

## Семантический WEB для всех

Аарон Сварц (с помощниками)

### The Semantic Web In Breadth

[Aaron Swartz](#) with [much assistance](#)

<http://logicerror.com/semanticWeb-long>

**Замечание:** В данной статье говорится о различных частях семантического Web'a и о том, как эти части согласуются друг с другом. Более точное представление об этом можно найти в работе Сандро Хока "Семантический Web (в простом изложении)" [Sandro Hawke's The Semantic Web \(Put Simply\)](#). С другой стороны, если вы являетесь Web-разработчиком, интересующимся производством Web-сайтов или службами семантического Web'a, обратитесь к работе [The Semantic Web \(for Web Developers\)](#) – Семантический Web (для Web-разработчиков). А теперь к нашей простенькой статье.

#### Идентификаторы: Универсальный идентификатор ресурсов (URI)

Если надо что-то обсудить, это что-то необходимо сначала идентифицировать. В противном случае, откуда можно узнать, на что я ссылаюсь? Можно сделать это неявным образом: "Северная звезда", "Странный мужчина около универсама", "Очень кислые конфеты, которые всегда ест Боб". Но можно также сказать и более точно: "Полярная", "Джонатан Свифт", "Барбарис".

На Web'e для идентификации элементов мы также применяем идентификаторы. Так как мы используем унифицированную систему идентификаторов и так как каждый идентифицированный элемент рассматривается "ресурсом", эти идентификаторы называются "Унифицированными идентификаторами ресурсов" или для краткости URI (Uniform Resource Identifier). URI можно присвоить чему угодно, и если это что угодно обладает URI, то про него говорят, что оно находится "на Web'e": вы, купленная вами на прошлой неделе книга, жужжащая в ваше ухо муха и все, о чем вы можете подумать – у всего этого может быть URI.

URI это то, на чем базируется Web. В то время, когда почти каждая часть Web'a может быть заменена, URI – не может: он делает Web единым.

Одной из форм URI является URL (Uniform Resource *Locator*), унифицированный указатель ресурса. URL это адрес, позволяющий посетить Web-страницу, такой,

например, как: <http://www.w3.org/Addressing/>. Если разбить его на [составные части](#), то можно увидеть, что URL сообщает вашему компьютеру, где можно найти конкретный ресурс (в нашем случае, это W3C's [Addressing website](#)). В отличие от других форм URI, каждый URL одновременно и идентифицирует, и указывает. Сопоставьте это с URI, который идентифицирует сообщение email, но не может определить местоположение копии вашего сообщения.

Так как Web слишком велик для того, что одна какая-либо организация управляла им, все множество URI является децентрализованным. Ни один человек и ни одна организация не следит за тем, кто их создает или как их можно использовать. В то время как одни схемы URI (такие, как *http*;) зависят от централизованных систем (например, систем DNS), другие схемы (такие, как *freenet*;) являются полностью децентрализованными.

Это означает, что вам не нужно чье-либо разрешение для создания URI. Вы можете даже создавать различные URI для вещей, которые вам не принадлежат. Эта гибкость делает URI мощным средством, но и ставит не так уж мало новых проблем. Так как создать URI может кто угодно, мы обязательно столкнемся с тем, что одна и та же вещь будет представлена многими URI. И, что еще хуже, не будет никакого способа выяснить, ссылаются ли два URI в точности на один и тот же ресурс. Следовательно, мы никогда не сможем с уверенностью сказать, что в точности означает данный URI. Но это те издержки, на которые мы должны пойти, если ставим перед собой цель создать нечто огромное, подобное семантическому Web'у.

Обычно создание URI начинают с Web-страницы. Эта страница описывает подлежащий идентификации объект и говорит, что URL этой страницы является URI идентифицируемого объекта. Например, мне нужно было создать URI для моей копии книги Тима Бернерса-Ли "[Weaving the Web](#)". Сначала, я создал Web-страницу, [на которой описана моя копия](#). Затем я отметил на ней, что URL этой страницы будет являться URI для моей копии идентифицируемой книги. Результатом этих действий явилось то, что я связал это URI (<http://logicerror.com/myWeavingTheWeb>) с моей копией книги "Weaving The Web". Создать URI можно и таким простым способом.

