Webtrekk
Drive Your Profit.

Funktionsbeschreibung der Pixel Plugin Schnittstelle

Pixel-Version 3.2.7

Webtrekk
Drive Your Profit.

Inhaltsverzeichnis

Webtrekk
Drive Your Profit.

1 Allgemeines

Die aktuelle Version des Webtrekk Tracking-Scriptes (Version 3.2.x) ermöglicht das Einbinden beliebiger Plugins. Sämtliche JavaScript Funktionen, die dem Webtrekk Pixel übergeben werden, sind als Plugin zu verstehen. Diese können von Webtrekk zur Verfügung gestellt werden oder auch selbst entwickelt sein.

Mit deren Hilfe können beispielsweise 3rd Party Tracking-Scripte eingebunden werden, zusätzliche Webtrekk-Parameter befüllt werden oder auch zu trackende Daten vor dem Senden verändert werden.

Innerhalb der Plugin-Scripte haben Sie Zugriff auf sämtliche Webtrekk-Parameter und Funktionen des Webtrekk-Tracking-Scriptes, welche Ihnen die Arbeit erleichtern sollen.

Dieses Dokument beschreibt die Integration eines Plugins in die Webtrekk-Pixelumgebung sowie die Nutzung der Funktionalität.

2 Aktivieren der Plugins

Um die Pluginfunktionalität zu aktivieren, wird der Parameter „executePluginFunction“ verwendet. Dieser kann entweder in der Konfigurationsvariable (webtrekkConfig) oder als Parameter des Pixelobjekts gesetzt werden.

Der Variable „wt.executePluginFunction“ können eine oder mehrere Funktionsnamen zugewiesen werden und diese werden dann nacheinander ausgeführt.

Die Funktionen die dem Pixel übergeben werden, müssen global unter dem Objekt „window“ zu finden sein. Mehrere Plugins werden mit einem Semikolon voneinander getrennt.

Beispiel für eine vollständige seitenspezifische Konfiguration:

```
var wt = new webtrekkV3();
wt.contentId = "de.startseite";
wt.executePluginFunction = "plugin1;plugin2;plugin3;plugin4;plugin5";
wt.sendinfo();
```

Beispiel für eine vollständige globale Konfiguration:

```
var webtrekkConfig = {
        trackId : "111111111111111",
        trackDomain : "track.webtrekk.net",
        domain : "www.website.com",
        cookie : "1",
        contentId : "",
        executePluginFunction : "plugin1;plugin2;plugin3;plugin4;plugin5"
};
```

3 Aufbau einer Plugin-Funktion

Ihre Plugin-Funktion ist eine JavaScript-Funktion mit beliebigem Code. Die in Kapitel 4 beschriebenen Methoden sollen beim Schreiben der Funktionen behilflich sein.

Wichtig ist lediglich, dass Ihrer Funktion die Variable „conf“ übergeben wird. Sie enthält notwenige Informationen über den ausgeführten Pixelrequest. Häufig wird es so sein, dass Ihr Plugin nur bei bestimmten Requestarten z.B. Seitenrequests ausgeführt werden soll. Mit Hilfe der „conf“-Informationen ist es Ihnen möglich diese Informationen zu prüfen.

```
function pluginX(conf)
{
        …. my code …
}
```

Folgende Informationen sind in dem „conf“-Objekt enthalten.

Parameter	Datentype	Beschreibung	Beispiel
instance	webtrekkV3	Beinhaltet die aktuelle Pixelinstance	wt
mode	String	Art des Tracking-Requests	“link”, “click”, “page”, “heatmap” oder “form”
type	String	Beinhaltet die Informationen, ob das Plugin vor oder nach einem Track-Request ausgeführt wird	“before“ oder “after“
requestCounter	Integer	Beinhaltet eine Zahl die angibt, der wievielte Request seiner Art, ausgeführt wird	5

4 Parameterübersicht

Beim ausführen eines Plugins wird die jeweilige Funktion zu einer internen Funktion. Somit haben Sie die Möglichkeit mit „**this**“ oder „**conf.instance**“ auf alle internen Funktionen und Variablen zuzugreifen. In der nachfolgenden Tabelle werden alle Variablen und Funktionen beschrieben und erklärt.

