - Points d'arrêt
  - Tester sur plusieurs navigateur

LES DEVTOOLS

- Outils de développement de Chrome
  - Elements
  - Console
  - Sources (debugger)
  - Network
  - Application
  - ...
- Firefox, Safari, Opera, IE10/11 : outils similaires

Notes :

Les navigateurs récents proposent tous des outils destinés aux développeurs web. Ils ont tous des outils similaires, nous nous concentrerons sur les outils de Google Chrome.

Les outils se composent des éléments suivants :

- **Elements** : permet d'afficher le code HTML, CSS de la page en cours d'affichage, le modifier et voir les changements immédiatement
- **Console** : affiche les logs, erreurs du code js, voire les erreurs de chargement (404, etc.)
- **Sources** : affiche le code source et permet de mettre en pause l'exécution du code js à l'aide de breakpoints
- **Network** : log tous les appels réseaux effectués, permet de voir quel appel est long ou n'abouti pas
- **Application** : inspecte les cookies, le stockage local, ...

LES DEVTOOLS : DEMONSTRATION

Notes :

les différents outils des devtools :

CONSOLE

L'onglet console est certainement l'onglet le plus important lorsque l'on fait du JavaScript.

C'est dans cet onglet que sont affichées toutes les erreurs JavaScript qu'il s'agisse d'erreurs de syntaxe ou d'exception.

La console permet également d'écrire directement du JavaScript dans le même contexte d'éxécution que la page et donc de pouvoir tester du code JavaScript sans avoir besoin de recharger la page.

La console est intimement liée à l'inspecteur HTML et fourni des variables pratique comme "$0" qui correspond à l'élément HTML sélectionné dans l'inspecteur HTML.

ELEMENTS (INSPECTEUR HTML)

L'inspecteur HTML permet de voir le contenu HTML de la page.

Il est possible de déplacer ou de supprimer des éléments et même de leur ajouter, enlever ou modifier des attributs.

Le clic sur un élément permet d'inspecter les règles CSS qui s'y appliquent dans le volet de droite. Comme pour le HTML, il est possible de modifier les règles CSS à la volée afin de constater en direct l'impact visuel que cela entraîne.

SOURCES

L'inspecteur sources permet de visualiser les resources inclues par la page tels que les JS ou les CSS.

Cette partie de l'inspecteur permet entre autre de debugger le JavaScript en plaçant des breakpoint dans le code.

Un breakpoint permet de mettre en pause le programme lorsqu'il arrive à une ligne précise du code afin d'analyser le contexte d'exécution à un instant T. Il est possible ensuite d'exécuter le script pas à pas (ligne par ligne) afin d'analyser plus finement le flux d'exécution de l'application ce qui permet en général d'identifier plus facilement la cause d'un bug.

NETWORK (INSPECTEUR RÉSEAU)

L'inspecteur réseau liste tous les appels réseaux qui sont faits depuis la page qu'il s'agisse de chargement d'images, de CSS, de JS ou encore de requête AJAX. Pour chaque requête il est possible de visualiser l'en-tête HTTP envoyée au serveur ainsi que la réponse retournée.

APPLICATION

L'inspecteur de resources permet de voir le contenu des différentes base de données intégrées au navigateur comme le WebSQL, l'indexedDB, le Local storage, le Session storage ou les cookies.

Pour chaque type de base, il est possible d'y ajouter, d'y supprimer ou d'y modifier des données.

2. ENV. DE DEBUG

- Les devtools
- **Gérer les erreurs**
- L'objet console
- Points d'arrêt
- Tester sur plusieurs navigateur

GÉRER LES ERREURS

Différents types d'erreurs :

- **ReferenceError :** référence d'une variable invalide
- **SyntaxError :** code ne respectant pas la syntaxe js
- **TypeError :** variable ou argument du mauvais type
- ...

et leurs propriétés :

- **error.message :** texte associé à l'erreur
- **error.name :** nom de l'erreur
- **error.stack :** pile d'exécution

Notes :

Toutes les erreurs en js héritent de la classe `Error` et bénéficient donc des mêmes propriétés et méthodes.

DÉTECTER LES ERREURS

try / catch

```
try {
    // code à exécuter
} catch(error) {
    // traitement à effectuer en cas d'erreur
} finally {
    // traitement à effectuer dans tous les cas
}
```

Le callback : window.onerror

```
window.onerror = function( error ) {
    console.log(`new Error fired : ${error.message}
        in ${error.filename} at line ${error.lineno}`);
    fetch(
        'http://monsite.com/api/errors',
        {
            method: 'POST',
            body: JSON.stringify( error )
        }
    )
}
```

Notes :

On peut écouter l'événement onerror et ainsi envoyer l'info à une api qui fait ensuite le traitement adéquat (sauvegarde en bdd ou envoi d'email, etc.)

Il est aussi possible d'utiliser les outils de tracking (Google analytics / Matomo)

2. ENV. DE DEBUG

- Les devtools
- Gérer les erreurs
- **L'objet console**
- Points d'arrêt
- Tester sur plusieurs navigateur

L'OBJET CONSOLE

- tracer les actions : console.log(), console.warn(), console.error()
- Mesurer le temps d’exécution : console.time(), .timeEnd()
- Profiling automatique : console.profile(), .profileEnd()

2. ENV. DE DEBUG

- Les devtools
- Gérer les erreurs
- L'objet console
- **Points d'arrêt**
- Tester sur plusieurs navigateur

POINTS D'ARRÊT

Permettent de :

- Mettre en pause l’exécution
- inspecter les valeurs des variables
- Afficher la pile d’exécution
- exécuter le code pas à pas

2 techniques :

- Dans le code : debugger;
- Via l’inspecteur sources

HTML CSS JS

```
var button = document.querySelector('button');
var message = document.querySelector('div');
button.addEventListener('click', onButtonClick);
function onButtonClick( event ){
  debugger;
  showMessage( 'A girl has no name !' );
}
function showMessage( answer ){
  message.style.display = 'block';
  message.innerHTML += answer;
}
```

what is a girl's name ?