Вероятно вы заметили, что в приведенном примере URI (<http://logicerror.com/myWeavingTheWeb>) выступает в двух ролях: он представляет и книгу, как предмет, а также Web-страницу, которая эту книгу описывает. Об этом можно дискутировать. С одной из таких дискуссий можно ознакомиться на сайте [The Semantic Web Identification Problem](#). Разработчики семантического Web'a постоянно возвращаются к обсуждению этого вопроса.

Этот важный факт требует хорошего понимания. URI не является набором указаний вашему компьютеру, как на Web'e добраться до конкретного файла (хотя таковым и может быть). Это – имя "ресурса" (вещи). Этот ресурс может **быть или не быть** доступен по Интернету. URI может **предоставить или не предоставить** вашему компьютеру возможность получить дополнительную информацию об этом ресурсе. Да, URL это такой тип URI, который, безусловно, предоставляет возможность получить информацию о ресурсе, или, возможно, выбрать сам ресурс. Сейчас разрабатываются и другие методы предоставления информации об URI и ресурсах, которые они идентифицируют. Также верно, что способность сказать что-то полезное об URI является

важной частью семантического Web'a. Но вы не должны предполагать, что URI это нечто большее, чем идентификатор ресурса.

### Документы: расширяемый язык разметки (XML)

Язык [XML](#) был разработан в качестве простого средства пересылки документов по Web'у. Он позволяет каждому создать свой собственный формат документов и затем писать документы в этом формате. Эти форматы документов могут включать разметку, уточняющую смысл содержимого документа. Документ с разметкой "машиночитаем", то есть программы могут его читать и понимать. Включая в наши документы элементы, смысл которых понятен для машины, мы делаем эти документы значительно более мощными.

Рассмотрим простой пример: если в документе содержатся некоторые слова, отмеченные разметкой как "акцентированные", то способ выделения этих слов может быть согласован с контекстом документа. Web-браузер мог бы просто показать их на дисплее курсивом, а голосовой браузер (который читает Web-страницы вслух) мог бы выразить это выделение изменением высоты тона или громкостью голоса. Каждая программа может реагировать в соответствии со значением, закодированном в разметке. В противоположность этому, если бы я просто отметил эти слова "курсивом", то компьютер не знал бы, по какой причине я печатаю эти слова курсивом. Это сказано для эмфазы или просто для зрительного эффекта? И как бы звуковой браузер отреагировал на это?

Вот пример простого неразмеченного документа:

```
I just got a new pet dog.
```

Если говорить о компьютере, то для него это просто текст. Никакого конкретного смысла компьютер в нем не находит. Но теперь посмотрим на то же самое предложение, когда оно будет записано в некотором аналогичном XML языке разметки (создадим его для нашего примера):

```
<sentence>  
<person href="http://aaronsw.com/">I</person> just got a new pet  
<animal>dog</animal>.  
</sentence>
```

Заметим, что размеченный текст имеет тот же самый контент, но некоторые части этого контента помечены. Каждая метка состоит из двух "тегов" (tags): открывающий тег (например, <sentence>) и закрывающий тег (например, </sentence>). Имя этого тега ("sentence") является меткой контента, заключенного между тегами. Мы называем эту совокупность тегов и контента "элементом". Таким образом, элемент sentence в приведенном выше документе содержит предложение "Я приобрел новую собачку" (I just got a new pet dog.). Этот элемент сообщает компьютеру, что "I just got a new pet dog." это "sentence", но, и это более важно, он не говорит компьютеру, что это такое – sentence. Тем не менее, у компьютера теперь есть некоторая информация о документе, и мы можем с пользой употребить эту информацию.

Таким же образом компьютер теперь знает, что "I" это "person" (что это такое – неизвестно) и что "dog" это "animal".