Alle Variablen denen kein Wert zugewiesen wurde, werden mit dem Datentyp „Boolean“ und dem Standartwert „false“ deklariert.

Property	Datentype	Beschreibung	Beispiel
birthday	String	Geburtsdatum des Kunden (JJJJMMDD)	19901027
birthdayD	String	Geburtstag des Kunden (DD)	27
birthdayJ	String	Geburtsjahr des Kunden (DD)	1990
birthdayM	String	Geburtsmonat des Kunden (DD)	10
browserLang	String	Browsersprache	de
campaignAction	String	Aktion der Kampagne	„click“ oder „view“
campaignId	String	Beinhaltet die Kampagnen inklusive Mediacode	mc%3DNewsletter_2011_08
city	String	Stadt	Berlin

config	Objekt	Beinhaltet die Informationen für den aktuellen Track-Request, nach versenden wird das Objekt gelehrt	config : {
			contentId : "de.Startseite"
			}

contentGroup	Objekt	Beinhaltet die Seitenkategorien	contentGroup : {
			1 : "Startseite"
			}

contentId	String	Seitenname	de.Startseite
cookie	String	Genutzte Cookieeinstellung	„1“ oder „3“
cookieDomain	String	Domain, in der die Cookies gesetzt werden sollen	.webtrekk.com

country	String	Land	Germany
couponValue	String	Gutscheinwert ohne Währungsangabe	25
currency	String	Währungsangabe	EUR

customCampaignParameter	Object	Beinhaltet die Kampagnen Parameter	customCampaignParameter = {
			1 : "personalized"
			}

customClickParameter	Object	Beinhaltet die Aktionsparameter	customClickParameter = {
			1 : "action1",
			7 : "action7"
			}

customEcommerceParameter	Object	Beinhaltet die Produktparameter	customEcommerceParameter = {
			1 : "L;32",
			2 : "grün;blau",
			3 : "rechnung"
			}

customParameter	Object	Beinhaltet die Seitenparameter	customParameter = {
			1 : "green"
			}

customSessionParameter	Object	Beinhaltet die Sessionparameter	customSessionParameter = {
			1 : "logged_in"
			};

customerId	String	Beinhaltet die Kunden-ID	372d1a04d003eebc09e1
deactivatePixel	Boolean	Deaktiviert den kompletten Pixelversand	true
deactivateRequest	Boolean	Deaktiviert den Pixelversand des aktuellen Requests	true
delayLinkTrack	Boolean	Aktiviert die Verzögerung beim LinkTracking	true
delayLinkTrackTime	Number	Zeit in Millisekunden, für die Verzögerung beim	200

		LinkTracking	
disableOverlayView	Boolean	Deaktiviert die Ansicht des Overlays bzw. der Heatmap	true
domain	String	Beinhaltet die Domain der zumessenden Seite	www.website.com
eid	String	Beinhaltet die User-ID, die automatisch vom Pixel ermittelt wird. Dieser Wert darf nicht geändert werden.	2132385537700174775
email	String	E-Mail Adresse	test@tester.com
emailOptin	String	Erhalten eines Newsletters	„1“ oder „0“
emailRID	String	E-Mail RID	123456789
executePluginFunction	String	Beinhaltet die Namen der auszuführenden Plugins	plugin1;plugin2;plugin3
firstName	String	Vorname	Hans
forceHTTPS	String	Verschickt die Pixel-Requests immer mit SSL	1
form	String	Aktiviert das Formulartracking	1
formAnonymous	String	Anonymisiert alle Formulardaten	1
formAttribute	String	Dient zum auslesen des Formularnamens	rel
formFieldAttribute	String	Dient zum auslesen der Formularfeldnamen	id

formFieldDefaultValue	Object	Beinhaltet Standartwerte für vorausgefüllte Formularfelder	{
			fname: "Vorname"
			}

formFullContent	String	Angabe der Formularfeldnamen, bei denen der komplette Formularinhalt verschickt	ort;plz