Notes :

exemple d'utilisation de l'instruction `debugger;` : http://codepen.io/kumquats/pen/mEmkGO?editors=0010

2. ENV. DE DEBUG

- Les devtools
- Gérer les erreurs
- L'objet console
- Points d'arrêt
- **Tester sur plusieurs navigateur**

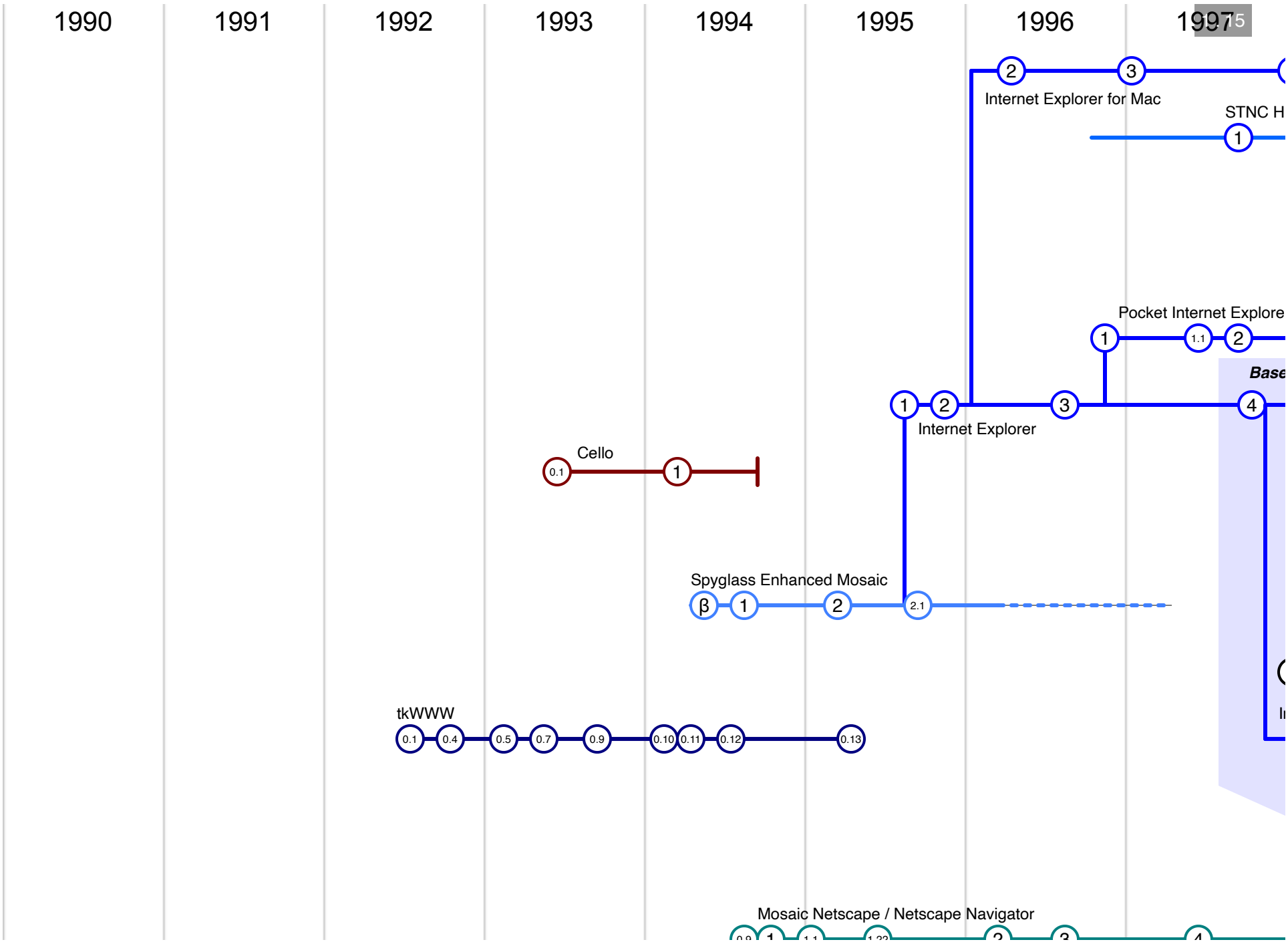
TESTER SUR PLUSIEURS NAVIGATEURS

- **1**/ Installer tous les navigateurs ?
- **2**/ Les machines virtuelles ?
- **3**/ outils en ligne ?

Notes :

Machines virtuelles vont permettre d'installer différents OS et différents navigateurs sur une même machine physique.

A ce propos, microsoft propose des VMs windows (7, 8.1, 10) gratuites avec edge/ie sur https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/



Notes :

L'Histoire des navigateurs source :
https://en.wikipedia.org/wiki/List_of_web_browsers#/media/File:Timeline_of_web_browsers.svg

OUTILS EN LIGNE :

- browserstack.com tests en live dans le navigateur
- comparium.app captures d’écran