Иногда полезно сообщить о контенте элемента больше информации, чем та, которая содержится только в имени элемента. Например, компьютер знает, что "Я" в приведенном выше предложении представляет "person", но он не знает, как узнать что-то об этом/этой "person". Мы можем предоставить информацию такого типа, добавив к нашему элементу *атрибуты*. У атрибута есть имя и значение. Мы можем, например, переписать наш пример следующим образом:

```
<sentence>
<person href="http://aaronsw.com">I</person> just got a new pet <animal type="dog"
href="http://aaronsw.com/myDog">dog</animal>.
</sentence>
```

Как вы, может быть, уже заметили, здесь мы сталкиваемся с серьезной проблемой. В своем языке разметки я использовал слова "sentence", "person", "animal". Но это очень обычные слова. Что если другие уже использовали эти же самые слова в своих собственных языках разметки? Что если эти слова в их языках имеют другой смысл? Возможно, что в других языках разметки "sentence" означает срок, который приговоренный преступник должен провести в исправительном заведении? Как это должен понимать мой компьютер?<sup>1</sup>

Чтобы не было путаницы, я должен однозначно **идентифицировать** свои элементы разметки. И что может быть лучше, чем идентифицировать их с помощью стандартного идентификатора ресурса? Поэтому каждому элементу и атрибуту поставим в соответствие некоторый URI. Сделаем это с помощью так называемого пространства имен XML ([XML Namespaces](#)). На этом пути каждый может создавать свои собственные теги и пользоваться ими наряду с тегами, сделанными другими. Пространство имен это просто способ идентифицировать часть Web'a (пространства), из которой мы узнаем значения этих имен. Создаем "пространство имен" для своего языка разметки, создавая для него URI. (Как мы уже обсуждали, возможно, создадим Web-страницу для описания языка разметки и воспользуемся URL этой страницы как URI пространства имен.)

Раз каждый тег, кому бы он ни принадлежал, имеет свой собственный URI, нам не нужно беспокоиться о конфликте имен тегов. Разумеется, XML предоставляет нам сокращенные формы записи, и у нас есть множества URI по умолчанию, так что нам не надо приводить их в тексте каждый раз:

```
<sentence
  xmlns="http://example.org/xml/documents/"
  xmlns:c="http://animals.example.net/xmlns/"
><c:person c:href="http://aaronsw.com/">I</c:person> just got a new pet
<c:animal>dog</c:animal>.</sentence>
```

Обратите внимание, что "http://example.org/xml/documents" это пространство имен по умолчанию для обсуждаемого документа. Именно здесь определены все элементы и атрибуты, перед которыми не находится символ "c": (в нашем случае так определенным элементом является только элемент "sentence").

---

<sup>1</sup> Английское слово "sentence" означает и "предложение", и "приговор" (прим. перевод.)

## Statements: Resource Description Framework (RDF) Утверждения: Общая схема описания ресурсов

Вот теперь мы начнем разбираться в том, что является сердцевинной семантического Web'a. Прекрасно, что умеем создавать различные URI и говорить о них на наших Web-страницах. Но было бы еще лучше, если бы мы могли говорить о них так, чтобы компьютеры смогли начать обрабатывать высказываемое нами. Например, одно дело сказать "Мне очень нравится книга "Weaving the Web" во время обсуждения Web'a на форуме. Другое дело, как это поймет компьютер?

RDF дает вам возможность формулировать наши утверждения в виде, **пригодном для обработки машиной**. Конечно, и теперь компьютер не "понимает" то, что вы сказали, но он может работать с этим так, как будто понимает. Например, я мог бы дать задание просматривать Web, отбирая все оценочные высказывания о книгах, создать средний рейтинг для каждой книги и потом поместить эту информацию обратно на Web. Другой Web-сайт мог бы воспользоваться этой информацией (списком усредненного рейтинга книг) и создать страницу "Десять самых рейтинговых книг".