		werden soll	
formPathAnalysis	Boolean	Aktiviert die Pfadanalyse für den Formular-Fokus	true
formSubmit	Boolean	Legt fest, ob ein Formular abgeschickt wurde	true
formValueAttribute	String	Legt ein alternatives Attribute zum auslesen des Formularfeldinhaltes fest	value
gender	String	Geschlecht	„0“, „1“ oder „2“
heatmap	String	Aktiviert die Heatmap	"1"
heatmapRefpoint	String	Referenzpunkt für die Heatmap	wt_refpoint
internalSearch	String	Beinhaltet den Suchbegriff bei der internen Suche	suchbegriff
isChrome	Boolean	Gibt an, ob es sich um einen Google Chrome Browser handelt	false
isIE	Boolean	Gibt an, ob es sich um einen Internet Explorer Browser handelt	true
isOpera	Boolean	Gibt an, ob es sich um einen Opera Browser handelt	false
isSafari	Boolean	Gibt an, ob es sich um einen Safari Browser handelt	false
linkId	String	Beinhaltet den Namen des geklickten Links	linkname
linkTrack	String	Aktiviert das Link-Tracking	„link“ oder „standard“
linkTrackAttribute	String	kann für die Aktionsnamengenerierung genutzt werden	name

linkTrackDownloads	String	Beinhaltet die Dateiendungen der Dokumente	doc;pdf;zip
linkTrackIgnorePattern	String	Regulärer Ausdruck, mit dem Links vom LinkTracking ausgeschlossen werden sollen	/^javascript:.+/
linkTrackParams	String	kann für die Aktionsnamengenerierung genutzt werden	teaser_id;page_id
linkTrackPattern	RegExp	Filtert einen Bereich aus der URL heraus und ersetzt diesen durch den Variableninhalt „linkTrackReplace“	new RegExp(";jsessionid=[a-zA-Z0-9\.]+[#|?|&]?","g")
linkTrackReplace	String	Ersetzt die Fundstelle vom Parameter „linkTrackPattern“ durch den angegebenen Wert	"?"
linktrackOut	Boolean	Gibt an, ob es sich um einen externen link handelt	true
maxRequestLength	Number	Maximale Requestlänge	7168
mediaCode	String	Beinhaltet den Namen des Mediacodes	mc
mediaCodeCookie	String	MediaCode nur einmal pro Session oder in einer bestimmten Zeit messen	sid oder eid
mediaCodeValue	String	Beinhaltet den Wert zu dem gefundenen Mediacodes	Newsletter_2011_08
noDelayLinkTrackAttribute	String	Attribute, um Links zum Verzögern zu markieren	data-wt-delay
optOut	Boolean	Deaktiviert den kompletten Pixelversand	true
optoutName	String	Name des OptOut-	webtrekkOptOut

		Cookies	
orderId	String	Enthält eine eineindeutige Bestellnummer	M-12345
orderValue	String	Enthält den Gesamtbestellwert	12.99
pixelSampling	String	Bei diesem Wert wird nur jeder n-te Besucher gemessen	10
plugins	Array	Beinhaltet alle Plugins, die der Browser unterstützt	["Adobe Acrobat", "Windows Media Player", "Shockwave Flash"]
postalCode	String	PLZ	12345
product	String	Beinhaltet den Produktnamen	pullover

productCategory	Object	Beinhaltet die Produktparameter	productCategory = {
			1 : "oberbekleidung",
			2 : "noname"
			}

productCost	String	Enthält den Produktpreis	99.90
productQuantity	String	Enthält die Produktanzahl	2
productStatus	String	Enthält den Status des Warenkorbs	„view“, „add“ oder „conf“
street	String	Straße	Teststraße
streetNumber	String	Hausnummer	12
telefon	String	Telefonnummer	0123456789
trackDomain	String	Enthält die Track-URL	q3.webtrekk.net
trackId	String	Enthält die fünfzehn Stellige Webtrekk Kunden-ID	111111111111111
updateCookie	Boolean	Webtrekk EID wird bei jeder PI auf neue 6 Monate Laufzeit gesetzt	true

urmCategory	Object	Beinhaltet die User-Parameter	urmCategory = {
			1 : "27",
			2 : "ledig"
			}

version	Integer	Enthält die Pixelversion	321
cookieManager	Constructor	Dient zum setzen und auslesen von Cookies	Siehe Kap. 5.6
include	Function	Dient zum nachladen von JavaScript Dateien	Siehe Kap. 5.2
loadAsynchron	Function	Dient zum nachladen von JavaScript Dateien mit angebe einer Callback Funktion	Siehe Kap. 5.3
registerEvent	Function	Dient zum registrieren von Events	Siehe Kap. 5.4
unregisterEvent	Function	Dient zum löschen von Events	Siehe Kap. 5.5
sendinfo	Function	Globale Funktion zum Versenden von Track-Requests	Siehe Kap. 5.1

5 Funktionsübersicht

Im nachfolgenden werden interne Webtrekk Funktionen aus dem Pixel beschrieben, die Ihnen das Entwickeln eines eigenen Plugins erleichtert.