RDF весьма прост. RDF-утверждение очень напоминает обычное предложение за тем исключением, что в нем почти все слова являются URI. У каждого RDF-утверждения три части: субъект, предикат и объект. Рассмотрим простое RDF-утверждение:

```
<http://aaronsw.com/> <http://love.example.org/terms/reallyLikes >  
<http://www.w3.org/People/Berners-Lee/Weaving/> .
```

Догадываетесь, что здесь утверждается? Первый URI – субъект. В нашем примере субъектом является "aaronsw (Аарон Сварц, автор статьи)". Второй URI – предикат. Он связывает субъект с объектом. В нашем примере предикат это "reallyLikes (очень нравится)". Третий URI – объект. Здесь объектом является книга Тима Бернерса-Ли "Weaving the Web". Поэтому приведенное RDF-утверждение говорит, что автору очень нравится "Weaving the Web".

Вы, возможно, заметили, что RDF-утверждение может выразить практически все, что угодно, и что не имеет никакого значения, кто это говорит. Нет ни одного официального Web-сайта, который бы рассказал все о книге Бернерса-Ли или обо мне. Это приводит нас к одному важному принципу RDF, а именно, "все, что угодно, может быть сказано кем угодно о чем угодно". Информация распространена на Web'e, и два человека могут говорить даже противоречащие вещи – Боб может сказать, что Аарон любит книгу Тима, а Джон – что ненавидит. Это – предоставляемая Web'ом свобода.

Приведенное нами утверждение написано на языке N-триплетов ([N-Triples](#)), позволяющем писать простые RDF-утверждения. Однако официальная RDF-спецификация определяет XML-представление RDF, которое несколько сложнее, но говорит то же самое:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:love="http://love.example.org/terms/"
>
  <rdf:Description rdf:about="http://aaronsw.com/">
    <love:reallyLikes
rdf:resource="http://www.w3.org/People/Berners-Lee/Weaving/" />
  </rdf:Description>
</rdf:RDF>
```

Писать документы RDF, аналогичные этому, не самое простое дело, и, надо полагать, не каждый начнет скоро разговаривать на этом новом странном языке. Откуда же мы можем надеяться получить эту RDF-информацию? Наиболее вероятным источником являются базы данных.

В мире существуют тысячи баз данных, во многих из них содержится интересная, пригодная для машинной обработки информация. Административные органы хранят в базах данных записи об арестах; компании – информацию об инвентаризации; многие компьютеризованные адресные книги хранят имена людей и номера их телефонов в – вы уже догадались! – базах данных. Когда информация хранится в базе данных, очень легко задавать компьютеру вопросы о данных: "Покажите мне всех, кто был арестован в последние шесть месяцев", "Покажите мне список ваших гостей, у которых низкие такие-то показатели". "Составьте для меня список номеров телефонов людей с фамилией Джонс".

RDF идеален для публикации этих баз данных на Web'e. И когда их помещают на Web'e, каждому элементу базы данных присваивают URI, так что другие люди могут также говорить об этом. И теперь, разумные программы могут начать сопоставлять эти данные. Используя имеющуюся информацию, компьютер может начать связывать Боба Джонса, телефонный адрес которого имеется в вашей адресной книге, с Бобом Джонсом, арестованным на прошлой неделе, и с Бобом Джонсом, кто только что сделал заказ на 100000 игрушек. Теперь мы можем задавать вопросы, касающиеся сразу всех этих баз: "Дай мне номер телефона каждого, кто сделал заказ более чем на 1000 игрушек и был арестован в последние шесть месяцев".

### Схемы и онтологии<sup>2</sup>: схемы RDF, DAML+OIL и WebOnt

При любой работе с базами данных предполагается, что данные в этих базах почти безупречны. Мало (если вообще есть) систем баз данных, готовых к происходящему на Web'e беспорядку. Каждая система, которая "жестко закодирована" на понимание определенных терминов, почти наверняка быстро устареет или, по крайней мере, будет иметь ограниченное применение, так как постоянно изобретаются и определяются новые термины. Что будет, если кто-то предложит новую систему, оценивающую книги по шкале 1-10, вместо того, чтобы сказать, что кто-то эти книги "очень любит". Программы,

<sup>2</sup> Онтология — это структурная спецификация некоторой предметной области, ее формализованное представление, которое включает словарь (или имена) указателей на термины предметной области и логические выражения, которые описывают, как они соотносятся друг с другом.

См. <http://www.uran.donetsk.ua/~masters/2005/fvti/anhina/libraries/ontology.htm> (прим. пер.).

построенные на принципах старых систем, не будут способны обрабатывать новую информацию.

Что еще хуже, у компьютера или человека нет никакого способа узнать, что означает конкретный термин или как его можно использовать. Все эти URI бесполезны, если мы не описываем, что они означают. Вот здесь нам и нужны [схемы](#) и [онтологии](#). Схема и онтология это средства для описания смысла и связи терминов. Эти описания (естественно, на языке *RDF*) помогают компьютерным системам более свободно работать с терминами и решать, как их преобразовывать друг в друга.

Две тесно связанные друг с другом системы: схемы [RDF Schemas](#) и [DARPA Agent Markup Language with Ontology Inference Layer](#) – язык разметки агентов DARPA + онтологический слой вывода (DAML+OIL) были разработаны для решения этой проблемы. Например, схема может утверждать, что:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

# A creator is a type of contributor:
dc:creator rdfs:subClassOf dc:contributor .
```

(Здесь мы пользуемся нотацией<sup>3</sup> ([Notation3](#)), как надмножеством [N-Triples](#), что позволяет использовать больше аббревиатур и делать тем самым запись более краткой.)

Эта схема говорит, что *creator* (создатель) является подклассом (типом) *contributor* (помощник). Что нам это дает? Ну, скажем, например, что вы пишете программу для выбора всех *creator* и *contributer* для документов различных типов. Ваша программа пользуется словарем (dc: creator и dc: contributor) для понимания найденной информации. В один прекрасный день наплыва новичков ([newbies](#)) из AOL<sup>3</sup> начинают создаваться RDF-документы. Никто из новичков ничего не знает о dc: creator, поэтому они вводят свои собственные термины: ed: hasAuthor (имеет автора).

```
# The old way:
<http://aaronsw.com/> is dc:creator of
<http://logicerror.com/semanticWeb-long> .

# The new way:
<http://logicerror.com/semanticWeb-long> ed:hasAuthor
<http://aaronsw.com/> .
```

При обычной работе ваша программа просто проигнорирует эти новые утверждения, потому что она не сможет их понять. Однако, найдется добрая душа, достаточно умелая для того, чтобы перекинуть мостик между этими двумя словами, задав информацию об их конвертации:

<sup>3</sup> AOL - America On-Line. См. <http://logicerror.com/aol>

```
# [X dc:creator Y] is the same as [Y ed:hasAuthor X]
dc:creator daml:inverse ed:hasAuthor .
```

Эти утверждения говорят вашей программе, что `ed:hasAuthor` инверсия `dc:creator`. Из этого следует, что все, что нужно сделать вашей программе, это поменять местами объект и субъект и заменить `ed:hasAuthor` на `dc:creator`. Так как ваша программа понимает онтологии DAML, она может воспользоваться этой информацией и использовать ее при обработке всех утверждений с `ed:hasAuthor`, которые она не могла понять раньше.

Что касается 2001-09-03, то [W3C](#) организует рабочую группу WOW-G ([Web Ontology Working Group](#)). Перед этой группой стоит задача подготовить язык Web-онтологий, базирующийся на работах по схеме [RDF](#) и [DAML+OIL](#). С интересом ожидаем результатов работы этой группы.

## Логика

Теперь я буду обсуждать те части семантического Web'a, которые пока еще не разработаны. В отличие от предыдущего обсуждения, я не рассматриваю конкретные системы, но анализирую общее понятие, которое может лечь в основу (и это происходит уже сейчас) многих различных систем.