5.1 Funktion – sendinfo

Die Funktion „sendinfo“ ist die zentrale Methode zum Versenden der Tracking-Pixel. Bei Übergabe des Konfigurationsobjektes „c“, werden nur die übergebenen Werte (im Objekt) verschickt. Sobald der Parameter nicht übergeben wird, wird die Konfiguration aus dem Tracking-Pixel herangezogen und versendet. Mit dem Parameter „p“, kann optional eine ContentId oder LinkId mitgeben werden. Unter „mode“ können Sie definieren, ob der Request als „click“, „link“ oder „page“ Request verschickt werden soll. Wenn der Parameter nicht angegeben wird, wird standartmäßig „page“ verwendet. Sobald eine LinkId definiert wurde, ist der standartwert

„click“. Mit dem Letzten Parameter „ep“, kann eine semikolonseparierte Liste eigener Parameter mitgebeben werden.

```
/**
 * @param {Object} [c]          Konfigurationsobjekt [optional]
 * @param {String} [p]          ContentId oder LinkId [optional]
 * @param {String} [mode]       „link“, „click“ oder „page“ [optional]
 * @param {String} [ep]         Semikolonseparierte Liste eigener Parameter [optional]
 */
sendinfo(c, p, mode, ep);
```

5.2 Funktion – include

Die Funktion „include“ lädt zusätzliche JavaScript Dateien nach.

```
/**
 * @param {string} url      URL der nachzuladenden JavaScript Datei
 * @return true/false
 */
include(url);
```

Im nachfolgenden Beispielplugin wird nach dem ersten Seiten-Request die JavaScript Datei „javascript.js“ nachgeladen. Dies geschieht asynchron.

Beispiel für das Nachladen einer JavaScript Datei:

```
function plugin1(conf)
{
        if(conf.mode == "page" && conf.type == "after" && conf.requestCounter == 1)
        {
                var url = "http://domain.de/demo/javascript.js";
                this.include(url);
        }
}
```

5.3 Funktion – loadAsynchron

Die Funktion „loadAsynchron“ arbeitet identisch zur Funktion „include“, wobei man bei dieser Variante nicht nur den Pfad zur JavaScript Datei angeben kann. Zusätzlich muss man noch eine Variable, Funktion oder Objekt angeben, die sich in der nachzuladenden Datei befindet. Somit kann geprüft werden, ob die JavaScript Datei tatsächlich nachgeladen wurde und vom Browser erfolgreich eingelesen bzw. ausgeführt wurde.

Optional kann noch eine Callback Funktion angegeben werden. Diese wird ausgeführt, wenn die Datei erfolgreich nachgeladen wurde oder die Standartzeit von 2000 Millisekunden abgelaufen ist. Der Funktion wird als erster Parameter „true“ oder „false“ mitgegeben, die angeben ob die JavaScript Datei erfolgreich nachgeladen wurde. Der zweite Parameter beinhaltet die aktuelle Pixelinstance.

Falls Ihnen die Standartzeit von 2000 Millisekunden zu gering ist, können Sie optional auch eine andere Zeit in Millisekunden angeben.

```
/**
 * @param {String} url                  URL der nachzuladenden JavaScript Datei
 * @param {String} search               Variable, Funktion oder Objektes, welches sich in der
 *                                      nachzuladenden Datei befindet
 * @param {Function} [callback]         Callback-Funktion, die nachträglich ausgeführt werden soll
 *                                      [optional]
 * @param {Integer} [timeout]           Zeit in Millisekunden, die auf die nachzuladenden Datei
 *                                      maximal gewartet wird [optional]
 */
loadAsynchron(url, search, callback, timeout);
```

Im unten gezeigten Beispielplugin wird eine JavaScript Datei vor dem ersten Seiten-Request nachgeladen. Wenn die Funktion das „demoObject“ gefunden hat, wird die Funktion „demoCallback“ ausgeführt und es wird „include true“ ausgeben. Falls die Datei nach 3000 Millisekunden nicht nachgeladen wurde wird „include false“ ausgegeben.