Хотя и приятно иметь системы, понимающие такие основные понятия как "подкласс", "инверсия" и т.д., было бы еще лучше, если бы мы смогли устанавливать любые логические принципы и дать возможность компьютеру рассуждать (делать логические выводы) с помощью этих принципов.

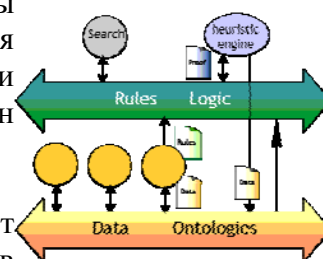
Вот пример: предположим, что одна компания решила, что если кто-то продает более 100 единиц её товара, то этот кто-то является членом клуба "Суперпродавец". Умная программа может прибегнуть к дедукции, используя следующее правило: "Джон продал 102 единицы товара, следовательно, он является членом клуба "Суперпродавец".

## Доказательство

Раз мы начинаем строить системы, отслеживающие логику, имеет смысл применить их для доказательства тех или иных утверждений. Большинство людей могут писать логические утверждения. В этом случае ваша машина может проследить эти семантические "связи" для построения доказательств.

Пример: Записи корпоративных продаж показывают, что Джейн продала 55 игрушек и 66 зубчатых колес. Система инвентаризации устанавливает, что игрушки и зубчатые колеса это различные продукты одной компании. Встроенные математически правила говорят, что  $55+66=121$ , и что 121 больше 100. А мы знаем, что каждый, кто продает больше 100 продуктов, является членом клуба "Суперпродавец". Компьютер объединяет все эти логические правила вместе и получает доказательство, что Джейн принадлежит клубу "Суперпродавец".

Создать такие доказательства очень трудно (это может потребовать отслеживания тысяч или даже миллионов связей в



семантическом Web'e), но проверить их не составляет труда. На этом пути мы приходим к построению сети информационных процессоров. Некоторые из них просто предоставляют данные для использования остальными. Другие, более находчивые, могут применить эти данные для построения правил. Самые находчивые и умные являются "эвристическими устройствами", которые прослеживают имеющиеся утверждения и делают заключения. Результат они помещают в сети как доказательство наряду с исходными данными.

### **Доверительность: Цифровая Подпись и WEB доверия**

Теперь вы, может быть, уже подумали, что весь этот план замечателен, но достаточно бесполезен, так как можно говорить что угодно. Кто будет полагаться на такую систему? Вы не пустите меня на свой сайт? О'кей, мне достаточно сказать, что я – Правитель мира и что у меня есть разрешение. Ввиду правила "кто угодно может говорить что угодно о чем угодно", разве можно меня остановить?

Вот здесь-то и появляются Цифровые подписи ([Digital Signatures](#)). Основанная на работах по математике и криптографии, цифровая подпись является доказательством того, что документ или утверждение написала (или с ними согласна) определенная персона. Ага! Итак, я снабжаю цифровой подписью все свои RDF-утверждения. Значит, вы можете быть уверены, что их написал я (или, по крайней мере, отвечаю за их аутентичность). Теперь вы просто говорите вашей программе, чьим подписям доверять, а чьим нет. Каждый может установить свой собственный уровень доверия или предоставить (иногда необоснованно) компьютеру право самому решать насколько верить тому, что он читает.

Считаю, что почти невероятно, чтобы вы доверили большинству людей использовать все возможности Web'a. Вот здесь к нам приходит на помощь "Web доверия". Вы говорите своему компьютеру, что вы доверяете своему лучшему другу Роберту. Ваш Роберт в Сети очень популярный человек, он доверяет очень многим людям. И, разумеется, все люди, которым он доверяет, доверяют другой группе людей. Каждый из этих людей доверяет аналогично своей группе и так далее. По мере того как от вас эти взаимоотношения доверия расходятся веером, формируется "Web доверия". Каждое из этих взаимоотношений имеет связанную с ним степень доверия (или недоверия).