Beispiel für das asynchrone Nachladen einer JavaScript Datei:

```
function plugin2(conf)
{
	if(conf.mode == "page" && conf.type == "before" && conf.requestCounter == 1)
	{
		this.demoCallback = function(result, instance)
		{
			if(result)
			{
				alert("include true");
			}
			else
			{
				alert("include false");
			}
		}

		var url = "http://domain.de/demo/javascript.js";
		var search = "demoObject";
		var callback = this.demoCallback;
		var timeout = 3000;
		this.loadAsynchron(url, search, callback, timeout);
	}
}
```

5.4 Funktion – registerEvent

Die Funktion „registerEvent“ registriert eine Funktion auf ein angegebenes Event. Der erste Parameter gibt das Objekt an, bei dem das Event registriert werden soll. Der zweite Parameter beinhaltet den Eventnamen. Als letzter Parameter wird eine Funktion angegeben die ausgeführt wird, wenn das Event eintritt.

```
/**
 * @param {Object} object                Objekt bei dem das Event registriert wird
 * @param {Object} event                 Event
 * @param {Function|String} function     Funktion die ausgeführt werden soll
 */
registerEvent(object, event, function);
```

Das unten gezeigte Beispielplugin wird jeden Link auf der Seite markiert. Sobald auf den Link geklickt wird, wird die URL des Links ausgegeben.

Beispiel für das registrieren eines Events:

```
function plugin3(conf)
{
	if(conf.mode == "page" && conf.type == "before" && conf.requestCounter == 1)
	{
		this.checkLink = function(e)
		{
			var a = document.all ? window.event.srcElement : this;
			alert(a.href);
		}
		var f = this.checkLink;

		for(var i = 0; i < document.links.length; i++)
		{
			this.registerEvent(document.links[i], "mousedown", f);
		}
	}
}
```

5.5 Funktion – unregisterEvent

Die Funktion „unregisterEvent“ ist das genaue Gegenstück zu „registerEvent“, es entfernt eine registrierte Funktion vom angegebenen Event. Der erste Parameter gibt das Objekt an, von dem das Event entfernt werden soll. Der zweite Parameter beinhaltet das Event. Als letzter Parameter wird eine Funktion angegeben, die entfernt werden soll.

```
/**
 * @param {Object} object                Objekt bei dem das Event entfernt werden soll
 * @param {Object} event                 Event
 * @param {Function|String} function     Funktion die vom Objekt entfernt werden soll
 */
unregisterEvent(object, event, function);
```

Das unten gezeigte Beispielplugin entfernt von jedem Link die zuvor gesetzte Funktion „checkLink“.

Beispiel für das registrieren eines Events:

```
function plugin4(conf)
{
	if(conf.mode == "page" && conf.type == "before" && conf.requestCounter == 1)
	{
		this.checkLink = function(e)
		{
			var a = document.all ? window.event.srcElement : this;
			alert(a.href);
		 }
		var f = this.checkLink;

		for(var i = 0; i < document.links.length; i++)
		{
			this.registerEvent(document.links[i], "mousedown", f);
			this.unregisterEvent(document.links[i],"mousedown", f);
		}
	}
}
```

5.6 Funktion – cookieManager

Der Cookie-Manager ist ein Key-Value Store für beliebige Strings, die für eine spätere Verwendung gespeichert werden sollen. Key-Value Store bedeutet, dass Sie einen Key (also einen eindeutigen Schlüssel in Form einer Zeichenkette) definieren, zu dem dann ein beliebiger Wert (ebenfalls in Form einer Zeichenkette) gespeichert wird.

Typische Einsatzgebiete sind das „Durchschleifen“ von Parametern, deren Informationen nicht auf allen Seiten zur Verfügung stehen, oder bspw. eine Laufzeitbeschränkung von einzelnen Parametern.

```
/**
 * @param {String} name                 Name des Cookies
 * @param {Number} expires              Laufzeit in Tagen
 * @param { String } [accessPath]       Pfad in dem das Cookie gesetzt wird [optional]
 *
 * @type Constructor
 */
cookieManager(name, expires, accessPath);
```

5.6.1 Methoden des Cookie-Managers

5.6.1.1 Methode – Set

Setzt den Wert „value“ zum Key „key“ und speichert das Cookie. Anschließend wird der Wert „value“ (abhängig vom mode) zurückgegeben.

Der optionale dritte Parameter definiert eine Laufzeit in Sekunden, für die der gesetzte Wert gültig bleibt. Wird kein Wert angegeben, bleibt der gesetzte Wert für ein Jahr gültig.

Der optionale vierte Parameter „mode“ kann auf „first“ oder „last“ (default) gesetzt werden. Bei der Angabe von „first“ wird der übergebene Wert nur beim ersten Aufruf innerhalb der Laufzeit gesetzt, bei weiteren Aufrufen wird der im Cookie gespeicherte Wert zurückgegeben. Bei „last“ wird immer der zuletzt übergebene Wert gespeichert und zurückgegeben.

```
/**
 * @param {String} fieldname          Name des Key innerhalb des Cokies
 * @param {String} fieldval           Value für den angegebenen Key
 * @param {Number} duration           Angabe in Sekunden
 * @param {String} [mode]             “first” oder “last”
 *
 * @returns {String|Boolean}          value oder false
 *
 */
set(fieldname, fieldval, duration, mode);
```

5.6.1.2 Methode – Get

Gibt den Wert zurück, der zum Key „key“ gespeichert wurde.

```
/**
 * @param {Object}  fieldname        Gesuchter Key im Cookie
 *
 * @return {String}                  Value
 */
get(fieldname);
```

5.6.1.3 Methode – GetSize

Gibt die Anzahl der Zeichen zurück, die im Cookie gespeichert wurden.

Ein Cookie kann ca. 4kB Daten speichern. Sollte der Speicherplatz für eine bestimmte Anwendung nicht ausreichen, kann mit Hilfe dieser Methode die Länge des Strings abgefragt werden, der im Cookie gespeichert wird. Wird die Größe von 4kB überschritten, kann die Anwendung ein weiteres Cookie zum Speichern der Daten erstellen.

```
/**
 * @returns {Number}       Anzahl der Zeichen des kompletten Cookies
 */
getSize();
```

5.6.1.4 Methode – Prepare

Die Funktionsweise ist analog der Methode „set“ aufgebaut. Sie dient dazu Informationen, die sich z.B. häufig ändern (wie das kontinuierliche Setzen einer Verweildauer) für den Schreibvorgang vorzuhalten. Um das tatsächliche Schreiben der Werte in das Cookie auszuführen, muss die Methode „write“ verwendet werden.

Auf diese Weise wird eine effektivere Datenverarbeitung sichergestellt, da der Schreibaufwand minimiert ist.

```
/**
 * @param {String} fieldname              Name des Key innerhalb des Cokies
 * @param {String} fieldval               Value für den angegebenen Key
 * @param {Number} duration               Dauer in Sekunden
 * @param {String} [mode]                 “first” oder “last”
 *
 * @returns {String|Boolean}              value oder false
*/
prepare(fieldname, fieldval, duration, mode)
```

5.6.1.5 Methode – Write

Mit Hilfe der Methode werden zuvor gesetzte Werte im Cookie gespeichert.

Gleichzeitig werden die Laufzeiten der im Cookie hinterlegten Werte überprüft. Falls die Laufzeit überschritten ist, werden die entsprechenden Werte aus dem Cookie entfernt. Sollte die Laufzeit aller Werte überschritten sein, wird das Cookie entfernt.

```
write();
```

5.6.1.6 Methode – Remove

Entfernt ein Cookie. Das Löschen der Daten ist endgültig, gelöschte Daten können nicht wiederhergestellt werden!

```
/**
 * @returns {Boolean}      wobei true bedeutet, dass das Cookie erneut ausgelesen
 *                         und somit nicht gelöscht werden konnte
 */
remove();
```

5.6.2 Cookie-Manager Beispiel

Gehen wir davon aus, dass bei einer Registrierung auf unserer Webseite für einen Monat kostenlos zusätzliche Funktionen angeboten werden, für die danach bezahlt werden muss. Nun soll dieser Status „free premium“ für diese Zeit übergeben werden.