Заметим, что недоверие может оказаться столь же полезным, как и доверие. Предположим, что ваш компьютер натолкнулся на документ, которому явно никто не высказывает своего доверия, но которому явно никто и не высказывает своего недоверия. Вполне вероятно, ваш компьютер будет доверять этому документу больше, чем тому, который явно помечен как не заслуживающий доверия.

Компьютер учитывает все эти факторы при решении вопроса, насколько можно доверять некоторой информации. Он может сделать этот процесс по вашему желанию прозрачным или нет. Например, вам может показаться достаточным простой просмотр на дисплее "вверх/вниз". Кто-нибудь возможно стал бы настаивать на сложных объяснениях, включая описание некоторых или всех факторов доверия, влияющих на решение.

Бернерс-Ли ([Tim Berners-Lee](#)) предложил кнопку "О, да?" ([an "Oh, yeah?" button](#)), которая при нажатии заставляет компьютер попытаться привести причины, по которым нужно доверять обрабатываемым данным. Но решаете ли вы сами или оставляете это на усмотрение компьютера, необходимая для принятия обоснованного решения информация, доступна вам через Web доверия.

## Заключение: захватывающие перспективы

Одним из прекраснейших свойств Web'a является то, что он предоставляет много различных вещей многим различным людям. Грядущий семантический Web тысячекратно умножит эту многосторонность. Для одних определяющими особенностями семантического Web'a будет свобода, с которой ваш PDA, ваш ноутбук, ваш настольный компьютер, ваш сервер и ваш автомобиль будут общаться друг с другом. Для других это будет автоматизация принятия корпоративных решений, что раньше требовало утомительного ручного труда. Для еще кого-то это возможность выбрать из документа на Web'e то, что заслуживает доверия, и исключительная легкость, с которой мы сможем получать ответы на свои вопросы - процесс, связанный в настоящее время со многими огорчениями.

Какова бы ни была причина, почти каждый может найти довод поддержать эту захватывающую перспективу семантического Web'a. Ну конечно, отсюда туда путь долог – и нет никакой гарантии, что нам удастся его пройти, – но мы все-таки кое-что сделали. Возможности бесконечны, и если даже нам не удастся реализовать их все, это путешествие будет, безусловно, само по себе наградой.

## Признательности

Моя благодарность Сину Пальмеру ([Sean B. Palmer](#)) за просмотр первого наброска этой статьи. С тех пор он опубликовал статью [The Semantic Web: An Introduction](#), (Семантический Web: введение), использующую материалы этого наброска. Джим Хендер ([Jim Hendler](#)) вместе со мной опубликовал статью в струе этого наброска: "Swartz, A. and Hendler, J.A Network of Content for the Digital City. Proceedings Second Annual Digital Cities Workshop, Kyoto, Japan, October, 2001." (Сеть контента для Цифрового Города). Чарлз Муна ([Charles F. Munat](#)) был весьма любезен в редактировании моего манускрипта, последняя версия обновлена дополнительной и новейшей информацией.

**Linked from:** [RDF Issues](#) | [The Semantic Web](#)

**Links to:** [Aaron Swartz](#) | [America On-Line \(AOL\)](#) | [Anatomy of an URL](#) | [DARPA Agent Markup Language with Ontology Inference Layer](#) | [Digital Signatures](#) | [Extensible Markup Language](#) | [Jim Hendler](#) | [N-Triples](#) | [Newbie](#) | [Notation3](#) | [Ontology](#) | [RDF Schema](#) | [Resource Description Framework](#) | [Sandro Hawke](#) | [Schema](#) | [Sean B. Palmer](#) | [The Semantic Web \(for Web Developers\)](#) | [Tim Berners-Lee](#) | [URI Schemes](#) | [Weaving the Web](#) | [WebOntology \(WebOnt\)](#) | [World Wide Web Consortium](#) | [XML Namespaces](#)

**Last Modified:** 2002-05-01 19:12:37

Part of [LogicError](#). Powered by [Blogspace](#), an [Aaron Swartz](#) project. [Email the webmaster with problems.](#)