Lösung:

Bei der Registrierung wird dieser Status mit dem Cookie-Manager für eine Laufzeit von 30 Tagen gesetzt. Auf den einzelnen Seiten wird dieser Wert als Seiten- oder Session-Parameter mit jedem Request versendet.

Auf der Bestätigungsseite der Registrierung wird folgender Code eingebaut:

```
var wt = new webtrekkV3();
var myCookie = new wt.cookieManager("registration", 30);
wt.customSessionParameter = {
    1 :  myCookie.set('status', 'free premium', 2592000, 'first');
};
wt.sendinfo();
```

Auf allen anderen Seiten wird folgender Code eingebaut:

```
var wt = new webtrekkV3();
var myCookie = new wt.cookieManager("registration",30);
wt.customSessionParameter = {
    1 :  myCookie.get('status');
};
wt.sendinfo();
```

Der Wert „free premium“ für den Key „status“ wurde in diesem Beispiel mit einer Gültigkeitsdauer von 2592000 Sekunden (entspricht 30 Tagen) gesetzt. In diesem Beispiel haben wir im Frontend den Session-Parameter 1 für diesen Status angelegt und schreiben dort den Wert hinein. Nach Ablauf der Gültigkeitsdauer wird der Aufruf myCookie.get('status'); einen leeren String zurückgeben und der Session-Parameter dadurch nicht mehr übermittelt.

Im Frontend können wir nun diesen Session-Parameter 1 nach dem Wert „free premium“ filtern und bekommen alle Kennzahlen der Nutzer mit diesem Status.

5.7 Ändern von internen Variablen

Im Plugin kann man mit „this“ auf alle Variablen zugreifen, die zuvor auf der Seite definiert wurde. Mit „this.config“ kann man auf alle Variablen zugreifen, die für den aktuellen Track-Request relevant sind.

In dem Beispiel wird gezeigt, wie man z.B. vor jedem Request die „contentId“ um die Browsersprache erweitern kann.

Beispiel für das Ändern der „contentId“:

```
function plugin5(conf)
{
        if(conf.type == "before")
        {
                var browserLang = "";
                if (typeof(navigator.language) == "string")
                {
                        browserLang = navigator.language.substring(0,2);
                }
                else if (typeof(navigator.userLanguage) == "string")
                {
                        browserLang = navigator.userLanguage.substring(0,2);
                }
                this.config.contentId = browserLang + "." + this.config.contentId;
        }
}
```

6 Coding-Rules

Im folgenden Kapitel wurden ein paar Coding-Rules festgelegt, an die Sie sich beim Entwickeln eines Webtrekk Pixel Plugins gehalten werden muss. Im Anschluss der Pluginentwicklung wird Webtrekk eine Abschließende Abnahme und Prüfung der Funktionalitäten vornehmen.

6.1 Verbotene Funktionsaufrufe

Das Plugin bzw. nachgeladene Dateien dürfen folgende Aufrufe nicht enthalten.

- document.write, document.writeln, document.open, document.close

6.2 Browser und Betriebssysteme

Das Plugin sollte von folgenden Browsern und Betriebssystemen unterstützt werden. Dabei sollten alle möglichen Konstellationen getestet und unterstützt werden.

- Internet Explorer 6 – 9, Firefox, Google Chrome, Safari, Opera
- Windows XP, Windows Vista, Windows 7, Mac OS, Linux

6.3 Nachladen einer Datei

Das Nachladen einer JavaScript Datei sollte nicht länger als 2000 Millisekunden dauern und darf dabei das Laden der Seite nicht verzögen.

6.4 Track-Request

Der Request-Versand des Webtrekk Pixels sollte nicht länger als 500 Millisekunden verzögert werden. Dabei ist zu beachten, dass Track-Requests von Drittanbietern erst nach dem Webtrekk Track-Requests geschickt werden dürfen.

Bei Plugins, die bei einer Aktion ausgeführt werden, darf der Request-Versand des Webtrekk Pixels nicht verzögert werden.

7 Kontakt

Wenn Sie Fragen zur Integration der Plugins haben sollten, stehen wir Ihnen selbstverständlich zur Verfügung:

Webtrekk GmbH
Hannoversche Straße 19
10115 Berlin

fon 030 - 755 415 - 0
fax 030 - 755 415 - 100
support@webtrekk.com

http://www.webtrekk.com